AN ATTRIBUTE-ORIENTED TASK SCHEDULING STRATEGY FOR IMPROVEMENT OF QUALITY OF SERVICE IN CLOUD COMPUTING

by

Jianying Miao
Bachelor of Science in Computer Science and Engineering
Jilin University of Technology, China, 1995

A thesis
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Science
in the Program of
Computer Science

Toronto, Canada, 2016

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

An Attribute-Oriented Task Scheduling Strategy for Improvement of Quality of Service in Cloud Computing

Jianying Miao
Master of Science in Computer Science
Ryerson University, 2016

Abstract

This thesis describes an innovative task scheduling and resource allocation strategy by using thresholds with attributes and amount (TAA) in order to improve the quality of service of cloud computing. In the strategy, attribute-oriented thresholds are set to decide on the acceptance of cloudlets (tasks), and the provisioning of accepted cloudlets on suitable resources represented by virtual machines (VMs,). Experiments are performed in a simulation environment created by Cloudsim that is modified for the experiments. Experimental results indicate that TAA can significantly improve attribute matching between cloudlets and VMs, with average execution time reduced by 30 to 50% compared to a typical non-filtering policy. Moreover, the tradeoff between acceptance rate and task delay, as well as between prioritized and non-prioritized cloudlets, may be adjusted as desired. The filtering type and range and the positioning of thresholds may also be adjusted so as to adapt to the dynamically changing cloud environment.

Acknowledgements

I would like to express sincere gratitude to my supervisor Dr. Vojislav Misic for his continuous support, encouragement and time throughout my graduate studies. Without his kind cooperation and support, completion of this thesis would not be possible.

I would like to thank my thesis committee members for their time, patience and proficiency in judging my thesis. Their valuable opinions were very helpful to improve my work.

I would also like to convey my gratitude to the faculty members of the Department of Computer Science, Ryerson University.

My sincere appreciation goes to staff members of Computer Science department and fellow graduate students for their continuous support.

Lastly, I am grateful for the continuous support and inspiration of my family.

Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Motivation

Cloud computing represents a rapidly growing trend in IT industry and it is progressing to allow users to share enormous computing resources over networks to achieve heavy and complex data processing in a cloud environment. Also, it may provide some attractive services such as the well-known pay-as-you-go mode that customers can simply buy computing services only when the services are needed, instead of buying and maintaining an expensive machine that is not often used.

Partly due to numerous potential attractions of cloud computing, optimistic promoters are very brave to promise customers a vast variety of services, and the promises often sound more than good enough. Nevertheless, fulfilling the promises strongly depends on the technical performance of cloud computing, especially on how to deploy a limited amount of cloud resources to a vast variety of or even unlimited tasks requested by customers that often demand fast service but low price. In other words, customers' satisfaction at the promised services has a strong reliance on the technical performance of cloud computing from the perspectives of resource allocation and task scheduling, although this sort of services do not require customers' knowledge of the technical detail of resources (e.g., a machine), because the resources have been friendly virtualized and interfaced by service providers (e.g., brokers or data centers) so that the customers are impressed that they are exclusively allocated a powerful machine.

For sake of description in the field of cloud computing, a task is often categorized as a cloudlet or divided into smaller cloudlets [1] [2] [3], and a resource is often expressed as a virtual machine (VM). Therefore, many past investigations have been conducted to allocate limited number of various VMs to enormous amounts of different cloudlets. However, most of them

mainly regard cloudlets and VMs as homogenous, i.e., the cloudlets or VMs are similar or uniform in size, urgency, priority, etc. Thus, relevant techniques are mostly conducted to balance the amount of cloudlets. Up to date, the varieties and distinctions of the characteristics and features of cloudlets and machines have been rarely considered for the VM allocation and cloudlet scheduling, to the best knowledge of this thesis author.

This thesis is, therefore, motivated to enrich the techniques of resource allocation and task scheduling by substantially considering the varieties and distinctions of the characteristics and features of cloudlets and machines. In this thesis, the varieties and distinctions are reflected by "attributes".

1.2 Objectives and Approaches

Consistent with the motivation, the overall objective of this thesis is to compromise the research scarcity by matching different cloudlets with different VMs based on different attributes and dynamic workload status of resources, so as to improve the quality of service for cloud computing from the perspectives of task scheduling and resource allocation. To this end, a strategy using thresholds with attributes and amount (TAA) is proposed in this thesis.

To approach the overall objective, the following sub-objectives and approaches are planned in a step-by-step way:

(1) Introduce and extend the concept of attribute and then design a VM allocation and cloudlet scheduling technique capable of filtering the attributes and amount of queuing cloudlets for optimal VM usage.

This is approached through setting up attribute-oriented thresholds upon the queuing system of the VM of interest. A simplified setup is a two-level attribute matching mechanism, i.e., a lower threshold (LoT) and an upper or higher threshold (HiT) that can match the attributes of tasks and virtual machines (VM) by comparing and filtering the attributes of cloudlets and machines. The maximum queuing capacity of the VM is also an important thresholding parameter. For example, if queue capacity is 100, we may set LoT to be 50 and HiT to be 80.

In detail, the two-level attribute matching is based on task attributes that can be prioritized according to dynamic requests and cloud environment. The tasks can be filtered into different groups according to their inherent or assigned attributes such as storage intensive, computation-intensive, GPU resources, and high-pay-for-priority, i.e., the attributes can be dynamically prioritized or custom defined according to user or provider's preference and dynamic status of the cloud environment.

The LoT provides a rough filtering of attributes and the HiT provides a stricter filtering. The functioning of these two thresholds can then provide suitable machines to different types of cloudlets. If the cloud environment is very busy, the HiT will be activated to make sure some very special cloudlets (e.g., with high pay or urgency) are always allocated the most suitable machine.

(2) Build a simulation environment to test the implementation of the proposed TAA strategy.

The algorithms for TAA are converted to codes which can successfully set up the abovementioned two-level thresholds with attribute filtering function and amount limitation for cloudlets. The codes are embedded in CloudSim, an open source tool kit widely used for cloud computing simulation. Parameters capable of characterizing the environment are selected to make the proposed TAA algorithm functional. Typical environment examples for arriving cloudlets are configured to make it analogous to a real world of cloud environment, so that cloudlet acceptance rate as well as average execution time using the TAA technique can be examined.

(3) Assess the performance improvement in resource allocation and task scheduling by using TAA strategy, so as to improve the QoS of cloud computing.

The assessment is based on the simulation results and a number of parameters such as acceptance rate reflecting the performance of attribute filtering, and average execution time representing the processing speed of a cloudlet of interest.

(4) Suggest future improvements and design rules for TAA, such as standardizing the attributes to form a protocol of building cloudlets. It is expected that the research of this thesis may provide inspirations to other investigators to further improve the performance of cloud computing.

1.3 Contributions

It is expected this thesis can deliver the following major contributions after approaching the aforementioned research objectives:

(1) Contribute an attribute-oriented VM allocation and cloudlet scheduling strategy (TAA) to the QoS branch of the cloud computing field, although the strategy is preliminary and far from mature.

(2) Improve the suitability between different cloudlets and different machines through using the TAA strategy to efficiently allocate the most suitable resources to the tasks with similar attributes.

(3) Quantify and assess the performance improvement in VM allocation and cloudlet scheduling by using TAA, although the assessment is based on a simplified version of TAA consisting of a LoT that roughly filters different attributes and a HiT that strictly screening different attributes. The performance improvement includes reduction of cloudlet execution time, and enhancement of the acceptance rate of cloudlets preferred by customers or brokers, etc.

As the contributions are quite interrelated with the conclusions obtained from the thesis research, so this section does not elaborate much on the detail of the contributions.

1.4 Organization of Chapters

Subsequent chapters are organized as follows:

Chapter 2 describes introductory information on resource allocation and task scheduling and some recently published research results in the area of cloud computing. The strengths and major challenges of their investigation are summarized, which considerably provide hints and inspirations to the research scope and objectives of this thesis.

Chapter 3 introduces the concept of attribute and proposes a strategy for VM allocation and cloudlet scheduling, which utilizes thresholds with attributes and amount (TAA). This chapter also presents the step by step methods and procedures to substantialize the TAA strategy.

Chapter 4 describes the preparation and creation of a simulation environment to host the experiments so as to assess the performance of TAA in resource allocation and task scheduling. Codes are prepared on the basis of an open source cloud computing simulation tool kit CloudSim. Based on the simulation environment, the proposed TAA policy is implemented.

Chapter 5 presents the performance improvement resulted from using the proposed TAA in task scheduling and resource allocation. Performance evaluation criteria, corresponding parameters, and related experimental results of the proposed techniques are discussed. This chapter includes the comparison of performance between before and after attributes are categorized and set as a task scheduling and resource allocation criterion. The significance and implications of the results for cloud computing are also discussed in this chapter.

Chapter 6 concludes the thesis work along with some proposals for future research.

Chapter 2. Literature Survey

2.1 Cloud Computing: Concept and Attractions

Cloud computing is viewed as a leading technology and represents a future development and commercial trend in IT industry. Recently, the drastic development of cloud computing has been bringing in great changes to the traditional IT business. It has influenced many entities in the globalized industry. It was reported in 2013 that the deployment of cloud computing solutions would remain a priority for U.S. Government, Departments and Agencies moving forward [4]. A growing number of small and medium businesses are moving to cloud for searching services. The number of service providers are also growing while the cost of services is expected to be decreased. Considering the growing importance of cloud computing, it can be concluded that improving cloud services is undoubtedly an area of concern and research focus.

Up to date, however, there is no a strict definition for cloud computing that has been accepted by all investigators and commercial operators. This does not mean cloud computing can be defined in a random way. We can understand the concept of cloud computing by referring some statements that are provided and widely accepted by different authors and entities.

For example, NIST proposed the following definition that highlights the "model" significance of cloud computing, which is widely referred to in the cloud computing community [5]: "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models*".

Another definition that is often cited emphasizes the "service" provided by the cloud computing to the public [6]: "*Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility (like the electricity grid) over a network (typically the Internet).*"

Some definitions pay more attention to the evolvement history and difference from other computing technologies, the following expression reflects this [7]: "*Cloud computing is developed from Grid computing and it combines the traditional concepts of Grid Computing, Distributed Computing, Utility Computing that coupled with virtualization.".*

There are also definitions focusing on the structure and constitution of clouds [8]:"*Cloud computing is defined as a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers*",.

In addition to a variety of definitions on cloud computing, there are also various types of clouds, including public cloud, private cloud, and hybrid cloud models. Based on these different models, different applications can be realized [9]". Although there are different cloud computing definitions and cloud types, it can be observed that a common principle of cloud computing is to distribute the computing requests of customers to many distributed computers rather than to local computers [10]. Also, cloud computing is usually viewed as a dynamic provisioning of information technology capabilities (hardware, software, or services) from third parties over a network [11].

The architecture of cloud computing is based on large amounts of data centers (or clusters) and networks. A cloud consists of a number of clusters which are called datacenters. Datacenters are further divided into a number of nodes (or hosts), and each node consists of a number of virtual machines (VMs) serving as an interface between users and the computers of a data center, and the VMs impress the users that they are exclusively using a computer of the data center. Datacenter offers on demand storage and computation services over the internet. Users can access the computing resources through the network interfaced by a virtual machine. The sharing of resources and service provision by a cloud provider are similar to a utility (like the electricity grid) over a network. The computing resources such as physical and virtual resources are dynamically pooled together to serve multiple customers via the multi-tenant model.

Regardless of the existing differences in definition, type, and architecture of cloud computing, it is often acknowledged that cloud computing may deliver the advantages or benefits to both personal and business users and service providers:

Scalability: This is the capability of a system to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth. Resources can be hardware, servers, storage space, and network. Users can quickly scale up or down the resources in cloud computing according to their needs without owning the resources as their property. In other words, the computing resources are scalable mean that the resources can be extended by the customers whenever they need and released once the tasks have been finished, and the consumption of resources can increase at any time to meet the requirements.

Virtualization: This gives users a feeling that they have an exclusive usage and view of available resources irrespective of actual arrangement in physical devices, with details about hardware variation intentionally hidden from users [12]. Providers, such as Amazon, provide users with access to a homogeneous set of commodity hardware with little knowledge and control of resources, because details of the hardware are obscured through virtualization technology. With virtualization, it is also possible to provide services to many users at the same time with fewer physical resources.

Mobility: This means users can access cloud applications through internet easily from anywhere and at any time. With the progress of cloud computing, customers including personal and business users can utilize or maintain the online resources.

Low Infrastructure Costs: No infrastructural maintenance or upgrade costs are required for end users. Moreover, the pay-per-usage model of cloud computing enables end users to pay for the resources in the cloud when needed, rather than make any investment on resources.

Increased Storage: End users in cloud computing can store more data in cloud than on private computer systems. The users don't need to buy extra storage hardware for their own computers but can still enjoy unlimited storage space provided by a cloud. In addition, the stored data or programs can be accessed anytime and anywhere through internet, as long as they are available in cloud.

2.2 Performance Issues of Cloud Computing

When talking about the attractions of cloud computing, its performance always matters. As presented in the above section, cloud computing can provide a number of advantages. Nonetheless, there are also a number of cloud performance issues (including both technical and commercial) that are compromising the attractions of utilizing cloud. The major technical issues include:

Resource Control: Controlling of resources in a cloud may vary among different cloud providers. Resource isolation is needed in cloud computing for easy usage and security consideration, but it may be very hard for the client to identify the exact resource. At the same time, resources may exhaust, so data or program may need to migrate to other resources. These are major and challenging issues in cloud computing. So controlling resources and distributing loads among different resources is a very essential operation of cloud computing.

Response Time and Latency: Applications are requested by cloud users and sent to a cloud for computation or storage. Examples of applications include digital media streams, data obtained from scientific instruments, and enterprise business database. The processing of these applications is demanding and resource hungry. Response time for these applications is very important, because low response time leads to unhappy experience to the users. Once the available resources cannot meet the requirements, further investment of resources is needed. To assure cloud performance, e.g., response time, within tolerance limit, it is important that available resources are used at an efficient and optimal level, for example, it is good to avoid some resources to be over-loaded, while other sources are leanly loaded. It is also preferred that resources are allocated to matching appropriate requests.

On the other hand, it is expected that the time taken by transferring data of customers to the processing machine of a cloud is negligible or small enough compared with the time taken by the data processing on the machine, although we also hope the latter one is also small. Thus, low latency has always been a major preference in cloud for transferring data. In other words, high latency is another issue in communication between client and provider, as cloud resources are accessed through the internet. For example, in some online games, a high latency can bring difficulty to determining which player has performed an action first (such as shooting a target, car racing, and responding a question). In playing computer-based musical instruments, it is difficult

for players to get the nearly instantaneous feedback if latency is too much to give a correct music rhythm.

Heterogeneity: By reviewing the currently existing task scheduling and resource allocation strategies, it can be found that current cloud computing infrastructure typically assumes a homogeneous collection of commodity hardware[13][14] [15] [16]. Nevertheless, different types of requests from end-users may need to obtain access a complex set of resources, such as different accelerators, machine architectures, and network interconnects to match various task requirements.

Non-uniformity exists widely because numerous cloud vendors provide different services with different or custom-built policies, infrastructures, platforms, and Application Program Interfaces (APIs). Even though computing resources are within the same data center, there are still different types of resources. For example, some resources fit better for storage, while others are better for computation. Such variations cause major challenges related to the non-uniformity of the available resources in cloud computing. This means a mechanism of matching tasks of customers with appropriate resources needs to be defined for the sake adaptation, as adaptation to different platform or hardware often plays an important role in the improvement of cloud computing performance.

2.3 Task Scheduling and Resource Allocation

2.3.1 Current Existing Strategies

It can be observed that all the above issues presented in the previous section are related to efficient and economical processing of users' requests (tasks) and distribution of cloud providers' resources. This is a significant reason why task scheduling and resource allocation is a major research direction in the improvement of cloud performance among many other investigations focusing on cloud performance. Within the limit of cloud environment, resource allocation strategy aims to utilize and allocate limited resources to meet the requirements of the cloud applications. In resource distribution, the allocation of a processor to tasks for execution is particularly important, which is known as task scheduling, which is carried out by a scheduler. Schedulers keep all

resources busy and allow multiple users to share the resources effectively so as to achieve a targeted quality of service. To improve the task scheduling and resource allocation, usually a third party, called service broker, is required for routing user requests to the most appropriate data centers and provide efficient performance, although the maintenance of the resources such as storage, processing, capacity planning, resource optimization, and bandwidth is controlled by a cloud provider. The process of sending a user request, processing it and returning the result to the user, involves a number of policies and algorithms at each logical layer [13].

In a scenario that only limited resources are available at a data center server, many requests may be submitted to one single processor simultaneously or asynchronously. Thus, a scheduler needs to arrange many tasks to execute on a single processor at a time. There may be many goals a scheduler aims at, for example, maximizing throughput, minimizing response time, maximizing fairness or load balancing. In practice, conflicts often happen for satisfying these goals simultaneously. Thus, a scheduler for a suitable compromise is preferred according to users' needs and objectives. A number of existing scheduling algorithms have been discussed by investigators, among them typical algorithms are:

Join the Shortest Queue (JSQ): In the JSQ, new requests are directed to the processor loaded with shortest queue of jobs. The entire requests for processors are directed through a centralized load balancer. Therefore, load balancers can easily track the status of current requests and responses to each processor. JSQ causes less overhead because only processors with their allocation are required to be maintained. [14]

Random algorithm: Jobs that are randomly selected are assigned for execution into virtual machines. The advantages of this algorithm are simple and not requiring maintenance of any resource information record, therefore, quite fast. Furthermore, it does not lead to heavy or low load as the VM position is not considered. [15]

Minimum Execution Time: This algorithm first finds out a resource by considering execution time. On the arrival of task, the task is allocated to the best resource with minimum execution time regardless whether resource is available or not at that time. [16]

First Come First Serve algorithm: Jobs are placed in a queue as soon as they arrive and the processing sequence is same as their arrival sequence. This algorithm is straightforward and quick.

Max-Min algorithm: Bigger tasks are chosen first to execute. Larger Jobs get executed sooner than smaller tasks. Smaller tasks will have to wait until all the larger tasks get executed. This may reduce the latency caused by small jobs and make the machines fully occupied at most of the running time. [16]

Round Robin: This policy picks a VM randomly from the group and assigns the VM to a new request. In this strategy, subsequent requests are processed in a rotational order. Each VM is allocated to a request for a fixed interval of time known as time slice. If the request is unable to be finished within the given slice time, it will have to wait for the next cycle to get it turn for execution. This will continue till submitted tasks are completed. Based on this algorithm, there is a modified policy called weighted round robin in which a weight is assigned to each VM depending on the capacity of each VM. For example, a VM is assigned with weight 2 if it is able to handle twice load. Higher weight VMs will be allocated more jobs each cycle. Weighted round robin is more reasonable and result in better performance, if the VMs have different processing capacities.

Throttled: Throttled load balancing policy maintains an index table of virtual machines and the state of each virtual machine, either BUSY or AVAILABLE. When a new request arrives, the index table will be parsed until the first AVAILABLE virtual machine is found. If more requests are presented in the data center than the number of available VMs, some of the requests have to be queued until the next VM is available. This technique ensures that only one cloudlet is allocated to a single VM at any point of time. [17]

The above paragraphs briefly summarize currently existing typical task scheduling and resource allocation algorithms. It can be found that reducing response time and data processing time, increasing throughput, load balancing of the system, and improving resource utility are expected for a good cloud performance researcher's aim at. These existing algorithms can improve performance in one or more facets, for example, join shortest queue (JSQ) algorithm can make relatively good load balancing and causes less overhead because only processors with their allocation are required to be maintained. Regarding Random load balancing algorithm, it is simple and fast because it does not need to maintain any resource information record. As for minimum execution time algorithm, it focuses on first finding out the resources by considering execution time, so its execution time is superficially low. As to Round Robin algorithm, can solve the problem of starvation by processing each task for that stipulated time period.

Based on these algorithms, researchers proposed some more methods with modifications and extensions. For instance, a hybrid approach based on Round Robin and Throttled algorithm was proposed to get better results in terms of response time [17]. Some investigators [18] [19] modified an algorithm based on Max-Min for cloud task scheduling to reduce waiting time, completion time or make-span. Some other investigators [20] modified a cloud task scheduling policy based on ant colony optimization algorithm so as to minimize the make-span of a given tasks set. In some investigations [21], an integrated task scheduling algorithm taking into account the issues such as VM management and Datacenter management was proposed to reduce the average execution and waiting time of tasks, and in the meanwhile improve the throughput of the cloud system. Some investigators [22] proposed a scheduling algorithm which is shortest-job-first to lower turnaround time. Another algorithm based on evaluating the completion time and sorting tasks by priority to achieve good performance and load balancing was examined [23]. In the research work of [24], an implementation of priority queue on broker side was proposed to be able to control cloudlets with priority and process them differentially. The results showed a faster completion time. Some investigators [25] proposed a task scheduling algorithm prioritizing tasks with close deadline to reduce task waiting time. In other works [26], tasks with highest priority is assigned to a Virtual Machine with highest processing power to lower execution time.

To improve the task scheduling and resource allocation, usually a third party, called service broker, is required for routing user requests to the most appropriate data centers and provide efficient performance, although the maintenance of the resources such as storage, processing, capacity planning, resource optimization, and bandwidth is controlled by a cloud provider. The process of sending a user request, processing it and returning the result to the user, involves a number of policies and algorithms at each logical layer [27].

2.3.2 Complexity and Challenges

Although there exist a number of strategies dealing with the task scheduling and resource allocation, it can be summarized that most of them are based on task amount rather than the diversity of task and resource types. In fact, types of customers' tasks (e.g., storage and computation) are as same important as task amount in the task scheduling and resource allocation.

A reason for this is because the type of resources (e.g., machines, tools, and software) provided by datacenters are also quite different, as presented in the previous section about the heterogeneity of resources. Hence, some resources are more suitable for some specific tasks. Complexity of loads and tasks exists not only among different service providers but also within a datacenter provided by a same service provider. Among different cloud service providers, different services with custom-built policies, infrastructures, hardware, platforms, and Application Program Interfaces (APIs) are offered. Even though within a datacenter, different server may still have different characteristics, for instance, some computers may be more appropriate for storage, while other computer may be more suitable for processing video and audio, and graphical processing unit (GPU) is great for processing graphical tasks.

Some researchers have realized the importance of considering the heterogeneity of resources into the task scheduling and resource allocation [12], they presented an approach for extending the traditional notions of cloud computing to provide a cloud-based access model to clusters that contain a heterogeneous architectures and accelerators. In other works, a scheduling scheme called TDS to schedule tasks of a directed acyclic graph (DAG) onto a heterogeneous system [27]. The primary objective of this scheme is to minimize schedule length and scheduling time itself.

In addition to the variety of tasks, priority is another factor that brings challenges of task scheduling and resource allocation. For example, medical and police emergency requests might have higher priority than other requests, although the processing of their requests by a machine of a cloud may not be as CPU intensive as other tasks like weather forecast data processing, and highly paid tasks may request higher priority than low paid jobs. In this situation, those task scheduling strategies that mainly focus on task amount will not give us a satisfactory performance. Although it is a very challenging work to define and classify dynamic priorities of cloud tasks, researchers are stepping forward with considering complex priorities of tasks. For example, some investigators have suggested using priority of tasks to conduct task scheduling so as to improve cloud performance and customers satisfaction [28] [29] [30]. In some investigations, tasks are divided into three priority levels, i.e., gold, silver, and bronze for different pay levels, wherein the services at different priority levels get paid for different prices [28]. Similarly, a priority class of VMs was suggested and users pay a higher price for better performance [30], and the resources

can be further classified according to three standards: computing capability, storage space and network bandwidth in [31].

2.3.3 Performance monitoring and simulation tools

To improve the performance of cloud computing, including the task scheduling and resource allocation, it is valuable if the performance can be monitored so that real time and historical performance can be tracked. This has promoted the development of performance tools. Currently, there are a number of automated performance monitoring (APM) tools to monitor the performance in cloud computing, which are able to provide the monitoring based on real time and/or historical data. These cloud monitoring tools are either provided by the cloud provider (such as cloudwatch by Amazon) or a third party such as RevealUpTime and Cloudstatus. The majority of them enable monitoring the detail of the status of task processing in the cloud by using wide variety of devices such as smart phones or laptop devices to be possible. The cloud providers may also gather and document the cloud performance monitoring data for future research and improvement.

For example, Google, Salesforce, Rackspace, and Terremark conducted their tests with the help of test agent and gathered the data including service response, network performance, CPU, and internal I/O to gradually improve their service. CloudHarmony staff conducted tests on the prescribed benchmarks for specific scenarios such as CPU performance, storage, I/O, memory I/O or video encoding. Other investigations suggested a new approach for infrastructure management to determine the performance in virtualized environment by monitoring infrastructure response time [32]. Performance factors or indicators were suggested to improve the cloud performance, e.g. latency, response time and execution time were used to evaluate the cloud performance [33].

To get a better cloud performance, another widely accepted option is to first utilize a cloud simulation tool to mimic the real cloud environment so as to test new techniques and solutions, then introduce them to the real cloud environment. Therefore, a convenient cloud simulation tool to rapidly evaluate the new algorithms with low cost before actual software being applied in an actual cloud environment is needed. Among various simulation tools, CloudSim is a widely used cloud simulation toolkit which allows modeling, simulation and experimenting on cloud computing environment [2]. It was originally developed in the GRIDS laboratory at the University

of Melbourne and has been proved to an efficient simulation toolkit, which supports both system and behavior modeling of cloud system components such as data centers, virtual machines (VMs) and resource policies [3] so as to simulate and assess resource allocation, provisioning, and scheduling. In this thesis, CloudSim will be used for evaluating the proposed algorithm.

2.4 Summary

It can be concluded from the above literature survey that cloud computing has many promising attractions and has been booming and represents a future trend of IT industry and business. To turn the promising attractions into practice, the performance of cloud computing must be improved and hence the performance issues such as resource control, heterogeneity, response time and latency must be resolved. All these issues are related to task scheduling and resource allocation, i.e., to allocate the resources (e.g., machines and software) of data centers in a cloud to the tasks (e.g., calculation and storage) of customers in an efficient, economical, and secure way. Many investigators have proposed a number of different strategies to improve the performance of task scheduling and resource allocation, such as Random, Minimum Execution Time, Round Robin, throttled, and different modified versions based upon them. Also, different performance monitoring and simulation tools are being used and investigated for the improvement of quality of service of cloud computing.

These strategies perform well under their preferred conditions. However, most of them mainly consider the amount of workload, i.e., task scheduling and resource allocation are mainly arranging tasks according to the queuing size of a machine (including virtual machine). The variety of task types and different characteristics of machines are not considered sufficiently. Although there are some strategies have taken into account the heterogeneity of resources, the matching of task varieties with the heterogeneity of resources has not been well studied. Some investigators introduced priorities into tasks so as to perform task scheduling to mitigate the weakness of only taking into task amount into account, but the matching issue is still unresolved.

This thesis aims to resolve the matching issues through introducing attributes into the customers' tasks and data centers' resources, which are expected to reflect the varieties of tasks and characteristics of resources, as well as provide insight of users' tasks and cloud resources. A

task scheduling algorithm called Thresholds with Amount and Attribute (TAA) will be proposed in this thesis to schedule tasks and allocate resources according to both the amount and attributes of tasks. A simplified version of this strategy is a two-level filtering structure that can schedule tasks to different resources based on the queuing size limit and attribute matching between tasks and machines. In order to test the strategy, an appropriate simulation tool is selected and TAA codes are embedded into the tool for implementation. These will be presented in detail in subsequent chapters.

Chapter 3. Our Proposed TAA Algorithm

As presented in the previous chapter, scheduling of tasks is essential in cloud computing. Appropriate load distribution and spreading to resources may improve resource utilization and cloud performance. Currently existing resource scheduling and allocation strategies discussed in the previous chapter of literature review are mainly focusing on task amount, and there have been rare investigations on scheduling strategy that is based on matching the variety and complexity of users' tasks and data centers' resources. For example, although scheduling routine is often composed of three steps of resource spotting, resource preference, and task submission [34], the three steps are mainly based on the amount of queuing loads.

In this thesis, the term "attribute" is adopted to reflect those factors causing the variety and complexity and matching thereof. Up to date, no standard definition describing attributes of tasks or jobs has been found and there have been rare investigations address this issue, to the best knowledge of the thesis author. Even though complexity of cloud environment and various attributes of tasks has been considered for improving cloud performance, there is no clear and uniformed categorizing standard for the attributes of tasks. This thesis tries to define a standard for categorizing different attributes and make two-level attribute definition, and applies a two-level filtered attribute matching into queuing system for arranging resource processing order.

The scope of task attributes may be very large. For instance, the tasks may be small or large, compute-intensive or storage-intensive, requiring uniprocessor or multiprocessor. Also, the task content may be online game, movie, or scientific data sheets. The tasks may be loosely coupled or tightly coupled. The attributes of requests from users need to be considered in the task scheduling and resource allocation. Selecting optimal node to execute a task according to its resource requirements and the fitness between resource nodes and tasks may improve task execution and resource utilization efficiency.

By adopting attribute, it is also expected that priority can then be dynamically assigned to different attributes. This way, other task scheduling and resource allocation strategies that have considered task priority can then be integrated with the TAA strategy proposed in this thesis.

To mitigate the investigation scarcity of attributes, this thesis work considers both load attributes and amount in the proposed algorithm for optimal task scheduling. Furthermore, attributes of tasks or jobs are incorporated into a cloudlet to constitute a part of the cloudlet for optimal matching of tasks and resources. By reading attributes of each cloudlet, task scheduling applies two-level attribute filtering and thresholds [35] on a queuing system that will eventually optimally arrange the processing of tasks having different attributes and reduce execution time of the cloudlets.

3.1 Strategy of Incorporating Various Attributes into a Cloudlet

3.1.1 Attributes Arising from Requests and Resources

Current cloud datacenters host a wide range of applications with various requests and features. For example, web application requests usually concern more about response time, and batch jobs often require being finished before deadline and hence concern the amount of CPU time and RAM allocated to the jobs. Requests from different users may correspond different resource demands: some workloads may be CPU-intensive whereas others are storage-intensive, and some of them may even require a dominating use of special hardware like GPUs to achieve dramatic performance gains [36].

In addition to the variety and complexity of requests, resources in cloud computing environment are also quite various and not uniform in many aspects, i.e., resources are inherently heterogeneous in cloud. For example, some machines are more suitable to store large data whereas others run faster for better computation capability [36]. Thus, the performance of the implementation of a job may have a dependence on the machine on which the job runs. Various configurations of machines are available in cloud with different capacities and performance, and cloud providers also like advertising and selling their uniqueness and differences of their clouds

and services, either from technical or pricing aspect. If each type of machine is arranged to offer the most suitable tasks or deliver a good cost-performance and trade-off for the job, the performance of cloud may be efficiently improved, e.g., requests are processed more efficiently and resources are used more economically.

The advantages of grouping or clustering of jobs assist in sorting identical jobs requesting for similar resources, accordingly the resource can be provisioned [37]. However, classification of attributes is very complex according to different requirement. Traditionally, attributes obtained through analyzing the jobs, because users submit a job to resource with a few hints, including memory requirement, ability to use special features like GPUs, and the deadline, if needed [36]. However, this is time consuming and complex. The differences among these different workloads further make the resource provisioning a challenging task. In this chapter, a method for clustering and categorization of attributes is proposed to render the attributes always accompany task content of a cloudlet so as to provide convenience for matching different jobs for different resources.

3.1.2 Cloudlet Consisting of Both Content and Attributes

The method proposed in this thesis is listing all the necessary attributes of a cloudlet and incorporating the attributes into a cloudlet. As a consequence, a cloudlet is composed of two parts: attribute part and content part. Figure 3.1 illustrates a simplified structure option of a cloudlet, wherein different attributes are arranged in a contiguous section.

Note that different attributes could also be distributed in many other sections of the cloudlet to improve the identification of the attributes, or even be ciphered first and the placed in the cloudlet to enhance the security or the privacy of the attributes. The size of attribute part can be designed small enough compared with the dimensions of content. This would make the attribute analyzing and grouping easier and more cost efficient. When a cloudlet of this type is received, the attributes of the cloudlet can be easily read according to the reading or deciphering protocol agreed to and acknowledged by cloud broker or data center, without disturbing the content of the cloudlet.

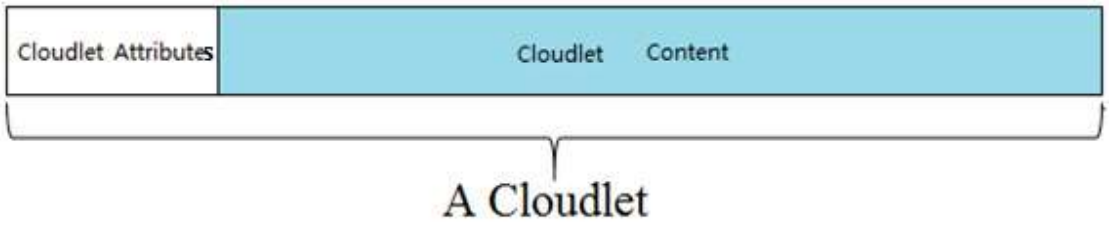

Figure 3.1: A cloudlet consisting of contiguous attribute part and content part

3.2 Thresholds of Attributes and Amount (TAA)

3.2.1 Concept of TAA

For the sake of arranging appropriate cloudlets to an available resource (e.g., a storage space or computing unit), a simple option for the categorization of attributes can be a two-level filtering strategy: a lower level filtering (level 1) and an upper level filtering (level 2) of attributes. The lower threshold performs a rough filtering of attributes and the higher level threshold imposes a strict filtering on coming cloudlets. So below the lower level threshold there are more types of attributes than between lower and upper thresholds. The setup of these two thresholds allows flexible prioritizing for some attributes. For example, a cloudlet with the completion deadline that is urgently approaching may be arranged a higher priority than other tasks with distant completion timeline, and military tasks may have higher priority than civil tasks. If users want to pay a higher price for cloud service, they may expect that their tasks should have a higher priority. Based on the two-level attribute threshold definition, all the tasks can be grouped and arranged the best suitable resources for them. Several examples of setting up a two-level attribute thresholds are in illustrated in Table 3.1. There could be many other options for this thesis will not focusing on how to define the level or priority of attribute.

Table 3.1: Examples of attributes based on different level of threshold

Below lower threshold	**Between lower and higher thresholds**
computation-intensive	Deadline far or near
storage-intensive	Military or civil
GPU-intensive	High pay or low pay

The TAA strategy can also be interpreted to TAA algorithm, i.e., to select the best suitable resource (virtual machine) for each cloudlet, prioritized attributes and workload amount are used as selection parameters. Two rules for the selection could be followed: (1) The amount threshold is to avoid a virtual machine to be overloaded. This is also to help ensure a cloudlet can be arranged a virtual machine; (2) The attribute threshold is to ensure that the cloudlets with special attributes (e.g., high priority is assigned to the attributes) will have privilege to be arranged to the best suitable resource if many cloudlets with other attributes are requesting the same resource.

For simplicity, an option for the proposed TAA algorithm could be based on First Come First Serve (FCFS) policy. In the TAA algorithm, each virtual machine has a waiting queue for incoming cloudlets before the cloudlets are processed by the virtual machine. The waiting queue consists of two sections created by the lower and upper thresholds, as illustrated in Figure 3.2 (a) (b). If the virtual machines are not busy, only rough filtering is performed so the cloudlets with various attributes can be allowed to enter the waiting queue until the lower threshold is fully occupied. As the queue size may be dynamic, so once the broker finds lower threshold is reached, then strict filtering is imposed on all requests and only selected tasks are allowed to enter the waiting queue between the lower and upper thresholds, until the upper threshold is reached. For simplicity, it is regulated that no cloudlet will be removed from the waiting queue once it has been allowed to enter the queue, although the cloudlet might have a lower priority than a cloudlet coming after it. When the upper threshold of a virtual machine is reached, no more requests will be accepted by the waiting queue of the virtual machine. This may help avoid overloading of the virtual machine.

Since each time a virtual machine only takes in one cloudlet for processing, so once the processing of the cloudlet has been completed, another cloudlet right after the cloudlet will be accepted exclusively into the virtual machine. In other words, when a cloudlet is allowed to enter

the waiting queue of a VM and arrives at the VM, it will be processed right away if the virtual machine is free; otherwise, the cloudlet will remain in the waiting queue somewhere in front of the VM. This is consistent with the processing order of "first come, first serve (FCFS)".

In fact, FCFS is not the only scheduling policy to be integrated with TAA strategy. Other scheduling policies [38] such as Shortest Job First (SJF) or Shortest Remaining Processing Time (SRPT) can also be combined with TAA strategy. In SJF, the next job to be served is the one with the smallest size. When it is integrated with TAA, the task with the smallest size queuing under either of the upper or lower thresholds can be picked up by the machine for processing. In SRPT, the next job to be serve is the one with the smallest remaining processing time queuing under either of the upper or lower thresholds can be picked up by the machine for processing. Since this thesis will focus on examining the feasibility of TAA, so only FCFS is used for a simplified integration.

The magnitude of upper and lower thresholds are decided by virtual machine's capacity, CPU speed and some other factors. With reasonable value of thresholds, the resource utilization may be maximized. Generally, the threshold magnitude of each virtual machine may be different because of various hardware and software configurations and capabilities. Figure 3.2 (a) illustrates a general schematic of different threshold magnitude for each different virtual machine.

In real cloud environment, a large amount of virtual machines exist, so it would be a big and complex work to determine the magnitude of each threshold for each different virtual machine. The purpose of this thesis is to examine the task scheduling and resource allocation performance of TAA, so only a simplified option is preferred for the feasibility study, that is, same values are set for the two thresholds of all virtual machines. Figure 3.2 (b) shows a simplified version with same thresholds values for each different virtual machine. Regarding how to determine the magnitude of thresholds, it is not the focus of this thesis but will also be examined from the aspect of queuing and thresholding dynamics in Chapter 5.

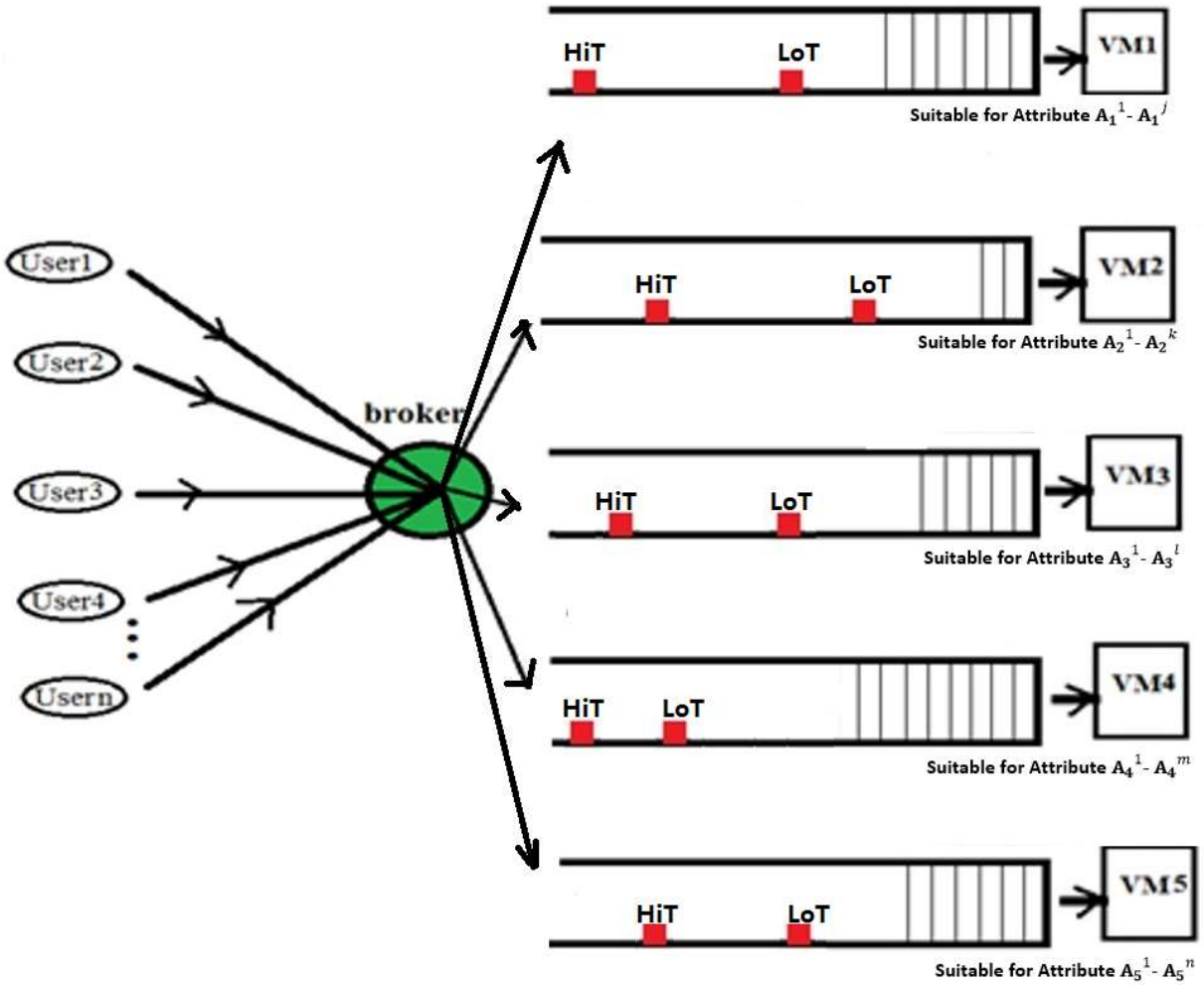

Figure 3.2(a): Thresholds and Attribute Awareness of TAA (a general cases)

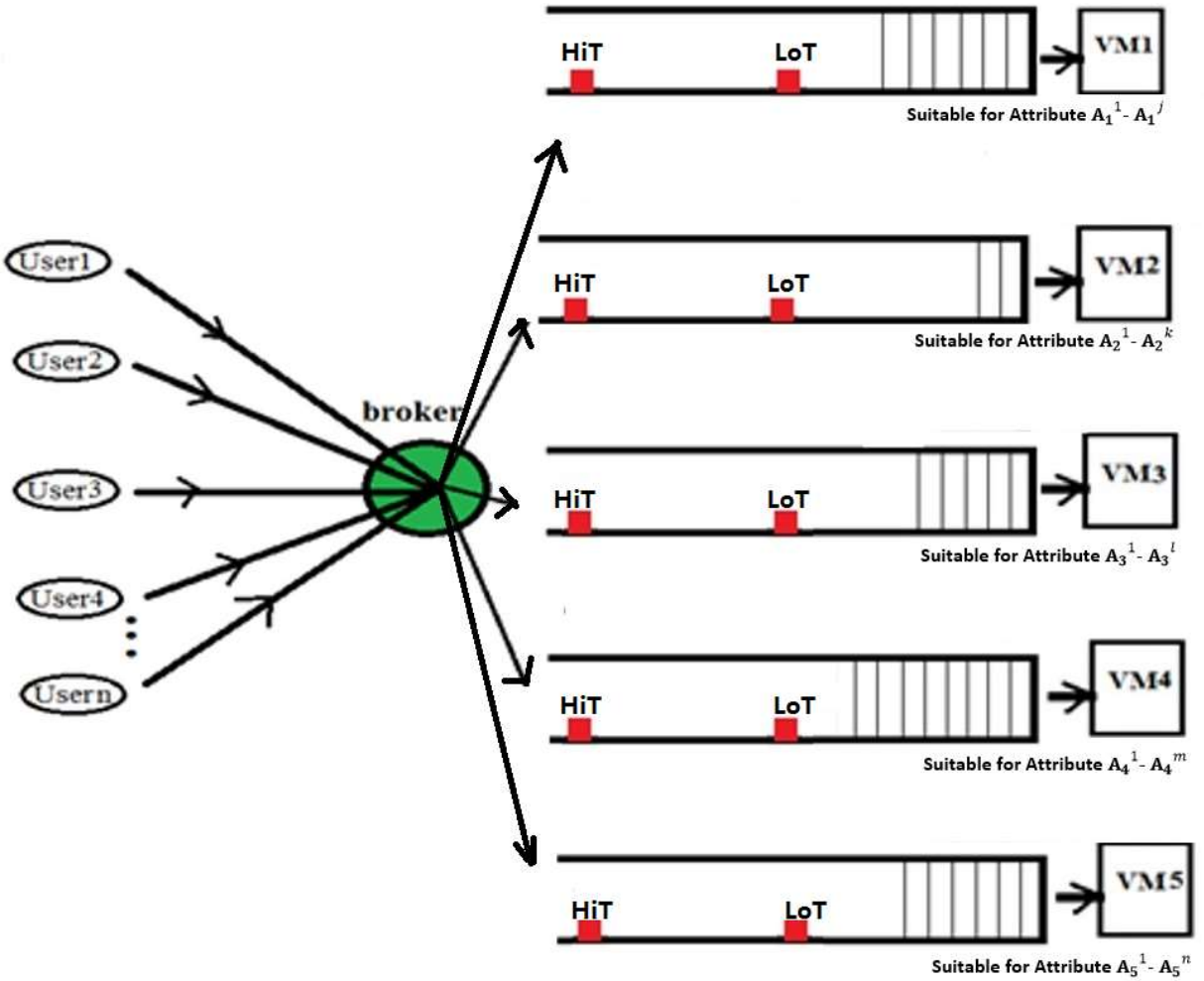

Figure 3.2(b): Thresholds and Attribute Awareness of TAA (a simplified special case)

Table 3.2: Notes for Figures 3.2(a) and (b)

VM1	More suitable for cloudlets with attributes $\mathbf{A_1}^1$- $\mathbf{A_1}^j$
VM2	More suitable for cloudlets with attributes $\mathbf{A_2}^1$- $\mathbf{A_2}^k$
VM3	More suitable for cloudlets with attributes $\mathbf{A_3}^1$- $\mathbf{A_3}^l$
VM4	More suitable for cloudlets with attributes $\mathbf{A_4}^1$- $\mathbf{A_4}^m$
VM5	More suitable for cloudlets with attributes $\mathbf{A_5}^1$- $\mathbf{A_5}^n$
HiT	Upper Threshold
LoT	Lower Threshold

3.2.2 Roughly Filtering Tasks under Lower level Threshold

When queue size is under lower threshold, it means there are not many customer tasks, i.e., cloudlets, requesting the usage of a virtual machine, so the virtual machine is relatively free. In this situation, cloudlets are roughly filtered and then join the queue. For example, for a virtual machine that is suitable for cloudlets with attribute of requiring computation, all the computation-intensive tasks will be accepted by the virtual machine, regardless of its completion deadline or pay for the service. In other words, other restricting attributes such as deadline near or far and high pay or low can then be ignored at this level, and these attributes are normally considered and prioritized when there are many tasks beyond the magnitude of the lower threshold. Figures 3.3(a) and (b) illustrate the filtering strategy of virtual machines with queue size under lower thresholds. Figure 3.3 (a) represents a general case of different threshold magnitude for each different virtual machine，and Figure 3.3 (b) represents a simplified case that all different virtual machines have the same maximum value for all lower thresholds, and Table 3.3 describes the attributes of VMs that are suitable for to help explain figures 3.3 (a) and (b).

An example of setting the attribute standard for the rough filter of the lower level threshold could be the primary or approximate types of cloudlets rather than detailed subtypes, corresponding to categories of parental and descendant attributes. In other words, the rough filter standard can be clear for primary types of cloudlets, but blurred for secondary types that belong to a primary types. For instance, “requesting storage” and “computation intensive” could be two primary types or parental attributes, and “storage of financial data”, “storage of photo”, and “storage of video” could be three subtypes or descendant attributes under the primary type or parental attribute of “requesting storage”. As for the parental attribute “computation intensive”, its descendant attributes could be “High pay computation” and “low pay computation”. It is noted that this categorization is mainly to help explain the filter of attributes at different levels of thresholds, this thesis will not discuss much about how to define the terminology of attributes, as this requires a much more comprehensive investigations.

Under the lower threshold, the cloudlets are directed to different VMs only based on the parental attributes rather than the descendant attributes. For example, those computation-intensive cloudlets that are requesting computation would be directed to those VMs that are more suitable for computation, regardless of high pay or low pay. Once the queue of a VM for computation is

reaching the full queuing capacity of the VM's lower threshold, then the higher threshold of the same VM will be activated and other late coming cloudlets requesting computation will be filtered more strictly, e.g., additional descendant "high pay" will be added to its parental "computation intensive" attribute to form a "high pay + computation intensive" filter. As a consequence, late coming "low pay + computation intensive" cloudlets may be directed to the queue of another VM for computation so as to allocate the best resource of the data center to the cloudlets with highest priority (high pay is an example of gaining highest priority). This might cause a decrease of usage efficiency of the most powerful machine. This tradeoff will also be briefly discussed in subsequent chapters regarding the queuing and thresholding dynamics.

Table 3.3: Notes for Figures 3.3(a) and (b)

VM1(under Lower threshold)	Cloudlets with Attribute $\mathbf{A_1}^1$- $\mathbf{A_1}^j$
VM2(under Lower threshold)	Cloudlets with Attribute $\mathbf{A_2}^1$- $\mathbf{A_2}^k$
VM3(under Lower threshold)	Cloudlets with Attribute $\mathbf{A_3}^1$- $\mathbf{A_3}^l$
VM4(under Lower threshold)	Cloudlets with Attribute $\mathbf{A_4}^1$- $\mathbf{A_4}^m$
VM5(under Lower threshold)	Cloudlets with Attribute $\mathbf{A_5}^1$- $\mathbf{A_5}^n$
LoT	Lower Threshold

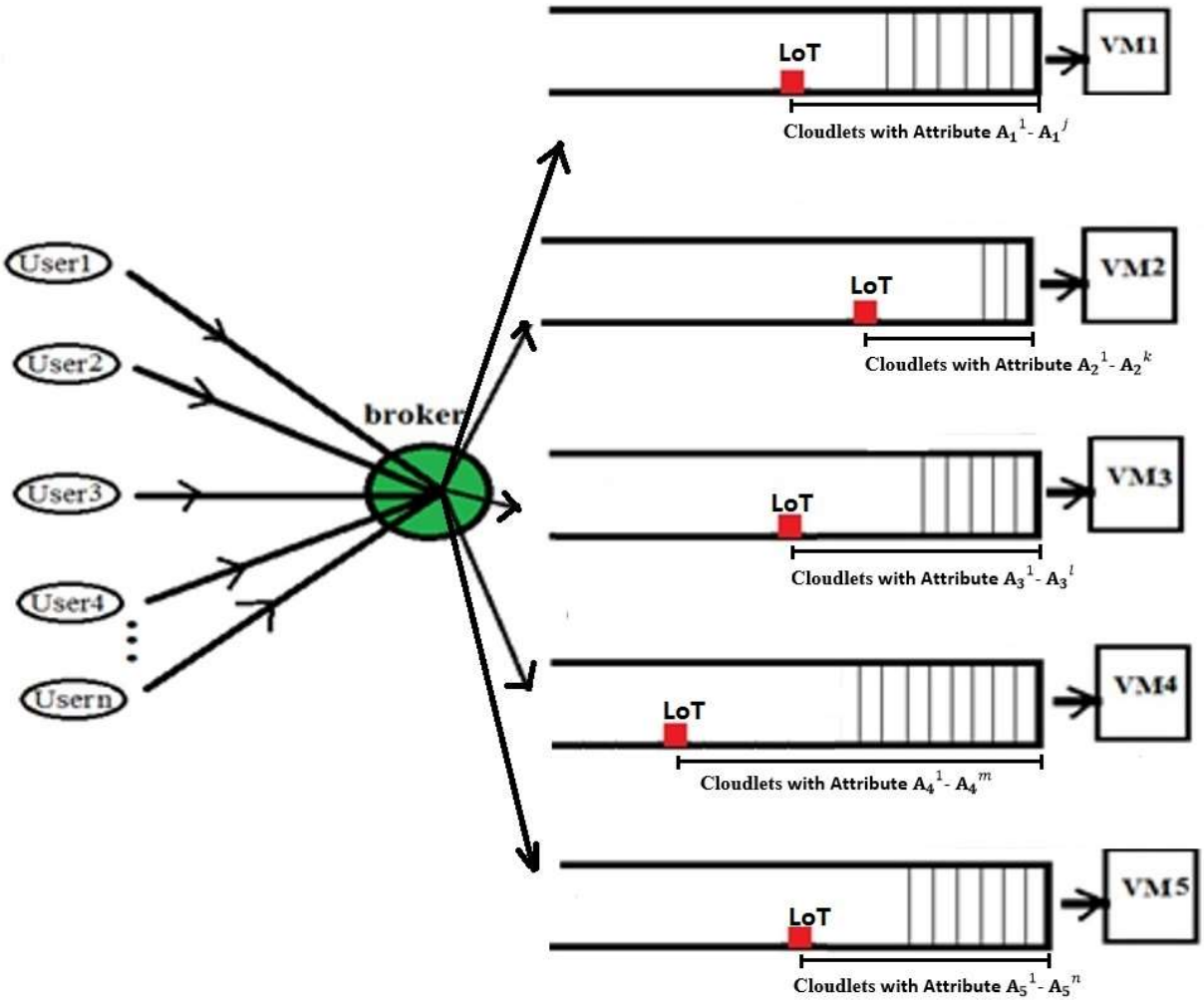

Figure 3.3(a): Roughly Filtering Tasks under Low level Threshold (General case)

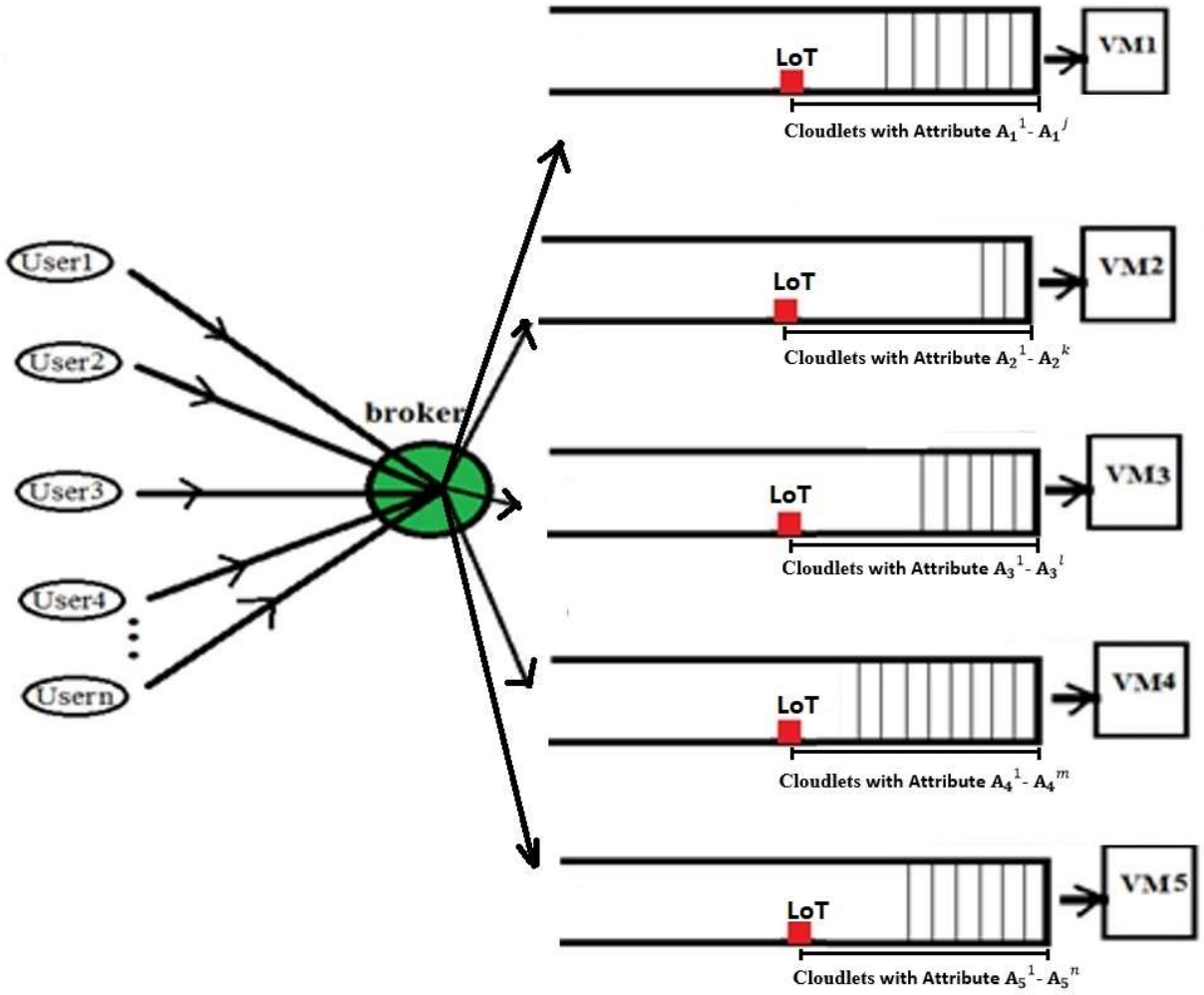

Figure 3.3(b): Roughly filtering tasks under low level threshold (a simplified special case)

3.2.3 Strictly Filtering Tasks between Upper and Lower Thresholds

When the queue of a VM exceeds its lower threshold, the virtual machine is becoming more busy. In this situation, another filtering mechanism of TAA will work, that is, cloudlets will be strictly filtered to enter the waiting queue of the VM to assure only a portion of the cloudlets have privilege to get this most suitable resource.

Figure 3.4 (a) shows a general case of different threshold magnitude for each different virtual machine, and Figure 3.4 (b) represents a simplified version that all different virtual machines have

the same maximum values for all lower and upper thresholds, and Table 3.4 describes the attributes of VMs that are suitable for to help explain figures 3.4 (a) and (b).

The privilege could be based on descendant attributes instead of parental attributes. As presented in previous sections, an advantage of introducing an upper level of attribute threshold is to make the prioritizing of attributes very convenient or dynamic, so when the cloudlet amount exceeds the lower level threshold, the attributes of incoming cloudlets could be prioritized, either dynamically or predetermined by the broker or customers. For example, both the cloudlets A and B are suitable for VM1, but the lower threshold of VM1 has been fully occupied. In this situation, if the completion deadline of cloudlet A is nearer and deadline of cloudlet B is farther, then, cloudlet A can be dynamically prioritized to a higher level and then accordingly allocated to VM1. As a consequence, cloudlet B will be filtered out of VM1 and conducted to another VM. Another instance is that high or low pay tasks can also be dynamically prioritized to narrow down the entry gate to the queue between the upper and lower thresholds when the lower threshold of the VM has been fully occupied.

Table 3.4: Notes for Figures 3.4(a) and (b)

VM1(between Lower and Upper thresholds)	Cloudlets with Attribute $\mathbf{A_1}^1$- $\mathbf{A_1}^{j-u}$
VM2(between Lower and Upper thresholds)	Cloudlets with Attribute $\mathbf{A_2}^1$- $\mathbf{A_2}^{k-v}$
VM3(between Lower and Upper thresholds)	Cloudlets with Attribute $\mathbf{A_3}^1$- $\mathbf{A_3}^{l-x}$
VM4(between Lower and Upper thresholds)	Cloudlets with Attribute $\mathbf{A_4}^1$- $\mathbf{A_4}^{m-y}$
VM5(between Lower and Upper thresholds)	Cloudlets with Attribute $\mathbf{A_5}^1$- $\mathbf{A_5}^{n-z}$
HiT	Upper Threshold
LoT	Lower Threshold

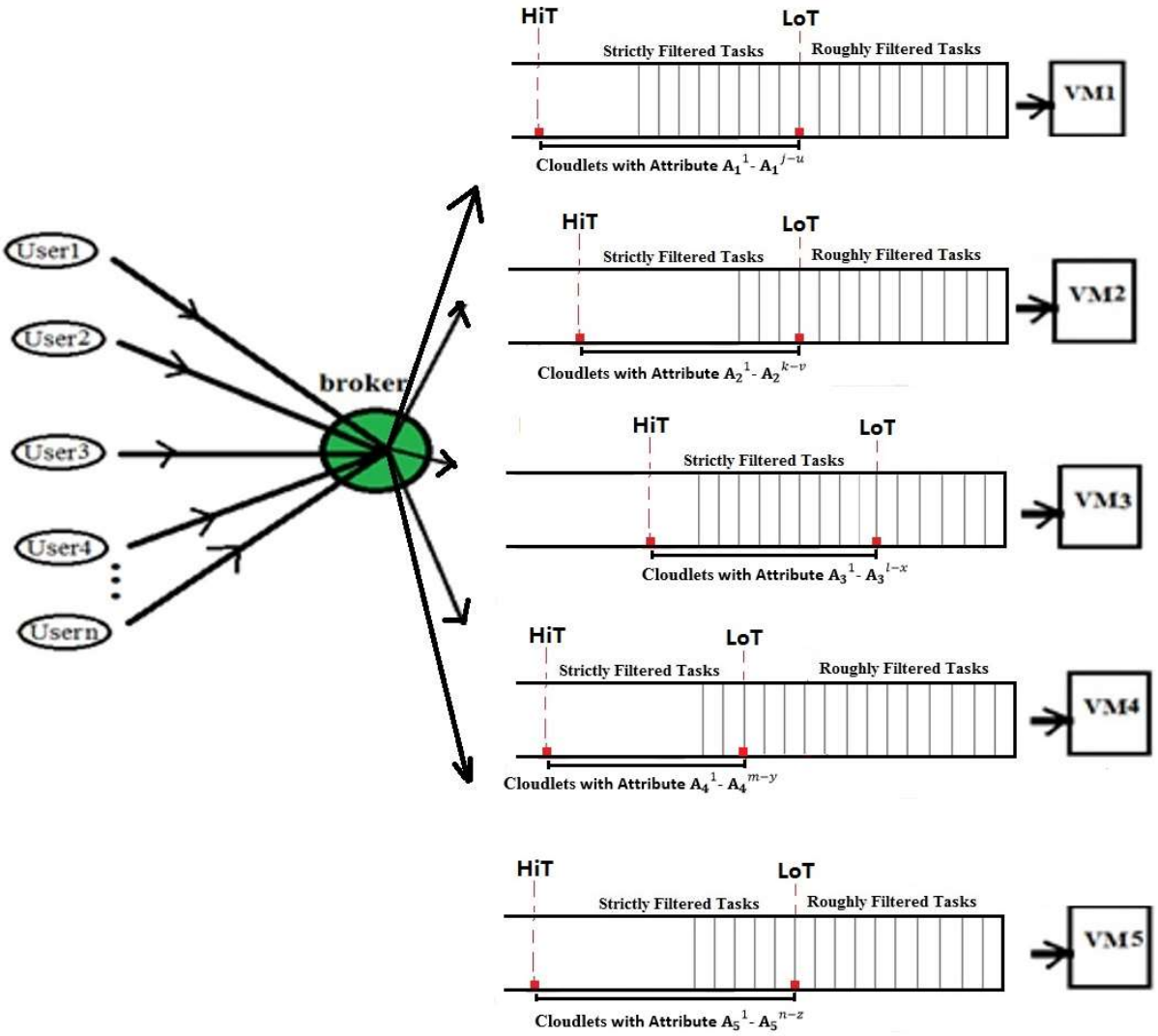

Figure 3.4(a): Strictly filtering tasks between low and high Thresholds (general case)

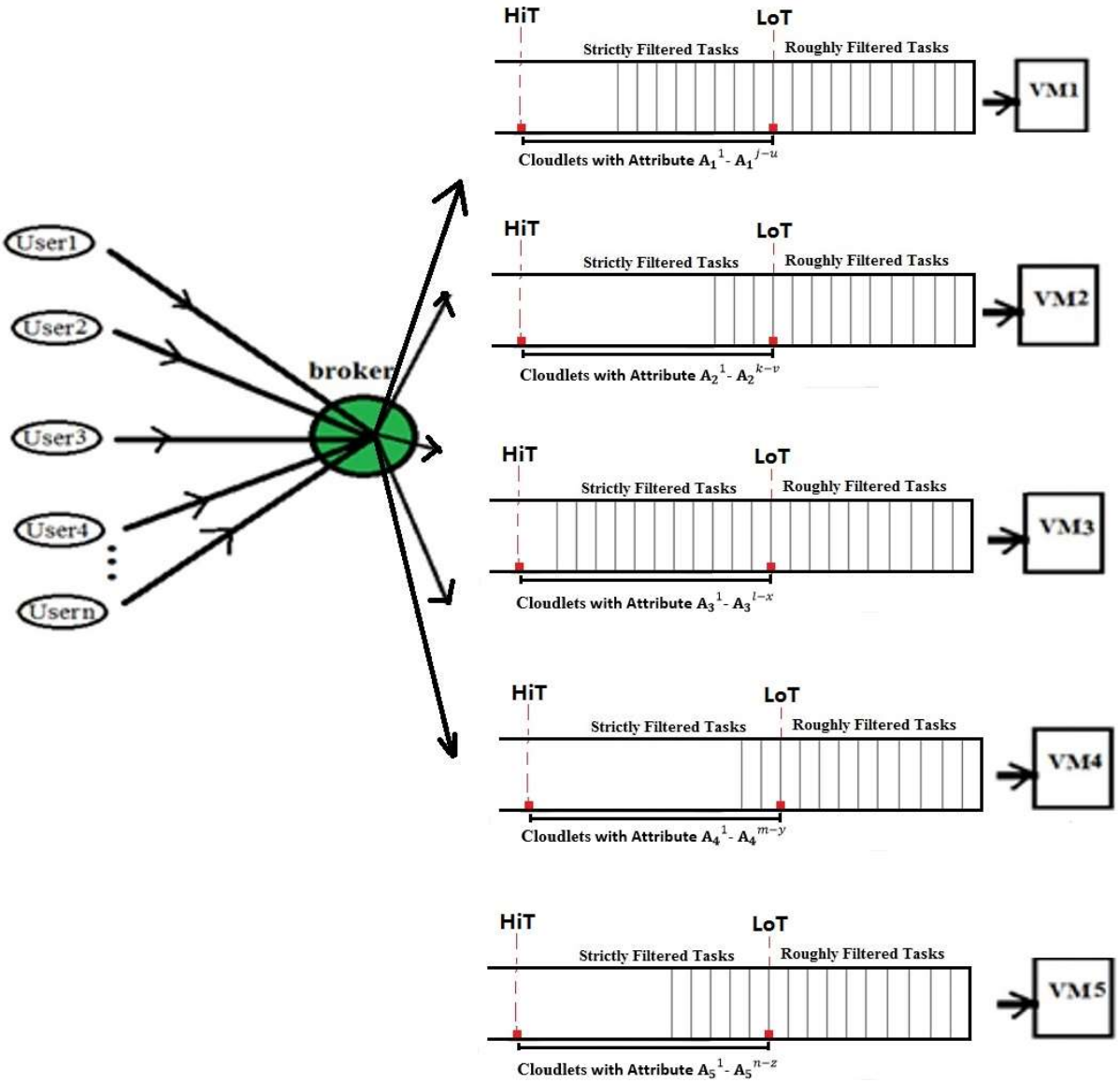

Figure 3.4(b): Strictly filtered tasks between low and high threshold (a simplified special case)

Chapter 4. Preparation for Implementation of TAA Strategy in a simulated Environment

Whether the TAA strategy proposed in the previous chapter would practically improve the performance of cloud requires tests in a real cloud environment. However, it is costly and fairly unrealistic to test an immature strategy that has not been engineered in a real cloud environment. Fortunately, as presented in the previous literature survey, some cloud simulation tools have been developed to test cloud performance. This chapter will select a cloud simulation tool to mimic a real cloud environment so as to test the performance of task scheduling and resource allocation with the proposed TAA strategy before introducing it to the real cloud environment. Among various simulation tools, CloudSim is selected due to its flexibility of inserting developed code into it, as well as convenience of clearly exhibiting experimental results to researchers. In subsequent sections, the major entities and functionalities of CloudSim as well as service providing process will be introduced, and then the related flow chart and pseudo code of TAA strategy will be introduced.

4.1 Introduction to CloudSim

CloudSim was originally developed in GRIDS laboratory at the University of Melbourne, which has been viewed as a typical cloud simulation toolkit that allows modeling and simulation of, as well as experimenting with, cloud computing environments [3]. The toolkit supports both system and behavior modeling of cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies [21]. Recently, CloudSim has been widely used for evaluating various algorithms including resource allocation, provisioning, and scheduling in recent year, and has been proved to be an efficient simulation toolkit.

Major entities or classes included in CloudSim tool are presented as follows [3]:

Datacenter broker: it is designed as an intermediate between cloud users and providers to deploy services over the clouds. It performs dual roles: on one hand, it is primarily responsible for the VM management within a single data center and load balancing of VM's within that single data center; on the other hand, the broker manages the routing of user requests among data centers based on different policies.

Cloudlet: it is defined for modelling cloud-based application services (e.g., content delivery, social networking, and business workflow) in CloudSim. Briefly, a cloudlet is defined as a job or task submitted to cloud.

Cloud Information Service (CIS): this is an entity that provides services such as cloud resource registration, indexing and discovery. The cloud resource table or list (host list, VM list) informs their readiness to process cloudlets by registering themselves. Other entities such as broker can contact CIS to get a list of registered resource IDs for resource discovery service. In analogy, CIS acts like a yellow page service.

Datacenter: it contains and models hardware and software services in the cloud which is managed by cloud service providers.

Host: a host is a computing component of a datacenter. It can host virtual machines and has a defined policy for provisioning CPU, memory and bandwidth to virtual machines. Note that the meaning of the term "computing" here is not limited to calculation, it includes all functions that a computer can do, e.g., storage, just like the cloud "computing" is not only for cloud calculation.

Virtual Machine (VM): it is not a real machine although it runs inside a host that processes cloudlets. This processing happens according to a policy for submitting cloudlets to VMs to be executed. A host can be used to create multiple virtual machines to interface multiple users.

DatacenterCharacteristics class: this represents the characteristic of a datacenter containing the information about OS, CPU type and System Architecture of the datacenter. In addition, the cost of using CPU, memory, processing or storage can also be defined in a datacentercharteristics object.

It can be observed that the Cloudsim has provided most entities or classes to build a cloud environment, which in turn can further provide the following different kinds of functionalities [3].

- Support for modeling and simulation of large scale cloud computing data centers.
- Support for modeling and simulation of datacenter network topologies and message-passing applications.
- Support for customized policies for resource allocation and task scheduling.
- While using CloudSim features, developers or researchers don't need to think about the lower level details of cloud based infrastructure.

A schematic description for the hierarchy and process of how Cloudsim simulates the provision of cloud services to clients over the network will be very helpful for us to have a large picture view of a cloud. However, no such a schematic description has been reported, although Cloudsim has been adopted by some investigators for the simulation of different aspects of cloud computing. After thoroughly reading the code and flow of Cloudsim, this thesis summarized the hierarchy and process schematically, as illustrated in Figure 4.1.

The process begins with a user request and this request is taken as an input by the cloud system and termed as a cloudlet in CloudSim. A service broker representing a client checks resource information through an entity called Cloud Information Service (CIS), determines suitable resources and then submits the request to the selected resources for processing. During the resource selection and submission procedure, certain resource allocation and task scheduling policies are used for routing user requests to the appropriate virtual machines configured with CPU, memory, storage and bandwidth.

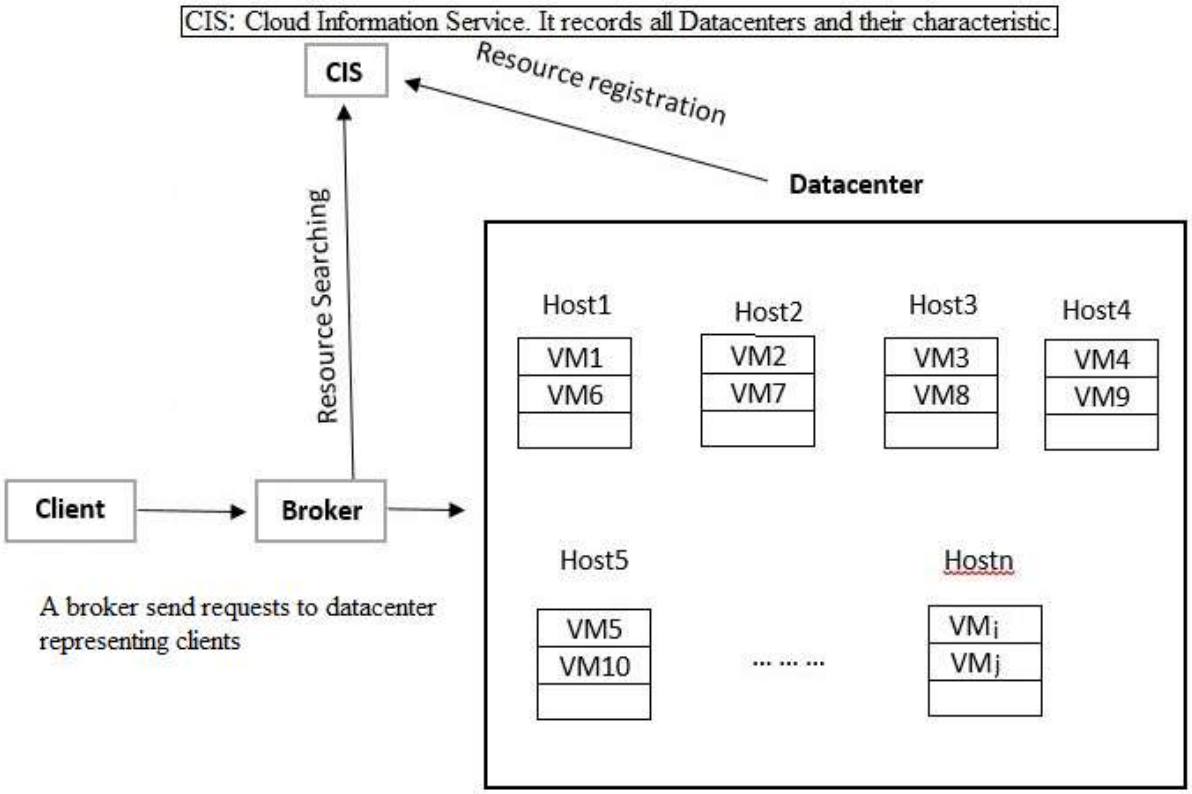

Figure 4.1: Hierarch and process of simulating the provision of services to clients in CloudSim

4.2 Extending CloudSim to include TAA

To implement the proposed TAA algorithm, we need to consider extending Cloudsim to include TAA from the three aspects:

- Create Cloudlets and VM Clusters
- Apply lower and upper thresholds on waiting queue of each virtual machine to dynamically detect the load amount of each virtual machine.
- Select a most suitable virtual machine for each cloudlet and submit, through comparing attributes of cloudlets with the suitability of each virtual machine, as well as through detecting the load amount information of each virtual machine. This is performed by a broker.

4.2.1 Create Cloudlets and VM Clusters

The task requested by user may have different attributes, thus some specific resources are needed. For the experiments, this thesis uses two types to characterize tasks as well as virtual machine clusters: computation, storage. In order to reach the best efficiency in the execution of each task, tasks will be matched with virtual machine clusters when broker receives new requests from users.

To find out virtual machine groups to be matched for each cloudlet based on rough attribute filtering, the code that perform both amount and attribute filtering is inserted into broker class. This includes examine and assign attributes to both cloudlets and virtual machines (VMs) according to their types and characteristics. As to assigning attributes to VMs, they can be classified and grouped based on different characteristics, such as, computation and storage. Then, according to the assignment, different clusters or groups of VMs are created. Similarly, cloudlet class can be created. A simple example is adding two attribute fields to cloudlet class: 'attribute 1' and 'attribute 2', and attribute 1 can be used for lower threshold, and attribute 2 can be used for upper threshold. Based on the cloudlet class and VM clusters, this way, matching cloudlets with virtual machine is checked whether attribute1 of cloudlets match VM's characteristic. If attribute1 is 'computation-intensive', then the VMs that is good at computation matches. Based on roughly match, strictly matching require attribute 2 meet some requirements, such as close dead line or high pay.

The broker checks if a cloudlet attribute matches with the suitability of a resource, and then accordingly selects an appropriate virtual machine cluster for the cloudlet. The simulation steps are described as follows:

Step1: Create VMs that belong to two basic clusters. For example, storage, computation.

Step2: Create cloudlets that have two different attributes. For example, storage-intensive, computation-intensive.

Step3: Check the attributes of each cloudlet.

Step4: Check the available VM clusters.

Step5: Match cloudlets with VM clusters.

Table 4.1 illustrates an example of the process of roughly filtering clusters for the matching.

Cloudlet_ID	Roughly Matched VM groups
cloudlet1- cloudlet100	vm1,vm3
cloudlet101-cloudlet200	vm2,vm4

4.2.2 Apply Lower and Upper Thresholds

In order to apply lower and upper thresholds on waiting queue of each virtual machine, a new class that extends class `CloudletSchedulerSpaceShared` is created for adding two thresholds (LoT and HiT) and load status updating mechanism. To this end, this thesis makes use of a producer and consumer mechanism: on the one hand, the queue is accepting new cloudlets into the queue of an appropriate virtual machine, and on the other hand, the queue delivers a cloudlet to the virtual machine. Both of the import and export action may change load status of the virtual machine. Figures 4.2 (a) and (b) illustrate the flowchart how the import and export action as well as thresholds influence the load status of a virtual machine. In the figures, “LoT” means Lower Threshold, “HiT” means the higher or upper threshold, and “LoT full” means the queue under lower threshold has been fully occupied by queuing cloudlets, and “OK into LoT” means arriving cloudlets can still enter the queue because the queue below LoT has not been fully occupied. “HiT full” indicates the queue under the higher threshold has been fully occupied. The corresponding description along with flowchart in Figure 4.2(a) and (b) are described as follows:

A cloudlet joins the queue（producer）:

Step 1: Initial load_status as ’OK into LoT’.

Step 2: A new cloudlet joins the queue.

Step 3: Check current queue size. If the current queue size is greater than or equal to LoT, update load status of the virtual machine to ‘LoT FULL’; if the current size is greater than or equal to HiT update load status of the virtual machine to ‘FULL’.

Step 4: Repeat Step 1.

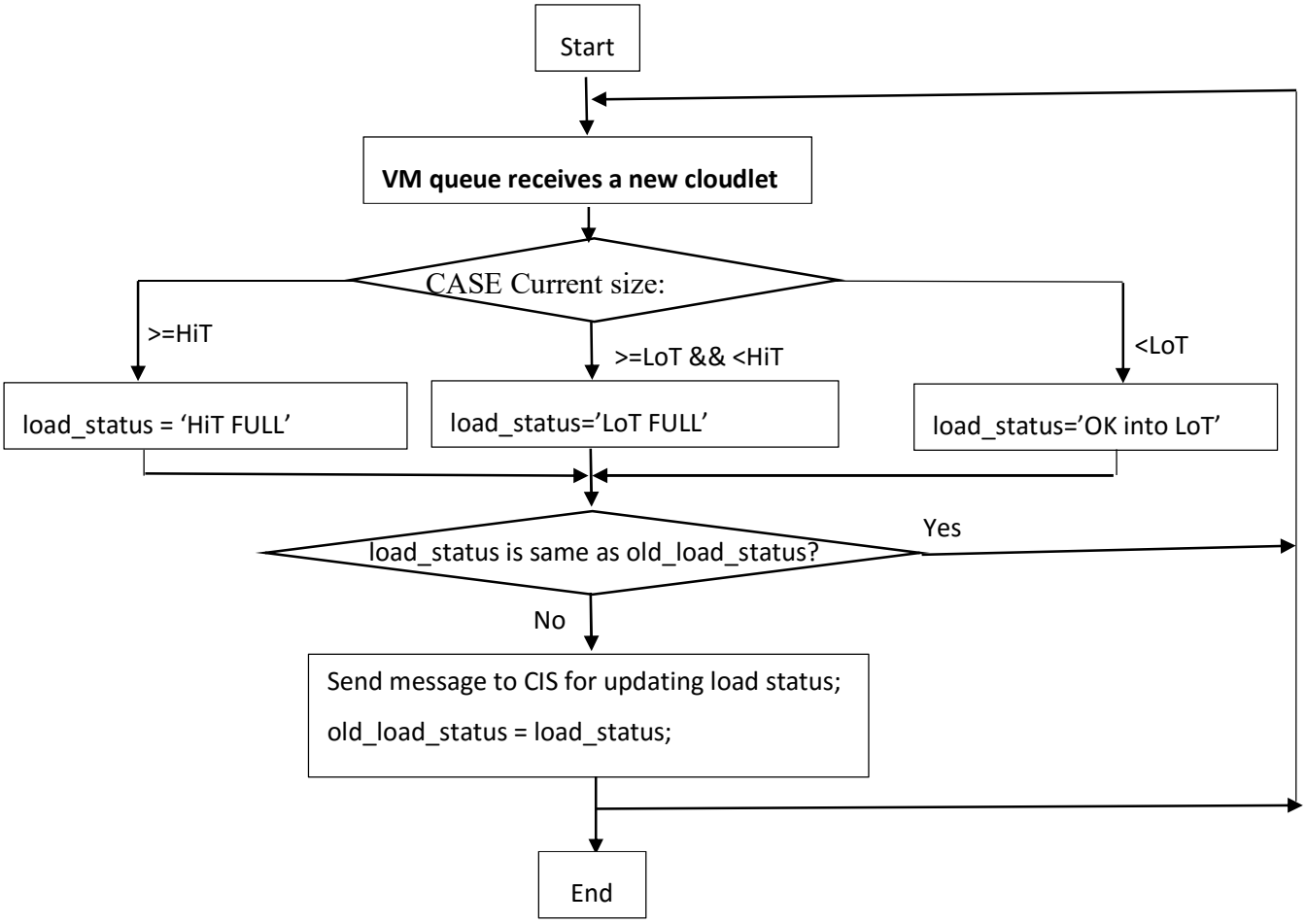

Figure 4.2(a): Flowchart showing "Cloudlets join the queue of a virtual machine" in TAA

A cloudlet ends its queueing and enters VM (consumer):

Step 1: A cloudlet at the queue head finishes its queueing and enters the virtual machine when the machine accepts the cloudlet.

Step 2: VM scheduler checks current queue size. If the current queue size is equal to LoT, check old load status, if old load status is not 'LoT FULL', then send message to CIS for updating load status of the virtual machine to 'LoT FULL'; if the current size is equal to HiT, check old load status, if old load status is not 'FULL', then send message to CIS for updating load status of the virtual machine to 'HiT FULL'.

Step 3: Repeat Step 1.

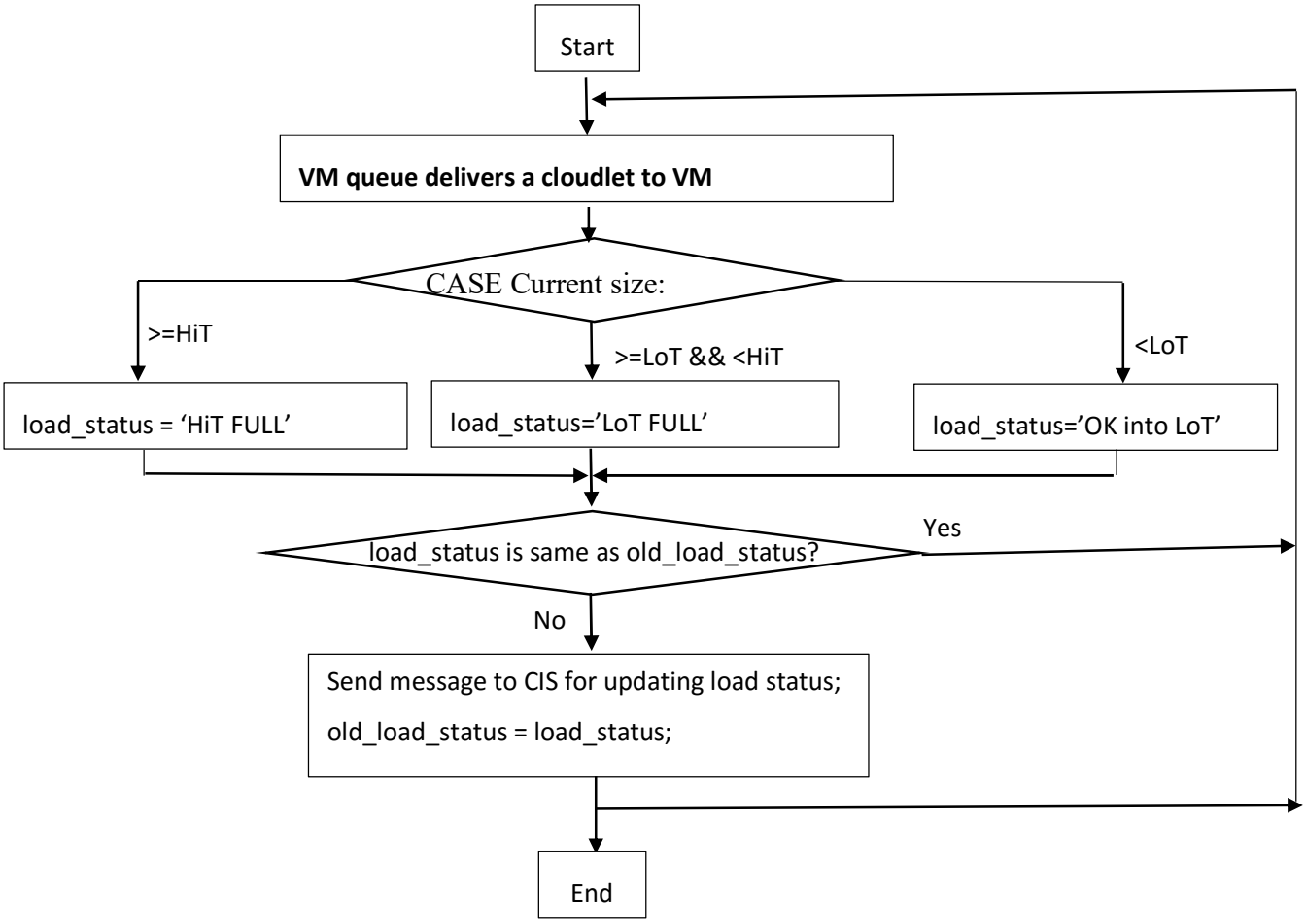

Figure 4.2 (b): Flowchart showing "Cloudlets finish queuing and enter the virtual machine" in TAA

We may wonder where the broker finds the load status information of each virtual machine. In fact, as mentioned in section 4.1, an entity called Cloud Service Information (CIS) in CloudSim save all related important information of all resources. Real time load status of each virtual machine is saved in CIS and can be read by the broker. The load status of each virtual machine is updated when the waiting queue size changes around the lower and upper thresholds of each virtual machine. Some examples showing load status of virtual machines are in Table 4.2.

Table 4.2: Examples showing load status of virtual machines

VM_ID	Load_status
vm1	'OK into LoT'
vm2	'LoT FULL'
vm3	'HiT FULL'
vm4	'OK into LoT'
vm5	' LoT FULL'

4.2.3 VM Selection and Cloudlet Submission

Allocating an appropriate virtual machine to cloudlets is based on load status of virtual machines and attributes of cloudlets, and the codes are inserted into broker class. The rules of allocation are briefly described as follows: (1). Allocate cloudlets to the most suitable virtual machine if the virtual machine is available; (2) If the best suitable virtual machine is full, then allocate the cloudlets to a can-do virtual machine, because a lower performance of the processing is better than not being processed.

Each time before cloudlet submission, the broker checks the load status (OK into LoT, LoT FULL, or HiT FULL) of virtual machine. If load status is 'OK into LoT', then the broker directs the cloudlets that meet the rough filter standard to this virtual machine. If load status is 'LoT FULL', then only the cloudlets that meet the strict filter standard are sent to the queue of the VM, the cloudlets which do not meet the strict filter standard will be allocated to another virtual machine that may be not very suitable but can do the job. If load status is 'FULL', find another virtual machine that may be not very suitable but can do the job. The corresponding flowchart and pseudo-code are illustrated in Figure 4.3 and Figure 4.4.

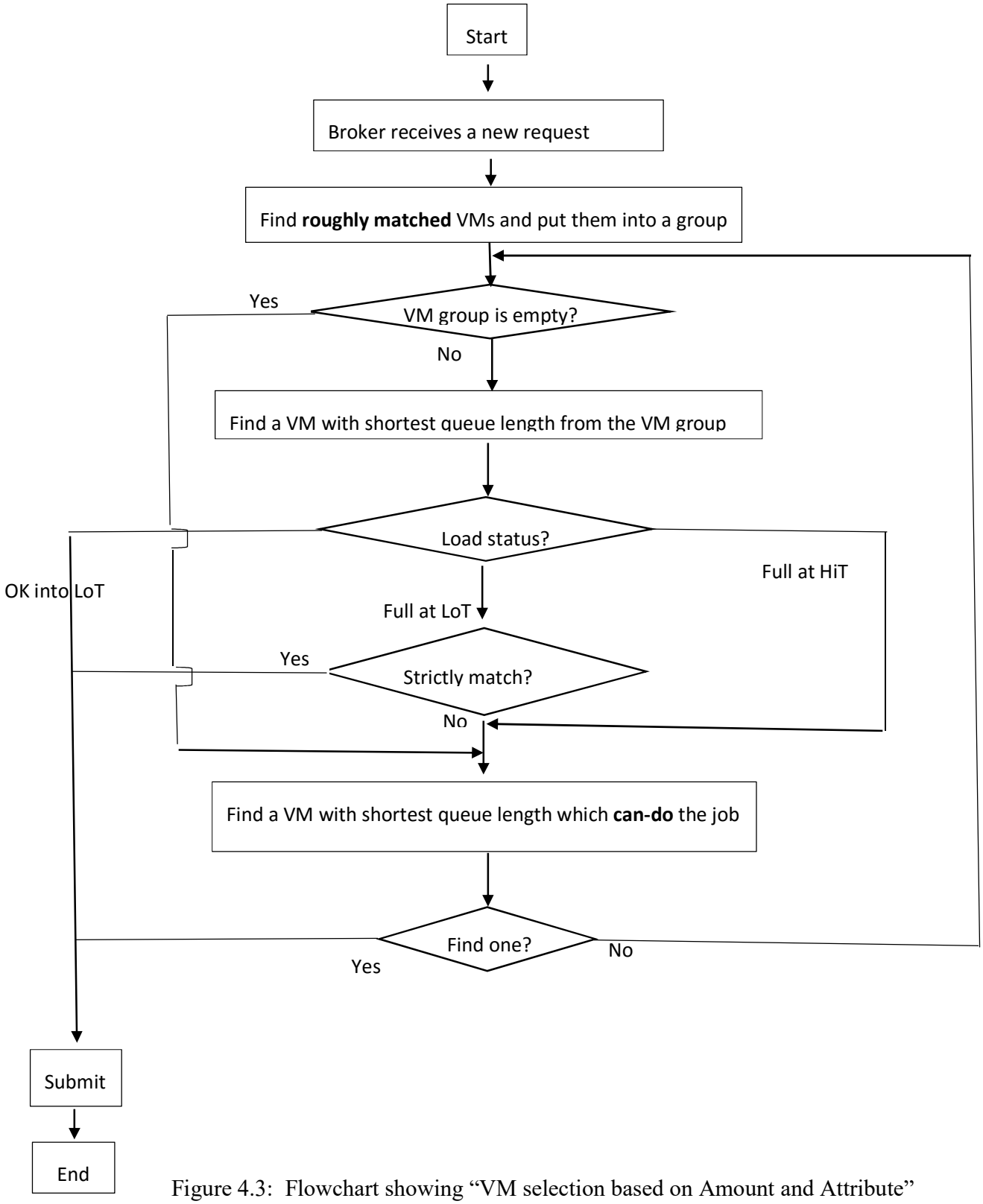

Figure 4.3: Flowchart showing "VM selection based on Amount and Attribute"

```
Input: a cloudlet and virtual machines
Output: submit (true) or not (false)

Boolean Function SendCloudletToVM(Cloudlet cloudlet)
{
    /*Find a roughly matched VM that is under LoT */
    for (all VMs which is roughly matched){
        if (VM is “OK into LoT”) {
            Submit cloudlet to the VM;
            Return true;
        }
    }

    /*If all roughly matched VMs exceed LoT, find a strictly matched VM which is between LoT and HiT
*/
   for (all VMs which is roughly matched){
        if (VM is “LoT FULL” and “strictly match”) {
                    Submit cloudlet to the VM;
                    Return true;
        }
    }

    /*If filtered out by the last two steps, find a VM which has the most slight load and can do the job */
    for (all other VMs that can-do){
        Find a VM with shortest queue length {
            Submit cloudlet to the VM;
            Return true;
        }
    }

   Return false;
}
```

Figure 4.4: Pseudo-code of VM selection based on Amount and Attribute

4.3 Function Streamlining of the Insertions

The inserted codes must be streamlined into Cloudsim to assure the compliance with both TAA algorithm and Cloudsim's flow. To approach this streamlining goal, this thesis adopts the following flow: users submit their tasks to Datacenter Broker, which acts as a dispatcher between user and data center and helps schedule tasks on virtual machines, and in the data center there are a number virtual machines on hosts and tasks are scheduled on those VMs according to the proposed TAA algorithm. In Figure 4.5, the overall simulation procedure of the proposed algorithm is schematically illustrated.

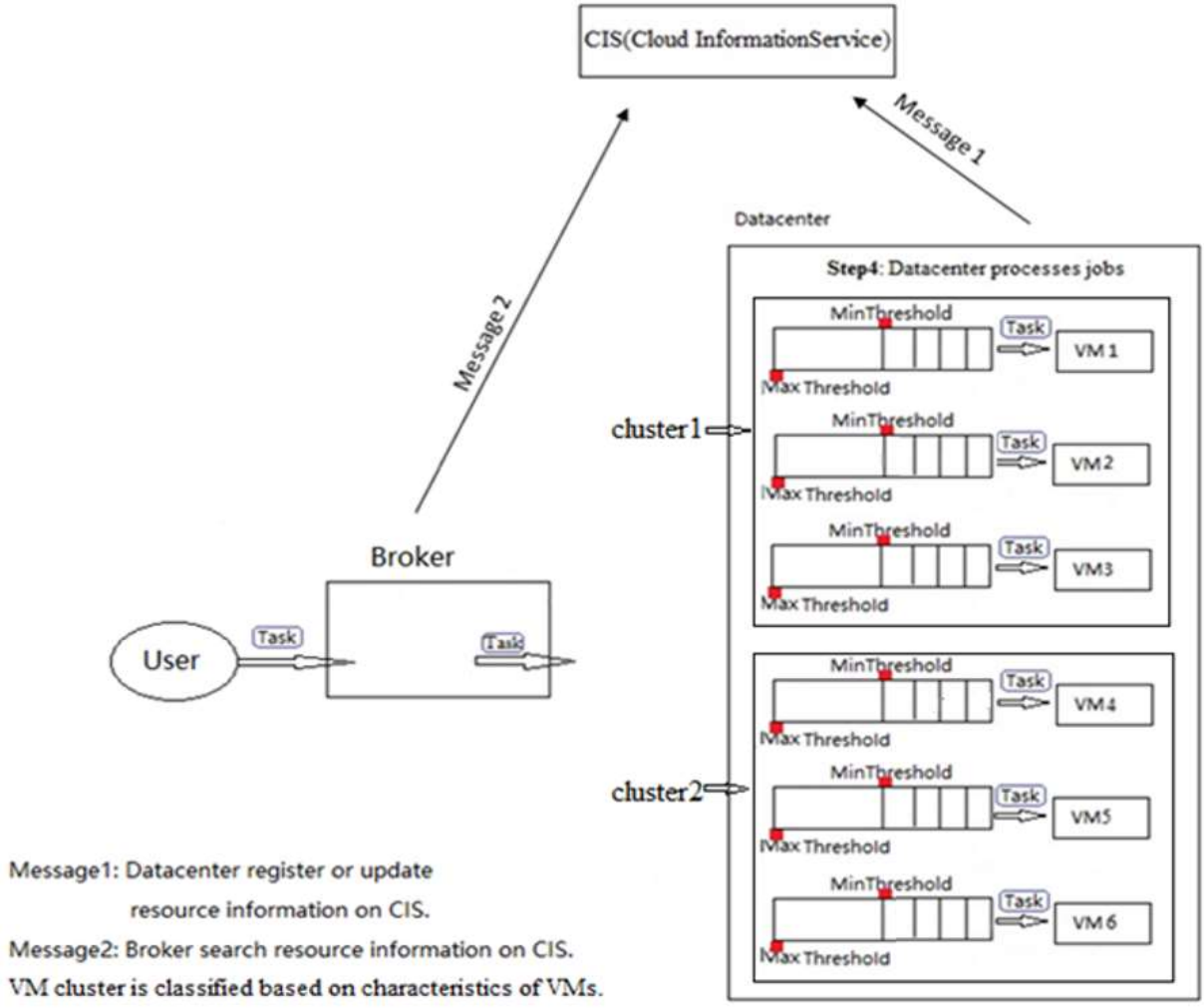

Figure 4.5: Overall Simulation procedure of the proposed algorithm in CloudSim

4.4 Process of Task Scheduling and Resource Allocation

It is also very helpful if the simulation process can be schematically explained as much as possible. To this end, this section illustrates the procedure how tasks are sent from the broker to virtual machines in Figures 4.6 (a) and (b), by using a limited number of cloudlets and VMs, although there are much more cloudlets and VMs in a real cloud environment for task scheduling and resource allocation. The distribution of cloudlets based on attributes and thresholds is also illustrated in Figures 4.6 (a) and (b). Figure 4.6 (a) shows the situation in which all virtual machines are still under lower thresholds because the total amount of cloudlets (from 1st to No. 9th cloudlets) is small, so the cloudlets are only roughly filtered by the lower thresholds of the VMs. This is why although VM1 is best suitable for cloudlet 3 (e.g., with "high pay + computation intensive" attributes), other cloudlets (e.g., "low pay + computation intensive") are also allowed to enter the queue of VM1, because the cloudlets all share the same parental attribute (e.g., computation intensive) that can be used for a rough filtering by the lower threshold. Note that the underlined cloudlets in Figure 4.6 (a) and (b) are the cloudlets that are more suitable for the VM than other cloudlets in the same queue of the VM.

When a VM is processing cloudlets, more cloudlets (from No. 10 to No.80) may request the usage of the VM, then the queue size may exceed the limit of the VM's lower threshold. In this situation, its stricter filter will be imposed on incoming cloudlets, i.e., higher threshold will perform this filter function. For a very busy data center, VMs' lower thresholds may be fully occupied by their specifically suitable cloudlets, Figure 4.6 (b) illustrates this situation: the cloudlets are filtered into the queue of each VM between the VM's lower and upper thresholds.

Because the full filling of a lower threshold is dynamic, i.e., when some cloudlets are coming into the queue of a VM, some other cloudlets are disappearing in the queue because they have been processed by the VM. To illustrate the dynamic filling and queuing of the lower threshold as well as the activation of the higher threshold, VM1 is taken as an example and the dynamic events are enlarged in Figure 4.7. After VM1's higher threshold is activated, late coming cloudlets are strictly filtered for being accepted into the queue of VM1, so only the cloudlets numerated 18, 20, 23, 24, … …, 34, 69 can be arranged to the queue of VM1 among the total 90 different types of cloudlets, because only these cloudlets match the strict filtering standard of VM1's higher

threshold. The thresholding and queuing of other VMs are similar to VM1, which are not described again in this chapter.

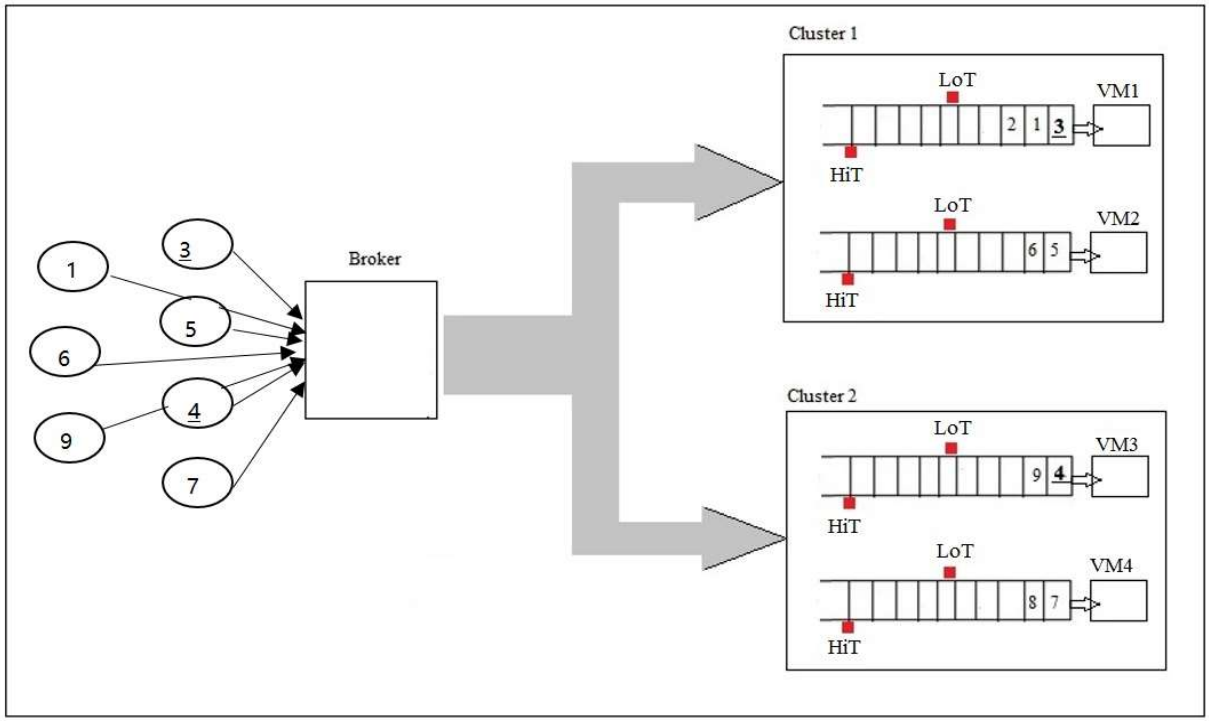

Figure 4.6 (a): Jobs Distribution based on Attribute and Thresholds (under LoT)

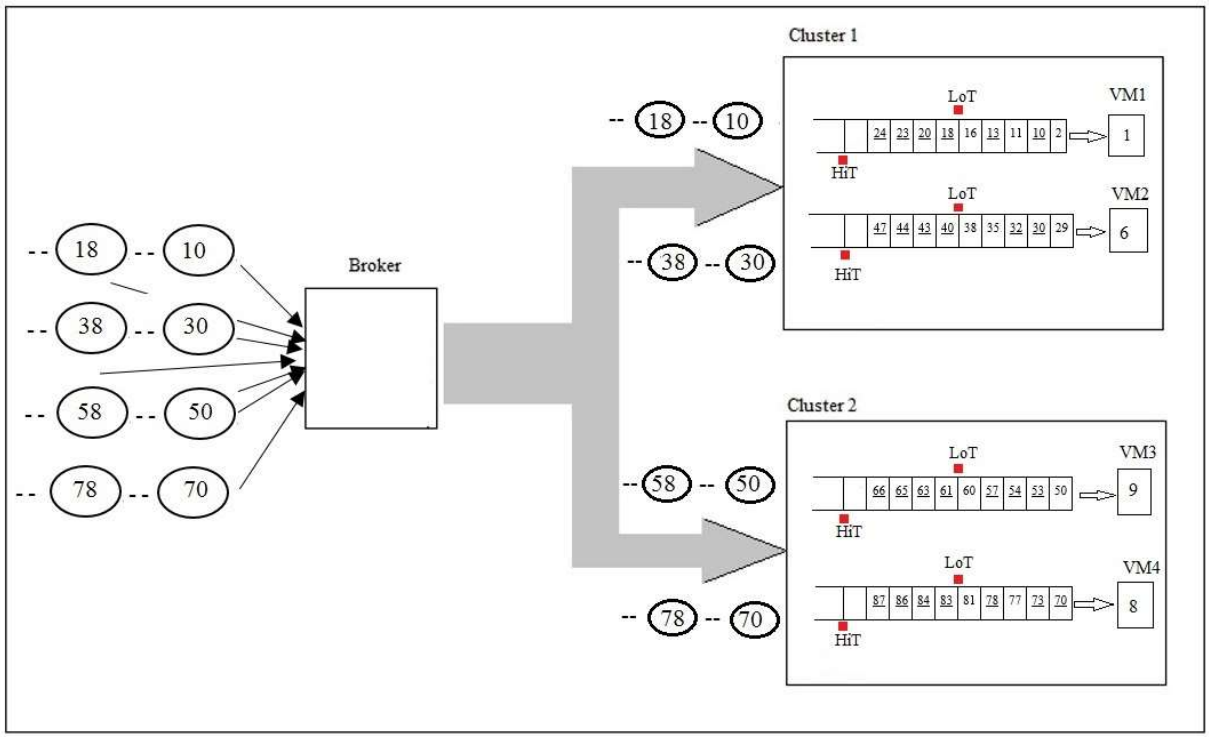

Figure 4.6 (b): Jobs Distribution based on Attribute and Thresholds (between LoT & HiT)

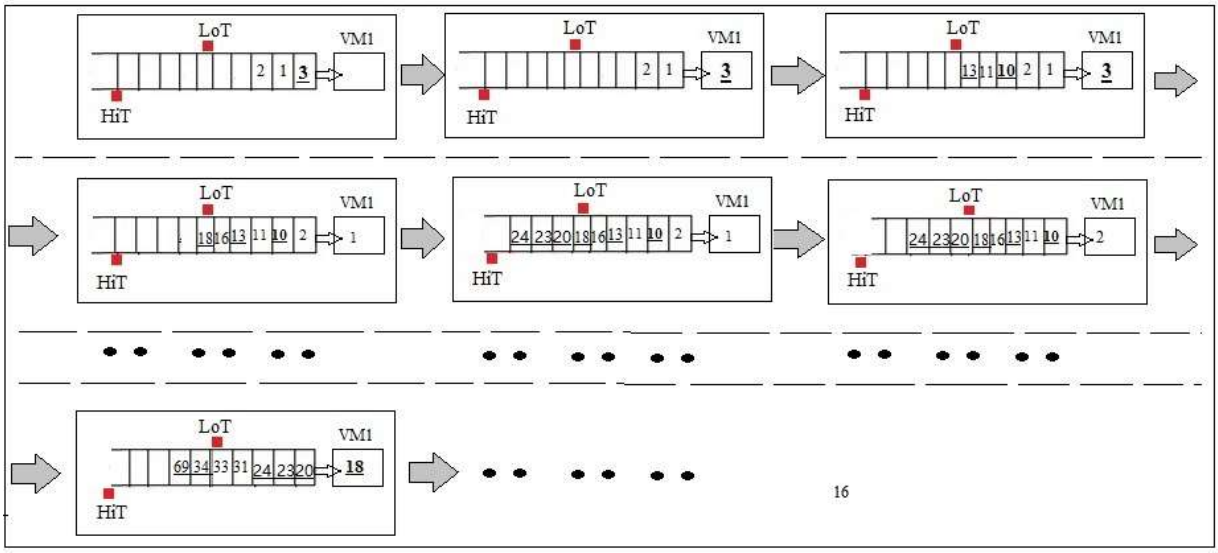

Figure 4.7: Process of more cloudlets entering the queue and being filtered by upper threshold of VM1

Chapter 5. Experimental Results and Discussion

5.1 Performance Evaluation Parameters and Criteria for TAA Algorithm

As presented in previous sections, this thesis highlights a new task scheduling strategy based on two-level attribute filtering and workload amount. It is expected that the TAA strategy may eventually allocate tasks to the most appropriate resource and at the same time all resources are utilized to a maximum extent so as to improve the cloud performance. To assess the performance of the proposed TAA algorithm, some parameters for describing evaluation criteria are needed. In this thesis, the following parameters are adopted:

Arrival Time（AT）: It is defined as the instant when a cloudlet (i) arrives at the waiting queue of a virtual machine (VM).

Arrival Interval (AI): It refers to the ratio of the total time to the total amount of cloudlets that have arrived at the data center within the total time. It is noted AI is an average time which does not depend on the arriving sequence of cloudlets. For example, if totally 1,800 cloudlets of type A arrive at the datacenter within 18,000 seconds, then the AI for type A is 10s, although a cloudlet may arrive at the data center 30 seconds later than its adjacent earlier arriving cloudlet. In real world, cloudlets arrive at irregular intervals. Thus, we simulate the arrival interval following a Poisson distribution.

Starting Time（ST）: It is defined as the instant when a cloudlet finishes queuing and starts to be processed in a virtual machine. In detail, whenever a cloudlet (i) is accepted by a virtual machine (VM), if there were no other cloudlets to be assigned to that VM before, then the starting time of the cloudlet (i) is zero; otherwise, the starting time of the cloudlet is defined as the finishing time of the previous cloudlet (i-1) on that VM. Subsequently, the finishing time of cloudlet (i) is the starting time of next assigned cloudlet (i+1). The definition of the starting time of a cloudlet is mathematically expressed in Eq. (5.1):

ST(i) = 0; [if VM(j) is in free]

ST(i) > 0; [otherwise, ST(i) = finishing time of cloudlet(i-1)] .....(5.1)

Acceptance rate: It is the ratio of the amount of cloudlets actually accepted by a VM to the total amount of cloudlets that are expected to be accepted by the VM.

Execution Time (Exec): It is the time length that is defined as the ratio of MI (Million Instructions) constituting the cloudlet being processed to the MIPS (Million Instructions Per Second) of the allotted VM. The calculation of the execution time of a cloudlet is expressed in the Eq. (5.2):

Exec(i) = MI of Cloudlet(i) / MIPS of VM(j) .....(5.2)

Table 5.1: Parameters used in the experiments of a data center

Parameters reflecting attributes		**Values**
VM1 (best for computation)	CPU speed	3000 mips
	Storage	50GB
VM2 (best for storage)	CPU speed	1000 mips
	Storage	1000GB
VM3 and VM4 (acceptable for computation and storage)	CPU speed	2000 mips
	Storage	300GB
dimensions of thresholds	LoT for all VMs	500 cloudlets
	Between HiT and LoT for all VMs	300 cloudlets
Cloudlet	Amount of computation-intensive cloudlets (including high_pay and low_pay)	1800
	Amount of computation-intensive & high_pay cloudlets	180 (10% of total computation-intensive cloudlets)
	Amount of storage-requesting cloudlets (including high_pay and low_pay)	1200
	Amount of storage-intensive & high_pay cloudlets	120 (10% of total storage-requesting cloudlets)
	Average arrival interval	Follow Poisson curve

The development environment for the simulation experiments has been described in the previous chapter, which is based on CloudSim 3.0.3, JDK 8 and Eclipse Kepler Server Release 1. Using the parameters listed in Table 5.1, the experiments are performed on a Window 8.1 (64-

bit), AMD E1-2500 APU with Radeon(TM) HD Graphics 1.40GHz Processor, 4.0G installed memory.

5.2 Experimental Results and Discussion

To perform simple but typical experiments, 4 virtual machines and 3000 cloudlets are experimented. Among the 4 VMs, VM1 is more suitable for cloudlets requesting computation service, and VM2 is more suitable for cloudlets requesting storage service, and VM3 and VM4 can provide both computation and storage service, but their ability and speed for computation and storage processing are significantly lower than VM1 and VM2. In the experiments, VM1 and VM2 will be the focus of discussion for the performance assessment of TAA, because their attributes are more distinguished from one another, and the migration of cloudlets filtered out of VM1 and VM2 to VM3 and VM4 is an important process of load balancing, which is not the focus of this thesis. Nonetheless, the experimental results for VM3 and VM4 will still be briefly discussed, as a goal of setting up VM3 and VM4's significantly lower ability than VM1 and VM2's is to test the situation in which some cloudlets requesting high capable machines can be distributed to a machine with lower ability (VM3 and VM4 in this experiment) when the amount of cloudlets requesting VM1 and VM2 exceed the maximum size of VM1 and VM2, so as to avoid lengthy waiting time in front of a machine with higher ability (VM1 and VM2 in this experiment).

In the experiments, each cloudlet has two attributes: attribute1 and attribute 2. Attribute 1 is either 'computation-intensive' or 'storage-intensive'; and attribute2 is either 'high_pay' or 'low_pay'. Rough filter corresponding to lower level threshold (LoT) is only exerted on attribute1, and strict filter corresponding to higher level threshold (HiT) is imposed on both attribute1 and attribute 2.

Among the 3000 cloudlets, 1800 cloudlets have attribute1 of 'computation-intensive' and the other 1200 cloudlets have attribute1 of 'storage-intensive'. The cloudlets with 'attribute1=compution_intensive' are picked out by rough filter for VM1, and 'attribute = storage_intensive' for VM2. Regarding strict filter, 'attribute1=computation_intensive && attribute2=high_pay' are picked for VM1 and 'attribute1 = storage_intensive && attribute2 = high_pay' for VM2. After the filtering by VM1 and VM2, the rest of the cloudlets are conducted

to VM3 and VM4 to assure all resources in the data center are utilized with no waste or idling, and all cloudlets are being processed to avoid impressing customers that their requests are not served.

To discuss the experiment results, two situations should be considered: virtual machine queue size is under lower threshold, or virtual machine queue size exceeds lower threshold. We discuss the results from the two aspects because: under lower threshold, rough filtering mechanism is applied; and once lower threshold is reached, strict filtering mechanism will be triggered.

5.2.1 Applying Lower Thresholds to Cloudlets

In the experiments based on the parameters listed in Table 5.1, it is observed that when the 1,153th cloudlet of the total 3,000 is sent out of the broker to the queues of VMs, the lower threshold of VM1 will be fully occupied by cloudlets, while other VMs' lower thresholds are still accepting cloudlets. If the starting time is the instant at which the first cloudlet is sent out of the broker to the queues of the VMs, then the instant the lower threshold of VM1 is fully occupied is 6,546s. At this instant, the total amount of processed and queuing cloudlets of VM1 is 696, which is larger than 500, the maximum capacity of the lower threshold of VM1. This is not surprising, because in the meanwhile of the broker sending a cloudlet to the queue of VM1, another cloudlet is being processed by the VM1. The experimental results are listed in Table 5.2. Similar situations are occurring for all other VMs and subsequently arriving cloudlets, which will not be listed in detail in this section.

At what instant the queue of the lower threshold of a VM is fully occupied by cloudlets is influenced by both the duration of a cloudlet occupying a VM (execution time) and the interval of two adjacent cloudlets entering the queue of the VM (arrival interval). This is related to the dynamics of the queuing and thresholds, which will be discussed in detail in later sections. The section here mainly discusses a general case of how the lower threshold of VM1 is applying both attribute and amount to filter and distribute cloudlets in the simulation experiments.

To simulate and discuss a general situation, this thesis examines the early arriving 924 cloudlets of the 1153 in Table 5.2. Among the 924 cloudlets, 550 are computation-intensive and 374 are storage-intensive. It is noted that the selection of 924 is because it is smaller than 1153 and 550 is also smaller than 696 in Table 5.2. Hence the queuing is not flooding out of any threshold

of the 4 VMs, which is convenient to study the functioning of lower thresholds. In fact, any other amount different from 924 can also be selected if it is convenient for examining the lower threshold.

Table 5.2. The events before the lower threshold of VM1 is fully occupied

Time	**Events of cloudlets by applying TAA**
At instant 0s	The first cloudlet of the 3,000 enters one of the queues of the 4 VMs
At instant 6546s	The 1,153th cloudlet of the total 3,000 is sent out of the broker to the queues of VMs.
	VM1 has accepted 696 cloudlets of the 1,153. Among these 696 cloudlets, 196 cloudlets have been processed and 500 cloudlets are queuing and making VM1 fully occupied.

Table 5.3 lists the acceptance rate of each virtual machine in the simulated environment. Through rough filtering mechanism, 100% computation-intensive cloudlets are accepted by VM1, 100% storage-intensive cloudlets are accepted by VM2, and no cloudlets are allocated to VM3 and VM4. In other words, this results verified the assumption: VM1 and VM2 can 100% accept the most suitable cloudlets if amount limit of the cloudlets is under their lower thresholds, and there is no need to utilize the less capable VMs (VM3 and VM4), because VM1 and VM2 have sufficient computation and space to suitably arrange all cloudlets.

The results are not surprising, because the lower thresholds of VM1 and VM2 roughly filter the 924 cloudlets, so 550 computation-intensive cloudlets should be accepted by VM1 which is illustrated in Figure 5.1(a), 374 storage-intensive cloudlets should be accepted by VM2 which is illustrated in Figure 5.1(b).

Table 5.3: Acceptance Rate of the Early Arriving 924 Cloudlets by the Lower Thresholds of VMs (the maximum queuing capacity of each lower threshold = 500 cloudlets)

Attribute of Cloudlets	Amount of accepted cloudlets	Acceptance Rate of VM1	Acceptance Rate of VM2
Computation-intensive	550	100%	0%
Storage-intensive	374	0%	100%

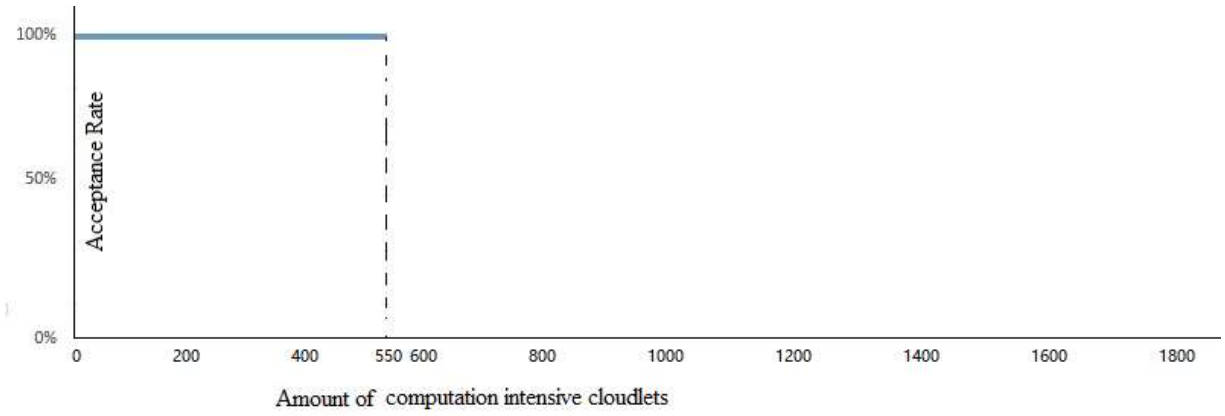

Figure 5.1(a): Acceptance Rate of VM1 to the Early Arriving 924 Cloudlets (550 are computation-intensive) by the Lower Thresholds of VM1 (the maximum queuing capacity of lower threshold = 500 cloudlets)

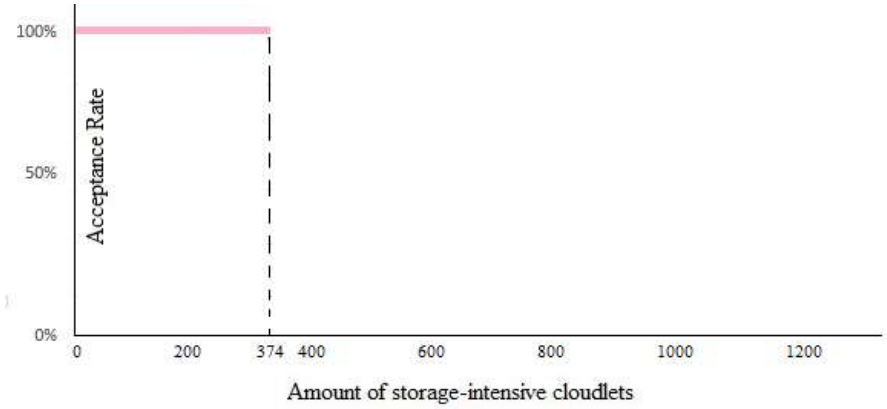

Figure 5.1(b): Acceptance Rate of VM2 to the Early Arriving 924 Cloudlets (374 are storage-intensive) by the Lower Thresholds of VM2 (the maximum queuing capacity of lower threshold = 500 cloudlets)

The above experimental results and discussion mainly focus on the advantage of optimal matching of cloudlet and resource attributes, and obviously this advantage is brought by the TAA's filter ability resulted from using lower threshold. Can this ability bring more advantages, e.g., reducing the average execution time for those cloudlets that are more important or urgent than other ones, and consequently make the VM more lucrative, valuable, or efficient? As a data center or cloud environment may have limited resources that can be equally allocated to all different cloudlets with different importance and urgency, so focusing on reducing the execution time of the more urgent or valuable tasks is often a more realistic and efficient option, although reducing execution time might be preferred by all requests. For instance, computation-intensive cloudlets may prefer faster execution more than storage-intensive cloudlets, as in the minds of the customers, they may haunt themselves by thinking "can my calculation jobs be done as soon as possible?" In comparison, the customers requesting storage may not be same haunted by the how fast question.

Reducing the execution time of a cloudlet means the cloudlet enjoyed a faster execution. So a quantitative comparison of the execution time difference between applying and not applying lower thresholds to cloudlets would be helpful to examine the improvement of task scheduling and resource allocation. Table 5.4 lists the distribution of computation-intensive cloudlets on 4 VMs by using TAA and another policy without filter. In the non-filter policy for both the general and the special cases of Table 5.4, once the broker receives a cloudlet, it will immediately send the cloudlet to VMs in a rotating sequence, which is described as follows:

(1) When the broker adopting the non-filter policy receives a cloudlet, it will send the cloudlet to VM1, regardless of the attribute of the cloudlet;
(2) When the broker receives a second cloudlet, it will send the cloudlet to VM2, regardless of the attribute of the cloudlet;
(3) when the broker receives a third cloudlet, it will send the cloudlet to VM3, regardless of the attribute of the cloudlet;
(4) when the receives a fourth cloudlet, it will send the cloudlet to VM4, regardless of the attribute of the cloudlet;
(5) For more future cloudlets, repeat the rotation of steps (1) and (4).

This way, the non-filter policy may immediately process all coming cloudlets and make full use of all VMs. This will be fairer for the comparison with TAA that filters the attributes first and then allocates a most appropriate VM to different cloudlets rather than rotationally utilizing all VMs.

Generally, computation-intensive and storage-requesting cloudlets may come to the broker in a randomly alternating sequence, for example, in a time window, 5 computation cloudlets may come to the data center continuously and then 2 storage-requesting cloudlets come, then 3 computation-intensive cloudlets follow. This is also a dynamic feature of task scheduling and resource allocation. Considering this, Table 5.4 lists a general case in which the arrivals of 550 computation-intensive cloudlets come are often intermixed with the arrivals of the 374 storage-requesting cloudlets. Because all four VMs are rotationally arranged to all cloudlets, as a result of the random arrival sequence of different cloudlets, the 550 computation-intensive cloudlets are not equally allocated to the 4 VMs. VM1 shares 100 of the 550 cloudlets, VM2 shares 135, VM3 shares 210, and VM4 shares 105. It is noted if the arrival sequence is changed, the above allocation may change accordingly, therefore there may be many other distributions. This section only uses the distribution listed in Table 5.4 to discuss the execution in subsequent section.

A very special case in applying the non-filter policy is that all the 550 computation-intensive cloudlets come to the broker before all the storage-requesting 374 cloudlets, then the 550 cloudlets will be equally allocated to all VMs. This very special case is also listed in Table 5.4 so as to provide a more comprehensive comparison with the performance TAA. This special case is

mathematically analogous to the situation that the broker performs sorting among 924 cloudlets and first process the 550 computation-intensive cloudlets, then process the rest 374 storage-requesting cloudlets, although the broker received the 924 cloudlets in a random sequence. Again, because all four VMs are rotationally arranged to all cloudlets, as a result of the special arrival sequence, the 550 cloudlets are equally spread to the 4 VMs, i.e., each VM shares 137 or 138 of the 550 cloudlets. As half (0.5) cloudlet is not allowed, so there is a slight difference between 127 and 128, instead of allocating 137.5 cloudlets to each VM. The execution time of this very special case will be also examined and compared to the TAA proposed in this thesis.

Table 5.4: VM allocation for computation-intensive cloudlets by using the LoT of TAA and another policy without filter

Policy		VM allocation of total 550 computation-intensive cloudlets
Non-filtering policy	General Case (all computation-intensive and storage-requesting cloudlets come in random sequence)	100→VM1, 135→VM2, 210→VM3, 105→VM4
	Special Case (all the 550 computation-intensive come before all the 374 storage-requesting cloudlets)	137→VM1, 137→VM2, 138→VM3, 138→VM4 (equally allocated to VM1, VM2, VM3, and VM4)
TAA policy		550→VM1

Figures 5.2 (a) and (b) compare the average execution time resulted from applying TAA and non-filter policy to computation-intensive cloudlets for the general and special cases of Table 5.4. It can be observed that the average execution time for each cloudlet by using TAA algorithm is only 56% (general case) or 57% (special case B) of the counterpart of by using the policy without

filtering mechanism. This means, TAA performs much better than the non-filter policy, regardless of whether the 550 computation-intensive cloudlets are randomly or equally spread to 4 virtual machines in the non-filter policy. Also, the average execution time resulted from using TAA does not depend on the order of the cloudlets being sent out of the broker, this is because in TAA the broker will anywise perform filtering, so all the 550 computation-intensive cloudlets are eventually allocated onto VM1.

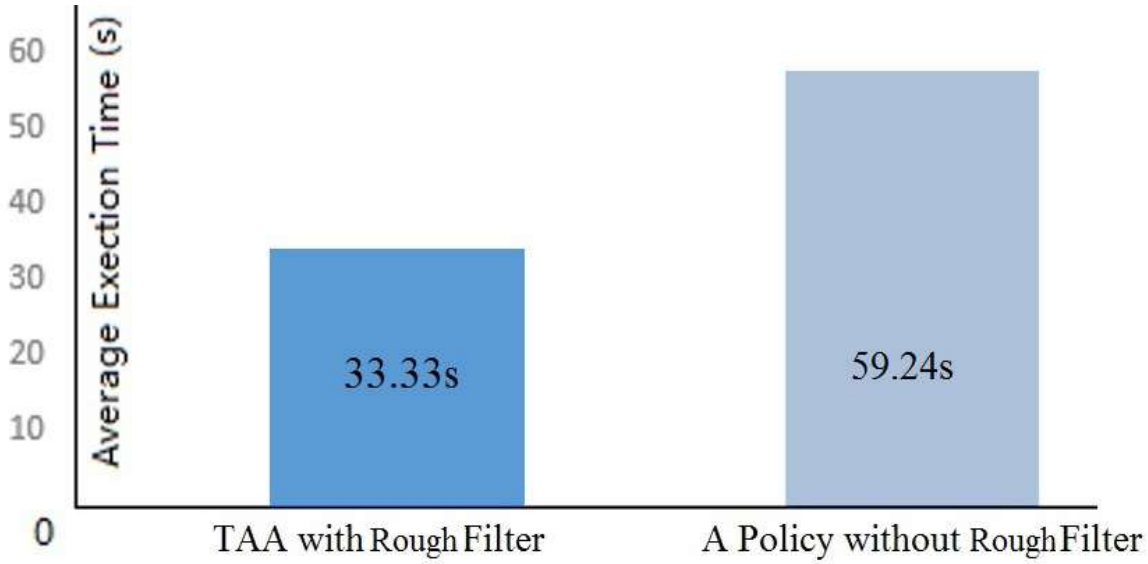

Figure 5.2(a): Comparison of execution time (General case: computation-intensive and storage-requesting cloudlets arrive in random sequence)

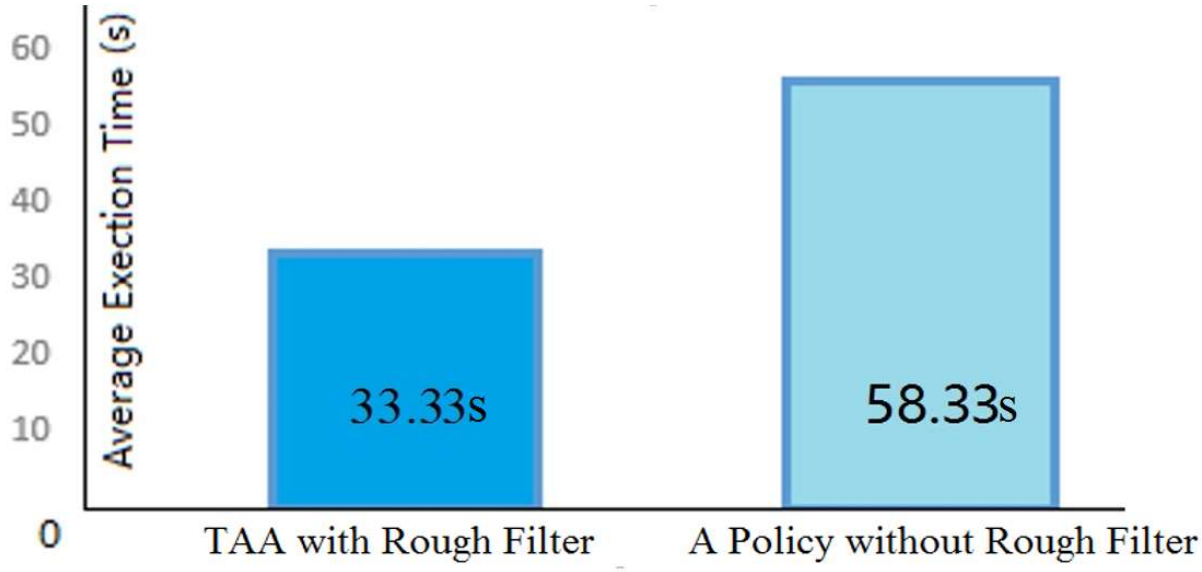

Figure 5.2(b): Comparison of execution time (Special case: all computation-intensive come before all storage-requesting cloudlets)

The results are within expectation, because under lower threshold, TAA algorithm allocates all computation-intensive cloudlets (550) to VM1 which has a more powerful CPU than other VMs. If not using an algorithm with the filter mechanism, some cloudlets would be allocated to other VMs with less power CPU and as a consequence, these cloudlets would occupy more execution time. Therefore, the average execution time of computation-intensive cloudlets by using TAA is much shorter than using a resource allocation mechanism which does not consider attributes.

Because VM2 will often be occupied by computation-intensive cloudlets if the cloudlets is not conducted to VM1 that has a faster CPU, so the execution time reduced on the computation-intensive cloudlets will in turn save the time of some cloudlets that request storage in VM2, this is an advantage of using the filter mechanism of lower thresholds. However, this does not mean the average execution time of all cloudlets requesting storage will be reduced, because a portion of the cloudlets to be processed by VM1 (faster CPU) are reallocated to VM2 (slower CPU) due to the functioning of lower thresholds, thus it is difficult to assess whether the execution time of this portion of cloudlets is reduced.

Fortunately, as customers requesting storage are more interested if their data can be stored reliably or if they can be provided a large storage capacity, so once the broker receives their data,

the customers would usually believe their data has been stored, so the CPU speeds of the machines of the data center do not matter so much to them as to those customers requesting intensive computation.

5.2.2 Applying Higher Thresholds to Cloudlets

When more and more cloudlets are submitted by broker to VMs, the queues of all virtual machines may become busy and eventually exceed the lower threshold of one VM or more thresholds of VMs. Once the lower threshold of a VM is exceeded, the stricter filtering mechanism of the VM's higher threshold will start. To examine the functioning of higher threshold, this section will extend last section's lower threshold simulation experiments to higher threshold simulation through continuing sending cloudlets to the broker. Similarly, the experiments here will also examine the acceptance rate and execution time at a random instant at which both VM1 and VM2 are working between LoT and HiT.

The simulation environment is same as for examining lower thresholds in last section (for example, at instant 0s, the first cloudlet of the 3,000 enters one of the queues of the 4 VMs). Following the early arrived 924 cloudlets discussed in Table 5.3 and Figure 5.1, more and more cloudlets are submitted by the broker to the VMs continually and when VM1 receives its 696th cloudlet at instant 6546s, its queuing capacity of allowing a maximum of 500 cloudlets is fully utilized and 196 cloudlets have been completed by VM1, as described in Table 5.5.

At this instant, the broker would take a precautious measure to avoid congestion of VM1, so the higher threshold will be imposed on the queuing for VM1. In addition to the attribute "computation-intensive" that is used in lower threshold, another attribute "high pay" will be added to the filter of the higher thresholds. This means a "high pay for high performance" service is provided in this situation. It is noted that any attribute such as "cyclone in weather forecast" other than "high pay" can also be adopted by the broker for the additional strict filtering. Since this paper focuses on examining the two-level thresholds rather than on defining many different attributes, so "high pay" is used for simplicity and common sense. Other computation-intensive cloudlets with low-pay attributes filtered out of VM1 will be allocated to VM3 and VM4, as they have a faster CPU than VM2 has, as indicated in Table 5.1.

Table 5.5. The events after the higher thresholds of VM1 and VM2 are started.

Time	**Events of cloudlets by applying TAA**
At instant 6546s (VM1's HiT starts to function)	Totally 1,153 of the 3,000 cloudlets for experiments have been sent by the broker to the queues of VMs, which are distributed as: 696→VM1 (500 queuing + 196 processed), its LoT is just full and HiT just starts to function. 457 →VM2, LoT is not full
At instant 8553s (VM2's HiT starts to function)	Totally1,517 of the 3,000 cloudlets for experiments have been sent out of the broker to the queues of VMs, which are distributed as: 756→VM1, its LoT has been full and HiT has been started for 2007 s. 586→VM2 (500 queuing + 86 processed), its LoT is just full and HiT just starts to function. 175→VM3 and VM4, below LoT.
At instant 9003s (randomly selected to examine the TAA performance)	Totally1,612 of the 3,000 cloudlets for experiments have been sent out of the broker to the queues of VMs, which are distributed as: 771→VM1, its LoT has been full and HiT has been started for 2457 s: 29 filtered in by HiT + 742 filtered in by LoT and processed by VM1. 592→VM2, its LoT has been full and HiT has been started for 450 s: 4 filtered in by HiT + 588 filtered in by LoT and processed by VM1. 249→VM3 and VM4, still below LoT.

When VM1 is just full and activates its upper threshold, VM2 is still under its threshold and accepting storage-intensive cloudlets until at instant 8553s its queue is full of 500 cloudlets and 86 cloudlets have been processed. At this instant, VM2 also activates its upper threshold by utilizing "high pay" as a screening standard in addition to the attribute "requesting storage". This is similar to the functioning of VM1's upper threshold and also described in Table 5.5. It is noted that at this moment, VM1 and VM2's queues of lower thresholds are both full and VM1's LoT has started for 2007 seconds.

To examine the functioning of the higher thresholds, this thesis randomly picked out an instant (9003s in Table 5.5) at which both the upper thresholds of VM1 and VM2 have been functioning for some time. It is noted that other instants are also fine for the discussion. Figures 5.3 (a) and (b) illustrate the acceptance rate of VM1 and VM2 for strictly matched cloudlets after their upper thresholds are activated, and Table 5.6 gives the brief description for the figures. Among all the cloudlets, 29 cloudlets are computation-intensive and high pay, 4 cloudlets are storage-requesting and high pay.

The results show that 100% strictly matched cloudlets are accepted by the best suitable virtual machines. In other words, strict filter mechanism imposed by the upper thresholds may guarantee all the strictly matched cloudlets are allocated to the best suitable virtual machines. The results are within expectation and the explanation is similar to that for the functioning of lower thresholds, which is not repeated in this section.

Table 5.6: Acceptance rate of cloudlets by the VMs after their upper thresholds are activated.

Attributes of Cloudlets	Amount of cloudlets	Acceptance rate by VM1	Acceptance rate by VM2
Computation + high pay	29	100%	0%
Storage + high pay	4	0%	100%

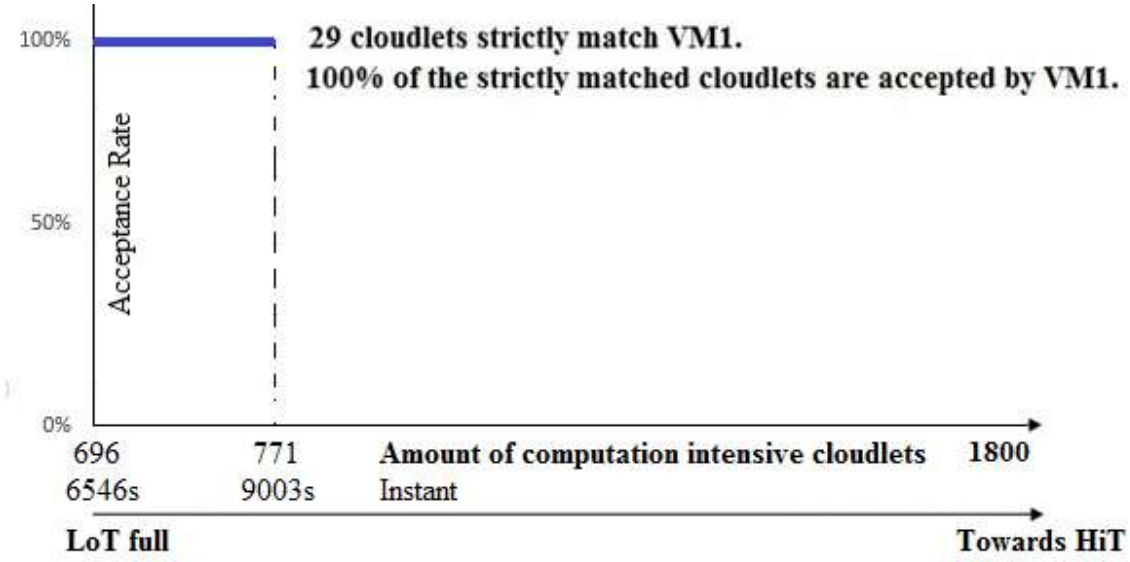

Figure 5.3(a): Acceptance rate of strictly matched cloudlets by VM1's upper threshold.

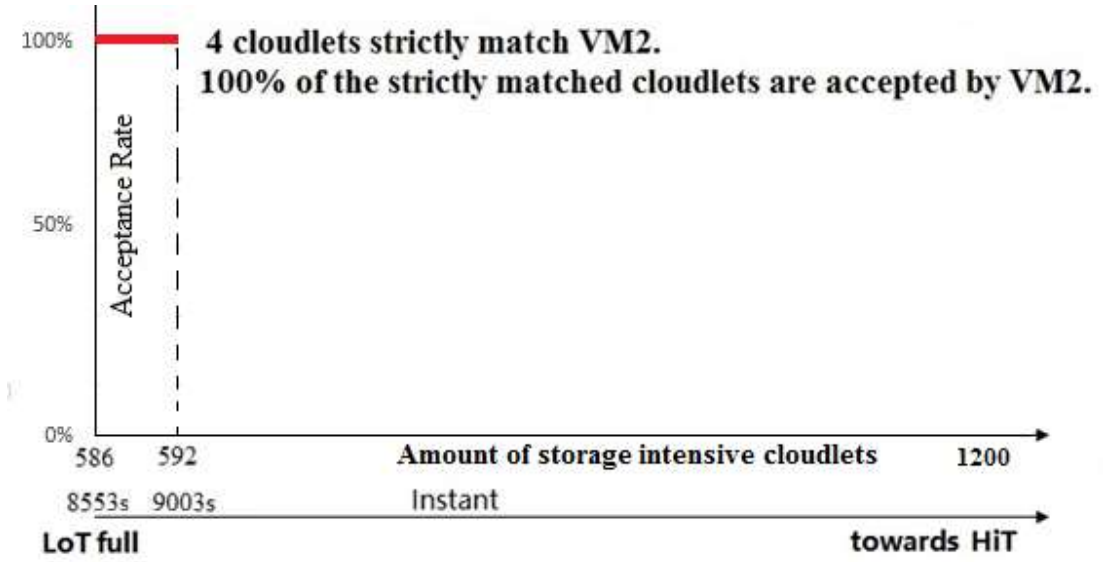

Figure 5.3(b): Acceptance rate of strictly matched cloudlets by VM2's upper threshold.

Even though 100% strictly matched cloudlets are accepted by the best suitable virtual machines, the overall acceptance rate is decreased when higher threshold of the VM is activated as illustrated in Figure 5.4. The reason is late coming "low pay + computation intensive" cloudlets

are filtered out by strict filtering mechanism of the VM. This tradeoff will also be discussed later in subsequent section regarding the queuing and thresholding dynamics.

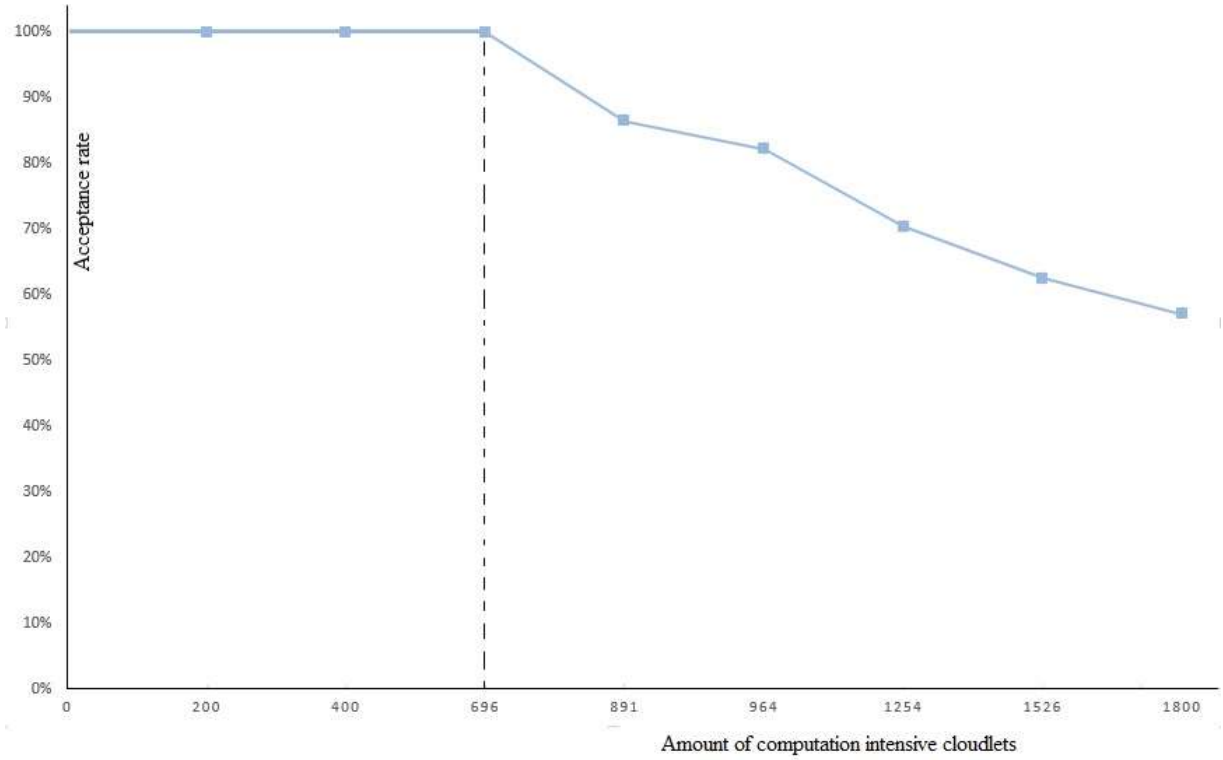

Figure 5.4: Overall Acceptance rate of VM1 to computation-intensive cloudlets

Similar to using execution time for the performance evaluation for lower thresholds in Figures 5.2 (a) and (b), the performance of higher thresholds is also assessed with average execution time. Figures 5.5(a) and (b) show the average execution time of computation-intensive and high pay cloudlets by using TAA algorithm and non-filter policy which is a policy that does not consider attributes of cloudlets, and Table 5.7 lists detailed information for the figures.

Figures 5.5 (a) and (b) and Table 5.7 also consider two cases: a general case in which the arrivals of 29 “high pay + computation” cloudlets are often intermixed with the arrivals of the 4 “high pay + storage” cloudlets, and a very special case in which all the 29 “high pay + computation” cloudlets come to the broker before all the 4 “high pay + storage” cloudlets and consequently the 29 cloudlets are equally allocated to all VMs.

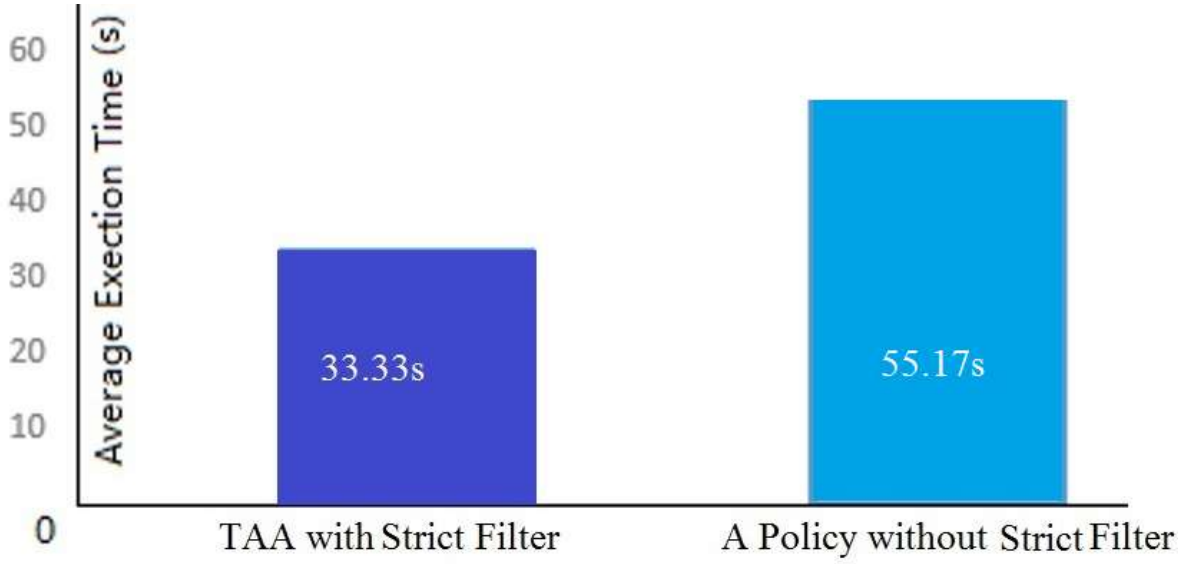

Figure 5.5(a): Execution time comparison between TAA and non-filter policy in a general case (computation-intensive and storage-requesting cloudlets arrive in random sequence)

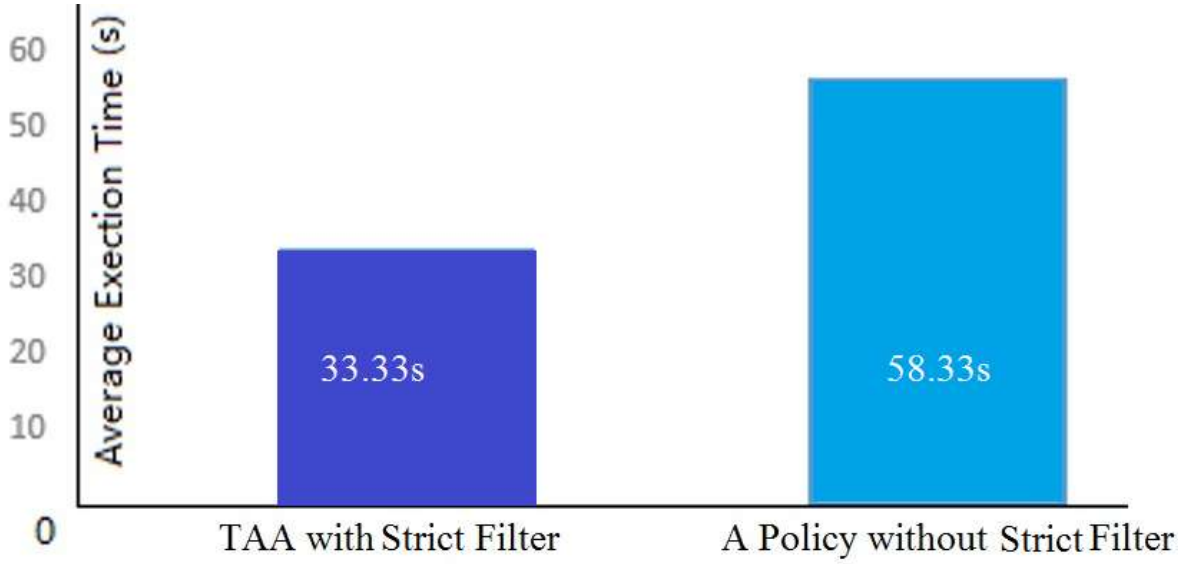

Figure 5.5(b): Execution time comparison between TAA and non-filter policy in a special case (all computation-intensive come before all storage-requesting cloudlets)

Table 5.7: VM allocation for “computation + high pay” cloudlets by using the HiT of TAA and another policy without filter

Policy		VM allocation of total 29 “computation + high pay” cloudlets
Non-filtering policy	General Case (all computation-intensive and storage-requesting cloudlets come in random sequence)	3→VM1, 4→VM2, 22→VM3, 0→VM4
	Special Case (all the 550 computation-intensive come before all the 374 storage-requesting cloudlets)	8→VM1, 7→VM2, 7→VM3, 7→VM4 (equally allocated to VM1, VM2, VM3, and VM4)
TAA policy		29→VM1

The results of Figures 5.5(a) and (b) illustrate that the average execution time for each cloudlet by using the higher threshold of TAA is 50% (general case, 33.33s vs. 55.17s) or 57% (special case 33.33s vs. 58.33s) of non-filter policy. This experimental finding is similar to the conclusion drawn from the previous section regarding the utilization of TAA’s lower thresholds, although the quantitative performance improvement is different. Similarly, the average execution time resulted from using TAA has no dependence on the arrival sequence. It is noted that in the high threshold experiments, more computation-intensive than requesting-storage cloudlets are examined for execution time, this is because this thesis assumes that computation-intensive customers are more demanding on time compared with the customers that concern more on large storage capacity. These similar findings and assumptions have been explained in the previous section for TAA’s lower thresholds, which are not repeated here.

5.2.3 Queuing and Thresholding Dynamics in TAA

By analyzing the experimental results in the above two sections, it is concluded that TAA algorithm may outperform the policy that does not consider attributes of cloudlets, for example, TAA algorithm can reduce execution time and increase the suitability of cloudlets and virtual machines. However, the performance improvement extent depends on the dynamic changes in a real cloud environment and the previous sections have not elaborate on the dynamics, which will be examined in this section based on the total 3000 cloudlets listed in Table 5.1 in the previous section.

An exhibition of the queuing and thresholding dynamics is that while cloudlets are being processed by a VM and hence disappearing in the queue of the VM, other cloudlets are joining the VM's queue. As a consequence, the following situation may occur in a real cloud environment: after the upper threshold (HiT) is activated due to the fullness of the VM's queue under lower threshold (LoT), some late coming cloudlets that match LoT filtering standard but does not match HiT standards may be directed to another less suitable VM, because the functioning of HiT will filter out these cloudlets.

For example, because the HiT of VM1 only accepts "high pay + computation intensive" cloudlets, so the late coming "low pay + computation intensive" cloudlets may be rejected by VM1 for the queuing, although VM1 is most suitable for these cloudlets and the queue below the LoT of VM1 is now processing "high pay + computation intensive" cloudlets instead of "low pay + computation intensive". This is partly caused by the continuous change and fullness of VM1's LoT: at the time some "low pay + computation intensive" cloudlets that are closer to and entering VM1 are processed and hence disappearing in the LoT queue, some "high pay + computation intensive" enter the LoT queue because the HiT has been activated before the abovementioned "low pay + computation intensive" cloudlets disappear in the LoT. Consequently, the "high pay + computation intensive" entering the LoT will still make the LoT full and thus HiT is still functioning, this may cause late coming "low pay + computation intensive" cloudlets to be rejected by VM1. As a result, the acceptance rate of "low pay + computation intensive" cloudlets by VM1 may be reduced to a number below 100%.

Figure 5.6 illustrates the overall acceptance rate of the "computation intensive" cloudlets (including both "high pay" and "low pay") by VM1, and Table 5.8 gives a more detailed description of the experimental conditions and results for the figure. The overall acceptance rate

is based on the totally experimented 3000 cloudlets listed in Table 5.1 of the previous section. As listed in Table 5.1, there are 1800 computation-intensive cloudlets consisting of 90% of "low pay" and 10% "high pay", and 1200 storage-requesting cloudlets consisting of 90% of "low pay" and 10% "high pay" as well.

Considering the cloudlet arrival interval may be varying in a real cloud environment, Figure 5.6 and Table 5.8 also illustrates the experimental results for arrival intervals of 15s and 20s that are 1.5 and 2 times of the 10s for the experiments of the previous sections. As Figure 5.6 and Table 5.8 only consider computation-intensive cloudlets, so the arrival interval here refers to the ratio of the total time to the 1800 computation-intensive cloudlets that have arrived at the data center within the total time. It is noted that within the total time, all 1200 storage-requesting cloudlets have also arrived at the datacenter in a sequence randomly intermixing with the sequence of computation-intensive cloudlets.

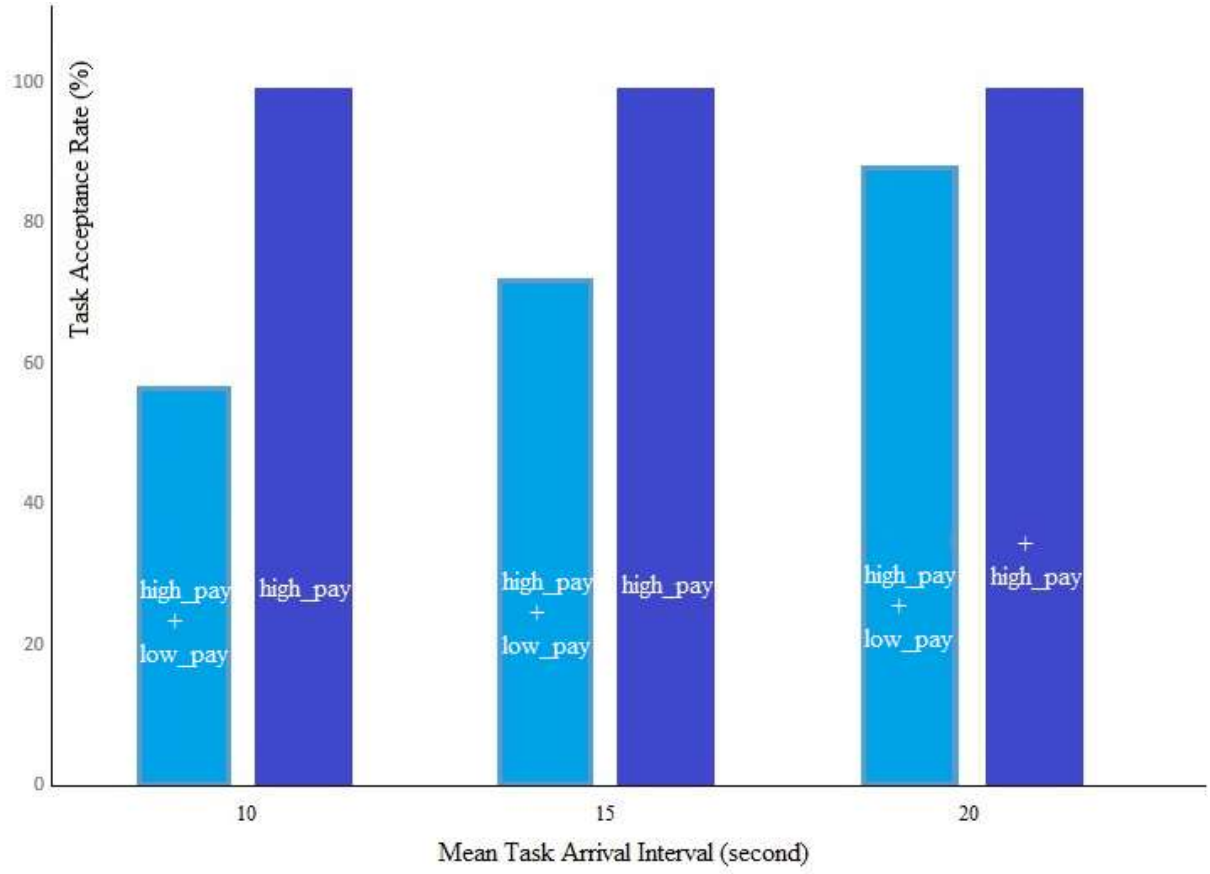

Figure 5.6: Overall acceptance rate of computation-intensive cloudlets by VM1

Table 5.8: Acceptance rate of computation-intensive cloudlets by VM1 having the fastest CPU that is best suitable for computation (33.33s is needed to process a cloudlet)

Arrival Interval (total time / total computation-intensive cloudlets)	Overall acceptance rate of computation-intensive cloudlets by VM1	Distribution of the total 1800 computation intensive cloudlets
10s (17,591s / computation intensive 1,800 cloudlets)	57.11%	1028→VM1 (low + high pay) 0→VM2 (low + high pay) 772→VM3, VM4 (low pay)
15s (26,900s/1,800 computation intensive cloudlets)	72.61%	1307→VM1 (low + high pay) 0→VM2 (low + high pay) 493→VM3, VM4 (low pay)
20s (36,756s/ 1,800 computation intensive cloudlets)	89.06%	1603→VM1 (low + high pay) 0→VM2 (low + high pay) 197→VM3, VM4 (low pay)

It can be observed from Figure 5.6 and Table 5.8 that the acceptance rates for all 1,800 computation-intensive cloudlets are less than 100% when the cloudlet arrival intervals are 10s, 15s, and 20s, respectively, although the acceptance rate illustrated in Figure 5.1 for the early arriving 550 computation-intensive cloudlets is 100%. As discussed in the beginning paragraphs of this section, the acceptance rates for intervals 10s are caused by the screening of the lower threshold of VM1: the cloudlets coming after the 696th computation-intensive (the 550 cloudlets are earlier coming ones) that match LoT filtering standard but do not match HiT filtering standards are directed to other less suitable VMs (VM3 and VM4), because at the instant the 696th computation-

intensive enter the queue of VM1, its lower threshold is just full and its upper threshold starts to function, as indicated by the events of 6546s in Table 5.5.

When the cloudlet arrival interval increases, the fullness of lower threshold of VM1 will be retarded, i.e., the activation of higher threshold will be postponed, so the probability of the late coming "low pay + computation intensive" cloudlets being filtered out by the higher threshold will decrease. As a consequence, the acceptance rate of "low pay + computation intensive" cloudlets will increase. If the cloudlet arrival interval is more slowed down, the higher threshold may not have a chance to be activated. However, if the amount of total cloudlets exceed 3000, the HiT may have a chance to be activated and thus the acceptance rate of "low pay + computation intensive" cloudlets by VM1's LoT may be decreased. Consequently, the overall acceptance rate of computation intensive cloudlets (including low pay and high pay) will be lower than 100%, even if the acceptance rate of "high pay + computation intensive" cloudlets by VM1's HiT is 100%.

Similar conclusions can be drawn from the experimental results for the storage-requesting cloudlets, as illustrated in Figure 5.7 and Table 5.9.

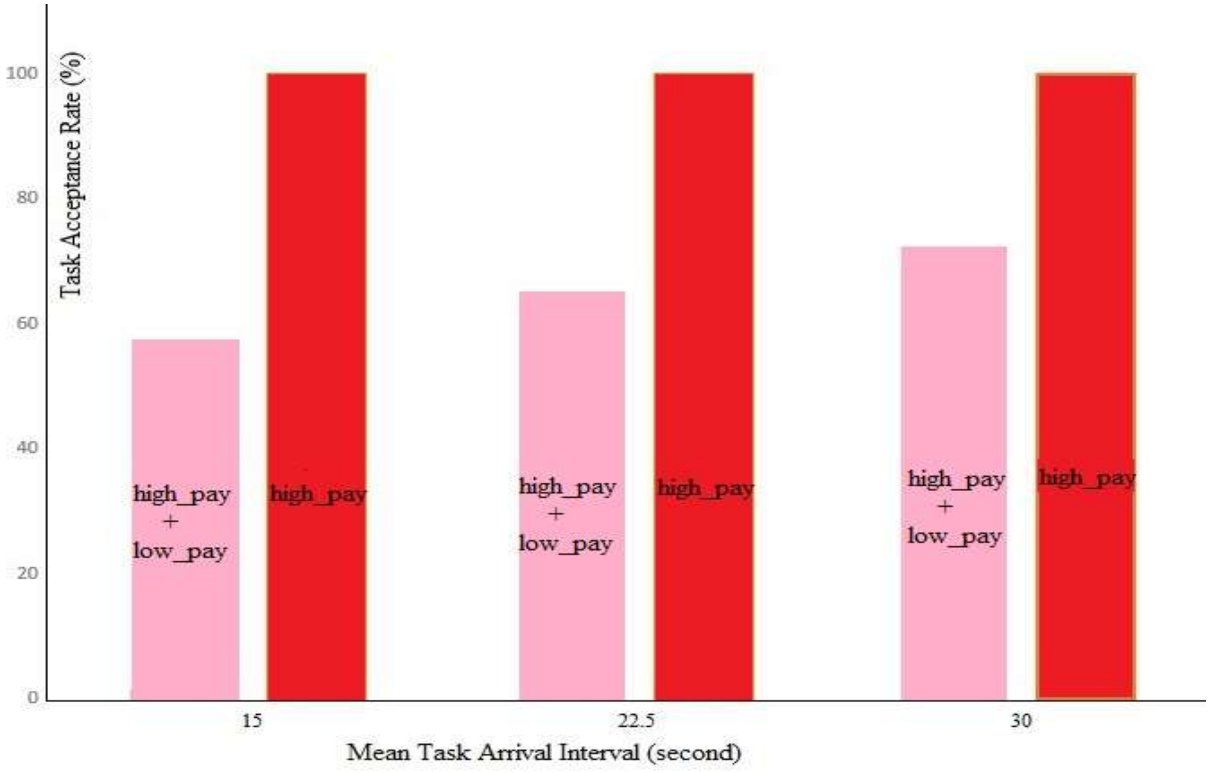

Figure 5.7: Overall acceptance rate of storage-requesting cloudlets by VM2

Table 5.9: Acceptance rate of storage-requesting cloudlets by VM2 having the biggest space that is best suitable for storage

Arrival Interval (total time / total storage-requesting cloudlets)	Overall acceptance rate of storage-requesting cloudlets by VM2	Distribution of the total 1200 storage-requesting cloudlets
15s (18,343s / 1,200 storage-requesting cloudlets, corresponding to 18,000s / 1,8 00 computation-intensive cloudlets)	57.00%	0→VM1 (low + high pay) 684→VM2 (low + high pay) 516→ VM3,VM4 (low pay)
22.5s (27,105s/ 1,200 storage-requesting cloudlets, corresponding to 27,000s / 1,8 00 computation-intensive cloudlets)	64.25%	0→VM1 (low + high pay) 771→VM2 (low + high pay) 429→ VM3,VM4 (low pay)
30s (35,736s/ 1,200 storage-requesting cloudlets, corresponding to 36,000s / 1,8 00 computation-intensive cloudlets)	71.50%	0→VM1 (low + high pay) 858→VM2 (low + high pay) 342→VM3,VM4 (low pay)

Table 5.8 - 5.9 and illustrates how acceptance rate are affected by arrival interval and virtual machine processing ability.

It is noted that even if the acceptance rate is lower than 100%, it does not mean the TAA's performance is lower than a non-filter policy, because the higher thresholds are always working to fit cloudlets to appropriate VMs, this is still much better than the VMs are often occupied by unsuitable cloudlets.

It can be found that the acceptance rate is influenced by how frequent cloudlets go to the broker and also how fast the cloudlets are processed by VMs. For example, if more cloudlets

arriving at the data center than the amount of cloudlets processed by VMs in the same time window, then the queues of the lower thresholds of VMs will be earlier full or the higher thresholds will be earlier activated, and vice versa. In other words, the instant at which LoT is just full or HiT is just activated depends on the relative speed of cloudlet arrival and being processed. In order to quantitatively examine the dependence, Figure 5.8 illustrates the relationship between the relative speed and the instant. The relative speed indicated by the X-axis is in fact a dimensionless number that may indicate an extent of generalization, which is defined as follows:

$$R_S = \frac{S_A}{S_P} = \frac{1/AI}{S_P} \tag{5.4}$$

where R_S is the relative speed, S_A is the average arrival speed of cloudlets representing the average amount of cloudlets arriving at the data center per unit time, so its units are s^{-1} suggesting arrival frequency. In fact, it is the reciprocal of arrival interval (AI) that was often utilized for discussion in the previous sections. In equation (5.4), S_P is the processing speed of cloudlets by a VM, which represents the average amount of processed cloudlets by the VM per unit time, so its units are also s^{-1} suggesting the cloudlet processing frequency and CPU's speed of the VM.

The curve shown in Figure 5.8 represents the boundary or transition line between the HiT and LoT, i.e., LoT is functioning below the line and HiT is above it. This means, future engineers can use this line to select or determine the activation instant and hence the maximum queuing capacity of LoT according to their specific parameters of their cloud environment such as CPU and cloudlet coming speeds. A lot of information can be obtained from Figure 5.8. For example, it can be observed that the activation of HiT will be advanced with increasing the ratio of arrival speed to processing speed. This verifies the above reasoning: when there are increasingly more cloudlets arriving at the data center than the amount of cloudlets processed by the VM within the same time, i.e., if the average arrival speed of cloudlets is increasingly larger than the processing speed by the VM, then the queue of the lower threshold of VM1 is earlier full and hence the higher threshold is earlier activated. The X-axis of Figure 5.8 does not show the values below 1.0, because if the arrival speed of cloudlets is lower than the speed of being processed by VM1, then it HiT will not have a chance to be activated, so the curve does not exist. This means, if the data center provider always provide more than the customers' requests, then the task scheduling and resource

allocation will be much easier. However, this is rarely the case in real world, so this thesis does not elaborate on the situation wherein the speed ratio of arrival to processing is smaller than 1.

Similar to the experiments for the dependence of transitional instant on the speed ratio, Figure 5.9 illustrates the experimental results for acceptance rate by VM1, from which similar conclusions can be made as from Figure 5.8 for transitional instant: when the average arrival speed of cloudlets is increasingly larger than the processing speed by the VM, the queue of the lower threshold of VM1 is earlier full and hence the higher threshold is earlier activated. As a consequence, the acceptance rate decreases because more "low pay" computation-intensive cloudlets are filtered out by the earlier activation of VM1's HiT.

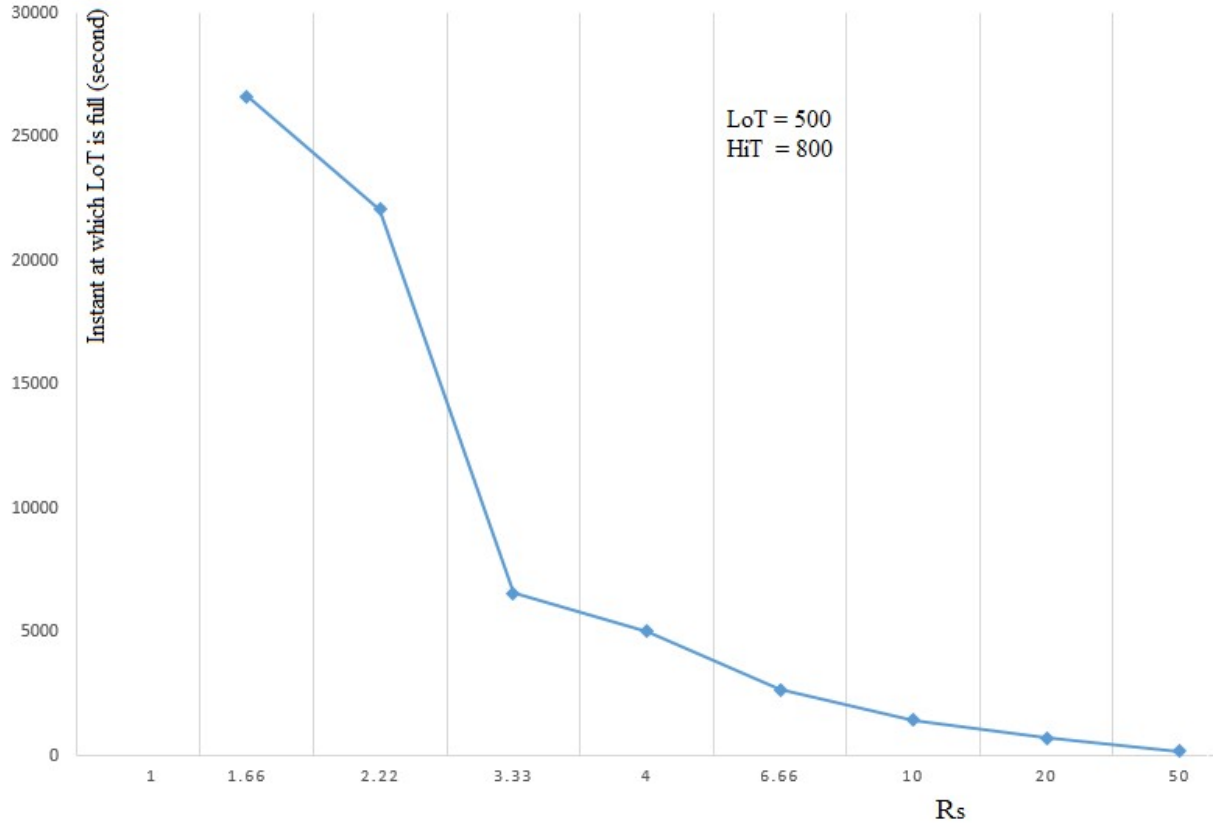

Figure 5.8: Dependence of activation of higher threshold of VM1 on the relative speed of cloudlet arrival and being processed.

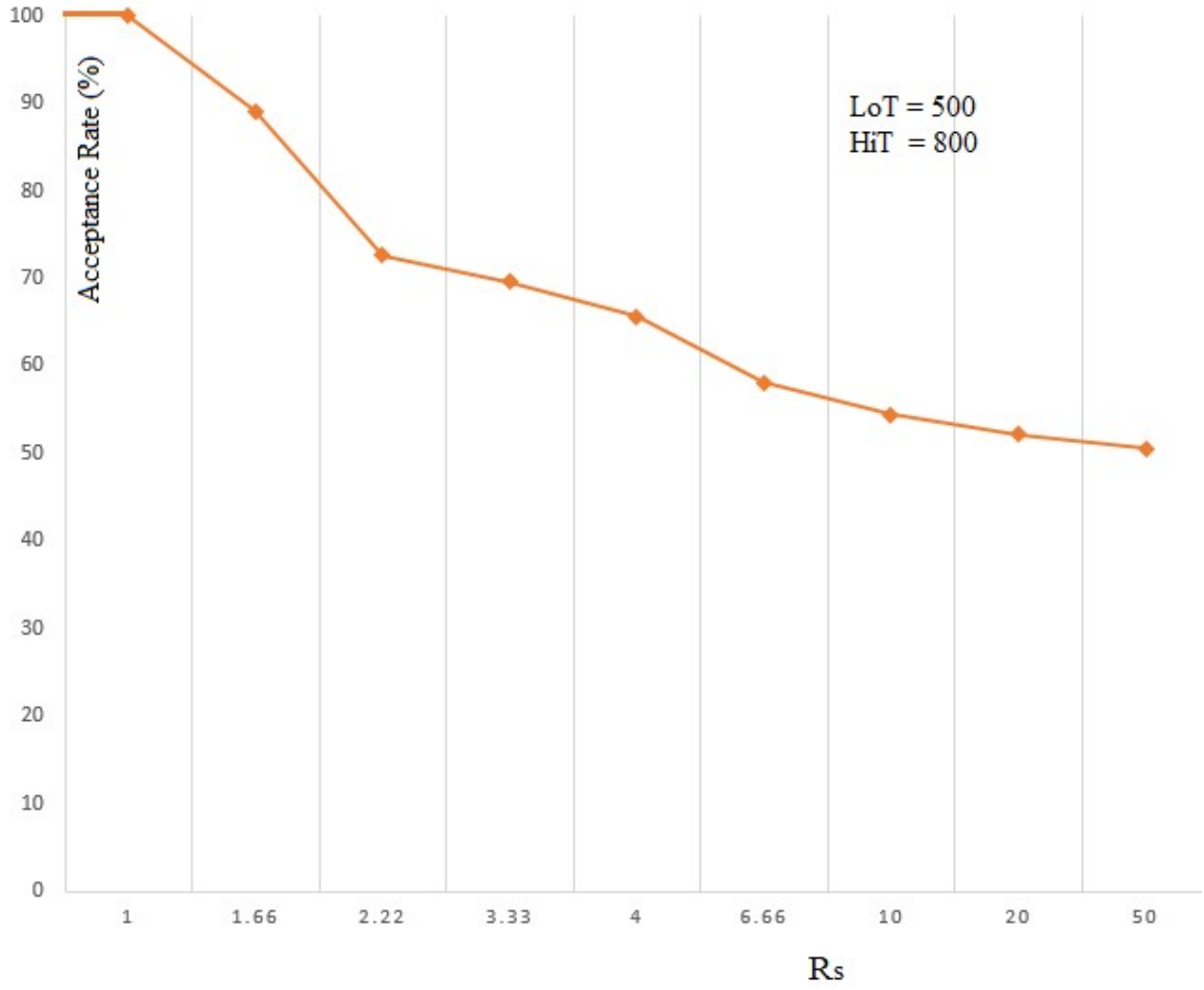

Figure 5.9: Dependence of acceptance rate on the relative speed of cloudlet arrival and being processed.

Both Figures 5.8 and 5.9 suggest that if the maximum queuing capacity of LoT is increased, i.e., the maximum amount of cloudlets accommodated by LoT is increased, the transitional instant from LoT's functioning to HiT's may be retarded. Figure 5.10 verifies this reasoning by illustrating the comparative view of different queuing capacities for the same VM: when the maximum queuing amount of cloudlets allowed by the LoT of VM1 increases from 500 to 700, the instant at which the LoT is just full and the HiT is just activated is postponed. So the transitional curve for LoT =700 is above the curve for LoT=500. This may improve the acceptance rate of low-pay

cloudlets, but the cost is the delay of processing high-pay cloudlets. The trade-off is complex, which will be examined in future works.

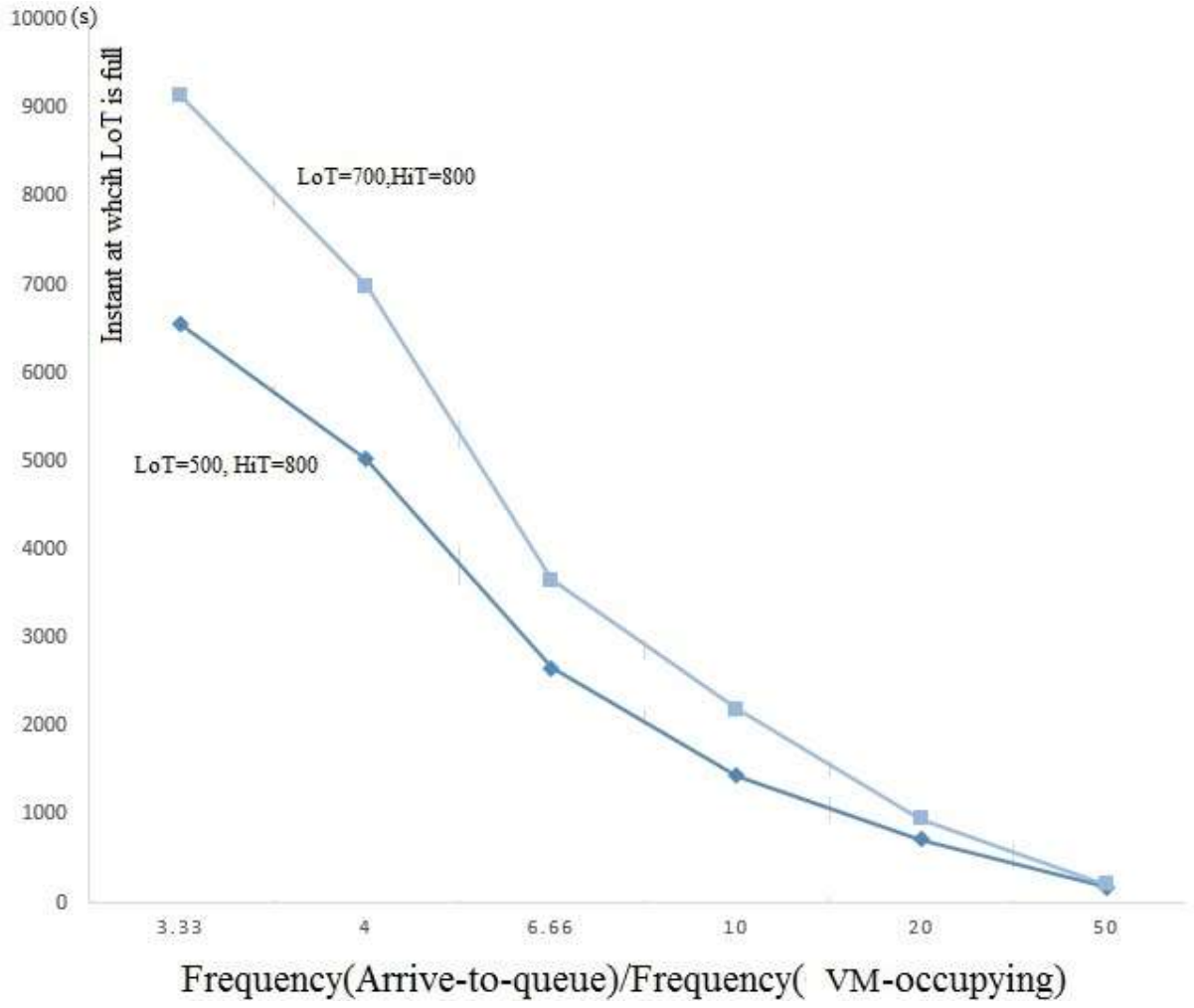

Figure 5.10: Comparative view of the dependence of HiT's activation on different queuing capacities for the same VM

Chapter 6. Conclusion and Future Work

Through the literature survey on a number of methods of improving the quality of service (QoS) for the booming cloud computing industry, this thesis concludes that the performance of almost all methods is substantially influenced by the performance of task scheduling and resource allocation. Since a task in cloud computing is often termed as a cloudlet or divided into cloudlets, and a resource is often represented by a virtual machine (VM), so this thesis focuses on the development of a strategy to optimize the matching between cloudlets and virtual machines.

It is found that most past task scheduling and resource allocation strategies regard tasks and resources as homogenous and thus the strategies mostly focus on processing the amount of cloudlets. In real cloud, however, different tasks and machines may have different features or attributes. Unfortunately, attribute-based task scheduling and resource allocation strategies have been rarely reported in the field of cloud computing.

To shrink the research blank, this thesis develops a strategy of TAA (Thresholds with Attributes and Amount) that is able to channel different cloudlets to different suitable VMs based on the cloudlets' attributes as well as the dynamic amount of queuing cloudlets in front of the VMs. In the TAA strategy, attribute-oriented thresholds or filters are set for the acceptance or rejection of cloudlets to the queue of a VM. To examine the performance of TAA, this thesis proposes a simplified TAA strategy consisting of a lower threshold (LoT) and a higher threshold (HiT), and the HiT imposes a stricter screening of attributes than LoT on cloudlets applying for the occupation of a powerful VM. When the amount of queuing cloudlets exceeds the maximum capacity of LoT, a stricter filtering of attributes will be activated for the HiT of the VM so as to ensure the cloudlets with very special attributes (e.g., high priority or urgency) can be anywise arranged a suitable resource or stick to session.

To estimate the performance of the TAA strategy in a real cloud environment, an open source cloud simulation tool kit (CloudSim) is utilized to provide a simulated cloud environment and implement the proposed TAA, through incorporating a number of codes developed in this thesis into the Cloudsim. Also, a number of parameters such as acceptance rate, execution time, arrival interval, arrival speed, and processing speed are adopted to quantify the performance of task scheduling and resource allocation.

It is concluded from the simulation results that using the TAA can significantly improve the performance of task scheduling and resource allocation, particularly in terms of the optimal matching between cloudlets and VMs. In summary, the improvement can be reflected by the following experimental findings:

(1) If the LoT of a VM is not full, then the cloudlet acceptance rate by the VM can be as high as 100%, although the VM only imposes a rough filtering of cloudlet attributes (e.g., parental attributes like "computation intensive" and "storage intensive"). This may significantly improve the channeling of cloudlets with different attributes to the VMs that best fit the attributes.

(2) The average execution time of a cloudlet channeled to its appropriate VM is also significantly reduced. Under the experimental conditions of this thesis, the average execution time using TAA is only 50-70% of a typical non-filter policy.

(3) When a VM's LoT is full and consequently the VM's HiT starts to function and a stricter screening of attributes is imposed on late coming attributes, then the acceptance rate of the cloudlets with very special attributes (e.g., descendant attributes like "high pay + computation intensive") is also as high as 100%. This may guarantee a suitable resource can be always allocated to very special cloudlets, or help a large task consisting of numerous cloudlets to stick to the same resource.

(4) Similar to the execution time reduction resulted from the functioning of LoT, the execution time resulted from the functioning of HiT is only 50-60% of a typical non-filter policy under the experimental conditions of this thesis.

(5) Trade-off is resulted from the activation of HiT: some late coming cloudlets that are suitable for a VM may be directed to another VM, because these cloudlets do not match the stricter screening standards of the HiT (e.g., high pay + computation intensive) and

thus are not allowed to enter the queue of the suitable VM, although they match the rough filtering standards (e.g., computation intensive but with no high pay). As a consequence, the overall acceptance rate may not reach 100%.

(6) The trade-off does not mean the performance of TAA in terms of suitability is lower than a non-filter policy, because the higher thresholds are always working to fit cloudlets to appropriate VMs, this is still much better than the VMs are often occupied by unsuitable cloudlets. As a consequence, the overall acceptance rate is still much higher than a typical policy that does not consider cloudlet attributes.

(7) The acceptance rate of cloudlets is affected by the transitional instant at which a VM's LoT is just full and HiT is just activated, and the transitional instant depends on the speed of cloudlets arriving at the data center as well as the speed of being processed by the VM. The acceptance rate is also influenced by the maximum queuing capacity of LoT. With decreasing the arrival speed or increasing the maximum queuing length of LoT, the transitional instant will be retarded and the overall acceptance rate will be increased. It is found that these influences are very dynamic.

(8) Relative speed defined as the ratio of arrival speed to processing speed is adopted in this thesis to partly quantify the thresholding and queuing dynamics and deliver a generalization. It is concluded from the simulation experiments that with decreasing the relative speed, the transitional instant will be retarded and the overall acceptance rate will be increased. However, the acceptance of highly prioritized and late coming cloudlets (e.g., high pay + computation intensive) will be postponed by lowly prioritized but early coming cloudlets (e.g., low pay + computation intensive), because the lowly prioritized are still allowed to enter the queue due to the retarded activation of HiT which will filter out lowly prioritized cloudlets once otherwise being activated.

(9) Historical resource status and attribute matching information can be potentially used by broker or load balancer for early arrangement of resource allocation and design improvement of future cloud tools. This technique may also potentially avoid migration of cloudlets belonging to the same big task from one node to another node, hence help achieve stick-to-session goal and in the meanwhile save the network transmission time for load balancing purpose in cloud computing system.

(10) The programing of TAA is not complex because a large portion of the programming is for searching and sorting among tasks and virtual machines, which can borrow many mature and open codes from many sources.

In conclusion, the TAA can significantly improve the performance of task scheduling and resource allocation for data centers, brokers and customers in cloud environment. Nonetheless, numerous future investigations are needed to improve the TAA, this thesis suggests the following near-future works the author believes achievable:

(1) Attributes should be defined comprehensively and representatively for both tasks and resources, as matching depends on the similarity of attributes. For example, the attributes could be categorized to a structure of a family tree, from parental to descendant attributes. Even an attribute-based protocol could be standardized for the communication among the ends of brokers, data centers, and customers.
(2) The maximum queuing capacity of LoT or HiT can be designed to be dynamically adjustable so as to adapt to varying requests and cloudlets. This may help the optimization of acceptance rate and execution time of different cloudlets.
(3) Multiple thresholds for TAA can be arranged to adapt to varying cloud environment, instead of using only a two-threshold TAA. This may help a data center or broker to ramp up or down to dynamic requests of customers and status of the resources of the data center.
(4) Mathematical modelling can be developed to formulate the queuing and thresholding dynamics, which can be used as a generalized tool to predict the dynamics of cloudlets and resources, as well as design the task scheduling and resource allocation algorithm.

Bibliography

[1] Satyanarayanan, Mahadev, et al. "The case for vm-based cloudlets in mobile computing." Pervasive Computing, IEEE 8.4 (2009): 14-23.

[2] Rodrigo N. Calheiros, Rajiv Ranjan, César A. F. De Rose, RajkumarBuyya, “CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. IEEE, 2009.

[3] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, C´esar A. F. De Roseand Raj Kumar Buyya,"CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resourceprovisioning algorithms”, SOFTWARE –PRACTICE AND EXPERIENCESoftw. Pract.Exper.2011; 41: 23–50, 2010. In Wiley Online Library (wileyonlinelibrary.com).

[4] Julie Anderson, AG Strategy Group, http://safegov.org/2012/2/16/the-president’s-budget-making-cloud-computing-a-priority-for-the-future as on Sep. 2012. AG Strategy Group

[5] Peter Mell, Timothy Grance, “The NIST Definition of Cloud Computing”, Special Publication 800-145 NIST Special Publication 800-145, 2011

[6] Foster, I; Yong Zhao; Raicu, I.; Lu, S. “Cloud Computing and Grid Computing 360-Degree Compared”, published in Grid Computing Environments Workshop, 2008. GCE '08 IEEE DOI 12-16 Nov. 2008.

[7] Kalagiakos P & Karampelas P., “Cloud Computing Learning.” 2011 5th International Conference on Application of Information and Communication Technologies [AICT], pp. 1-4.

[8] Rajkumar Buyya, et al., “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility”, Future Generation Computer Systems 25 (2009), pp. 599-616

[9] Sun Microsystems, Inc.”Introduction to Cloud Computing Architecture” Whitepaper, Ist Edition, June 2009.

[10] Qi Zhang, Lu Cheng, Raouf Boutaba, “cloud computing: state of-the-art and research challenges”, 20th April 2010, Spinger, pp. 7-18.

[11] Bhasker Prasad Rimal, Eummi Choi, Lan Lump (2009) “A Taxonomy and Survey of Cloud Computing System”, 5th International Joint Conference on INC, IMS and IDC, IEEE Explore 25-27Aug 2009, pp. 44-51

[12] Crago, Steve, et al., "Heterogeneous cloud computing." Cluster Computing (CLUSTER), 2011 IEEE International Conference on. IEEE, 2011.

[13] Harmandeep Singh Brar, Vivek Thapar, “An Efficient Priority Based Load Balancing Algorithm for Cloud Environment”, International Journal of Computer Science Trends and Technology (IJCST) –Volume 2 Issue 4,July-Aug 2014, www.ijcstjournal.org, pp. 99 -103

[14] Foley, R. D.; McDonald, D. R. Join the shortest queue: stability and exact asymptotics. Ann. Appl. Probab. 11 (2001), no. 3, 569--607. doi:10.1214/aoap/1015345342. http://projecteuclid.org/euclid.aoap/1015345342.

[15] Walrand, J., with Libin Jiang and Shah, D., Jinwoo Shin. “Distributed Random Access Algorithm: Scheduling and Congestion Control.” Information Theory, IEEE Transactions on 56.12 (2010): 6182-6207. Copyright © 2010, IEEE

[16] Hemamalini, M. "Review on grid task scheduling in distributed heterogeneous environment." International Journal of Computer Applications 40.2 (2012): 24-30.

[17] Subasish Mohapatra, K.Smruti Rekha, Subhadarshini Mohanty, “A Comparison of Four Popular Heuristics for Load Balancing of Virtual Machines in Cloud Computing”, International Journal of Computer Applications (0975 – 8887) Volume 68– No.6, April 2013, pp. 33 - 38

[18] Gao Ming and Hao Li, "An Improved Algorithm Based on Max-Min for Cloud Task Scheduling", Yunnam University, China, 2012

[19] Upendra Bhoi1, Purvi N. Ramanuj2, “Enhanced Max-min Task Scheduling Algorithm in Cloud Computing”, ijaiem.org , Volume 2, Issue 4, April 2013 ISSN 2319 – 4847

[20] Medhat A. Tawfeek, Ashraf El-Sisi, Arabi E. keshk, Fawzy A. Torkey, "Cloud Task Scheduling Based on Ant Colony Optimization", 2013 IEEE, pp. 64-69

[21] R.K. Yadav, Veena Kushwaha, "An Energy Preserving and Fault Tolerant Task Scheduler in Cloud Computing", IEEE International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), August 01-02, 2014

[22] Monica Gahlawat, Priyanka Sharma, "Analysis and Performance Assessment of CPU Scheduling Algorithms in Cloud using Cloud Sim", International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 5 – No. 9, July 2013 – www.ijais.org, pp. 5-8

[23] Xiaonian Wu, Mengqing Deng, Runlian Zhang, Bing Zeng, Shengyuan Zhou, "A Task Scheduling Algorithm based on QoS-Driven in Cloud Computing", January 2013, Volume17(IssueComplete) pp. 1162-1169

[24] Jun-Kwon Jung, Nam-Uk Kim, Sung-Min Jung and Tai-Myoung Chung, "Improved CloudSim for Simulating QoS-Based Cloud Services", Y.-H. Han et al.. (eds.), Ubiquitous Information Technologies and Applications, Lecture Notes in Electrical Engineering 214, DOI: 10.1007/978-94-007-5857-5_58, _ Springer Science+Business Media Dordrecht 2013, pp. 537 - 545

[25] Garima Gupta, Vimal Kr.Kumawat, P R Laxmi, Dharmendra Singh, Vinesh Jain, Ravinder Singh, "A Simulation of Priority Based Earliest Deadline First Scheduling for Cloud Computing System", 2014 IEEE, 2014 First International Conference on Networks & Soft Computing, pp. 35-39

[26] R. Vijayalakshmi, Mrs. Soma Prathibha, "A novel approach for task scheduling in cloud", IEEE – 31661, 4th ICCCNT 2013 July 4-6, 2013, Tiruchengode, India, pp.

[27] Ranaweera, A., Agrawal, D.P.: A Task Duplication based Algorithm for Heterogeneous Systems. In: Proc. of IPDPS, May 1-5, 2000, pp. 445–450 (2000)

[28] M. Macias, O. Fito, and J. Guitart, “Rule-based SLA Management for Revenue Maximisation in Cloud Computing Markets,”in Proceedings of the 2010 International Conference on Network and Service Management, pp. 354–357, 2010.

[29] Rongxian Chen, Yaying Zhang, Dongdong Zhang, “A Cloud Task Scheduling Algorithm Based on Users’ Satisfaction”, 2013 Fourth International Conference on Networking and Distributed Computing, pp. 1 – 5

[30] Rachel Householder, Scott Arnold, and Robert Green, “Simulating the Effects of Cloud-based Oversubscription On Datacenter Revenues and Performance in Single and Multi-class Service Levels” in 2014 IEEE International Conference on Cloud Computing, pp. 562-569, 2014

[31] ZHAO Yong*, CHEN Liang, “Efficient Task Scheduling for Many Task Computing with Resource Attribute Selection”, China Communications • December 2014, pp. 125- 140

[32] B. Harzog,"Infrastructure Performance Management for Virtualized Systems",

White Paper, APM Experts, 2010 pp. 1 - 18.

[33] Bannerman, P.L. Cloud computing adoption risks: state of play. In proceedings of the 17th Asia Pacific Software Engineering Conference (APSEC2010) Cloud Workshop. Sydney, Australia, 30Nov-03Dec, 2010.

[34] Nallakumar. R1, Dr. N. Sengottaiyan 2, Sruthi Priya K.S 3, “A Survey on Scheduling and the Attributes of Task Scheduling in the Cloud”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 10, October 2014, pp. 8167-8171

[35] S. Floyd and Van Jacobson, “Random Early Detection Gateways for Congestion Avoidance”, IEEE Transactions on networking, vol.1, No.4 (Aug 1993), pp.397-413.

[36] Lee, Gunho, Byung-Gon Chun, and Randy H. Katz. "Heterogeneity-aware resource allocation and scheduling in the cloud." Proceedings of HotCloud (2011): 1-5.

[37] M. Vaidehi, T. R. Goplalakrishnan Nair, V. Suma,”An Efficient Job Classification Technique to Enhance Scheduling in Cloud to Accelerate the Performance”, CT and Critical Infrastructure:

Proceedings of the 48th Annual Convention of Computer Society of India- Vol I Volume 248 of the series Advances in Intelligent Systems and Computing pp 593-603

[38] Remzi Arpaci-Dusseau and Andrea Arpaci-Dusseau., “Operating Systems: Three Easy Pieces”, 2014.