A New Energy Efficient VM Scheduling Algorithm for Cloud Computing based on Dynamic Programming

Kepi Zhang, Tong Wu, Siyuan Chen, Linsen Cai, Chao Peng
Shanghai Key Lab of Trustworthy Computing, School of Computer Science and Software Engineering,
East China Normal University, Shanghai, China
Email: 51141500059@ecnu.cn, cpeng@sei.ecnu.edu.cn

***Abstract*—As a new computing paradigm, cloud computing has significantly contributed to the rapid development of massive data centers. However, the corresponding energy issue becomes increasingly challenging. In this paper, we focus on the energy saving issue for virtual machine (VM) selections on an overloaded host in a cloud computing environment. We analyze the energy influencing factors during a VM migration, then design energy efficient VM selection algorithms based on greedy algorithm and dynamic programming method. We conduct experiments with CloudSim and results show that the proposed algorithm in this paper can effectively reduce energy consumption while satisfying the SLA constraints.**

Keywords—cloud computing, green computing, greedy algorithm, dynamic programming, CloudSim

I. INTRODUCTION

Cloud computing [1] is a new computational model that enables on-demand rationing and dynamic expansion of resources. It is based on large-scale distributed system which provides resources through virtualization and network technology. With cloud computing, users can request and release a variety of computing resources quickly, and use computing, storage, network and other resources like using water and electricity in daily life. However, as the scale of cloud computing data centers expanding, the server and other infrastructure's energy consumption has increased dramatically, which has not only raised the operating costs in data centers, but also led to serious environmental problems.

Literature [2] reported that, in 2013, data centers in the US consumed about 910 billion KWh electricity, which is equivalent to the sum of 34 thermal power plants (which generates 500 megawatts annually), these electricity was enough for all New York residents to consume for two years. By 2020, the power consumption of data center is expected to reach 1400 billion KWh, which is equivalent to the sum of electricity output of 50 thermal power plants. This means data centers in the US will spend $ 13 billion a year on electricity and cause 150 billion metric tons of carbon dioxide emissions.

The energy cost of data centers has become an important operating cost that can't be ignored, so effective energy management of data centers turns out to be a hot research topic in both academia and industry. Research on energy-saving technology in cloud environment is of great significance.

The current energy-saving technologies of data center include chip-level energy-saving technology, infrastructure-level energy-saving technology, system-level energy-saving technology [4] and application layer energy-saving technology [5]. Chip-level energy-saving technology saves energy mainly through dynamic regulating the CPU frequency and voltage and using special low power consumption components; infrastructure-level energy-saving technology mainly through using efficient cooling system and heat dissipation technology; system-level energy-saving technology mainly through dynamic adjusting the system workload and task queue on the CPU; while application layer energy-saving technology achieves the same goal through selecting the executing plans of minimum energy consumption after estimating energy consumptions of such application programs.

In this paper, we focus on energy efficient algorithms from system-level considerations and our goal is to optimize the VM selection on an overloaded host before VM placement.

II. RELATED WORK

The original intention of cloud computing is to reduce the customer's operating costs, while as the size of the data center grows, the energy cost of cloud service providers is exacerbated. So both academia and industry are paying close attention to data center energy efficiency research. In literature [6], the energy consumption level and energy efficiency level of data centers in China are studied. The results show that China's data centers not only has amazing power consumption but also has low energy efficiency level and there is great potential for energy saving. Through the research on virtual cloud computing platform's energy management, Ye K J et al [7] introduced the energy monitor approach and energy profiling analysis method, and summarized the current energy management mechanism of the progress made from the virtualization layer and the cloud platform layer. Ruan S L et al. [8] proposed an overall energy optimization method through task scheduling for virtualized data centers.

In recent years, there are already lots of researchs on cloud computing energy consumption model and energy-saving scheduling algorithm. In the study of the heuristic resource-aware scheduling method of cloud data center, Belogazov A et al.[9] proposed an energy consumption model based on the maximum power of the server and current CPU's utilization rate to estimate the current power consumption of the server.

978-1-5090-6644-5/17 $31.00 © 2017 IEEE
DOI 10.1109/CSCloud.2017.46

What' more, they proposed a MBFD (Modified Best Fit Decreasing) VM placement algorithm and a MM(Minimization of Migration) VM selection algorithm to optimize the energy consumption in data center. Tan Y M et al. [10] proposed an energy optimization management method based on task scheduling, which aims to reduce idle hosts in the cloud computing system since the idle hosts consume considerable energy and there is "luxury" energy waste caused by unreasonable task scheduling, however, the method has not been verified under real cloud environment or cloud simulation tools like CloudSim[11]. Cheng Chenlin et al. [12] established a two-level resource scheduling model under cloud computing environment on the basis of MapReduce programming model, and they combined various states like in work, idle, dormancy of the host to establish energy consumption model, and finally they make the energy-saving scheduling experiments in CloudSim with genetic algorithm. However they presumed that the power of host under different states (dormancy, idle, work, storage, communication) is the same, which is actually changed from time to time and should be calculated by integral computation rather than the direct multiplication of power and time. Luo Liang et al.[13] proposed the energy model in cloud data center based on performance counter and system utilization respectively, and then compared the prediction accuracy of multiple linear regression and nonlinear regression, finally they come to a conclusion that under the influence of new hardware and software, accuracy prediction of energy consumption model is difficult to achieve through simple linear regression method, while polynomial lasso model is relatively more accurate. Meikang Qiu et al [22] present a genetic-based optimization algorithm for *chip multiprocessor*(CMP) equipped with *Phase-change memory*(PCM) in green clouds, the experiment show that the algorithm can significantly reduce the maximum memory usage and improve the efficiency of memory usage. Keke Gai et al [23] proposed a *dynamic energy-aware cloudlet-based mobile cloud computing model* (DECM) which focused on solving the additional energy consumptions during the wireless communications by leveraging *dynamic cloudlets* based model, they examine the model by a simulation of practical scenario and provide solid results for the evaluations.

Due to the flexibility and convenience brought by cloud computing technology, public or private cloud computing technologies are developing rapidly. Now there are already many commercial public cloud platforms such as Ali cloud, Tencent cloud at China and American Amazon Web Services, Windows Azure and so on. The well-known open source cloud operating system OpenStack [14] has played an important role in the development of private cloud, because of its excellent architecture design, users can easily expand their required functions. Currently the energy related subproject of OpenStack is ceilometer[15][16], Francois Rossigneux et al. [17] extended ceilometer's energy collection plug-in which named KWAPI[18] through the IPMI(Intelligent Platform Management Interface) on modern server, this laid the foundation of subsequent research on the energy problem under OpenStack cloud environment.

III. ENERGY-SAVING VM SCHEDULING ALGORITHM

Resource allocation in cloud data center includes tasks assignment and VMs assignment. The allocation of tasks is to decide which VM should be assigned to run the task. The allocation of VM is to decide which physical node should be assigned to run the VM. In this paper, we mainly study the allocation of VMs. VM scheduling involves four steps: (1) detection of the overloaded physical node; (2) detection of the underloaded physical node; (3) select the appropriate VMs which need be migrated from the overloaded physical node; (4) place the VMs selected in step(3) and all VMs from underloaded physical node to appropriate physical node. In this paper, we mainly focus on step (3). We use the static CPU threshold[20] to detect overloaded and underloaded physical nodes, and MBFD[9] as the VM placement algorithm.

A. VM selection model on an overloaded physical node

Authors in [19] show that VM's migration energy is related to the size of the VM(mainly memory) and physical node's available bandwidth. The relationship between the above two factors and the VM's migration time is shown in (1):

$$t_{mi} = \frac{m_i}{B_j} \tag{1}$$

t_{mi} represents the migration time of VM *i*, m_i represents the current used memory of VM *i*, VM *i* runs on physical node *j*, B_j represents the current available bandwidth of physical node *j*. The energy consumption increment of physical node *j* during VM *i* migration is calculated by(2):

$$\Delta E_{mi} = \int_{t_0}^{t_0+t_{mi}} \Delta P_{mi}\, dt \tag{2}$$

ΔE_{mi} represents the energy increment during VM *i* migration,t_0 represents the time VM *i* started to migration, ΔP_{mi} represents the power increment during VM *i* migration.

From (1) and (2), it can be concluded that the migration time reduction of the VM is obviously able to reduce the energy consumption during VM migration. To reduce the VM migration time we can reduce the total migrate size or increase the physical node's available bandwidth. Since the latter is difficult to deploy in real environment, we reduce the total memory of VM to be migrated. So the problem can be modeled as : select a VM set from an overloaded physical node that satisfies the equation(3) under the premise of satisfying equation (4):

$$min\ VM_{size} = \sum_{i=1}^{m} x_i \times m_i \tag{3}$$

$$s.t \sum_{i=1}^{m} x_i \times c_i \geq C_{j\_used} - upperThreshold \times C_{j\_total} \tag{4}$$

VM_{size} represents the total memory of VMs which are selected as migration VM, c_i represents the current CPU used by VM *i*, $upperThreshold$ represents the upper CPU utilization threshold of a physical node, C_{j_used} represents the current used CPU of physical node *j*, C_{j_total} represent the total CPU of physical node *j*, x_i is defined as follow:

$$x_i = \begin{cases} 0, VM\ i \text{ is } not\ selected\ as\ migration\ VM \\ 1, VM\ i \text{ is } selected\ as\ migration\ VM \end{cases} \quad (5)$$

B. VM selection algorithm based on greedy

Literature[20] has mentioned a greedy algorithm to select a VM set with minimum migration time. The basic idea of greedy algorithm is to select a VM with the minimum memory in the remain VM list. So, we first sort the VM list on overloaded physical nodes in ascending order based on VM's memory, then add the VM with minimum memory to migration list, after that check whether the physical node is in underloaded status or not, if not repeat this process until physical node is in underloaded status, finally return the selected migration VMs. Fig.1 shows an example of the migration VMs selected by greedy algorithm.

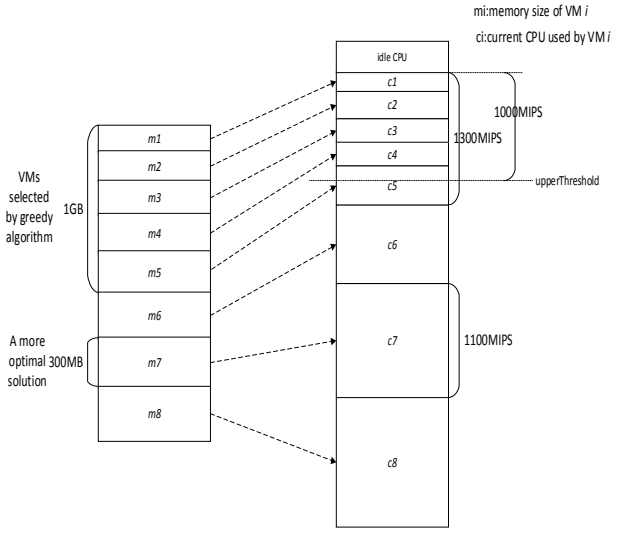

Fig.1. Migration VMs selected by greedy algorithmWe can see that VM set {1,2,3,4,5} is selected by greedy algorithm, but obvious VM 7 is a more optimal solution. So greedy algorithm is not the best method to solve this problem, but its time complexity is $O(n)$(n is the VM numbers running on the physical node).

C. VM selection algorithm based on dynamic programming

Dynamic programming can't directly solve the problem described by expression(3) and expression(4), we need to do some modifications to meet the dynamic programming requirements. So in this paper we will go to compute the reserved VM set which needn't to be migrated, then remove these VMs from the VM list, the rest is the final migration VM set. The reserved VM set can be described by expression(6) and expression(7):

$$max\ VM_{size} = \sum_{i=1}^{m} x_i \times m_i \quad (6)$$

$$\text{s.t} \sum_{i=1}^{m} x_i \times c_i \le upperThreshold \times C_{j\_total} \quad (7)$$

m_i, c_i, C_{j_total} and *upperThreshold* are defined above, x_i is defined as following:

$$x_i = \begin{cases} 0, VM\ i \text{ is } not\ selected\ as\ reserved\ VM \\ 1, VM\ i \text{ is } selected\ as\ reserved\ VM \end{cases} \quad (9)$$

After transformation, we can found that it is a classical 0/1 Knapsack problem. We treat the CPU threshold of overloaded physical node as the capacity of the backpack, the current CPU usage of the VM as the volume of the item, and the current memory usage of the VM as the value of the item.

Assume that overloaded physical node k has n VMs running in it, *M(i,j)* represents the total minimum memory of the first $i(1 \le i \le n)$ VMs running in physical node with CPU threshold $j(1 \le j \le upperThreshold \times C_{k\_total})$, we can obtain the following state transition equation:

$$M(i,j) = \begin{cases} 0, & i = 0 \\ 0, & j = 0 \\ M(i-1,j), & j < c_i \\ max\{M(i-1,j), M(i-1,j-m_i)+m_i\}, & j > c_i \end{cases} \quad (10)$$

When there are no VM running on physical node k ($i=0$), or the physical node k can't supply any CPU capacity for VM ($j=0$), *M(i,j)* equals 0, there are no VMs need to migrate. When the CPU resource required by VM i is greater than the current CPU threshold(jMIPS) on physical node $k(j<c_i)$, VM i can't run on physical node k, then *M(i,j)* equals to *M(i-1,j)*, which means the minimum total memory of the first i VMs reserved in physical node k with CPU threshold j is equivalent to minimum total memory of the first *i-1* VMs. On the other hand, if the CPU resource requirement of VM i can be satisfied by physical node $k(j>c_i)$, there are two cases: a) VM i is selected as a reserved VM; b) VM i has not been selected as reserved VM, we should choose the greater one as *M(i,j)*.

Dynamic programming algorithm can be implemented in a bottom-up or top-down manner. Since the top-down implementation may requires a relatively large system stack space and causes stack overflow, in this paper we use a bottom-up style to implement the algorithm.

Algorithm: dynamic programming selects VMs to migrate from overloaded host
Input: overloaded host
Output: migration VM list

```
1:  vmReserveList ← NULL
2:  get vmList from host
3:  vmNum ← vmList.size()
4:  thresholdCapacity ← upperThreshold× hostCpuTotal
5:  define array M[vmNum+1,thresholdCapacity+1]
6:  M[i,0] ← 0
7:  M[0,j] ← 0
8:  for i ← 1 to vmNum
9:      vmCpu ← vmList[i].getCurrentUsedCpu()
10:     vmMem ← vmList[i].getCurrentUsedMemory()
11:     for j ← 1 to thresholdCapacity
12:         if j < vmCpu then
13:             M[i,j] ← M[i-1,j]
```

```
14:          else
15:              M[i,j] ← max{M[i-1,j],M[i-1,j-vmCpu]+vmMem}
16:          end if
17:      end for
18:  end for
19:  i ← vmNum, j ← thresholdCapacity
20:  while i > 0 and j > 0 do
21:      vmCpu ← vmList[i].getCurrentUsedCpu()
22:      vmMem ← vmList[i].getCurrentUsedMemory()
23:      if j < vmCpu then
24:          i ← i-1
25:          continue
26:      end if
27:      if M[i,j] equals M[i-1,j-vmCpu]+vmMem then
28:          add vmList[i] to vmReserveList
29:          i ← i-1, j ← j-vmCpu
30:      else
31:          i ← i-1
32:      end if
33:  end while
33:  vmMigrationList ← vmList.removeAll(vmReserveList)
34:  return vmMigrationList
```

Line 8~18 fills out array *M[i,j]* , line 20~33 selects the migrated VMs. The time complexity of the algorithm is *O(vmNum×thresholdCapacity).*

IV. Experiment and Results Analysis

We choose the CloudSim[21] as a simulation platform, as it is a modern simulation framework aimed at Cloud computing environments.

A. Experiment configuration

In our experiment we create a data center with 800 physical nodes, the physical node type is *HP ProLiant ML110 G4* or *HP ProLiant ML110 G5,* and we use the corresponding power model as our power model, we create four types of VMs, the number of VMs equals to the task number of each workload data, the configuration of each physical node is shown in Table 1, the configuration of each VM is shown in Table 2. We test CPU utilization *upperThreshold* equals to 50%, 55%, 60%, 65%, 70%, 75%, 80%, 85%, 90%, 95%, 100% respectively, and comparing the energy consumption, numbers of VM migration, SLA violation rate, VM selection mean time between greedy and dynamic programming.

Table 1 Configuration of each physical node

Parameter	Value
Number of PE	2
Performance of each PE(MIPS)	1860 or 2660
RAM size(GB)	4
Bandwidth(Gbit/s)	1
Storage size(TB)	1

Table 2 Configuration of each VM

Parameter	Value
Number of PE	1
Performance of each PE(MIPS)	2500 or 2000 or 1000 or 500
RAM size(MB)	870 or 1740 or 613
Bandwidth(Mbit/s)	100
Storage size(GB)	2.5

B. Experiment results

We show the experiment results in Fig.2 to Fig.6. Here we define the SLA violation rate as: for some time period, if the VMs' total demand of CPU resource exceeds the CPU capacity of the physical node, then we say that the physical node is in SLA violation status, the percentage of violation time in total run time is called SLA violation rate.

First we use five different workload data from CloudSim to generate the data center's workload. The CPU's utilization threshold is set to 0.8, and the workload of data center remains the same for the same workload data during running. The energy consumption of the five tests is shown in Fig.2. We can see that the energy consumption based on dynamic programming is less than that based on greedy, regardless of the workload.

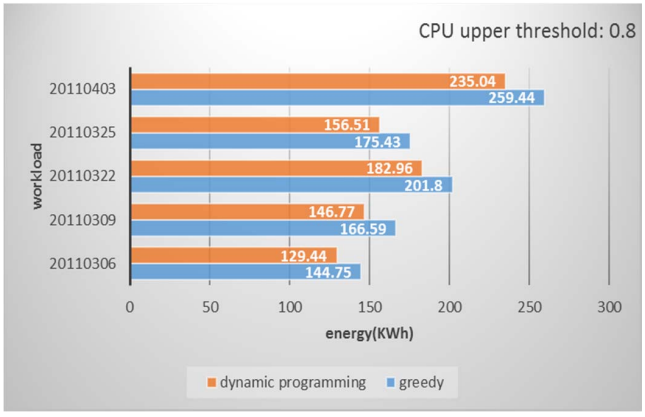

Fig.2. Energy consumption for different workload

Fig.3 show that the energy consumption based on dynamic programming is less than the energy consumption based on the greedy algorithm, regardless of the CPU usage threshold. It is proved that the VM selection algorithm based on dynamic programming is superior to the VM selection based on greedy algorithm in energy-saving.

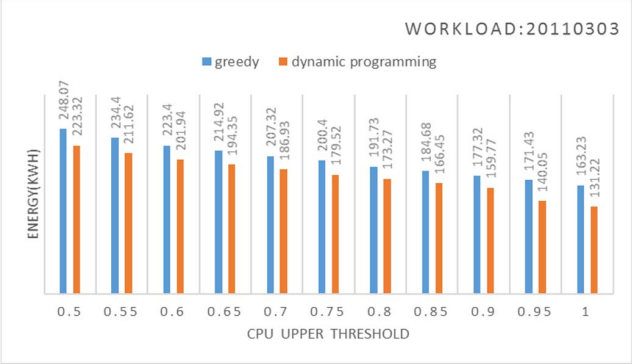

Fig.3. energy consumption for different CPU upper threshold

It can be seen from Fig.4 that the dynamic programming based algorithm has an obvious reduction in the number of VM migrations compared with the greedy algorithm, which can also be explained from Fig.1. The greedy algorithm will select the smallest VM to migrate multiple times until the physical machine's CPU utilization is less than the threshold, while dynamic programming method may only need to choose one VM to lower CPU utilization and to minimize memory at the same time.

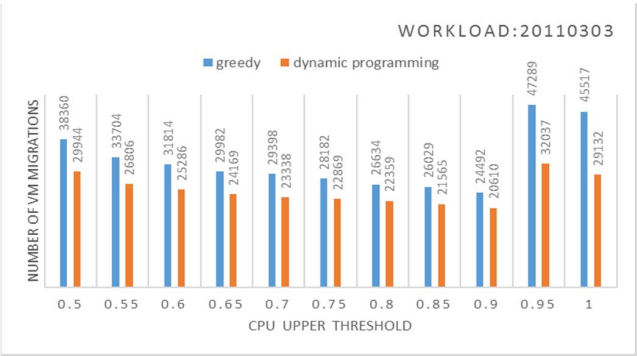

Fig.4. Number of VM migrations for different CPU upper threshold

From Fig.5, we can find that the total SLA violation of the algorithm is greater than that of the greedy algorithm. This is because the physical node's CPU utilization is lower after VM migration based on greedy than dynamic programming

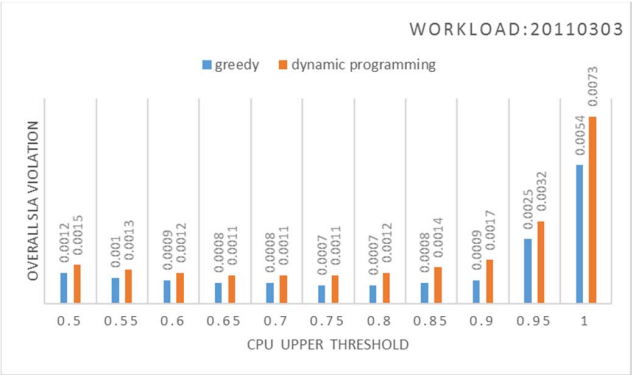

Fig.5. Overall SLA violations for different CPU upper threshold

It can be seen from Fig.6 that the VM selection mean time of the algorithm is significantly higher than that of the greedy algorithm, this is because the dynamic programming algorithm consumes pseudo polynomial time to solve the 0/1 knapsack problem, while the time complexity of the greedy algorithm is a polynomial function of the input size. However, no matter it is greedy algorithm or dynamic programming algorithm, it is millisecond-level time complexity, because in the actual situation, the input size is generally not large, so the time complexity does not matter so much here.

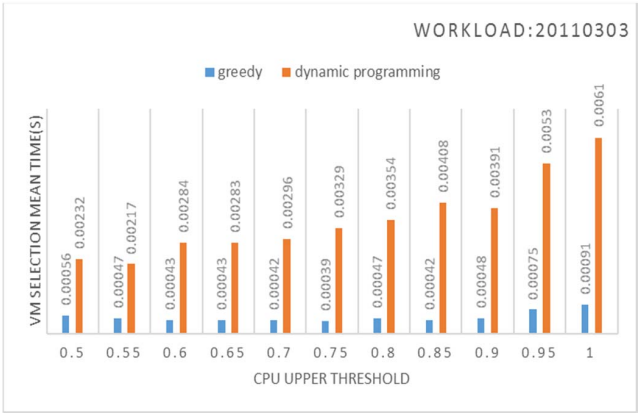

Fig.6. VM selection mean time for different CPU upper threshold

V. SUMMARY AND OUTLOOK

So far, data center energy consumption is still a very hot topic. High energy consumption brings huge operating costs not only to cloud service providers, but also has a bad impact on the global climate. In this paper, the dynamic programming method is used to optimize the selection of VMs to be migrated. The experiment results show that our algorithm achieves good energy-saving effect.

As for future work, so we will continue studying the scheduling of tasks and VM placement in future research.

ACKNOWLEDGMENT

This research is supported by the Shanghai Municipal Natural Science Foundation (14ZR1412400), the Innovation Program of Shanghai Municipal Education Commission, the Natural Science Foundation of China under Grant No.91118008, the Shanghai Knowledge Service Platform Project (No.ZF1213). Corresponding author is Chao Peng.

VI. REFERENCES

[1] NIST Cloud Computing Program, http://www.nist.gov/itl/cloud/

[2] Data-Center-Issue-Paper-final826, http://anthesisgroup.com/wpc-Ontent/uploads/2014/08/Data-Center-Issue-Paper-final826.pdf

[3] Lv T. Deep analysis and outlook based on China Data Center in our country[J]. The World of Power Supply, 2012, 12:6-8.

[4] Zeng Y. Comprehensive Review of Server Power Saving and Energy Efficiency Evaluation Technologies[J]. Information Technology & Standardization, 2008.

[5] Wang J, Feng L, Xue WW. A Survey of Energy Efficiency in Computer Servers[J]. Computer Engineering & Software, 2011.

[6] Gu LJ, Zhou FQ, Meng H. Research on national energy consumption and efficiency of data center[J]. Energy of China, 2010, 11:42-45+29.

[7] Ye KJ. Power Management of Virtualized Cloud Computing Platform[J]. Chinese Journal of Computers, 2012, 35(6).

[8] Ruan SL, Caiwu LU. Global Scheduling Method for Energy Consumption Optimization for Virtualized Data Center[J]. Computer Engineering, 2013.

[9] Belogazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing[J]. Future Generation Computer Systems,2012, 28(5):755-768.

[10] Tan YM, Zeng GS, Wang W. Policy of Energy Optimal Management for Cloud Computing Platform with Stochastic Tasks[J]. Journal of Software, 2012, 23(2):266-278.

[11] CloudSim Project, http://www.cloudbus.org/cloudsim/

[12] Cheng CL, Yu P, Zhang DY. Energy saving resource scheduling algorithm in cloud environment[J]. System Engineering & Electronics, 2013, 35(11):2416-2423.

[13] Luo L, Wu WJ, Zhang F. Energy Modeling Based on Cloud Data Center[J]. Journal of Software, 2014,07:1371-1387.

[14] OpenStack Project, http://www.openstack.org/

[15] Telemetry Project, https://wiki.openstack.org/wiki/Telemetry

[16] Dongmyoung B, Bunchul L. Analysis of telemetering service in OpenStack[C]// International Conference on Information and Communication Technology Convergence. IEEE, 2015.

[17] Rossigneux F, Lefevre L, Gelas JP, et al. Ageneric and Extensible Framework for Monitoring Energy Consumption of OpenStack Clouds[C]// Big Data and Cloud Computing (BdClo-d), 2014 IEEE Fourth International Conference on. IEEE, 2014:696-702.

[18] Kwapi Project, http://kwapi.readthedocs.io/en/latest/

[19] Dhanoa I S, Khurmi S S. Analyzing energy consumption during VM live migration[C]// International Conference on Computing, Communication & Automation. IEEE, 2015.

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

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

[22] Qiu M, Ming Z, Li J, et al. Phase-Change Memory Optimization for Green Cloud with Genetic Algorithm[J]. Computers IEEE Transactions on, 2015, 64(12):1-1.

[23] Gai K, Qiu M, Zhao H, et al. Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing[J]. Journal of Network & Computer Applications, 2016, 59(C):46-54.