2013 International Conference on Cloud Computing and Big Data

Research of Scheduling Strategy on OpenStack

Bin Hu
College of Engineering
Bo Hai University
Jinzhou,China
hubinustb@gmail.com

Hong Yu*
School of Computer and Communication Engineering
University of Science and Technology Beijing
Beijing,China
yuhong@ustb.edu.cn

***Abstract*—Cloud computing can provide Virtual Machine (VM) computing resources to meet the growing computational demands. Efficient and flexible resource management is a critical issue in cloud computing context. We present a scheduling study on OpenStack VMs scheduler using CloudSim. We focus it on from a more practical viewpoint, the simulation experiments indicate that each of the different scheduling policies suited to different applications in Cloud computing.**

Keywords—Cloud computing; PSO; GA;VM scheduling; OpenStack

I. INTRODUCTION

Cloud computing architectures have gained more and more attention in recent years and have got some solutions to improve the efficient of infrastructures. These solutions offer more effective computational resources to service providers with less cost. As a result, the Green Computing is put forward and the Cloud service providers are looking for efficient techniques to reduce the total power consumption of data centers. Cloud computing and IaaS in particular has adopted hardware virtualization as a key enabler for efficient resource usage. Vendors like Rackspace and HP base their cloud offerings on open source software such as OpenStack attempt to reduce execution time of multiple Virtual Machines (VMs) on physical host for less power consumption [1]. How to achieve an efficient, flexible and dynamic resource management at the IaaS is one of the most important key problems on IaaS Cloud platform.

On cloud computing platform, sharing virtual machine (VM) computing resources are provided as service by needs. Therefore, how to schedule tasks to the remote sharing VMs properly have become the issues to be better solved.

The remaining part of this paper is organized as follows. In Section 2, we present a succinct overview of the related work. We present an overview of the software architecture of OpenStack in Section 3. We present in Section 4 its scheduler of OpenStack Compute. We detail in Section 5 our methodology. In Section 6 we present and interpret the main results of the experiments. Finally, we conclude our paper in Section 8.

II. CORRELATION RESEARCH

There have been a lot of algorithms put forward for task scheduling in cloud computing fields. In [2] , it made the island parallel model and the multi-start parallel model with a new parallel bi-objective hybrid Genetic Algorithm to solve the minimum makespan and energy consumption problem. In [3] , it proposed Genetic Algorithm presented in a task whole accomplish time and average accomplish time double fitness Genetic Algorithm. In [4], it gave the queuing model and system cost function strategy and algorithm to get the approximate optimistic value of service for each job in the corresponding no-preemptive priority queuing model. In [5], it presented a resource-scheduling algorithm based on dynamic load balance. The algorithm, just as Greedy algorithm, selected the "better" node to complete the task to improve the efficiency of cloud computing and minimize the average response time of tasks. In [6], it presented the dynamic multi-stage resource pool of combining the resources scheduling strategy, which can effectively reduce the resources free time and improves the utilization rate of resources. In [7], it provided the cost and grouped tasks of the algorithm. In [8], it put forward an integrated approach that adopted three-layered resource controllers with different analytic techniques with the feedback control theory, statistical machine learning and system identification. In [9], it presented the Genetic algorithm to address grid scheduling. Finally, OpenStack is attracting an interest in the cloud computing research community. The authors of [10] present a detailed comparison of features of OpenStack with another Cloud platform, OpenNebula. In [11] the authors propose a survey of Cloud offerings in term of IaaS, including OpenStack.

III. OPENSTACK ARCHITECTURE

OpenStack is cloud computing technologists producing the ubiquitous open source cloud computing platform for public and private clouds. The software consists of several independently developed software components, organized into a separate subproject for each independent service, providing a wide range of functionality needed to build an IaaS cloud. It is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through well-defined Application Programming-Interfaces (APIs) exposed as RESTful web services. This most notably include functionality for the provisioning of compute, storage and network resources. The project rests on the following principles of openness; open design, open development and open community as stated on the OpenStack website.

This work was supported in part by the Program for Liaoning Excellent Talents in University (Grant No. LJQ2012095).

The OpenStack cloud operating system enables enterprises and service providers to offer on-demand computing resources, by provisioning and managing large networks of virtual machines[12]. Compute resources are accessible via APIs for developers building cloud applications and via web interfaces for administrators and users. The compute architecture is designed to scale horizontally on standard hardware, enabling the cloud economics companies have come to expect.

OpenStack is architected to provide flexibility as you design your cloud, with no proprietary hardware or software requirements and the ability to integrate with legacy systems and third party technologies[13]. It is designed to manage and automate pools of compute resources and can work with widely available virtualization technologies, as well as bare metal and high-performance computing (HPC) configurations. Administrators often deploy OpenStack Compute using one of multiple supported hypervisors in a virtualized environment. KVM and XenServer are popular choices for hypervisor technology and recommended for most use cases.

IV. THE VM ALLOCATION MODEL IN OPENSTACK

The scheduler's role in OpenStack compute is to direct incoming instance requests to suitable compute nodes that in turn will serve the requests handed to them by launching and making available new virtual machines sized according to the specifications of those requests. The scheduler uses a pluggable design, which let it load one of a set of drivers that does placements according to some scheduling policy. The scheduler makes use of host state information, which is sent to it periodically by the compute nodes as the basis of any scheduling decisions it makes. Finally, complete content and organizational editing before formatting [14].

In general, the compute tasks from different cloud users are relatively independent. We consider there are *n* independent tasks, as T_1,T_2, …,T_n. And there are m VMs as VM_1, VM_2,…,VM_m. The scheduler sends a task T_i to the queue of a VM by the scheduling decisions it makes periodically.

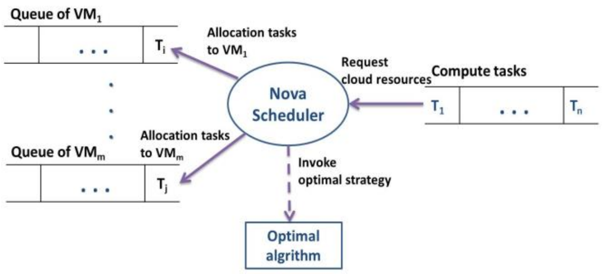

Fig. 1. Openstack scheduler model

OpenStack Compute includes three types of Scheduler: Filter, Chance and Simple [13].

When a request is made to launch a new VM, the nova-compute service contacts the nova-scheduler to request placement of the new instance. The nova-scheduler uses the scheduler, by default the Filter Scheduler, named in the nova.conf file to determine that placement. First, a filtering process is used to determine which hosts are eligible for consideration and an eligible host list is created and then a second algorithm, Costs and Weights, is applied against the list to determine which compute node is optimal for fulfilling the request.

By default, there are three filters that are used:

•The AvailabilityZoneFilter filters out hosts that do not belong to the Availability Zone specified when a new VM launch request is made via the Horizon dashboard or from the nova CLI client.

•The RamFilter ensures that only nodes with sufficient RAM make it on to the eligible host list. If the RamFilter is not used, the nova-scheduler may over-provision a node with insufficient RAM resources. By default, the filter is set to allow over commitment on RAM by 50 percent, i.e. the scheduler will allow a VM requiring 1.2 GB of RAM to be launched on a node with only 1 GB of available RAM. This setting is configurable by changing the "ram_allocation_ratio =1.2" setting in nova.conf.

•The ComputeFilter filters out any nodes that do not have sufficient capabilities to launch the requested VM as it corresponds to the requested instance type. It also filters out any nodes that cannot support properties defined by the image that is being used to launch the VM. These image properties include architecture, hypervisor and VM mode.

Next, the Filter Scheduler takes the hosts that remain after the filters have been applied and applies one or more cost function to each host to get numerical scores for each host. Each cost score is multiplied by a weighting constant specified in the nova.conf config file. The weighting constant configuration option is the name of the cost function, with the "_weight" string appended.

The Chance Scheduler chooses randomly an available node regardless of its characteristics while the Simple scheduler tries to find an available node with the least load.

In general, OpenStack compute scheduler adopts the random scheduling strategy.

V. DESIGN OF SIMULATION

We experimented with the CloudSim toolkit to simulate the scheduling algorithms in the existing popular IaaS platforms [15]. CloudSim is a cloud computing emulator developed by the University of Melbourne, Australia [16]. Its primary objective is to cloud infrastructure (software, hardware, services), and for different applications and services scheduling and allocation model to quantify the performance of strategic and comparison to control the use of cloud computing resources purposes. CloudSim provides novel support for modeling and simulation of virtualized Cloud based data center environments such as dedicated management interfaces for VMs, memory, storage, and bandwidth. CloudSim manages the instantiation and execution of core entities (VMs, hosts, data centers, application) during the simulation period [17].

For a more fully tested scheduling algorithm efficiency, we choose the suitable scheduling algorithms. We simulate five kinds of VMs allocation algorithms under different situation such as the number of virtual machines, performance of virtual machine, the number of tasks and task load size [18].

A. The experimental method

On one hand, we evaluated several common scheduling algorithms in the last completion time and average time of whole tasks under different conditions. On the other hand, we compared the standard GA (Genetic Algorithm) with PSO (Particle Swarm Optimization) algorithm in the execution speed of the algorithm.

B. The experimental settings

In simulation experiments, we represent the computing ability of VMs and the load of tasks with *MIPS* [19]. There are some mainly parameters setting of CloudSim simulator as shown in TABLE Ⅰ. The computation workload of task is from 10000 to 50000 *MI* (Million Instruction).

TABLE I. PARAMETERS SETTING OF CLOUD SIMULATOR

Type	Parameters	Value
Datacenter	Number of Datacenters	1
	Number of Hosts	5
Virtual Machine (VM)	Total number of VMs	10
	Number of PE per VM	1
	VM memory(RAM)	2048(MB)
	Bandwidth	1000bit
	Type of Manager	Time shared
Cloud Task	Total number of tasks	30
	Load of tasks	10000-50000(MI)

The performance of normal VMs is shown as TABLE II. The load of normal tasks is shown as TABLE III. The performance of high VMs is three times of the normal value. The performance of low VMs is one third of the normal value. The load task is also the same.

TABLE II. THE PERFORMANCE OF NORMAL VMs (MPS)

278	289	132	286	278
119	286	289	320	189

TABLE III. THE LOAD OF NORMAL TASKS (MPS)

19365	49809	30218	44157	16754
18336	20045	31493	30727	31017
20034	30128	18901	20067	30028
20121	10032	30012	20028	30067
31017	20034	30128	18901	20067
21017	10034	30128	28901	30067

VI. ANALYSIS AND DISCUSSION

The two performance metrics used for the experiments are the last completion time and average completion time of compute tasks.

A. Different Performance of VMs and the Same Tasks

We first evaluate the effect of hardware aware VM on the different performance of VMs. For these experiments, we were able to gain up to 300% in performance of VMs compared to scheduling algorithms. As we can see from Fig.2 and Fig.3, we can draw some conclusions as follows.

First, when VMs have higher performance, the last and the average completion time would shorter in all of five algorithms. Second, when VMs have the poorer performance, the Greedy and PSO algorithm are slightly dominant. Compared with the random algorithm, the Greedy shorten the time in more than 43%. Therefore, we can draw, when the underlying device has better performance, what kind of scheduling methods is not very important. So when vendors have high-performance hardware infrastructure, the Cloud Compute scheduler can adopt the stochastic scheduling algorithm. Furthermore, when a server performance is poor, the last completion time of Random, FIFO and GA algorithm are nearly double to that of PSO and Greedy algorithm. When the server has very poor performance even broken, the FIFO and Random scheduling algorithm cannot perform the tasks, while the GA, Greedy and PSO algorithms can successfully complete the task allocation.

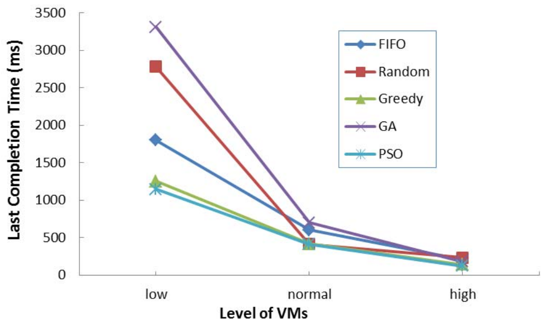

Fig. 2. Different Performance of VMs and the Same Tasks

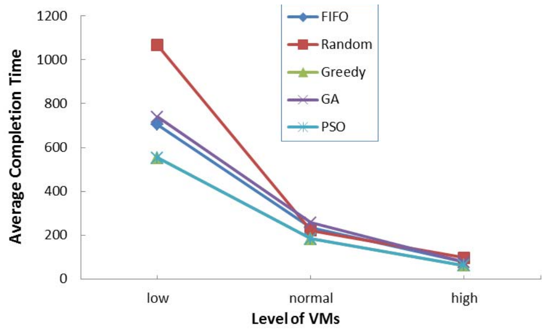

Fig. 3. Average Completion Time under Different Performance of VMs and the Same Tasks

B. Different Number of VMs and the Same Task

Similarly, we grow number of virtual machines in the same compute tasks conditions to evaluate the impact of number of VMs. The execution time of algorithms with different virtual devices is shown in Fig.4 and Fig.5.

With the fewer VM resources, the execution time of algorithms are longer. Of which GA algorithm has a maximum execution time, Greedy and PSO algorithm have the minimum completion time. At the same time, the Greedy and PSO algorithm have the closer efficiency. On the other hand, the more VMs resource, algorithms have more efficient. With virtual resources increase, the average and last completion time of GA and PSO algorithm are nearly stable, regardless of the number of Virtual Machines.

It can be concluded that when the volume and complexity of the task is relatively fixed, the Greedy and PSO algorithm have better efficient. When the number of VM resources is large, the effects of the other algorithms are also ideal.

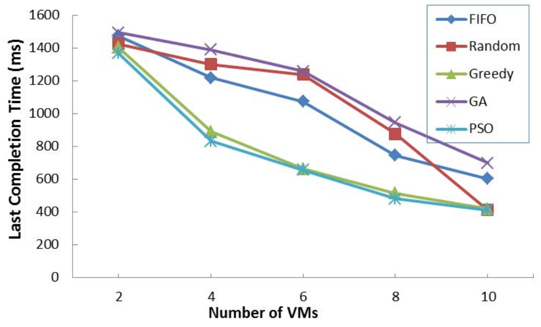

Fig. 4. Last Completion Time under Different Number of VMs and the Same Tasks

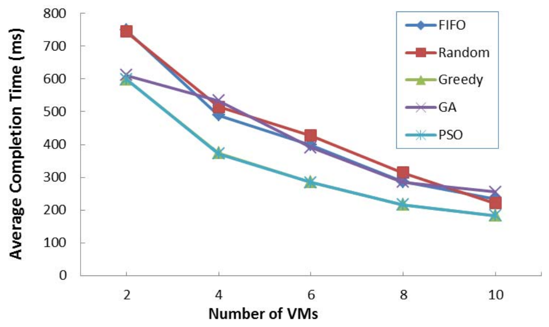

Fig. 5. Average Completion Time under Different Number of VMs and the Same Tasks

C. Different Number of Tasks and the Same VMs

In these experiments, we varied the number of Tasks from 10 to 30 with the same VMs for different circumstances. When the number of tasks increases in Cloud, the GA, PSO and Greedy algorithm have obvious advantages as seen in Fig.6 and Fig.7. Swarm intelligence optimization algorithm and the Greedy algorithm execution time is nearly half of other algorithms.

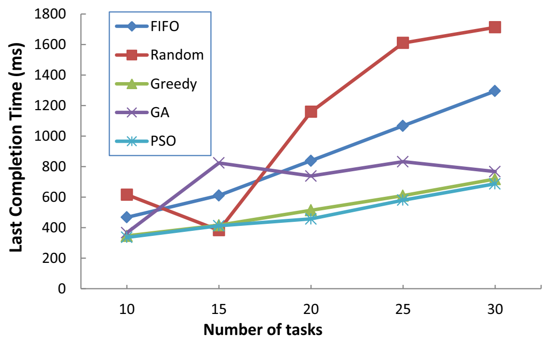

Fig. 6. Last Completion Time under Different Number of Tasks and the Same VMs

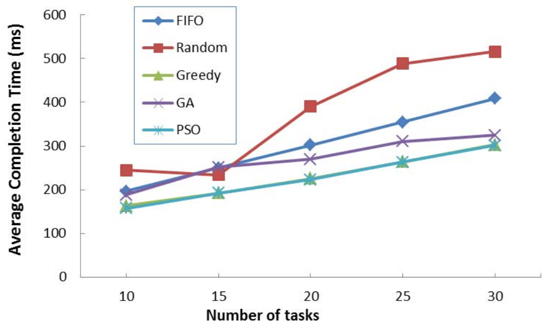

Fig. 7. Average Completion Time under Different Number of Tasks and the Same VMs

D. Comparison GA with PSO in Iterations

In PSO algorithm, the parameter ω used is recommended from Shi and Eberhart [20] with a linearly decreasing, which changes from 0.9 to 0.4. The acceleration constants *c1*, *c2* are both 2.0 which control the maximum step size that the particle can do. The r_1 and r_2 are two random functions in the range [0, 1] which the particles can search randomly. Initialization number of particles is 20. The maximum number of generations is set as 1000.

The standard particle swarm algorithm, which is the original form of particle swarm optimization developed, is very simple [21]. Let x_i denote the *i*th particle in the swarm and represented by n numbers of dimensions as $x_{id}=[x_{id1}, x_{id2},\ldots,x_{idn}]$ as in (2),where x_{idj} is the position value of the *i*th particle with respect to *j*th dimension(j=1,2,…,n). Let v_{id} is the velocity of particle i at iteration d *as in (1)*. It can defined as v_{id}= $[v_{id1},v_{id2},\ldots,v_{idn}]$, where v_{idj} is the velocity of particle i at iteration d with respect to the *j*th dimension. Parameter ω is to control the impact of the previous velocities on the current velocity. At the beginning of the search, it is used to enhance the global exploration while it is reduced for better local exploitation. The particles are manipulated according to the following equations.

$$v_{id} = \omega * v_{id} + c_1 * r_1 \left(p_{id} - x_{id}\right) + c_2 * r_2 * \left(p_{gd} - x_{id}\right) \qquad (1)$$

$$x_{id} = x_{id} + v_{id} \qquad (2)$$

In GA algorithm, the number of initial individuals is set as 1000. The crossover and mutation probabilities were taken as 0.2 and 0.005 respectively.

As the actual needs of the resource scheduling in cloud environment, our goal is to shorten the last accomplish time of the tasks, we also take into account the average accomplish time of the overall tasks. For this reason, we design the fitness function as shown in formula (3).

Objective function:

$$f(x) = Max\_Fitness - \max \sum_{i=1}^{k} (E_i + W_i) \qquad (3)$$

The *f(x)* is a fitness function about whole tasks. *Max_Fitness* is a maximum constant. E_i is denoted as the execution time of task *i* in VM, and the W_i is the waiting time before execution in a queue. From above, *F* is an overall function for cloud tasks.

In the virtual machine and the tasks are the same circumstances, we compared the GA with PSO algorithm to calculate the optimal time. As it can be seen from Fig.8, PSO iteration time is a little longer than the GA algorithm in the calculation of the optimal solution. Furthermore, we compare the results of PSO and GA algorithm from the standard deviation of the data shown in Fig.9. It can be seen that deviating from the arithmetic mean extent of PSO is less than that of GA.

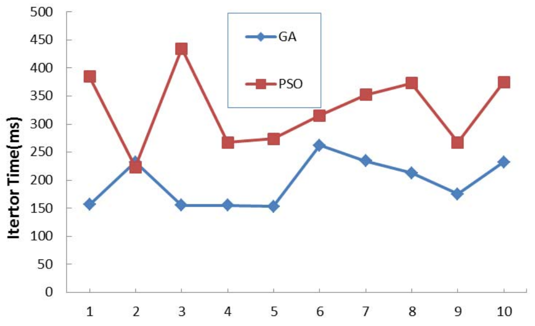

Fig. 8. Comparsion GA with PSO

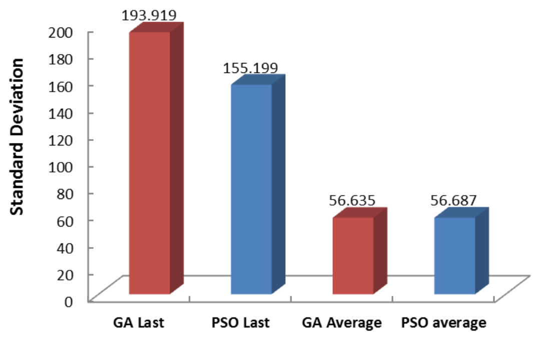

Fig. 9. Comparsion GA with PSO

In short, different algorithms are simulated and compared through different situations in a cloud environment. Greedy algorithm is the overall ability to execute the best scheduling policy. The number of tasks and processor performance with a linear relationship, and can deal with a great amount of tasks and extreme poor processor performance. In contrast, the randomized algorithm for better server performance cloud computing environment; FIFO algorithm is applicable to fewer cases of physical resources; PSO algorithm is close to the experimental results and Greedy, but cannot solve the case of poor resources scheduling.

Conclusions and future work

In this paper, we demonstrated the scheduling mechanism on Openstack. We evaluated the popular algorithms in different cloud situations with CloudSim simulator. In particular, we analyses that these algorithms suit to the corresponding situation to meet different cloud compute requirements. The experimental results illustrate that the Greedy, PSO and GA scheduling methods have the ability to complex Cloud compute situation. And the FIFO and stochastic strategies suit to the simple Cloud compute environment.

Recent studies have revealed that data centers consume unprecedented amount of electrical power, hence they incur massive capital expenditure for day-to-day operation and management. For instance, a data center hosted in a location where power cost is low and has less hostile weather conditions, would incur comparatively lesser expenditure in power bills. As future work, we are planning to research how to extend the scheduling optimize algorithms to reduce power consumption in Cloud data center.

References

[1] Wuhib F, Stadler R, Lindgren H. Dynamic resource allocation with management objectives—Implementation for an OpenStack cloud[C]//Network and Service Management (CNSM), 2012 8th International Conference on. IEEE, 2012: 309-315.

[2] Mezmaz M, Melab N, Kessaci Y, et al. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems[J]. Journal of Parallel and Distributed Computing, 2011, 71(11): 1497-1508.

[3] M. Mezmaz, Y. K. Melab, Y. C. Lee, E. G. Talbi, A. Y. Zomaya, and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," Journal of Parallel and Distributed Computing, vol. 71, pp. 1497-1580, 2011.

[4] L. Jian-feng and P. Jian, "Task scheduling algorithm based on improved genetic algorithm in cloud computing environment," Journal of computer applications, vol. 31, pp. 184-186, 2011.

[5] L. Li, "An Optimistic Differentiated Service Job Scheduling System for Cloud Computing Service Users and Providers," in Proceedings of the 2009 Third International Conference on Multimedia and Ubiquitous Engineering Qingdao,China, 2009, pp. 295-299.

[6] H. Chang and X. Tang, "A load-balance based resource-scheduling algorithm under cloud computing environment," in 2010 international workshop on cognitive-based interactive computing and web wisdom Shanghai, China, 2010, pp. 85-90.

[7] X. Song and L. Gao, "Job scheduling based on ant colony optimization in cloud computing," in 2011 International Conference on Computer Science and Service System (CSSS) Nanjing, China, 2011, pp. 3309-3312.

[8] Q. Li, Q. Hao, L. Xiao, and Z. Li, "An Integrated Approach to Automatic Management of Virtualized Resources in Cloud Environments," The Computer Journal, vol. 54, pp. 905-919, 2011.

[9] J. O. A. J. Abdulal W., "Genetic Algorithm for Grid Scheduling using Best Rank Power," in IEEE World Congress on Nature & Biologically Inspired Computing Coimbatore, 2009, pp. 181-186.

[10] P. Kumar and A. Verma, "Scheduling using improved genetic algorithm in cloud computing for independent tasks," in Proceedings of the International Conference on Advances in Computing, Communications and Informatics, Chennai, India, 2012, pp. 137-142.

[11] G. Xuan and R. Chen, Genetic Algorithms and Engineering optimization. Beijing: Tsinghua University Press, 2004.

[12] Litvinski O, Gherbi A. Experimental Evaluation of OpenStack Compute Scheduler[J]. Procedia Computer Science, 2013, 19: 116-123.

[13] Openstack,"http://www.openstack.org", Rackspace Cloud Computing, 2013.

[14] Wuhib F, Stadler R, Lindgren H. Dynamic resource allocation with management objectives—Implementation for an OpenStack cloud[C]//Network and Service Management (CNSM), 2012 8th International Conference on. IEEE, 2012: 309-315.

[15] Buyya R, Ranjan R, Calheiros R N. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit:

Challenges and opportunities[C]//High Performance Computing & Simulation, 2009. HPCS'09. International Conference on. IEEE, 2009: 1-11.

[16] Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments

[17] Wen X, Gu G, Li Q, et al. Comparison of open-source cloud management platforms: OpenStack and OpenNebula[C]//Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on. IEEE, 2012: 2457-2461.

[18] Mahjoub M, Mdhaffar A, Halima R B, et al. A comparative study of the current Cloud Computing technologies and offers[C]//Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on. IEEE, 2011: 131-134.

[19] Hu Y, Wong J, Iszlai G, et al. Resource provisioning for cloud computing[C]//Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research. IBM Corp., 2009: 101-111.

[20] Shi Y, Eberhart R. A modified particle swarm optimizer[C]//Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on. IEEE, 1998: 69-73.

[21] Pandey S, Wu L, Guru S M, et al. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments[C]//Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. IEEE, 2010: 400-407.

[22] Tayal S. Tasks scheduling optimization for the cloud computing systems[J]. International Journal of Advanced Engineering Sciences And Technologies (IJAEST), 2011, 5(2): 111-115.