electronics

MDPI

Article

Dual Threshold Adaptive Dynamic Migration Strategy of Virtual Resources Based on BBU Pool

Cheng Wang [1,*], Yushi Cao [1], Zhili Zhang [2] and Weidong Wang [1]

[1] School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; caoyushi@bupt.edu.cn (Y.C.); wangweidong@bupt.edu.cn (W.W.)

[2] The 54th Research Institute of China Electronics Technology Group Corporation, Hebei 500081, China; zhangzl_221@163.com

* Correspondence: wangcheng@bupt.edu.cn; Tel.: +86-1861-832-2537

Received: 15 January 2020; Accepted: 8 February 2020; Published: 11 February 2020

Abstract: The rapid development of mobile communications and the continuous growth of service needs lead to an increase in the number of base stations (BSs). Through virtualization and cloud technology, virtual Baseband Units (BBUs) are deployed on a virtual machine (VM) to build a BBU pool to achieve hardware resource sharing, which not only saves BS construction costs but also facilitates management and control. However, too high or too low server resource utilization in the pool not only affects the performance of the virtual BBU but also increases the maintenance cost of the physical equipment. In this paper, BBUs are virtualized to construct a virtual BBU pool based on the OpenStack cloud architecture and a dual threshold adaptive dynamic migration strategy is proposed in this scenario. Establish upper and lower threshold of resource utilization of the servers in the pool and the strategy determines whether the dynamic migration is triggered according to the resource utilization of each compute node. If the migration is triggered, the strategy selects the virtual resource to be moved out and the target node to realize the dynamic migration to achieve the purpose of balancing the server load and saving energy consumption. The migration strategy proposed in this paper is simulated on Cloudsim and the experimental results show that the strategy can effectively reduce the number of migrations and migration time on the basis of reducing energy consumption and SLA violations. This paper successfully deployed the strategy on the OpenStack platform, which implements dynamic migration autonomously to save the overall energy consumption of the BBU pool, instead of manual operations.

Keywords: BBU Pool; dynamic migration; OpenStack; Cloudsim

1. Introduction

With the development of mobile communication technology and the commercial use of 5G, mobile operators ensure the normal use of user services in their coverage areas and meet the need for a higher data rate, fast and efficient network services [1] by deploying a large number of base stations (BSs). The increasing number of BSs will lead to increasing problems. Firstly, in the traditional mobile communication system, each BS is relatively independent and cannot share resources [2]. Each BS only undertakes the processing of access user services within its coverage area, so the deployment and maintenance of a large number of BSs require significant construction costs. Secondly, the auxiliary equipment such as air conditioners equipped during the operation of BSs will also cause energy consumption problems. To solve the challenges mentioned above, the concept and architecture of centralized access network are proposed. Cloud-radio access networks (C-RAN) is a promising 5G mobile network architecture [2], which implements the functional separation of traditional BS into two parts—the baseband unit (BBU) and the remote radio head (RRH). It allows BBUs to be geographically

separated from RRHs [3,4]. BBU performs baseband signal digital processing together with all upper layer functions. RRH acts as the wireless signal transmission and reception with antennas. Each RRH does not belong to any specific physical BBU and a virtual BS can process the radio signals from /to a particular RRH. C-RAN can reduce capital expenditure (CAPEX) and operating expense (OPEX) of 5G by separating RRH from BBUs based on network functions virtualization (NFV) concepts [5]. IBM, Huawei and other equipment manufacturers have also put forward a centralized architecture implementation scheme, combined with software defined network (SDN) and NFV, to complete the unified management and allocation of BS computing resources on cloud computing platforms. Applying cloud computing as the computing paradigm of the centralized BBU pool can reduce the power consumption and improve hardware utilization, through resource sharing and virtualization, that is, a server can be further virtualized into many virtual machines (VMs) [6] running multiple isolated virtual BBU instances.

The continuous arrival or end of communication services leads to great differences in the load distribution of different BBUs in the BBU pool. To ensure the normal operation of the user services in the coverage area, the no-load or low-load BBUs must maintain the same running state as the high-load BBUs. On the one hand, the physical servers with these virtualized BBUs may degrade the quality of service because of the too high utilization of resources. On the other hand, the low utilization of resources wastes the physical resources while maintaining the power cost. BBUs are deployed in the server cluster centrally through virtualization to form a BBU pool and realizes the dynamic scheduling of virtual resources according to the utilization of real-time resource, which cannot only share physical resources but also improve the utilization of cluster server processing resources in the BBU pool, improve the overall operation reliability of the pool and reduce the power consumption. Since virtual BBUs are deployed on the VMs, the dynamic migration method of the VMs in the cloud environment can be adopted to improve the resource utilization rate and achieve load balancing.

Load balancing is a migration process of the load from over utilized nodes to underutilized nodes to reduce the wastage of the resources in a cloud environment [1]. It can be done by VM selection and migration that appropriate VMs are selected either from the overloaded or under-loaded host for migration. Yazir et al. implemented a threshold to determine whether migration is required by monitoring the resource utilization of the server and performs a migration operation once it detects that its resource utilization exceeds the high load threshold [7]. The authors of References [8,9] adopt the algorithm called Minimum Migration Time (MMT), which selects a VM based on the value of the migration time, the less, the better. Reference [10] selects the VM selection strategy with maximum utilization for migration and VM placement algorithm using the Minimum Power High Available Capacity strategy to find new places of VM has been developed. Reference [11] calculates total utilization (TU) combined with CPU and memory to choose the smallest TU as destination host. However, if the threshold is exceeded, the migration is triggered immediately and unnecessary migration may be triggered because of an instantaneous peak value, resulting in unnecessary system overhead. Razali et al. in Reference [12] make the historical data as a training set, establish a prediction model and predict the CPU usage for the next moment. In Reference [13], a prediction based on fuzzy logic is used to predict future resource utilization. The authors in Reference [14] use threshold and time sequence prediction technique, which does not perform migration operations immediately when it is detected that server resource utilization exceeds the threshold but continues to observe several cycles to determine whether to initiate migration. This method determines the migration time and avoids invalid migrations, which can save the overall energy consumption of the data center. However, the running communication service is sensitive to the quality of service in the BBU pool. If the virtual BBU does reach the continuous high load state at this time, long observation time of several consecutive cycles may lead to the sharp deterioration of the quality of service of the communication service in the BS.

Based on the architecture of a centralized access network, this paper uses virtualization technology on x86 architecture servers to virtualize the functions of traditional BBUs (such as physical layer

baseband processing and high-level protocol processing) on VMs, to build a centralized BBU pool and use OpenStack cloud platform for unified control and deployment of virtual resources in the pool. In this scenario, this paper proposes a dual threshold adaptive dynamic migration strategy, which sets the upper and lower threshold for the physical nodes' resource utilization in the BBU pool and realize adaptive dynamic migration. The main contributions of this paper are summarized as follows:

(1) We virtualize the BBU and build a virtual BBU pool combined with OpenStack cloud architecture. In the OpenStack-based BBU pool, all the processing resources provided by physical servers can be managed and allocated by a unified real-time virtual operating system.

(2) We propose a dual threshold adaptive (DTA) dynamic migration strategy divided into three parts. The first part is the adaptive migration trigger strategy combined with the Kalman filter algorithm, which can avoid a certain instantaneous peak of server resource utilization to trigger unnecessary migration because of too high or too low and effectively determine the time to trigger migration. The second part is selection strategy of VM, based on the number of migrations and the migration time. The last part is selection of target server node, combined with prediction and resource utilization to make a comprehensive judgment.

(3) The simulation is carried out first by the CloudSim platform to evaluate the proposed migration strategy, which can improve energy consumption and SLA violation and significantly reduce the total number of migrations and migration time. Then the strategy is implemented on the OpenStack and the results are finally analyzed that this strategy can avoid invalid migration, instead of manual, realize adaptive dynamic migration and achieve the load balancing purpose.

The rest of the paper is organized as follows. Section 2 presents a system scenario of BBU pool based on OpenStack and introduces related works of migration and parameter definition. In Section 3, we describe the migration strategy in detail. Section 4 shows the simulation results on Cloudsim and OpenStack. Then conclusions are drawn in Section 5.

2. System Scenario and Related Works

2.1. Virtual BBU Pool based on OpenStack Architecture

This paper adopts the distributed architecture of BBU and RRH, as shown in Figure 1. BBUs are clustered as a BBU pool in a centralized location. RRHs support high capacity in hot spots, while the virtualized BBU pool provides large-scale collaborative processing and cooperative radio resource allocation [15]. The pool has at least one control management to control the performance of all virtual resources.

Through virtualization technology, a virtual layer is built on a general server cluster of the X86 architecture. Multiple VMs are run on the virtual layer and a virtualized BBU pool is constructed through the running VMs deployed on many common servers. Integrate the BBU pool with the OpenStack and remotely connect to the RRH. The resources used by each VM will be planned according to the resource allocation plan of the BBU pool and support different functions of baseband processing units. The OpenStack cloud platform solves the problem of the automatic creation of virtual resources. Figure 2 shows the virtual BBU pool architecture based on OpenStack, which can manage and deploy the pool and implement dynamic migration. With the OpenStack cloud architecture, physical servers can be deployed as compute nodes, these nodes and the virtual BBU pool deployed on these nodes can be managed and controlled by a controller node (also deployed on a physical server).

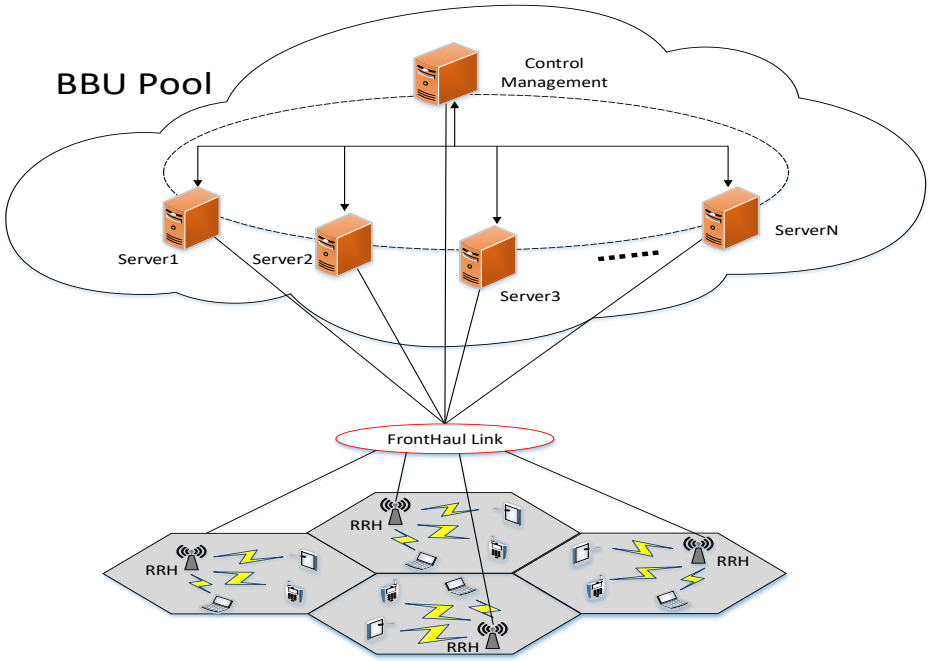

Figure 1. Baseband unit (BBU) and remote radio head (RRH) distributed architecture.

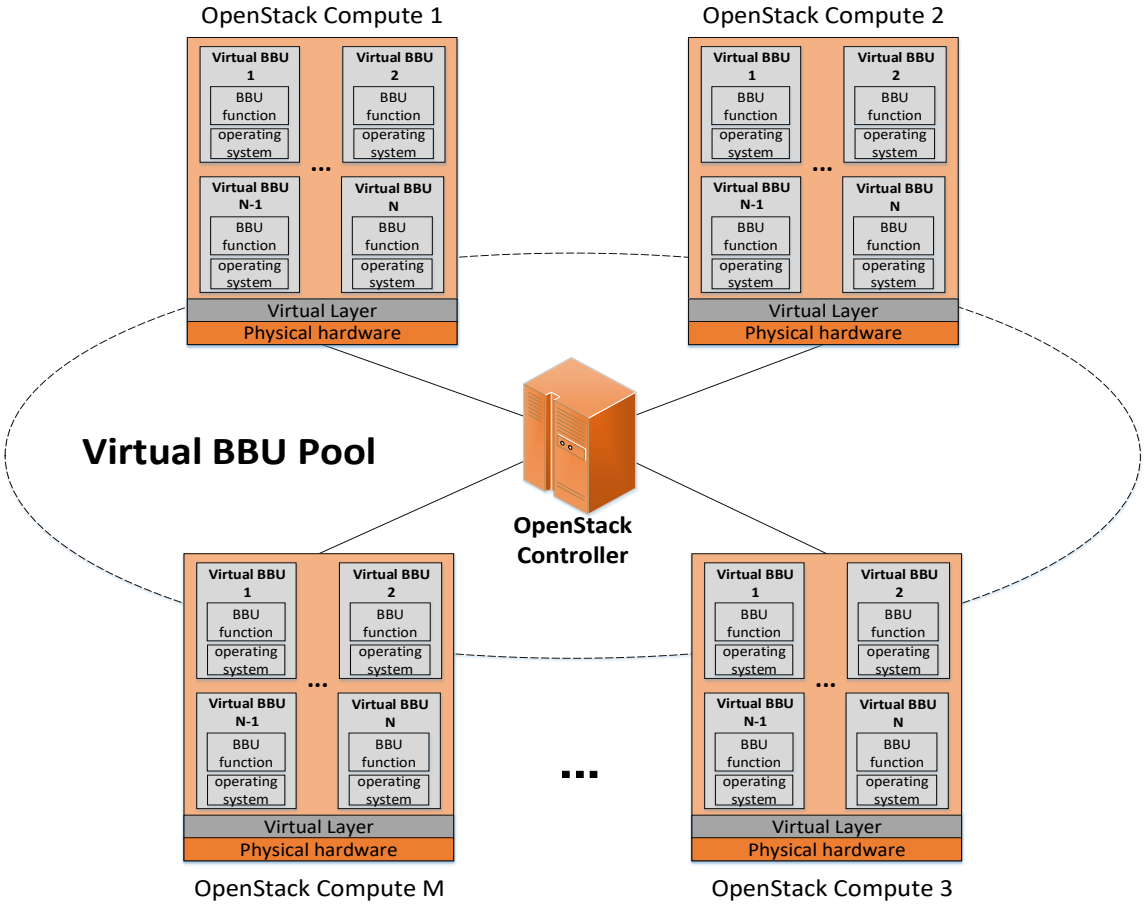

Figure 2. Virtual BBU pool architecture based on OpenStack.

2.2. Related Model Definition

2.2.1. Model of Power

The research in Reference [16] shows that power consumption by physical machines can be described by a linear relationship between power consumption and CPU utilization. The study also points out that a free physical machine uses approximately 70% of its power consumption on average when it is fully utilized. Define the power consumption as a CPU utilization function from Equation (1).

$$P(u) = k \times P_{max} + (1 - k) \cdot P_{max} \times u, \tag{1}$$

where P_{max} is the maximum power of a server in the running state, k is the percentage of power consumed by an idle physical server and u is the CPU utilization. As the utilization of CPU changes over time due to the workload variability and $u(t)$ is a function of the time. The total energy consumption can be defined by Equation (2). According to this model, the energy consumption is determined by the CPU utilization.

$$E = \int_t P(u(t))\, dt. \tag{2}$$

2.2.2. Cost of VM Dynamic Migration

Dynamic migration of VMs allows transferring VMs between physical nodes with short downtime. However, dynamic migration has a negative influence on the performance of applications running in a VM during a migration. Studies in Reference [17] found that a reduction in performance and downtime depends on the behavior of applications. The length of a dynamic migration depends on the total amount of memory used by the VM and available network bandwidth. In order to avoid performance degradation, a VM migration cost model is adopted in Reference [17] to help choose migratable VMs. As the authors say in Reference [18], a single VM Migration can cause performance degradation and can be estimated by an extra 10% of CPU utilization and this implies that each migration may cause SLA violations. Therefore, it is crucial to minimize the number of VM migrations and select the VM using the least memory. Thus in the migration strategy from this paper, the performance degradation of VM j is defined in Equations (3) and (4).

$$U_{d_j} = 0.1 \int_{t_0}^{t_0 + T_{m_j}} u_j(t) dt \tag{3}$$

$$T_{m_j} = \frac{M_j}{B_j}, \tag{4}$$

where U_{d_j} is VM j's total performance degradation, t_0 refers to the time when the migration starts, T_{m_j} is the time taken to complete the migration, $u_j(t)$ is the CPU utilization at time t. M_j is the amount of memory used by VM j and B_j is the network bandwidth. The smaller the memory of VM to be migrated, the smaller the migration time and the smaller the impact on the system.

2.2.3. SLA Violation (SLAV) Metric

In a cloud environment, since many users are competing for resources, each cloud service provider needs to ensure that the application requirements are met, which is usually defined in the form of SLA [19]. It is defined as a violation of the SLA between the resource provider and the user when the performance request of the resource exceeds the available capacity and the Equation (5) as follows:

$$SLAV = \frac{1}{J} \sum_{x=1}^{J} \frac{T_{sx}}{T_{ax}} \cdot \frac{1}{N} \sum_{i=1}^{N} \frac{Cd_i}{Cr_i}, \tag{5}$$

where J is the number of hosts, T_{sx} is the total time that utilization of host x reach to 100% and T_{ax} is a lifetime (total time that the host is active) of the host x. When host utilization reaches 100%, the application performance is bounded by the host. N shows a number of VMs, Cd_i estimated as 10% CPU utilization of VMi in all migrations. Cr_i is the total CPU requested by VMi.

Table 1 shows the parameter definitions that will be used in Section 3.

Table 1. Parameter Definition.

Parameter	Description
H_i_[CPU/Mem]util	CPU or memory utilization for a compute node in the migration trigger section
H_i_Pre[CPU/Mem]util	CPU or memory utilization for a compute node after the prediction in the migration trigger section
D_{CPU}/D_{Mem}	Difference between overload resource utilization and the upper threshold
OverloadHost	Overload compute node
VMtomigrate	Selected VM to be migrated from Overload Host
VM_k_[CPU/Mem]	CPU or memory utilization of VM_k
$VM_{k-[CPU/Mem]}$	Difference between VM_k_[CPU/Mem] and D_{CPU}/D_{Mem}
Hostlistidle	Compute nodes in the threshold range after prediction
hj−[CPU/Mem]	CPU or memory utilization for computing node from Hostlistidle
hostidx	Selected target compute node

3. Migration Strategy Description

A dual threshold adaptive (DTA) dynamic migration strategy proposed in this paper is divided into three parts: (1) adaptive trigger migration strategy based on Kalman filter algorithm prediction; (2) the selection of VM to be migrated; and (3), the selection of target physical node. These parts will be discussed in the following sections.

3.1. Adaptive Trigger Migration Strategy based on Kalman Filter Algorithm Prediction

It is necessary to monitor the running state of each physical compute node and set migration conditions for the resource state to determine whether the dynamic migration should be triggered or not. In this paper, the upper threshold is set to alleviate the problem that the physical server increases energy consumption and SLA violations because of the high load, so as to ensure that the resource utilization of the running node in the pool meets the demand on the basis of saving energy consumption. Set a lower threshold and migrate the VM on the compute nodes elsewhere and shut down or sleep the physical server to reduce the energy consumption of the physical cluster, when the server utilization is below the threshold. Because the resource utilization of physical servers is not stable, the CPU or memory utilization at a certain time may cause unnecessary migration due to too high or too low condition, result in a waste of system energy consumption. In this paper, the Kalman filter algorithm is used to predict the resource utilization, which only a small amount of data is needed to get the predicted starting point (the more data will make the result better). It can adjust itself and automatically set the parameters from continuous observation. Because the server resource utilization is instantaneous, the use of prediction technology can effectively prevent the invalid migration of virtual resource. The predicted migration trigger strategy based on the Kalman filter algorithm is used to determine whether the server resource utilization is higher or lower than the threshold.

Kalman filter takes the least mean square error as the best criterion to find a set of recursive estimation models. The basic idea is to use the state space model of signal and noise, to update the estimation of state variables by the estimated value of the previous time and the observed value of the present time and to find the estimated value of the occurrence time. The estimator is considered as a linear system and $\gamma(k)$ represents the resource utilization of the server in the migration strategy at the current time.

Firstly, the predicted value $\widetilde{\gamma}(k)$ is set as the optimal estimation of the previous state with Equation (6), where $\hat{\gamma}(k-1)$ is the last estimated resource utilization.

$$\widetilde{\gamma}(k) = \hat{\gamma}(k-1). \tag{6}$$

Then, the estimated value of the current time is calculated by Equation (7). Where $\overline{\gamma}(k)$ is the observation of the current time. $Kg(k)$ is the Kalman gain and calculated by the Equation (8) and Equation (9):

$$\hat{\gamma}(k) = \widetilde{\gamma}(k) + Kg(k)[\overline{\gamma}(k) - \widetilde{\gamma}(k)] \tag{7}$$

$$Kg(k) = \frac{\widetilde{P}(k)}{\widetilde{P}(k)+R} \tag{8}$$

$$\widetilde{P}(k) = P(k-1)+Q, \tag{9}$$

where R and Q are the variances of system noise and observation noise, respectively. $P(k)$ represents the deviation of the filter and is updated by Equation (10) to continue iteratively filter.

$$P\left(k\right) = (1-Kg\left(k\right))\widetilde{P}(k). \tag{10}$$

The migration strategy proposed in this paper is to judge the relationship between resource utilization and threshold combined with prediction. A detailed process is shown in Figure 3 as a trigger flow. In the proposed trigger strategy, the load utilization of each compute node is polled first to monitor and the CPU and memory utilization are used as the trigger migration criteria. Then judge the relationship of utilization and the threshold. If one or more parameters are not within the threshold range, use the prediction model to determine the relationship between the load utilization and the threshold at the next moment. If the predicted value exceeds the upper threshold, the compute node is considered overloaded and calculate D_{CPU}/D_{Mem}, which is the difference between overload resource utilization and the upper threshold. Otherwise, the strategy enters the underload judgment process with prediction model. If the predicted value is below the lower threshold, the node is identified as the underload node.

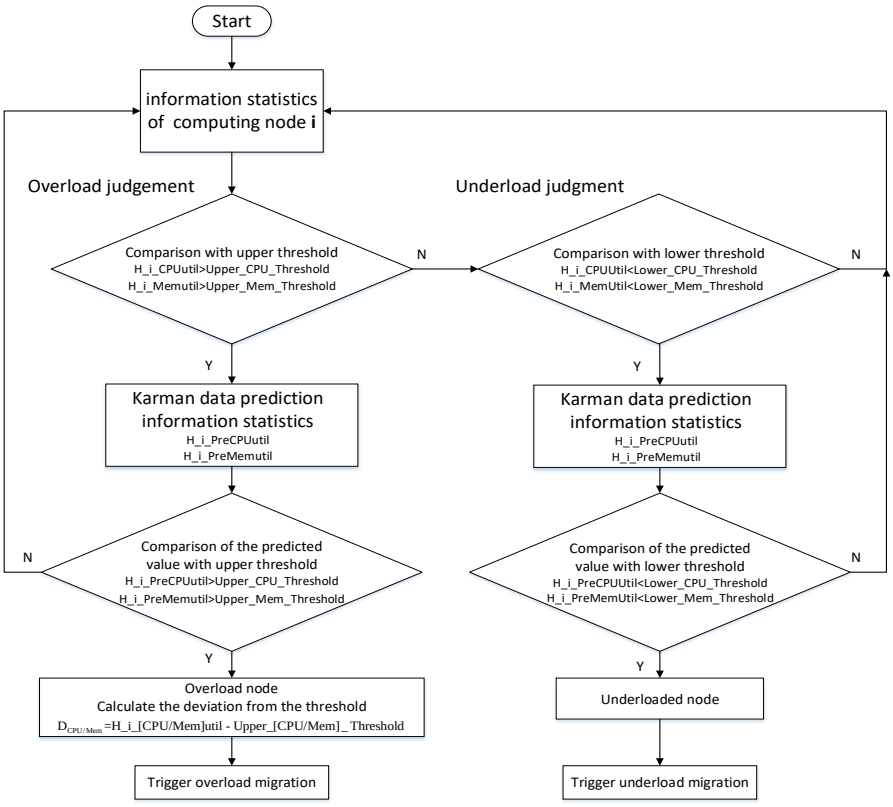

Figure 3. Dual threshold migration trigger process based on Kalman filter algorithm.

To implement a migration strategy, the VM to be migrated is selected from the overload/underload compute node according to the VM selection strategy from Section 3.2 and then the target server node to be migrated is determined from Section 3.3 to realize dynamic migrations. The strategy proposed defines the triggering of overload as three cases. (a) type 0: both CPU and memory utilization exceed the upper threshold, (b) type 1: only CPU utilization exceeds the upper threshold and (c) type 2: only memory utilization exceeds the upper threshold. If one or more utilization parameters are below the lower threshold, the dynamic migration is triggered and the underload node is shut down to reduce energy consumption. If all the parameters are within the threshold range, it is considered normal and then enters the next round of judgment process.

3.2. The Selection of VM to be Migrated (DTA-VM Selection)

Once the overload compute node is detected, the next step is to select VMs to migrate from the node to avoid performance degradation. In this paper, a VM selection strategy (DTA-VM selection) is proposed. Firstly, the VM with the least number of migrations is selected and then, in the selected VM subset, the strategy selects the VM with the minimum migration time. The migration time can be estimated as the number of RAM used by VM divided by the network bandwidth available to the server node. The proposed VM selection strategy shows in Algorithm 1.

VM_k_CPU and VM_k_Mem represent the difference between the utilization of VM's CPU and memory and D (D_{CPU} or D_{Mem}). If the difference is greater than or equal to 0, it is indicated that only one VM needs to be migrated to lower the resource utilization rate of the server below the upper threshold and this VM will be stored in the VMlist_count subset. In the subset of VMs that satisfy the minimum migration counts in the VMlist_count, the memory usage of each VM is sorted and the smallest memory is selected as the final VM to be migrated. If the list VMlist_count is empty, the strategy will judge the resource utilization by calculating f from algorithm 1 for each VM on the overloaded host and select the VM, which can alleviate the server load to the greatest extent.

Algorithm 1 DTA-VM selection strategy

Input: OverloadHost, $D(D_{CPU}$ or $D_{Mem})$
Output: VMtomigrate
foreach VM **in**OverloadHost **do**
type 0: $VM_{k-CPU} = VM_k\_CPU - D_{CPU}, VM_{k-Mem} = VM_k\_Mem - D_{Mem}$
type 1: $VM_{k-CPU} = VM_k\_CPU - D_{CPU}$
type 2: $VM_{k-Mem} = VM_k\_Mem - D_{Mem}$
if (**type** 0: $VM_{k-CPU} >= 0$ and $VM_{k-Mem} >= 0$; **type** 1: $VM_{k-CPU} >= 0$; **type** 2: $VM_{k-Mem} >= 0$)
VMlist_count$\leftarrow VM_k$
else
type 0: $w_1 = \frac{D_{CPU}}{D_{CPU}+D_{Mem}}, w_2 = \frac{D_{Mem}}{D_{CPU}+D_{Mem}}$
type 1: $w_1 = 1, w_2 = 0$
type 2: $w_1 = 0, w_2 = 1$
$f_k = w_1 \cdot VM_{k-CPU} + w_2 \cdot VM_{k-Mem}$ VMlist_f$\leftarrow VM_k, f_k$
end if
if VMlist_count =!NULL
sort VMlist_count by memory
VMtomigrate = VMlist_count[0]
else
sort VMlist_f by f_k
VMtomigrate = VMlist_f[0]
end if
end for
return VMtomigrate

When the compute node triggers migration because of too low load, the VM selection is not needed because all the VMs in the node are then directly migrated and this node will be closed in order to save energy.

3.3. The Selection of Target Physical Node (DTA-Target Selection)

When the Nova_Scheduler module selects the target node in the OpenStack, it selects the node which meets the requirements of VM resources and has the most memory surplus, regardless of other factors. Therefore, the strategy of selecting the target server node proposed in this paper takes into account not only the current CPU and memory usage of the server node but also the resource usage of the target node after the migration of the VM. The target compute node selection strategy is proposed based on Kalman filter prediction, which gives priority to select the nodes in the set within the normal threshold range to reduce the computational workload.

The target node selection strategy shows in Algorithm 2, for each other server node in the cloud-based BBU pool, to avoid unnecessary migration, prediction technology is used to predict the resource utilization of the running nodes that can accept migration in the pool. Only when the predicted values are within the threshold range, the size of the decision value F from algorithm 2 of CPU and memory utilization will be sorted and the sorted server list will be returned. The server node listed first is the most suitable target node. The calculation of the F value is similar to f in the previous section and there are also three cases. The selection of the target node in the case of underload is performed according to type 0.

Algorithm 2 DTA-Target selection strategy

Input: VMtomigrate, Hostlistidle, D(D_{CPU} or D_{Mem})
Output: hostidx
foreach host **in** Hostlistidle $= \{h_1, h_2, , , h_j\}$ **do**
if h_j_CPU + VM_CPU < Upper_CPU_Threshold and
h_j_CPU + VM_CPU < Upper_CPU_Threshold
for Hostlistidle $= \{h_1, h_2, , , h_j\}$ **do**
type 0: $w_1 = \frac{D_{CPU}}{D_{CPU}+D_{Mem}}, w_2 = \frac{D_{Mem}}{D_{CPU}+D_{Mem}},$
type 1: $w_1 = 1, w_2 = 0$
type 2: $w_1 = 0, w_2 = 1$
$F_j = w_1 \cdot h_{j-CPU} + w_2 \cdot h_{j\ -Mem}$
min = max.*value*
if $F_j <$ min **then**
min $= F_j$
hostidx $= h_j$
end if
end for
end if
end for
return hostidx

4. Simulation and Results

4.1. Experimental Environment

The DTA dynamic migration strategy proposed in this paper is firstly simulated and compared on Cloudsim and then applied to the OpenStack cloud platform for concrete implementation. CloudSim [20] is a cloud environment simulation software that can simulate virtualized and cloud-based entities, such as data centers, VMs and physical hosts, to model and simulate a cloud computing system. Based on Cloudsim, we can implement different resource allocation strategies and evaluate strategy performance. OpenStack [21] is an open source project developed jointly by NASA and

Rackspace to provide software for building and managing public and private clouds through which any company or individual can build a cloud computing environment. OpenStack is the software of the cloud computing IaaS layer, which provides an infrastructure solution for the needs of scalable private and public clouds of all sizes.

(1) The hardware used in the Cloudsim simulation experiment is a laptop with a pre-installed Windows 10 (64-bit) operating system (CPU type: Intel Core i5-7300HQ, memory 8G). Because CloudSim is based on Java, Eclipse is selected as its operating platform. In this paper, JDK1.8 and Eclipse 4.1.1 are downloaded and simulated with the version of CloudSim5.0. A virtual BBU pool, including 300 physical servers and 1000 VMs, is created based on the constructed simulation platform. The processing power per server node is based on 2000 MIPS and 4000 MIPS and the processing power allocation per VM is based on 1000 MIPS, 2000MIPS and 3000 MIPS. VMs are assigned to odd ordinal hosts in order and then to even ordinal hosts. The number of tasks submitted by user is 1500 and is assigned to the VM in sequence. The detailed experimental parameters are shown in Table 2.

Table 2. Detailed experimental parameters.

Parameter	Value	Unit
Number of hosts	300	/
Hosts' CPU capacity	2000, 4000	MIPS
Hosts' memory size	4096, 6144	MB
Number of VMs	1000	/
VMs' CPU capacity	1000, 2000, 3000	MIPS
VMs' memory size	256, 512, 1024	MB
Number of tasks	1500	/
Length of tasks	108,000,000	/

The simulation set running time is 12 hours and the scheduling interval is 300 seconds and system resource usage update, operation information collection and VM scheduling are performed. In order to verify the effectiveness of the proposed strategy, it is necessary to compare and analyze the existing VM selection and the target node selection strategy. In this experiment, firstly, the following three VM selection strategies are compared with DTA-VM selection strategy: Maximum Correlation (MC) strategy, Minimum Utilization (MU) strategy and Random Selection (RS) [9] strategy. Then, the proposed DTA-Target node strategy is compared with the IQR, LR and the most memory remaining strategy for OpenStack.

(2) In this experiment, the BBU pool based on the OpenStack cloud environment was built composed of four physical servers. One of them acts as a cloud controller and the other three servers act as compute nodes in this cloud environment. The experimental architecture is shown in Figure 4.

The controller node acts as the whole controller, responsible for controlling resources and the compute nodes act as a resource computing and storage resource node in the cloud environment. Through multi-node deployment, the server hardware failure can be dealt with without much difficulty or the dynamic migration of VM can be performed without interfering with the client. Keystone, Nova, Glance, Neutron, Horizon and Cinder components of OpenStack are deployed on the control node, where Nova can implement management of virtual BBU pools such as creating VMs and implementing dynamic migration operations. OpenStack uses the web graphical interface or command line to control the entire cloud-based BBU pool. In this experimental deployment architecture, each server is configured with two network interfaces. The private network segment is made up of interface1, allowing each component of OpenStack to communicate with each other and the service provided by the cloud environment is provided by interface2, realizing the connection with the Internet.

In the cloud system, node clusters implement SSH mutual access without password and dynamic migration is achieved through shared storage. In this scenario, VM instances are stored in shared storage and migration is mainly the migration of instance memory status, which greatly improves the migration speed. In this experiment, the dynamic migration mode is set to the pre-copy mode and

there is no real implementation of BBU related protocols on the VMs. BBU performs operations such as baseband processing or RRH return signal processing, which has a load effect on the VMs. This effect can be simulated by the script running on the VMs. The script implements CPU and memory pressure operation, simulates the dynamic load changes brought by BBU performing signal processing protocol and other functions. In the implementation, controller and compute nodes are installed in the ubntu16.04 64-bit operating system. In the OpenStack experimental environment, the controller node is deployed on an intel i5-6500, 12G memory, x86-architecture server with dual network cards. Twelve VMs (ubntu16.04 64-bit OS) are set up and Table 3 shows the deployment of VMs and compute nodes before implementing the migration strategy proposed in the paper.

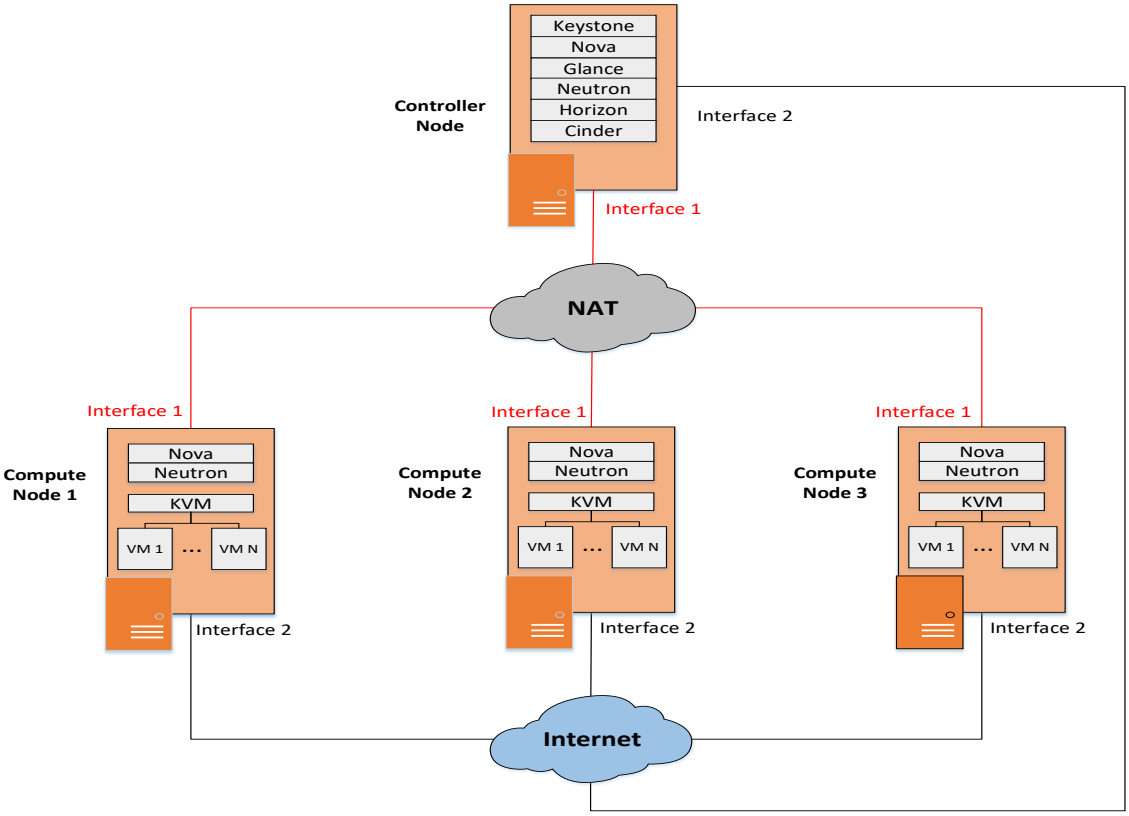

Figure 4. Experimental architecture.

Table 3. The deployment information of VMs and compute nodes.

Compute Node	VM ID	VCPU	Ram(M)	Disk(G)
Compute1 CPU: Intel Core i7-8700 RAM:16G x86 architecture	1	2	1024	10
	2	2	1024	10
	4	1	2048	20
	7	1	1024	5
Compute2 CPU: Intel Core i7-4770 RAM:8G x86 architecture	3	2	1024	10
	6	1	512	5
	10	1	1024	10
	12	2	512	5
Compute3 CPU: Intel Xeon E5-2643 RAM:128G x86 architecture	5	1	1024	20
	8	2	2048	10
	9	1	1024	10

4.2. Experimental Results and Analysis

4.2.1. Simulation of Migration Strategy on Cloudsim

In order to compare the performance of the proposed migration strategy with that of the existing algorithms, we consider four indicators: the total energy consumption of the BBU pool when performing the work, SLA violation, the total migration number when the dynamic migration occurs and the dynamic migration time.

The first group: First Fit is the default target host placing method. Under the host detection strategy (Mad, Iqr, Lr, Thr), DTA-VM selection is compared with the following three VM selection strategies—MC, MU and RS. The simulation results are shown in Figures 5–8, compared with the existing strategy, DTA-VM selection effectively reduces energy consumption, SLA violations, the number of migration and migration time. In terms of SLA violations and the number of migrations, DTA-VM selection is reduced by more than 50% compared to MU and the effect is reduced by about half in terms of migration time compared to the other three selection algorithms. Because DTA-VM selection achieves load balancing with as few number of migration and migration times as possible, the consumption caused by migration and SLA violations can be reduced.

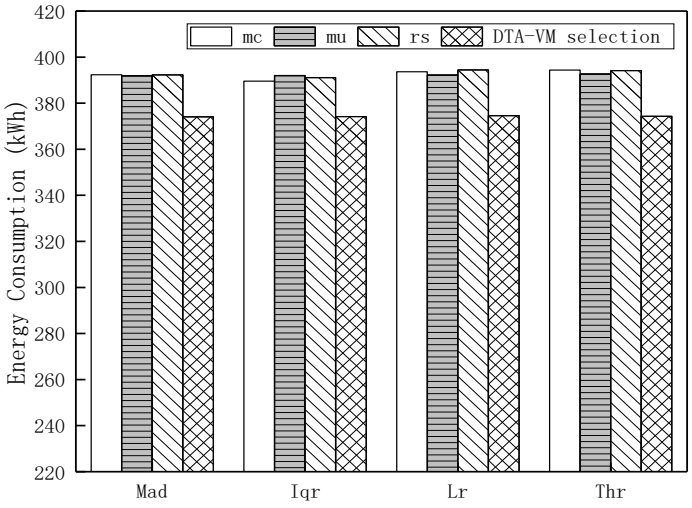

Figure 5. Energy consumption of DTA-VM selection.

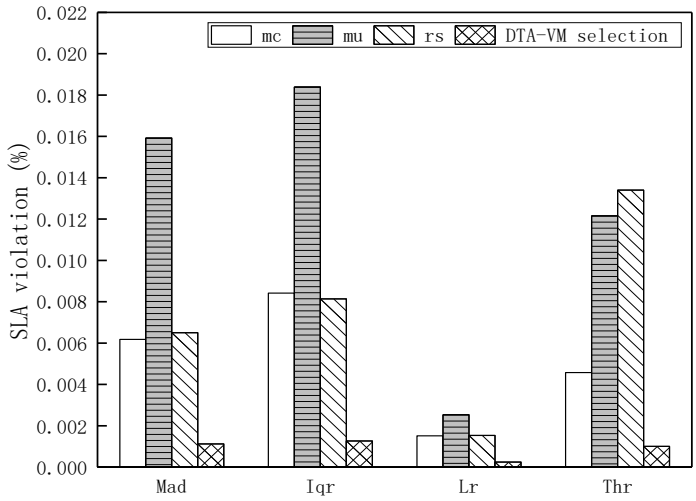

Figure 6. SLA violation of DTA-VM selection.

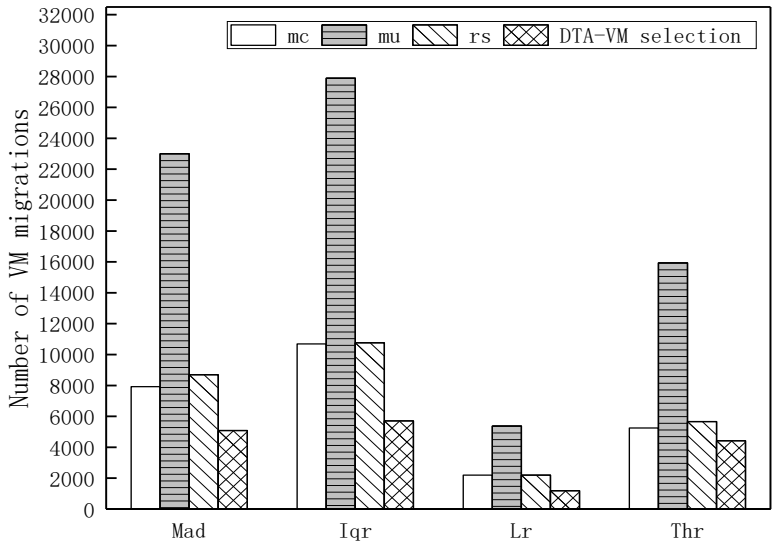

Figure 7. Number of migrations of DTA-VM selection.

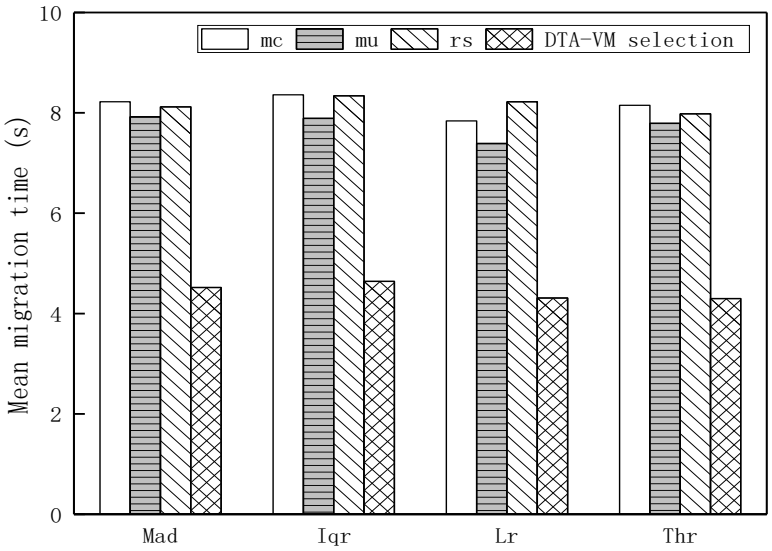

Figure 8. Migration time of DTA-VM selection.

The second group: based on the proposed strategy (DTA-VM selection), MC, MU and RS VM selection strategy, the proposed DTA-Target selection and First Fit and OpenStack "maximum memory remaining" strategy are compared and analyzed for performance. As shown in Figures 9–12, the DTA-Target selection proposed is not much different in terms of energy consumption. However, the performance of the target node is reduced by more than 50% in terms of SLA violation; and compared with the performance of the number of migration and migration time, DTA-Target selection is far lower than the existing two strategies. This is because DTA-Target selection can comprehensively consider the resource state and predict the status of the VM migration to the target node to reduce consumption and SLAV caused by repeated migration.

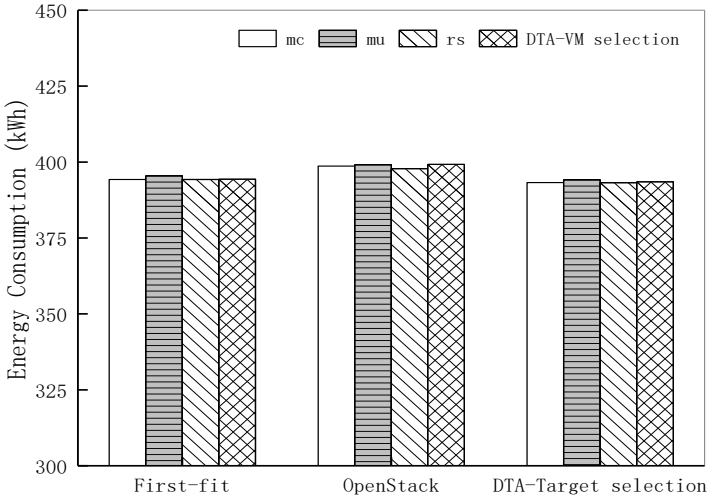

Figure 9. Energy consumption of DTA-Target selection.

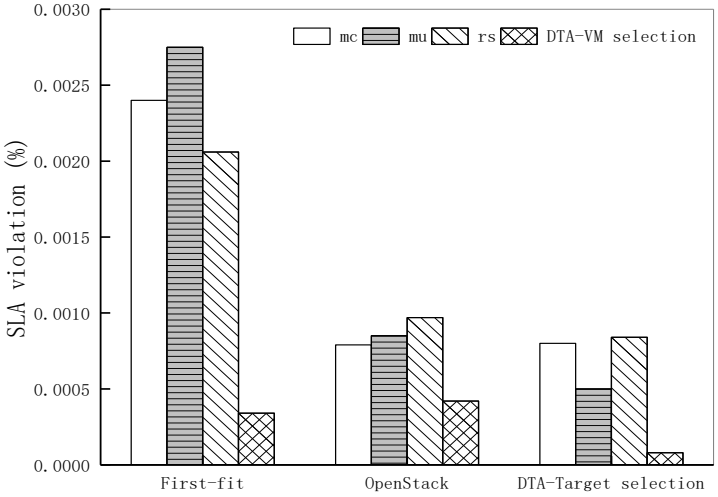

Figure 10. SLA violation of DTA-Target selection.

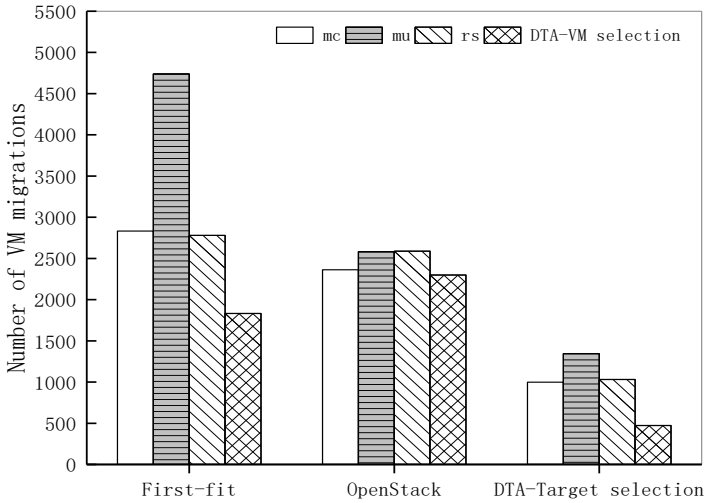

Figure 11. Number of migrations of DTA-Target selection.

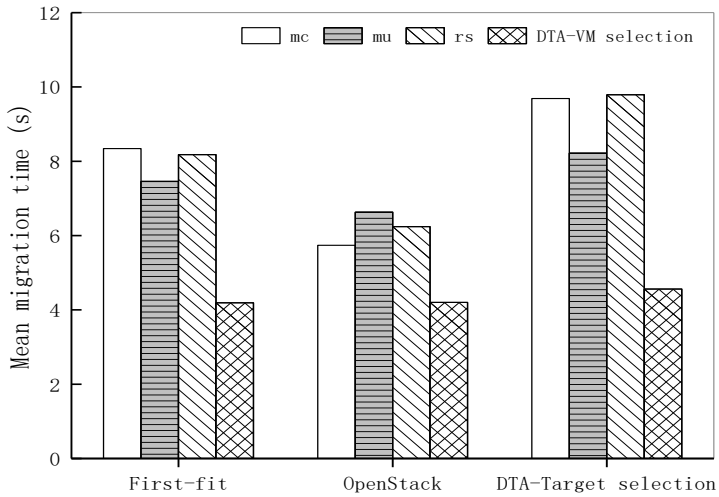

Figure 12. Migration time of DTA-Target selection.

In summary, when the DTA-VM selection and DTA-Target selection proposed in this paper work together, the above four performances can obtain more desirable effects. The simulation on Cloudsim can prove that the DTA dynamic migration strategy has a significant improvement in performance.

4.2.2. The experiment of Migration Strategy on OpenStack

In order to simulate the effect of dynamic change of load, this experiment based on the OpenStack cloud platform, the VMs with the ID number of 1–6,8 and 12 are tested for irregular CPU or memory stress tests, respectively. The experimental observation time is 270 minutes and the migration strategy deployed on the controller node polls and monitors the resource utilization of compute nodes every 60 seconds. According to the prediction of the Kalman filter algorithm and the manually set upper and lower limit threshold, it is determined whether dynamic migration is needed or not. In this paper, two groups of experiments are carried out, namely, the upper threshold overload migration and the lower threshold underload migration. The upper and lower threshold values are 80% and 10%, respectively. The dynamic variation of CPU and memory utilization over time is shown in Figures 13–15, which show the experimental results of the dynamic migration of upper and lower threshold in the migration strategy, respectively. This experiment is mainly to verify the feasibility of the dynamic migration strategy. There will be a short and unavoidable interruption in the migration process of the VMs, which will be studied in detail in the future work.

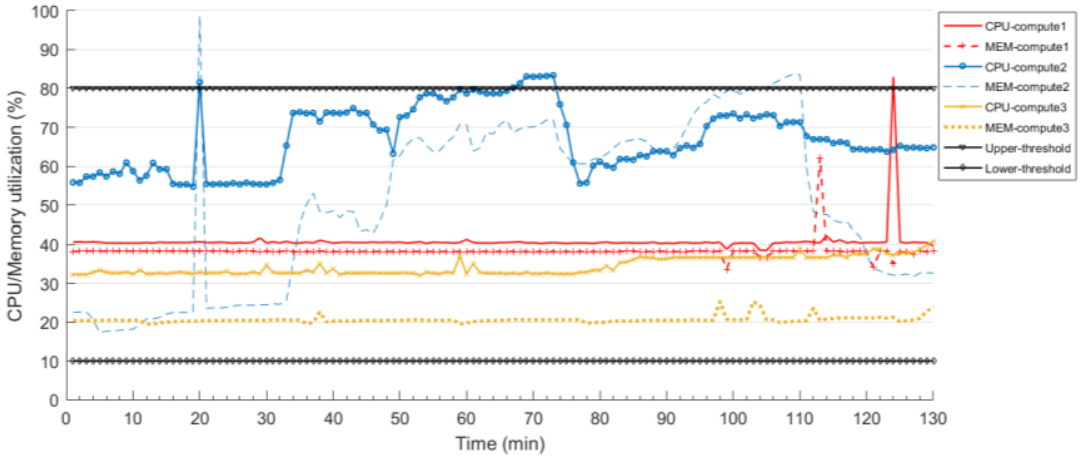

Figure 13. Compute node resource utilization based on upper threshold migration (0–130 min).

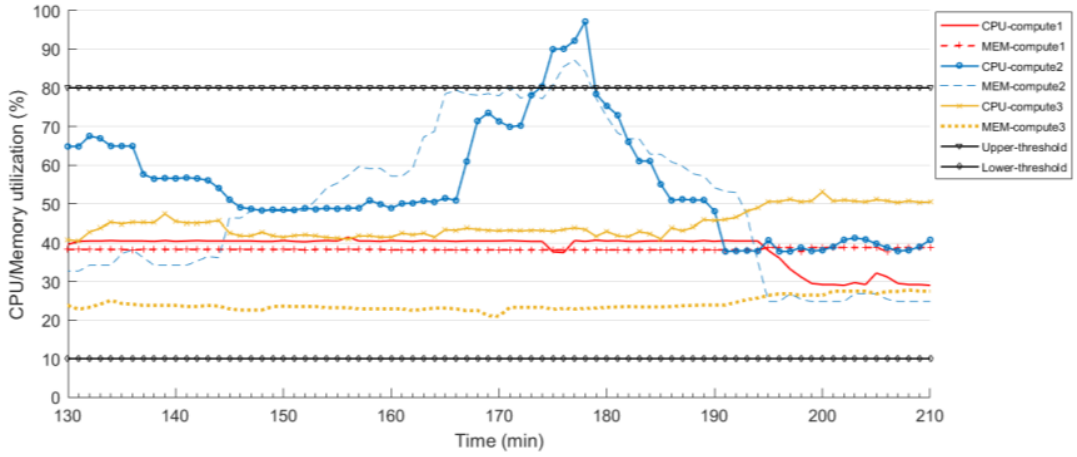

Figure 14. Compute node resource utilization based on upper threshold migration (130–210 min).

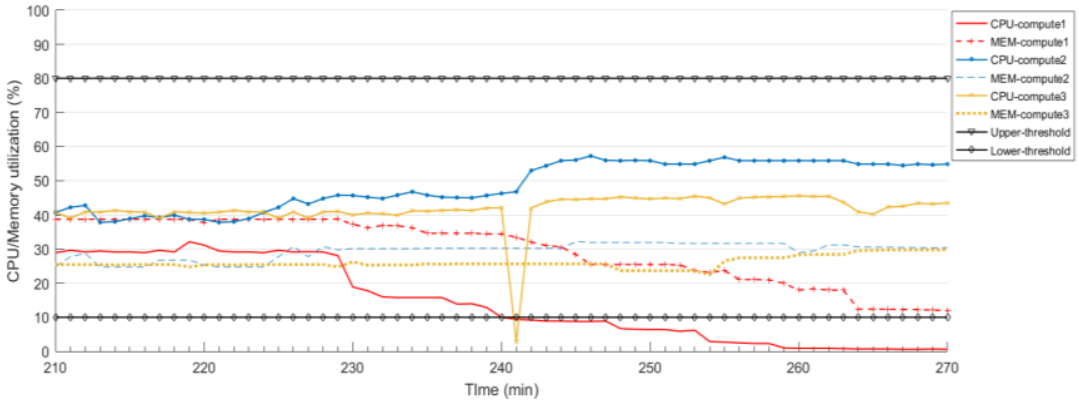

Figure 15. Compute node resource utilization based on lower threshold migration.

In the initial stage of the experiment, continuous CPU and memory stress test scripts are executed for all the VMs on the nodes. As can be seen from the Figure 13, the resource (CPU and memory) utilization of the three compute nodes is in a steady state in the first 30 minutes. After 30 minutes, keep compute1 and compute2 unchanged, perform a more intense load pressure script on compute2 and the CPU and memory utilization of compute2 are on the rise. Both CPU and memory utilization of compute2 exceeds the upper threshold in about 20 minutes and it does not migrate immediately. Similarly, in about 125 minutes, the CPU utilization of compute1 exceeds the upper limit threshold but does not migrate immediately according to the prediction. This is because the resource utilization exceeds the upper threshold is temporary at these two times and the unnecessary migration operation is avoided based on the Kalman prediction algorithm. Starting from about 70 minutes, the CPU utilization of compute2 exceeded the upper threshold again. Combined with Kalman's prediction, the dynamic migration of VM is carried out in accordance with the requirements of the migration strategy. The load utilization of compute2 has gradually increased and the CPU utilization of compute2 exceeded the upper threshold again at around 70 minutes. Combined with the prediction model to determine that compute2 is an overload node, the dynamic migration of VM is carried out in accordance with the requirements of the migration strategy. At about 105 minutes, the memory utilization of compute2 exceeded the upper threshold and a dynamic migration was performed by Kalman's prediction. After the migration, the memory utilization rate of compute2 drops significantly and the load utilization of compute2 returns to within the normal threshold range.

Reducing the load pressure on compute2 in about 130 minutes, as can be seen in Figure 14, the load situation of the computing node gradually becomes stable. From about 160 minutes, the load pressure of compute2 is gradually increased again and both compute2's CPU and memory utilization

are above the upper threshold at about 175 minutes. Through the prediction, a dynamic migration operation of the VM was performed. After the dynamic migration, the VM of compute2 migrates to other idle compute nodes. After the dynamic migration strategy selection, the VM of compute2 migrates to compute3 node. It can be seen that the resource utilization of compute2 drops below the upper threshold after the dynamic migration. Since the physical resources of the compute3 node are very abundant, the load utilization of this node after the migration of the virtual machine has increased slightly but it is not obvious.

Figure 15 shows resource utilization of compute nodes based on lower threshold migration of OpenStack. In the experiment, the intensity of the load pressure script executed by the VMs on compute2 and compute3 is not changed and compute1 gradually reduces the load pressure intensity. From about 240 minutes, the CPU utilization of compute1 is below the lower threshold and the under-load migration is determined. All VMs on the node are migrated to other nodes within the threshold range. During the dynamic migration process, the VMs are migrated to compute2 and compute3. After the migration process, the load utilization of the other two nodes has increased and the CPU utilization of compute1 is close to 0 and the node is shut down to reduce the energy consumption of the cloud-based BBU pool. The CPU utilization of compute3 is below the lower threshold in approximately 242 minutes and the dynamic migration is not required by the Kalman prediction. The VMs on compute1 are migrated to compute2 and compute3 and the resource utilization of these two compute nodes increases.

Due to the limited experimental conditions, there are only three compute nodes in the OpenStack environment but through the above experiments, it can be seen that the system and strategy designed in this paper are feasible. Combined with the simulation of the migration strategy used in the system in the third section, the effectiveness of the migration strategy proposed in this paper can be proved.

5. Conclusions

In the virtual BBU pool, it is necessary to reduce energy consumption without affecting the destruction of SLA. This paper proposes a dual threshold adaptive (DTA) dynamic migration strategy, including a triggered migration mechanism based on Kalman filter prediction, DTA-VM selection strategy and DTA-Target selection strategy, which can more accurately determine the resource utilization of the servers in the BBU pool and trigger the migration. In the strategy, the dynamic migration VM, from the overloaded host is selected based on the minimum number of migration and minimum migration time to avoid the increase of energy consumption caused by the multiple migrations of VM in the BBU pool. The DTA-Target selection strategy predicts whether the resource performance of the VM after the migration meets the threshold requirements, comprehensively considers the CPU and memory utilization and selects the target node to reduce the migration failure caused by the lack of physical resources of the node after the VM migration. Through the DTA migration strategy, adaptive dynamic migration can be realized to reduce energy consumption and SLA violations in the BBU pool. The experimental results show that the DTA migration strategy proposed in this paper has better performance in the cloud center. The migration strategy is further realized in the BBU pool based on the OpenStack platform. Without manual intervention, the change of the workload can be automatically responded and the energy consumption is saved. In the future, the authors will do more in-depth research on the actual deployment of the integration of BBU functions with OpenStack.

Author Contributions: C.W. proposed the basic framework of the research scenario. In addition, C.W. was in charge of modeling the problem and proposed the migration strategy. Y.C. performed the simulations and wrote the paper. Z.Z. provided suggestions for platform building. W.W. gave some suggestions on the mathematical model and formula derivation. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Fundamental Research Funds for the Central Universities (2019RC05).

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

References

1. Mahapatra, B.; Kumar, R.; Kumar, S.; Turuk, A.K. A Heterogeneous Load Balancing Approach in Centralized BBU-Pool of C-RAN Architecture. In Proceedings of the 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, India, 6–8 April 2018; pp. 1–5.
2. Gao, Z.; Zhang, J.; Yan, S.; Xiao, Y.; Simeonidou, D.; Ji, Y. Deep Reinforcement Learning for BBU Placement and Routing in C-RAN. In Proceedings of the 2019 Optical Fiber Communications Conference and Exhibition (OFC), San Diego, CA, USA, 3–7 March 2019; 18618440.
3. Li, Y.; Bhopalwala, M.; Das, S.; Yu, J.; Mo, W.; Ruffini, M.; Kilper, D.C. Joint Optimization of BBU Pool Allocation and Selection for C-RAN Networks. In Proceedings of the 2018 Optical Fiber Communications Conference and Exposition (OFC), San Diego, CA, USA, 11–15 March 2018; p. 17855949.
4. Pizzinat, A.; Chanclou, P.; Saliou, F.; Diallo, T. Things You Should Know About Fronthaul. *J. Light. Technol.* **2015**, *33*, 1077–1083. [CrossRef]
5. Amani, N.; Pedram, H.; Taheri, H.; Parsaeefard, S. Energy-Efficient Resource Allocation in Heterogeneous Cloud Radio Access Networks via BBU Offloading. *IEEE Trans. Veh. Technol.* **2019**, *68*, 1365–1377. [CrossRef]
6. Tang, J.; Tay, W.P.; Quek, T.Q.S.; Liang, B. System Cost Minimization in Cloud RAN With Limited Fronthaul Capacity. *IEEE Trans. Wireless Commun.* **2017**, *16*, 3371–3384. [CrossRef]
7. Yazir, Y.O.; Matthews, C.; Farahbod, R.; Neville, S.; Guitouni, A.; Ganti, S.; Coady, Y. Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis. In Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, Miami, FL, USA, 5–10 July; pp. 91–98.
8. Melhem, S.B.; Agarwal, A.; Goel, N.; Zaman, M. Minimizing Biased VM Selection in Live VM Migration. In Proceedings of the 2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech), Rabat, Morocco, 24–26 October 2017. [CrossRef]
9. Beloglazov, A. Energy-efficient management of virtual machines in data centers for cloud computing. Ph.D. Thesis, University of Melbourne, Melbourne, Australia, February 2013.
10. He, K.; Li, Z.; Deng, D.; Chen, Y. Energy-efficient framework for virtual machine consolidation in cloud data centers. *Chin. Commun.* **2017**, *14*, 192–201. [CrossRef]
11. Li, Z.; Wu, G. ACM. Optimizing VM Live Migration Strategy Based On Migration Time Cost Modeling. In Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems, Santa Clara, CA, USA, 17–18 March 2016; pp. 99–109.
12. Razali, R.A.M.; Ab Rahman, R.; Zaini, N.; Samad, M. Virtual machine migration implementation in load balancing for Cloud computing. In Proceedings of the 2014 5th International Conference on Intelligent and Advanced Systems (ICIAS), Kuala Lumpur, Malaysia, 3–5 June 2014; pp. 1–4.
13. Raghunath, B.R.; Annappa, B. Dynamic Resource Allocation Using Fuzzy Prediction System. In Proceedings of the 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, India, 6–8 April 2018; pp. 1–6.
14. Yang, G.; Zhang, W.J. Research of the optimized resource allocation strategy based on Openstack. In Proceedings of the 2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 18–20 December 2015.
15. Peng, M.; Sun, Y.; Li, X.; Mao, Z.; Wang, C. Recent Advances in Cloud Radio Access Networks: System Architectures, Key Techniques, and Open Issues. *IEEE Commun. Surv. Tutorials* **2016**, *18*, 2282–2308. [CrossRef]
16. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Futur. Gener. Comput. Syst.* **2012**, *28*, 755–768. [CrossRef]
17. Fu, X.; Zhou, C. Virtual machine selection and placement for dynamic consolidation in Cloud computing environment. *Front. Comput. Sci.* **2015**, *9*, 322–330. [CrossRef]
18. Song, Y.; Wang, H.; Li, Y.; Feng, B.; Sun, Y. Multi-Tiered On-Demand Resource Scheduling for VM-Based Data Center. In Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai, China, 18–21 May 2009; pp. 148–155. [CrossRef]
19. Addya, S.K.; Turuk, A.K.; Satpathy, A.; Sahoo, B.; Sarkar, M. A Strategy for Live Migration of Virtual Machines in a Cloud Federation. *IEEE Syst. J.* **2019**, *13*, 2877–2887. [CrossRef]

20. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Int. J. Softw. Pract. Exp.* **2011**, *41*, 23–50. [CrossRef]
21. Open Source Software for Creating Private and Public Clouds. Available online: //www.openstack.org/ (accessed on 10 May 2019).