electronics

MDPI

Article

The High Performance of a Task Scheduling Algorithm Using Reference Queues for Cloud-Computing Data Centers

Chin-Ling Chen [1,2,3], Mao-Lun Chiang [4,*] and Chuan-Bi Lin [4,*]

1 School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China; clc@mail.cyut.edu.tw
2 School of Information Engineering, Changchun Sci-Tech University, Changchun 130600, China
3 Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung City 41349, Taiwan
4 Department of Information and Communication Engineering, Chaoyang University of Technology, 168 Wufeng District, Taichung City 41349, Taiwan
* Correspondence: mlchiang@cyut.edu.tw (M.-L.C.); cblin@cyut.edu.tw (C.-B.L.)

Received: 16 January 2020; Accepted: 19 February 2020; Published: 21 February 2020

Abstract: As the network technology continues to grow at a high rate of speed, the traditional network topology is improved with novel distributed topologies such as the Cloud computing network. A cloud computing environment consists of a huge number of processors and memories, high-speed networks, and various application services to provide a lot of services over the Internet for users. However, many services need to search for suitable service nodes, and the workload of each node can be unbalanced. Based on the reason above, the Reference Queue based Cloud Service Architecture (RQCSA) and Fitness Service Queue Selection Mechanism (FSQSM) are proposed to handle more tasks, lower the makespan and queue waiting time, and improve efficiency. Moreover, the tasks can be distributed more evenly to avoid overloading cluster managers and lower the efficiency of the system.

Keywords: cloud computing; dynamic election mechanism; load balance

1. Introduction

In the early days, the distributed system was composed of a certain number of servers. With the continued development of the Internet and increased user demands for network services, the service providers gradually become unable to fulfill the diverse and huge demands of their customers. In order to fulfill those demands, a new concept of grid computing is proposed based on the distributed system [1–7] as a variant of grid computing, called "cloud computing" [1–4,7,8]. However, both grid and cloud computing are an extension of distributed computing. Grid computing focuses on how to integrate many heterogeneous platforms. The cloud computing stresses on how to compose a huge distributed system by using remote resources on the Internet under the condition of limited local resources. And the computers from all over the world are cooperated to provide user-oriented services. The goal is to satisfy the increasingly large demands of Internet service users, which will be an important direction of Internet developments for the near future. Moreover, cloud computing revamps the old structure of the distributed system.

The cloud computing provides Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) by charging the user or rental. IaaS is a virtual computing infrastructure. The users can rent the IaaS from the setup interface and build systems on it. The most famous example is the AWS EC2 [1,3,4]. Such platform services enable users to use supplier-provided API (Application

Programming Interface) to develop applications without building the system platform or the operating system. For example, the Windows Azure users can design programs such as .NET and SQL without setting up Windows Server in the engine room. Another common example is Google App Engine, which provides environments of JAVA and Python language to the programmers who can deploy desired programs on the platform without worrying about the operation of the servers in the back end. Unlike the two examples above, SaaS is a developed cloud computing software used to provide services to users through the Internet [9]. Take Google Earth as an example, the users do not need to install the map files of the world on their local devices, they can just download maps from the server when needed.

Therefore, cloud computing is the most important trend currently. The service providers can supply large computing resources, development platforms, and services, as shown in Figure 1. In various fields, the type of cloud computing can be sorted into three categories according to its applications:

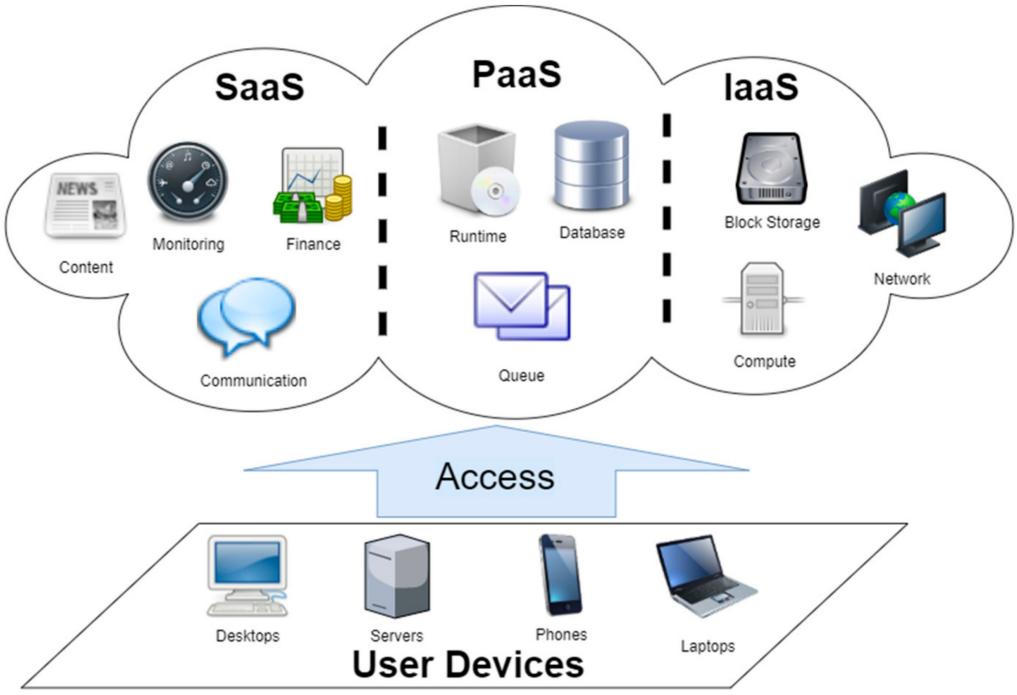

Figure 1. The infrastructure of cloud computing.

(1) Software as a Service (SaaS): The service providers bring the frequently used programs onto the Internet for the users, such as Google Map and Google Doc. The users do not need to install the software on their device, they just need to use the Internet as a platform to operate the software.

(2) Platform as a Service (PaaS): This service provides a specific computing platform for users to program and execute. For example, Google App Engine supports applications written by JAVA. Through PaaS, users only need to organize completed programs on the platform and obtain expected results.

(3) Infrastructure as a Service (IaaS): This service provides large computing or storage resources through virtualization technology. The users can rent those resources from the providers. The advantage of this service is that the users do not need to worry about the expensive cost of building servers. Instead, they only need to rent the resources.

As a result, the presence of cloud computing is going to create new network services in an innovative way. The business owners do not need to have a huge capital to invest in hardware anymore and can focus on innovation to create business values.

Cloud computing is a large and service-oriented distributed architecture. Therefore, how to enhance the computing power and the service efficiency of storage resources is an important subject. The goal of this study is to design a Reference Queue based Cloud Service Architecture (RQCSA) with

Fitness Service Queue Selection Mechanism (FSQSM) so that the system can provide more services and lower the waiting time and the total completion time in which the whole service efficiency can be improved.

Section 2 is about literature discussions that explain the cloud structures from past and related works. In the third section, the RQCSA and FSQSM are explained. The fourth section displays the experimental results. Finally, the fifth section gives the conclusion and lists future works.

2. Literature Review

This section discusses past works of literature. In Section 2.1, the discussion focuses on the recent and rapid development of related cloud computing. Many international giants in the field, such as Google, IBM, Microsoft, Yahoo, and Amazon are preparing for cloud computing and propose their own cloud services, respectively. Generally speaking, the clouds can be sorted into three categories according to its characteristics: the public cloud, private cloud, and the hybrid cloud. The differences between those clouds will be discussed in Section 2.2. Additionally, there will be a comparison between the past network structure and the currently popular hierarchical network in which the pros and cons of those structures will be shown. Finally, the related scheduling algorithms, such as the Opportunistic Load Balancing, Minimum Execution Time, Min-Min, Random, and Round-Robin, will be introduced in Section 2.3.

2.1. The Cloud Computing

Cloud computing network [1,3,4,7,8] is an extension from the grid computing environment [1,4,7] that is different from the traditional distributed computing environment. The main concept of cloud computing is to provide services in real-time according to users' demands by sharing hardware resources and software programs. In recent years, there have been many well-known companies involved in the field of cloud computing, such as Google, IBM, Microsoft, Yahoo, and Amazon [3,4,10–12]. Google, the number one Internet search engine in the US, utilizes the concept of cloud computing a long time ago on its own services, such as Gmail, YouTube, Google Docs, Google Talk, Google Calendar, and Google Gadget, etc. Furthermore, in October 2007, Google and IBM invested 15 million dollars together to build a large-scale clouding data center called Google 101. In 2008, Google set the cloud computing as its future development strategy, which was hinted by entering the communication industry and releasing the G-phone. From the aggressive actions and arrangements that Google put on the cloud applications, we believe that Google will continue to be a leader in the future market after the improved business model of connecting "end-to-cloud" is completed.

Besides Google, IBM is entering the cloud computing with its Blue Cloud. The main idea behind Blue Cloud lies not in providing services on the user end. Instead, it focuses on how to provide cloud-required hardware and software for enterprise customers so they can arrange their computing assignments into different modules and execute more efficiently. By achieving this, Blue Cloud can help enterprises to resolve the system loading issue during the peak and off-peak hours. By coupling Blue Cloud with supports from Google's clouding data center all over the world, the service will be more completed. On the other hand, Microsoft proposes a different operating model, which is called "Software + Service." Microsoft's new operating system, Azure, will combine Live Mesh to develop new features and integrates various Live Services, such as Windows Live Messenger, Live Search, etc. Moreover, Azure allows software developers to log onto Microsoft's data center and run their programs without the local servers. And those features will make Azure a foundation of Microsoft's online services and will prepare the company to move forward to cloud computing.

For Yahoo, it also introduces its cloud structure—Hadoop [3,13,14]. Hadoop is utilized on Yahoo's own two thousand servers to build the webpage index data to handle more than five petabytes of webpage content. Moreover, Yahoo positions its cloud product as Consumer Cloud Computing by providing online message services, such as Yahoo! Live, Yahoo! One Connect, and News Globe. Additionally, the upcoming Yahoo Application Platform provides an open platform for developers to

write and run their programs online. Last, Amazon couples its Amazon EC2 and Amazon S3 (a storage service) through virtualization technology to provide various formats of web hosting and storage as its web services. The developers can rapidly install and execute needed services and only pay for used resources and time in which businesses can lower their operating costs largely.

In addition, there are three types of clouds: the public cloud, private cloud, and hybrid cloud. Under the public cloud, service providers provide available resources (including applications and storage) over the Internet, such as Microsoft Datacenter. It uses the Windows Azure Platform and the API/Services within the platform. In such a PaaS structure, clients do not need to worry about the setup of the operating system and back-end environment. In this case, clients only need Internet-connected web browsers to obtain services, hence, the documents provided by the supplier is very important. For the private cloud, the clients have their own data center and usually hire a network and system professional to maintain and manage the servers and infrastructures. Therefore, the private cloud is usually designed and customized according to the client's specific needs. The advantage of customization is that the data will be more secured. On the other hand, the client will have a large expense in setting up the system and have few options for resources. The public cloud usually has the third parties to provide useful resources for the clients. The advantage of this is that its service is customizable and costs much less than the private cloud in terms of the construction, but the security of the data is questionable. The hybrid cloud is the combination of both the public and private clouds. There are a variety of structures in the Internet environment, and the hierarchical structure is one of the more suitable structures for cloud computing. In the next section, we will discuss the differences between many network structures.

2.2. The Hierarchical Virtualization Topology

Currently, every network structure has its own strengths and drawbacks. Among the common network, topologies are the star topology, ring topology, and hierarchical topology [3,15–18]. The star topology is built through servers to provide specific services to the customer end. Due to only one server-end equipment in a star topology, it is usually easy to become a bottleneck of the system. To solve this problem, many servers must be connected to form a ring topology and balance the load of the server end. This can also increase the number of serviced customer's ends. However, because of a longer routing time, the ring topology slows down the system performance when searching for information, and the hierarchical topology can avoid the drawback mentioned above. The main advantage of the hierarchical topology is that its upper layer nodes can control the data of its lower layer nodes and arrange jobs according to the capacities of the nodes of the lower layer. An example will be the hierarchical virtualization topology, which is presented as Figure 2 [7].

Under this network environment, the traditional Internet Service Providers (ISP) can be divided into two types: Infrastructure Providers (InPs) that manage the physical infrastructure, and network Service Providers (SPs) that create virtual networks to leverage resources obtained from InP to provide end-to-end network services. It provides a virtualization network and allows users or service providers to develop virtual networks that fit the application requirement of the user end. The nodes of the upper layers receive a request from the user end and distribute them to the suitable physical nodes in InPs group according to their resource statues.

In addition, Amazon Web Services also provides cloud computing infrastructure for businesses to rent. This platform has a huge node cluster and can respond to the requests from the user end [11]. For example, the computing power, storage capacity, and other applications are shown in Figure 3. The users can also purchase virtual hardware from Amazon Elastic Compute Cloud (Amazon EC2) based on their own needs and enter the cloud through Amazon EC2 Interface [3,18]. Then, the cloud controller will distribute jobs to the rented node clusters.

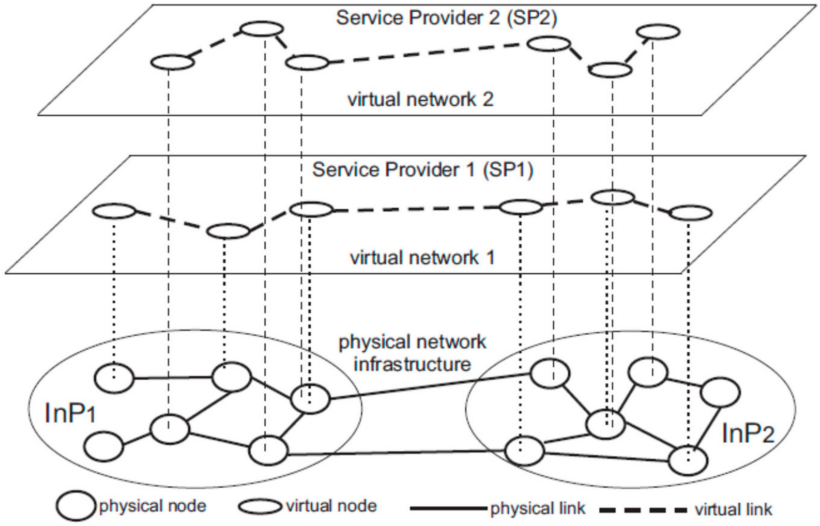

Figure 2. The example of hierarchical virtualization topology.

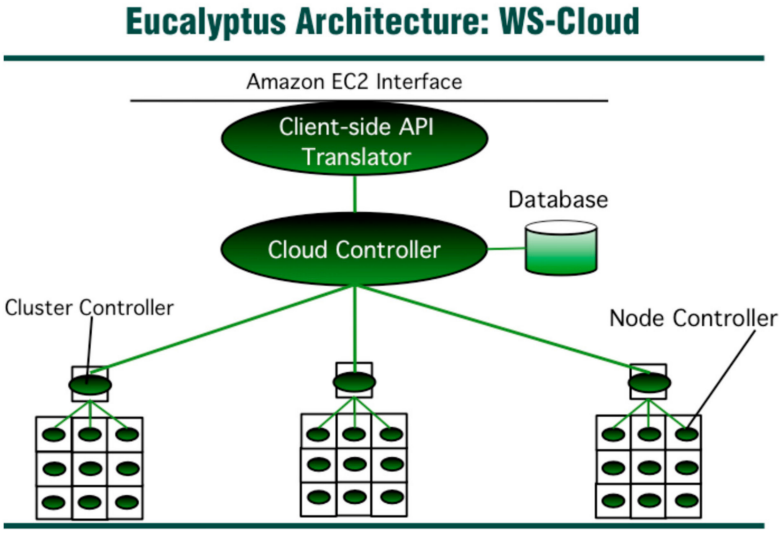

Figure 3. Amazon Elastic Compute Cloud.

Although the hierarchical structure distributes assignments evenly to the nodes in the next layer, the cluster manager or controller can still easily become a bottleneck of distributing. If there are enormous user requests sent into the system at the same time, the service node in the cluster will be depleted rapidly, which will cause the system to be fully loaded and unable to service. Additionally, the load of the system increases instability and involves a much longer waiting time. In order to improve the current hierarchical cloud structure, this study proposes the Reference Queue based Cloud Service Architecture (RQCSA) to enhance the overall performance of the system.

2.3. The Related Study of a Scheduling Algorithm

In past studies [19–21], researchers used various scheduling algorithms to enhance the efficiency of the system. In this section, we discuss the characteristics of the following algorithms: the Opportunistic Load Balancing, Minimum Execution Time, and Min-Min. These algorithms were frequently discussed not only in past studies regarding distributed computing but also in studies on applications of cloud computing. However, in those algorithms, the executing time needs to be known before the execution, which increases the system load and response time. The detailed explanation is in the next paragraph.

The Opportunistic Load Balancing (OLB) tends to keep every node busy by assigning jobs to available nodes without considering its current workload [15–17,22]. The biggest advantage of this algorithm is that it is quite fast. However, the makespan of it is longer than others because it does not consider the expected task execution time of each job.

The Minimum Execution Time (MET) tends to give each job the best node-support [15,23–25]. So, it does not consider the current workload of nodes and randomly assigns jobs to the nodes with the shortest execution time. However, this action may cause an unbalance between each node and is not suitable for the heterogeneity computer system.

The Min-Min establishes the minimum-minimum completion time for each unscheduled job and assigns them to nodes with the shortest completion time. In this case, the makespan will fall behind due to the extra computing cost in order to calculate the minimum-minimum completion time for each job [15,23–25]. However, the MET algorithm only considers the node with the shortest execution time and assigns a large number of jobs to it, and this action caused the unbalanced distribution and the increased queue waiting time. For the Min-Min algorithm, it can achieve a better makespan because it calculates the minimum execution time for each job before assigning them. However, the calculation cost of Min-Min will be much higher than others if the number of jobs is huge and all are sent into the system at the same time. Besides, the traditional scheduling algorithms do not consider the jobs based on the burst time. The process does not terminate due to there is the next waiting job in the queue. However, the jobs submitted by the user are of equal importance, so if they can be adjusted according to their needs, they will help to enhance the satisfaction of users. This is because the queue manager is part of cloud computing and it manages the utilization of all resources in the cloud. It is also an important role in balancing the workload of server nodes and tracking currently running jobs. Therefore, the concept of queue management [19] is used to our Reference Queues (RQ) to enhance the performance of the system.

From the above description, we know that all of those three algorithms need to obtain the completion time of the jobs before assigning them. However, in the cloud computing environment, users' requests could be sent into the system all at one time, and this will cause extra load for the system. Therefore, those algorithms may not be suitable for an environment that requires rapid process and fast response, such as the cloud computing environment. In order to be close to reality and improve those algorithms, this study uses a large number of random tasks to experiment and apply both the Random and Round-Robin algorithms to test the performances. The main idea is to be able to accept more jobs and lower the makespan and queue waiting time. The reason we chose the above algorithms is that those algorithms are often compared in past experiments. The main concept of the Random algorithm is that it does not take many factors into account and only distributes jobs randomly. The advantage of the Round-Robin algorithm is that it is much fairer than the MET scheduling because it distributes jobs in rotation so that the jobs are distributed evenly. However, there is a disadvantage of the Round-Robin: it does not take the workload of the service nodes into account. Therefore, this study proposes the Fitness Service Queue Selection Mechanism (FSQSM) to improve the distributions and lower the queue waiting time and makespan. The proposed method will be described in the next chapter.

3. The Study Method

There are two sections in this chapter. In Section 3.1, this study designs a structure called the RQCSA and expects that the cloud resources can be shared through the Reference Queues. In Section 3.2, we explain the FSQSM, which is based on the RQCSA structure and expects the FSQSM to improve the service efficiency, such as increasing the number of jobs by distributing jobs more appropriately. Moreover, the makespan and queue waiting time are also likely to be shorter. Last, the procedures of the algorithm will be demonstrated. The detailed description of the proposed cloud structure and algorithm is in the next section.

3.1. Reference Queue Based Cloud Service Architecture (RQCSA)

The RQCSA is proposed to enhance cloud efficiency by using a better-distributed algorithm. The architecture is shown in Figure 4. The following are explanations and functions of each layer:

- Cloud Interface (CI): For receiving requests from the users.
- Cluster Manager (CM): Selecting the head cluster from the cloud service cluster. The CM receives jobs from the CI and distributes jobs to the designated cloud service cluster for executing.
- Cloud Service Cluster (CSC): This is the main service node cluster that receives jobs from the CM and executes them.

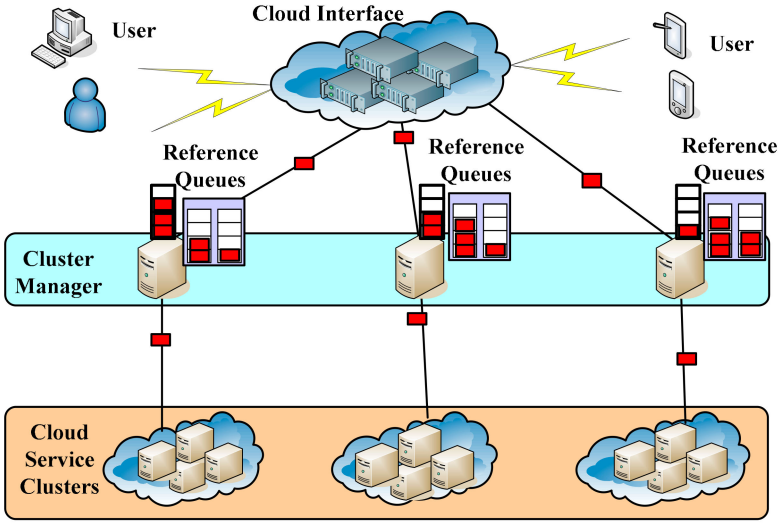

Figure 4. Reference Queue based Cloud Service Architecture (RQCSA).

The CI is the gateway that receives all jobs from the users. Then, the jobs are distributed to the second layer, which is the CM layer by the First Come First Service (FCFS). The main function of the CM layer is to save jobs in the queue temporarily, then distribute jobs through the Fitness Service Queue Selection Mechanism (FSQSM). Each CM in this layer not only has its own queue but also an n-1 Reference Queues (RQ). As shown in Figure 4, this study uses 3 CMs as an example, and each CM has 2 RQ, respectively. The main function of the RQ is to record the remaining number of other queues. In the CSC layer, there are service clusters that are composed of several nodes and are used to execute jobs. Therefore, by using the RQ function from the second layer and the FSQSM, every resource of CSC can be shared, and the system will be able to handle more jobs. In the next section, a detailed description of the FSQSM will be introduced.

3.2. Fitness Service Queue Selection Mechanism (FSQSM)

The architecture of the FSQSM is shown in Figure 5, and the parameters are listed in Table 1. When a job is sent to the CI, it is distributed to the CM and stored temporarily through the FCFS. Meanwhile, the remainder queue size q_i will be checked by the system. When $q_i = 0$, that means the queue of the CM_i is fully loaded. Then, the system will search for the most remaining queue size rq_{ij} from other $RQCM_{ij}$. If there is more than one rq_{ij} that has the same remaining amount, the Round Robin (RR) method will be applied to select an rq_{ij}. Moreover, the RR will distribute and temporarily save the Overload Task (OT) in the CM_i for the next available node. Last, the RR will update all the statues of CM_i and RQ.

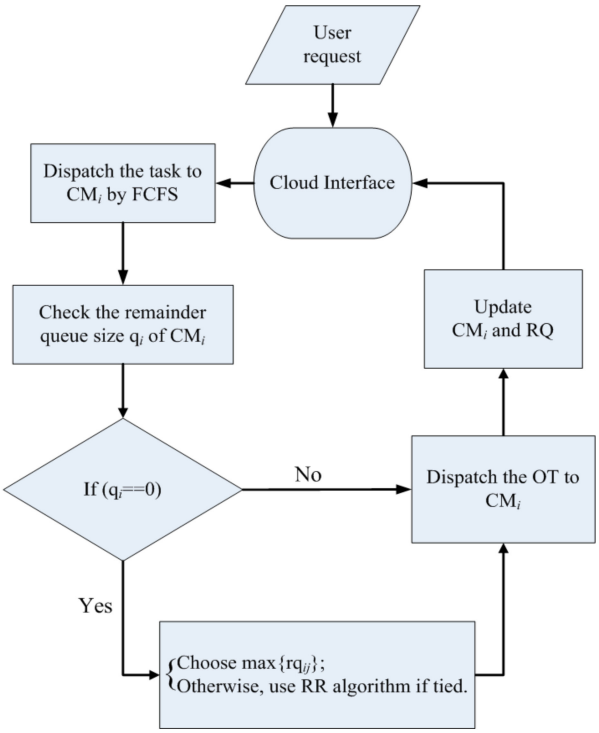

Figure 5. The Fitness Service Queue Selection Mechanism (FSQSM) flow chart.

Table 1. The FSQSM parameters and definitions.

Parameters	Definition
CM_i	Cluster manager i; ($1 \leq i \leq n$)
$RQCM_{ij}$	The RQ of CM_i; ($j = i-1$)
q_i	Remainder queue size of CM_i
rq_{ij}	Remainder queue size of $RQCM_{ij}$

In Figure 6, the following four rounds have demonstrated the task execution and scheduling process caused by new coming tasks. The first two rounds describe the distribution mechanism of the FSQSM and show the example of RQ searching when the CM is fully loaded. The third and fourth rounds illustrate how to select the RQ by rotation when the RQs have the same remaining amount. Moreover, this example not only assumes that the execution time of each task is randomly generated and marked on the task but also defines the unit time as the CSC execution time in each round. In Round 1, the CM_3 queue is fully loaded. The CSC1 and CSC3 are still executing the previous tasks and needs one-unit time. Now, a new coming task enters a fully loaded CM_3, so the RQ mechanism is started and searches for available nodes. In the case of Round 1, the task will be assigned to CSC_2 and perform services

Afterward, three new coming tasks are assigned to the CM_1 and CM_2 from the CI in Round 2. Meanwhile, the CSC1 and CSC3 have completed the previous tasks and the statuses are available, so the tasks from CM_1 and CM_3 have assigned to CSC1 and CSC3, respectively. For CSC2, a task cannot be assigned to it because it still needs a two-unit time to finish its work. Furthermore, the tasks will be waiting in the queue of CM_2 because the CM_2 is not fully loaded. Subsequently, the next round will show an example of rotation caused by the new coming tasks, so the detailed execution processes of CSCs will be omitted.

In Round 3, the tasks in CSC1 and CSC3 have been finished, but the CSC2 still needs a one-unit time to finish the previous tasks. The execution statues of CSCs are saved during there are no new coming task in this period. In addition, CSC2 is only capable to take two tasks at one time. However,

there are three new coming tasks are sent to the CSC2. Therefore, by checking the RQ, the two RQ with the same remaining amount will be assigned for the tasks based on the rotation defined by the algorithm. As a result, the task which requires a five-unit time to be finished will be sent to the CM_1.

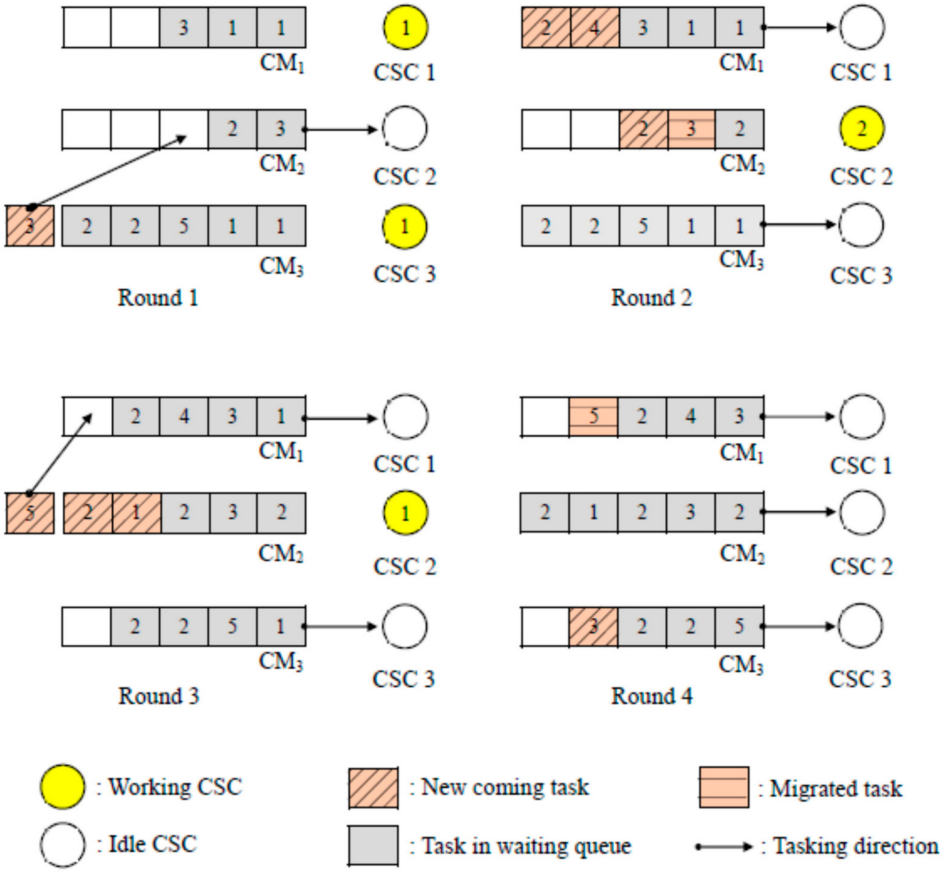

Figure 6. The operation of FSQSM.

In Round 4, the CSC2 already finished the prior task, and the CI distributes a new coming task to the CM3. In this case, because the CSC1, CSC2, and CSC3 are all in idle, the CM_1, CM_2, and CM_3 will distribute tasks to the CSC1, CSC2, and CSC_3, respectively.

Through the proposed Fitness Service Queue Selection Mechanism (FSQSM) and the use of RQ, the queue waiting time and makespan will be shorter. As a result, the system will be able to handle more tasks. In the next chapter, the proposed method will be verified by the performance analysis.

4. Experimental Simulation

The main focus of this chapter is to perform experimental simulation based on the proposed cloud architecture and scheduling algorithm and then analyze the results. This chapter can be divided into two sections. In Section 4.1, the experimental environment, including the specifications of the host device and analysis tools, will be explained. Additionally, the related experimental setting will be described. In Section 4.2, the performance analysis of the FSQSM, Random, and Round-Robin algorithms will be introduced. The analysis will be based on the number of tasks, queue waiting time, makespan and Mean Average Deviation. A detailed description will be provided in the next section.

4.1. The Experimental Environment

In this section, the RQCSA architecture will be analyzed based on the performance. In addition, a comparison of the performance will be done between the FSQSM and traditional algorithms, such as the Random and Round-Robin. Therefore, several analyses, such as the number of tasks, queue

waiting time, and makespan, will be executed. Moreover, this experiment even uses the Mean Average Deviation (MAD) to calculate the distributed task numbers and verify if the FSQSM distributes tasks to the CM evenly to avoid the overload of a single CM in which the overall efficiency will be lower [24–26]. The experimental host and analysis tools are listed in Table 2.

Table 2. The specifications of the experimental host and analysis tools.

Item	Specification
CPU	Intel Core2 Duo 2.53 GHz
Memory	2 GB
Operating System	Ubuntu 10.4
Language	C
Analysis Tool	KyPlot

In the experimental environment, there will be 16 CMs to manage all the service nodes in the cloud. And the queue size of each CM will be 1024 slots. Next, three different task amounts will be assigned: Low (<3000 tasks), Medium (4000–7000 tasks), and High (>8000 tasks), and the experiment will perform 1000 simulations on each amount level, respectively. In the trail environment, all execution time is randomly assigned and lost tasks will be distributed again.

In the calculation of the experiment, the number of tasks is defined by the number of executable tasks for each algorithm in the same amount of time. The queue waiting time is defined as the total time of every task wait in the CM queue. The makespan is defined as the total time that each task needs when traveling from the CI, through the CM queue, to the CSC.

4.2. Performance Analysis

In the first experiment, the FSQSM, Random, and Round-Robin are going through the queue waiting time test under the proposed RQCSA architecture. The results are shown in Figure 7.

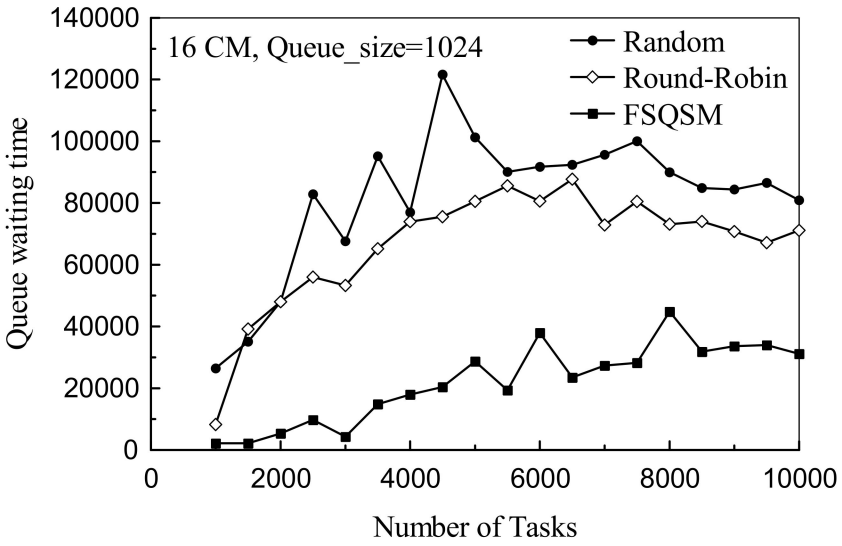

Figure 7. Queue waiting time, Cluster Manager (CM) = 16, queue size = 1024.

The first discussion will be on the Random and Round-Robin algorithms. When the task numbers gradually increase starting from 1000, the queue waiting time of those two algorithms is much longer than the FSQSM. The main reason for the long waiting time is that those two algorithms do not consider the remaining amount of CM queue, which frequent task reassignments and reduce the performance of the entire system. Moreover, the tasks are distributed by randomly selecting the CMs

when using the Random algorithm in which the queue waiting time of the Random will be higher than the Round-Robin. The FSQSM has the best queue waiting time performance because it takes the remaining amount of CM queue into account.

However, when testing the makespan, the experiment is performed based on the number of tasks between 2000 to 10,000 in order to obtain a more accurate analysis of the pros and cons of the three algorithms, as shown in Figure 8. The Random algorithm has a longer makespan because its random selecting method may select an RQ with no remaining amount when distributing tasks, in which the tasks may be missing and lead to a task re-distribution. Under the Round-Robin method, although it selects an RQ by rotation, it still does not take the RQ's current remaining amount into account. Therefore, the makespan of the Round-Robin is still longer than the FSQSM. The proposed FSQSM considers the remaining amount of all RQs and selects the RQ with the most amount to distribute tasks which makes the makespan much shorter.

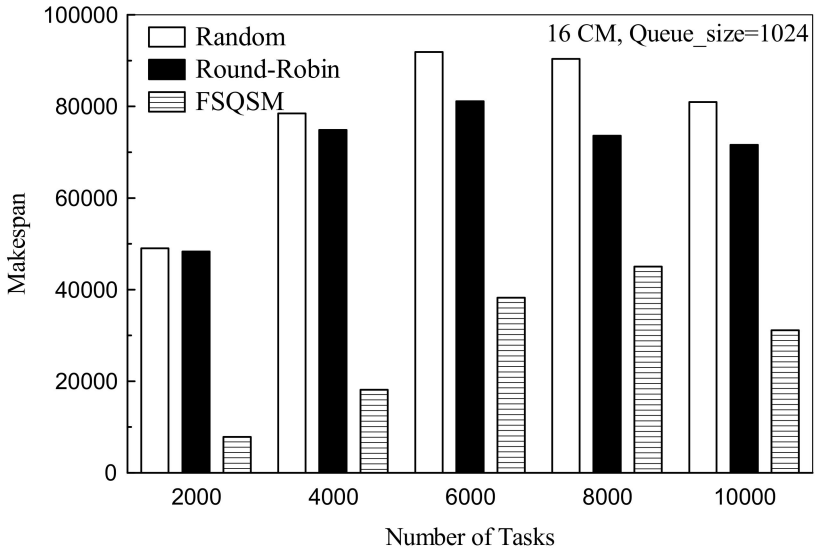

Figure 8. Makespan, CM = 16, queue size = 1024.

The goal of testing the number of tasks is to verify the number of executed/completed services of the proposed FSQSM, Random and Round-Robin in a specific period. As the simulating time increases, the Random and Round-Robin serves fewer tasks than the FSQSM, and the reason for this is that the first two algorithms do not take the current remaining amount of RQ into account when distributing tasks and the system will be slowed down because the selected RQ may be fully loaded and causes task re-distributions. On the other hand, the FSQSM takes the remaining amount of RQ to avoid the drawback. There are some inconsistent results that occur in Figure 9, and the reason for this is that the completion time of each task is randomly defined. That means that if the FSQSM happens to serve tasks with shorter completion times, it will be able to handle more tasks, otherwise, it handles fewer tasks.

Finally, this study uses the MAD value to determine whether these three algorithms evenly distribute tasks to CMs. The main purpose of the MAD value is to calculate the differences between the queue's remaining amount when an algorithm distributes tasks and the overall average queue's remaining amount. A larger MAD value indicates a more uneven distribution, and a smaller MAD value indicates that the tasks are distributed more evenly. In addition, a smaller MAD means that the system will not be overloaded due to an even distribution. Therefore, we need to design a task allocation experiment for MAD, and then use it to analyze the advantages and disadvantages of different task allocation algorithms. In order to find out the different effects of different tasks-amount levels (low, medium, high), the task execution time of this experiment is randomly generated, and the results of 1000 CM1–CM16 samples are accumulated to provide MAD for experiments. Therefore, the

remaining queue size of the CM_i under each of the three algorithms will be calculated in low, medium, and high task-amount levels, respectively. The parameters and calculation of the MAD are shown in Table 3 and Equation (1). Subsequently, the results of the experiment are shown in Figures 10–19 and are used to calculate the MAD values of each of the three algorithms. With the MAD values, the distributions of the three algorithms can be verified.

$$\sum_{i=1}^{n} |q_i - average(q_i)| \tag{1}$$

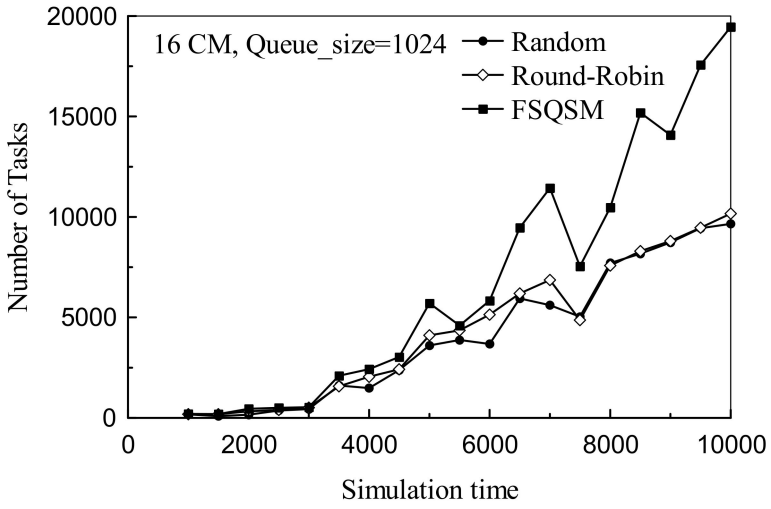

Figure 9. Number of tasks, CM = 16, queue size = 1024.

Table 3. The parameters of the Mean Average Deviation (MAD).

q_i	remainder queue size of CM_i
n	The number of CM
average(q_i)	Average of all q_i

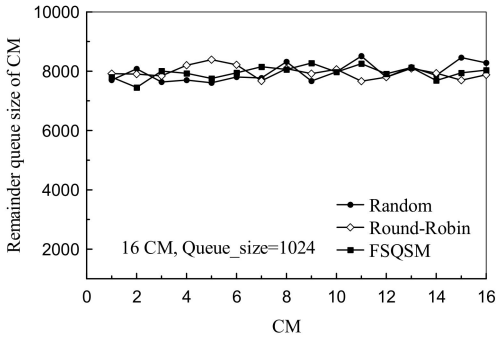

Figure 10. Remainder queue size of CM_i for 1000 tasks.

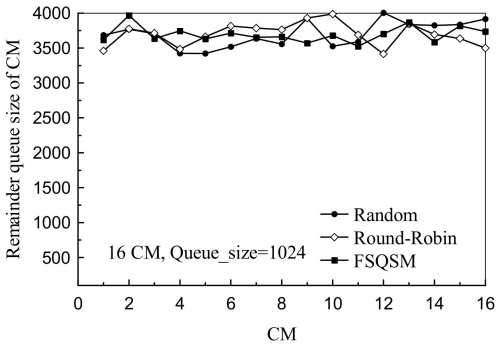

Figure 11. Remainder queue size of CM_i for 2000 tasks.

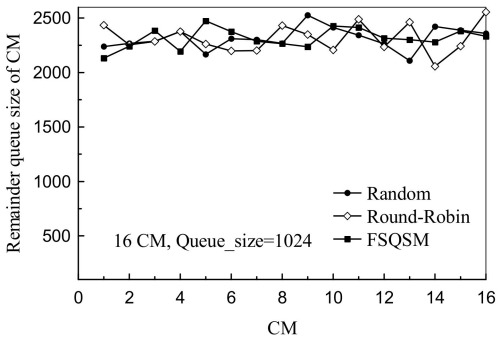

Figure 12. Remainder queue size of CM_i for 3000 tasks.

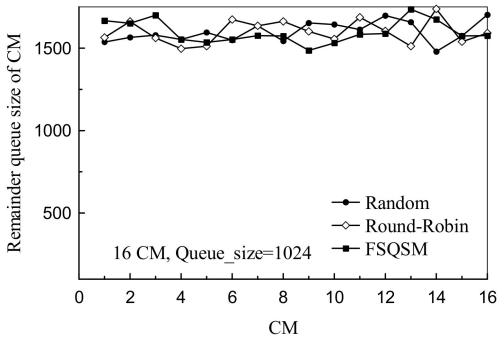

Figure 13. Remainder queue size of CM_i for 4000 tasks.

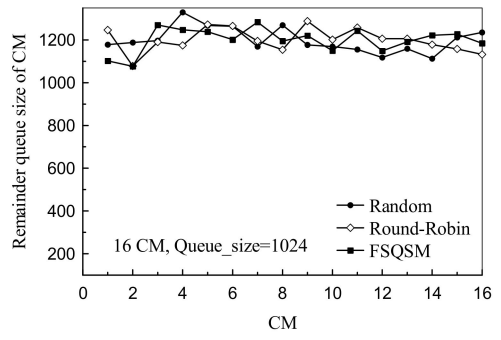

Figure 14. Remainder queue size of CM_i for 5000 tasks.

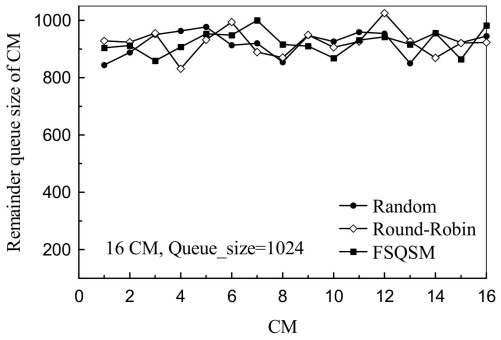

Figure 15. Remainder queue size of CM_i for 6000 tasks.

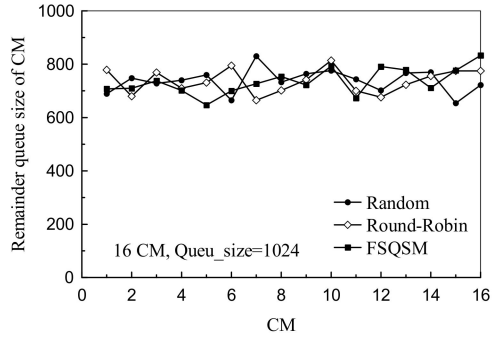

Figure 16. Remainder queue size of CM_i for 7000 tasks.

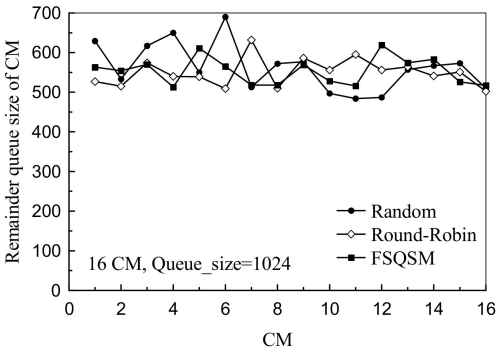

Figure 17. Remainder queue size of CM_i for 8000 tasks.

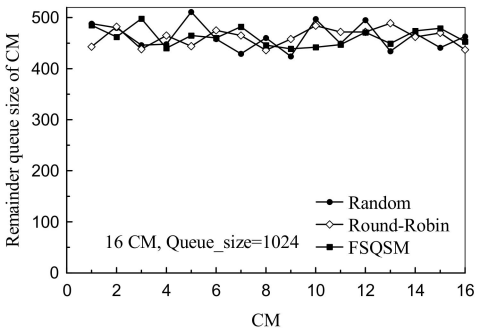

Figure 18. Remainder queue size of CM_i for 9000 tasks.

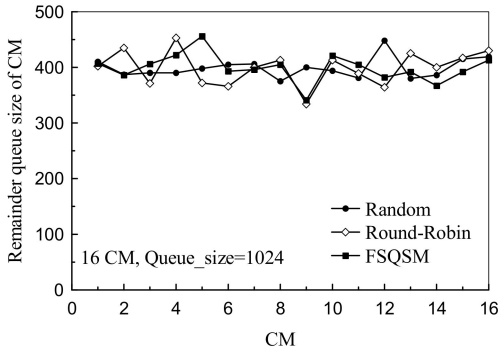

Figure 19. Remainder queue size of CM_i for 10,000 tasks.

In this study, the completion time is generated randomly, so the queue waiting time will be longer and cause a larger fluctuation of the remaining amount. Through the MAD values, the analysis can be more accurate and avoid the fluctuations. In Figure 20, the MAD values of the FSQSM are the lowest among the algorithms whether the task number is low, medium, or high. Therefore, the FSQSM is proven to be able to distribute tasks more evenly than others.

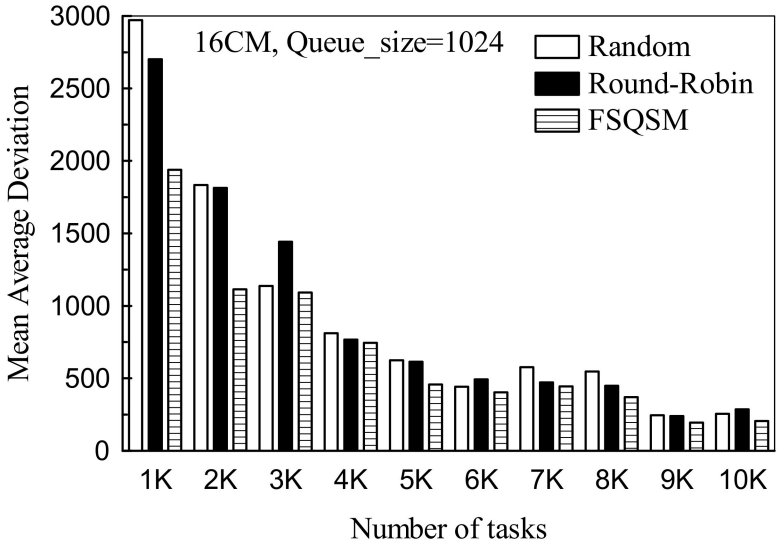

Figure 20. Mean Average Deviation (MAD) for all tasks

The results of the simulations show that the Random and Round-Robin algorithms are more similar. They all have worse performance than the FSQSM because of the random completion time and the ignorance of the RQ's remaining amount. Therefore, this study can utilize the FSQSM on the RQCSA architecture to increase the service amount, lower the makespan and queue waiting time, and improve the efficiency of the system. Moreover, the MAD values prove that the FSQSM distributes tasks more evenly and can avoid overloading CMs and lower the efficiency of the system.

5. Conclusions and Future Works

Due to the rapid development of the Internet and the incensement of the Internet service demands of users, the cloud concept based on the distribution system structure will be utilized widely. Therefore, how to provide better services to users and use lots of resources more efficiently is an issue that service providers cannot ignore.

Currently, the cloud computing environment has lots of resources. With that in mind, this study proposed the Reference Queue based Cloud Service Architecture and combines the architecture with the Fitness Service Queue Selection Mechanism to handle the huge cloud resources more efficiently. In the

study, we test the queue waiting time, the makespan, the number of tasks under the RQCSA structure, and use the MAD value to verify if the task distribution is even. The Random and Round-Robin algorithms do not take the remaining amount of RQ into account and may cause task re-distributions due to a fully loaded queue. In this situation, the queue waiting time will be affected and will lead to a longer makespan. The proposed FSQSM takes the remaining amount of the current RQ into account and distributes tasks based on the amount. This will not only lower the queue waiting time but also the makespan. Furthermore, the proposed FSQSM is able to execute more tasks at the same time and serves more users.

In conclusion, the proposed RQCSA and FSQSM are able to handle more tasks, lower the makespan and queue waiting time and improve efficiency. In addition, through the calculation of the MAD value, it is proven that the FSQSM can lower the differences in terms of the number of tasks each CM receives by taking the remaining amount into account when distributing tasks.

In the future, we will consider problems in which the service nodes are placed under a dynamic environment. For example, under the hierarchical network structure, the efficiency will be affected when the nodes are damaged and unable to distribute tasks more effectively. Therefore, how to maintain and manage service nodes effectively and enhance the tolerance, will be one of the directions worth exploring. Additionally, the current study only focuses on the distribution of the homogeneity nodes, so the task distribution and resource management of heterogeneous nodes are also topics that can be explored in the future.

Author Contributions: Conceptualization, M.-L.C.; methodology, C.-B.L. and M.-L.C. investigation, C.-L.C.; performance analysis, M.-L.C., C.-B.L. and C.-L.C.; writing—original draft preparation, M.-L.C.; writing—review and editing, C.-L.C. and C.-B.L.; supervision, C.-L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare re no conflict of interest.

References

1. Josep, A.D.; Katz, R.; Konwinski, A.; Gunho, L.E.E.; Patterson, D.; Rabkin, A. A view of cloud computing. *Commun. ACM* **2010**, *53*, 50–58.
2. Wang, J.; Chen, X.; Li, J.; Zhao, J.; Shen, J. Towards achieving flexible and verifiable search for outsourced database in cloud computing. *Future Gener. Comput. Syst.* **2017**, *67*, 266–275. [CrossRef]
3. Rajkumar Buyya, J.B.; Goscinski, A. *Cloud Computing Principles and Paradigms*; Wiley: Delhi, India, 4 April 2013.
4. Dinh, H.T.; Lee, C.; Niyato, D.; Wang, P. A survey of mobile cloud computing: Architecture, applications, and approaches. *Wirel. Commun. Mob. Comput.* **2013**, *13*, 1587–1611. [CrossRef]
5. Distributed Computing. July 2015. Available online: https://whatis.techtarget.com/definition/distributed-computing (accessed on 6 January 2020).
6. Guo, L.; Shen, H. Efficient approximation algorithms for the bounded flexible scheduling problem in clouds. *IEEE Trans. Parallel Distrib. Syst.* **2017**, *28*, 3511–3520. [CrossRef]
7. Duan, Q.; Yan, Y.; Vasilakos, A.V. A survey on service-oriented network virtualization toward convergence of networking and cloud computing. *IEEE Trans. Netw. Serv. Manag.* **2013**, *9*, 373–392. [CrossRef]
8. Yang, Y.; Zheng, X.; Chang, V. Semantic keyword searchable proxy re-encryption for postquantum secure cloud storage. *Concurr. Comput. Pract. Exp.* **2017**, *29*, e4211. [CrossRef]
9. Botta, A.; De Donato, W.; Persico, V.; Pescapé, A. Integration of cloud computing and Internet of things: A survey. *Future Gener. Comput. Syst.* **2016**, *56*, 684–700. [CrossRef]
10. Bailey, S.F.; Scheible, M.K.; Williams, C.; Silva, D.S.; Hoggan, M.; Eichman, C.; Faith, S.A. Secure and robust cloud computing for high-throughput forensic microsatellite sequence analysis and databasing. *Forensic Sci. Int. Genet.* **2017**, *31*, 40–47. [CrossRef] [PubMed]
11. Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & More. January 2020. Available online: http://amazon.com/ (accessed on 6 January 2020).

12. Microsoft Azure. January 2020. Available online: http://azure.microsoft.com/en-us/overview/ (accessed on 6 January 2020).
13. Shafer, J.; Rixner, S.; Cox, A.L. The hadoop distributed file system: Balancing portability and performance. In Proceedings of the IEEE International Symposium on Performance Analysis of System & Software (ISPASS), White Plains, NY, USA, 28–30 March 2010; pp. 122–133.
14. Hadoop Distributed File System (HDFS). January 2020. Available online: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (accessed on 6 January 2020).
15. Chen, S.L.; Chen, Y.Y.; Kuo, S.H. CLB: A novel load balancing architecture and algorithm for cloud services. *Comput. Electr. Eng.* **2017**, *58*, 154–160. [CrossRef]
16. Singh, S.; Chana, I. A survey on resource scheduling in cloud computing: Issues and challenges. *J. Grid Comput.* **2016**, *14*, 217–264. [CrossRef]
17. Kumar, N.; Singh, Y. Trust and packet load balancing based secure opportunistic routing protocol for WSN. In Proceedings of the 4th International Conference on Signal Processing, Computing and Control (ISPCC), Solan, India, 21–23 September 2017; pp. 463–467.
18. Toyoshima, S.; Yamaguchi, S.; Oguchi, M. Storage access optimization with virtual machine migration and basic performance analysis of Amazon EC2. In Proceedings of the 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, Perth, Australia, 23–23 April 2010; pp. 905–910.
19. Karthick, A.V.; Ramaraj, E.; Subramanian, R.G. An efficient multi queue job scheduling for cloud computing. In Proceedings of the 2014 IEEE World Congress on Computing and Communication Technologies, Trichirappalli, India, 27 February–1 March 2014; pp. 164–166.
20. Mathew, T.; Sekaran, K.C.; Jose, J. Study and analysis of various task scheduling algorithms in the cloud computing environment. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), New Delhi, India, 24–27 September 2014; pp. 658–664.
21. Salot, P. A survey of various scheduling algorithm in cloud computing environment. *Int. J. Res. Eng. Technol.* **2013**, *2*, 131–135.
22. Jyoti, A.; Shrimali, M.; Mishra, R. Cloud Computing and Load Balancing in Cloud Computing-Survey. In Proceedings of the 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 10–11 January 2019; pp. 51–55.
23. Pal, S.; Pattnaik, P.K. A Simulation-based Approach to Optimize the Execution Time and Minimization of Average Waiting Time Using Queuing Model in Cloud Computing Environment. *Int. J. Electr. Comput. Eng.* **2016**, *6*, 2088–8708.
24. Nayak, B.; Padhi, S.K.; Pattnaik, P.K. Static Task Scheduling Heuristic Approach in Cloud Computing Environment. In *Information Systems Design and Intelligent Applications*; Springer: Singapore, 2019; pp. 473–480.
25. Kumar, M.; Sharma, S.C. Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing. *Procedia Comput. Sci.* **2017**, *115*, 322–329. [CrossRef]
26. Sharma, S.; Tantawi, A.; Spreitzer, M.; Steinder, M. Decentralized allocation of CPU computation power for web applications. *Perform. Eval.* **2010**, *67*, 1187–1202. [CrossRef]