Dynamic Creation of Virtual Machines in Cloud Computing Systems

Fei Luo
School of Information and Engineering
East China University of Science and Technology, Shanghai, China
luof@ecust.edu.cn

Isaac D. Scherson
Donald Bren School of Information and Computer Sciences
University of California, Irvine, USA.
isaac@ics.uci.edu

Joel Fuentes
Department of Computer Science and Information Technologies
Universidad del Bío-Bío. Chillán, Chile
jfuentes@ubiobio.cl

***Abstract*—The creation of *virtual machine*s (VMs) is one of the procedures of resource scheduling which is the key technology in cloud computing systems. Currently it is rarely studied independently, and it is always set as a static model where the number and the type of VMs are predefined before scheduling. However, with the static model, it is difficult to consider the overall optimization for the scheduling and it is not user-friendly because of users' imprecise requirements. Therefore, in this paper a dynamic model for VMs' creation is proposed. With this model, users only submit fuzzy requirements for computing resources, while the necessary number and type of VMs are calculated according to the current environment and dynamically created for further scheduling. The model is implemented in CloudSim, where the key technologies are presented. Analysis and experiments were carried out to verify the extension of the model in a more wide research area, such as resource scheduling for multi-objective optimization and scheduling tasks with priorities.**

***Keywords*—dynamic creation; cloud computing; multi-objective optimization; resource scheduling; CloudSim**

I. INTRODUCTION

Cloud Computing is based on virtualization technologies and offers *infrastructure as a service* (IaaS) to users with reliability and flexibility, which led to wide and rapid adoption of cloud computing systems in the industry and research institutions, such as Google App Engine [1], IBM blue Cloud [2], Amazon EC2 [3], OpenStack [4], and so on.

Resource scheduling is one of the key technologies in cloud computing systems [5], where the traditional boundary of computing resources is blurred by virtualization [6]. In both academia and industry, the problem of cloud resource scheduling is seen to be as hard as a Nondeterministic Polynomial (NP) optimization problem [7].

There exist many studies to realize efficient resource scheduling to meet one or more objectives like QoS, costs, energy conservation, load balance, task migration, bandwidth balance, utilization, and so on [8-12]. However, as far as we know, few scheduling studies focus on the creation of VMs, which is part of the resource scheduling. In those studies, the provision of VMs is normally set as a static model, where the number and type of VMs are predefined by the cloud users or providers.

However, the static creation model not only reduces such extended QoS standards as the user friendliness, but also is not conductive to overall resource optimization. Therefore, a dynamic model for VMs' creation is proposed in this paper, where the number and type of VMs are conducted based on the users' fuzzy requirements for tasks and the current environment of the cloud computing system.

Because conducting research for new ideas on live cloud environments is extremely difficult [5], the novel model is further implemented in the popular cloud simulator, CloudSim [13], named as DCloudSim. Building cloud simulation systems allows researchers the evaluation of new ideas in life-like scenarios and as a result such simulators could pave the way for the new research results allowing their wide-spread adoption. As CloudSim is the most popular simulation tool available for cloud computing environment, it is set as the host environment for the dynamic VM creation model.

Not only the properties of the dynamic VM creation model is analyzed, but also the experiments in DCloudSim are carried out. Results show that the model extends the research area into scheduling in applications with priorities and those with multi-objective optimization.

The main contribution can be depicted as two aspects. One is that a dynamic VM creation model is proposed, which can be applied to further improve the user-friendly QoS and achieve the overall system resource optimization. The other is that the idea is implemented in CloudSim, which can be further extend the research area of resource scheduling in Cloud computing systems.

The rest of the paper is organized as follows. Related work is summarized in Section II. The dynamic model is described in Section III, which is based on the scheduling in cloud computing systems and the properties are analyzed. The implementation of the model in CloudSim is presented in Section IV, where the key technologies are depicted, such as the extended implementation of objects and the communication process. Experiments with different scenarios in DCloudSim are further carried out in Section V. The final conclusions are drawn in Section VI.

II. RELATED WORK

978-1-5386-0610-0/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSEng.2017.13

A. Resource Scheduling in Cloud

Cloud computing provides both platforms and applications on demand through the Internet or intranet [14]. One key technology plays an important role in the cloud datacenter is resource scheduling. Scheduling of resources in a cloud environment is challenging due to dynamic resources spread over geographical area and on-demand scalability [15].

At present, there are various studies focusing on resource scheduling, where they are focused on load balancing [16], dynamic placement [17], energy conservation [18], resource classification [19] and forecast [20], etc.

Virtualization is widely used in cloud computing to fully utilize the resources and improve the performance. Various VM scheduling methods [21-23] have been proposed to dynamically allocate and consolidate the VMs in cloud computing environment. The allocation algorithms can be mainly divided into two types, allocating VMs onto *P*hysical *M*achines (PMs) and assigning PMs to VMs. The consolidation is typically achieved by VM migrations [24].

Those scheduling policies can work well under a stable system environment, for example, constant user requests and fixed infrastructure resources, where a static VM create model is predefined. However, as far as we know, currently few studies consider the dynamic VM creation model.

B. Cloud Simulators

There are many challenging issues which requires a lot of deep research in the cloud computing systems. Because of high expensiveness and maintenance requirements in the cloud, the research on all those issues in the real environment is extremely difficult and impractical. Therefore, many studies are conducted using simulators which can simulate a real cloud environment.

To date there are many cloud simulators which vary in features like availability of GUI, licensing, base programming language, extensibility, etc. [25]. With the help of those simulators, great progress has been made in many research areas in the cloud, such as data distribution and resource scheduling with cost and energy reduction.

In [5], the popular open-source simulators are selected for further study, such as CloudSim, iCanCloud, GreenCloud and CloudSched. Because their source codes are public in details, new algorithms can be devised and implemented from them. iCanCloud implements parallel experiments but does not consider energy consumption or VM migration. Therein, GreenCloud models detailed energy consumptions for different physical components, while CloudSched can model lifecycle of requests, and provide different metrics for load-balance, energy efficiency and utilization etc. [26]

CloudSim is amongst the most popular cloud simulators, which introduced the cloud simulation mostly focusing on computational intensive tasks, data interchanges between datacenters, internal network communications, and even the datacenter with containers. Furthermore, based on CloudSim, several third parties offer extensions on top of CloudSim. Therefore, to verify the effect of the dynamic VM creation model, CloudSim is set as the default platform for the implementation, where experiments are further carried out.

III. Model Description

A. Description of Cloud Resource Scheduling

The traditional resource scheduling is defined to find an "optimal" mapping C: $T \times R \rightarrow \mathfrak{M}^z$, where a batch of M required tasks T = $\{T_1, T_2, \ldots, T_M\}$ are assigned onto N available resources R = $\{R_1, R_2, \ldots, R_N\}$. In the cloud computing system the provision of the resources *R* is further virtualized process V: $VM \times PM \rightarrow \aleph^X$, where each R_i ($0 \le i < N$) corresponds to a VM, and the necessary VM = $\{VM_1, VM_2, \ldots, VM_N\}$ are virtualized from the *Q* physical machines PM=$\{PM_1, PM_2, \ldots, PM_Q\}$.

The purpose of the resource scheduling is to achieve the fitness of z given objectives F = $\{F_1, F_2, \ldots, F_z\}$, which are minimized (or maximized) either collectively in weighting or individually in a Pareto sense within a given time frame. It is carried out by optimizing the mapping $\mathfrak{M}^z$ and the virtualized process $\aleph^X$, where the properties of the scheduling process should be considered, including those of the tasks, the scheduling objectives, as well as the VMs.

First, a task T_j ($0 \le j < M$)is characterized by the required computing capability, the necessary storage space, the dependency on the other tasks, the deadline of the completion, and so on. Especially, in this paper the exclusiveness is set as one of the task's characteristics in the cloud, which means that the dispatched VM for the task T_j will not be shared by the others during the current task's execution. This property is valuable to the security of the cloud. Furthermore, these properties of a task and the number *M* of a batch of tasks are predefined by the cloud users.

Second, each scheduling objectives include QoS, costs, energy conservation, load balance, task migration, bandwidth balance, utilization, reliabilities, scalability, etc., as well as negotiations. In different scheduling problems, the number and properties of the objective are different with each other, while they are definitely predefined for the problem.

Finally, a VM is usually characterized by the number of CPU cores, required CPU capacity per core (e.g., in million instructions per second (MIPS)), required RAM size (e.g., in gigabytes), required disk size (e.g., in gigabytes), and even requirements concerning the communication (bandwidth, latency) between pairs of VMs or a VM and the customer. All of a VM's resource requirements can vary over time based on the type of application(s) running on the VM.

The requirements to VMs are initialized by the cloud users, while those VMs are further provided by the *cloud providers* (CPs) according to the requirements. Commonly, users are not skilled in submitting precise and proper CPU, bandwidth, or memory requirements, which means that they normally have fuzzy requirements based on the task attitude. In contrast, the CP should be well aware of the current resource requirements of the VMs to optimize resource usage.

In the traditional resource scheduling policies, the property of each VM and the number of VMs are predefined, and they are normally specified by users. Such a policy is looked as a static VM provision policy in this paper. Because of the unskillness of VM properties, it is not user-friendly. Furthermore, in a public cloud setting, it is common that the CP offers standardized types of VMs, while users' specification of VMs in the static VM provision policy may be not compatible with the CP's provision.

Therefore, the dynamic VM creation model is proposed in the next section. It means that based on users' fuzzy requirements to tasks, the type of the VMs and the number of each type VM are dynamically decided and created.

B. Dynamic Model for VM Creation

In the cloud with dynamic VM creation model, the tasks are assigned with priorities. Especially, if no priorities are assigned, all the tasks are looked as having the same priority. It is further assumed that several types of standardized VM configuration are offered in the cloud by the CP. Each configuration specifies the properties of the VM, which are described in Section III.A.

The main idea of the model is that a batch of tasks with highest priorities are selected for execution in one turn, and the process is continued until all the tasks has been finished. In each turn, the necessary VMs are dynamically evaluated and further created from the physical resources, and then those tasks are dispatched to the created VMs for execution. The dynamic VM creation model is shown in Figure 1.

First the tasks with highest priorities, *H(T)*, will be first scheduled, which will be extracted by the function *GetHighestPrioiry()* from the task set *T* submitted by users. The number *a* of tasks in *H(T)*, calculated by the function *num()*, will be further checked. If $a > 0$, the tasks will be scheduled as follows; or the process terminates.

Then the current physical resources *PM* will be matched with the requirements of the tasks in *H(T)* through the function *match()*. It results in a two-tuple set <*Y, C*>, where each element $< y_i, c_i >$ is the combination of the current VM configuration type y_i and the number c_i of the corresponding type of VMs. According to the set <*Y, C*>, c_i VMs with configuration type y_i will be created into the available resource set *RA*, where 0<i< T_N and T_N is the maximum configuration type number of VMs.

Afterwards, the tasks in *H(T)* will be dispatched into the VMs in *RA*, and they are further locally scheduled by the scheduler of the current VM in RA. It means the completion of one round of scheduling process.

To prepare another round of scheduling process, *H(T)* and *a* are further refreshed by the function of *GetHighestPrioiry()* and *num()*. If $a > 0$, the process will be launched again.

```
H(T) = GetHighestPriority(T);
a = num(H(T));
while( $a > 0$ ) {
    <Y, C> = match(PM, H(T));
    RA = VMs created according to <Y, C>;
    Dispatching: $H(T) \rightarrow RA$;
    $\{lc(VM_i) \mid VM_i \in RA\}$;
    T = T-H(T);
    H(T) = GetHighestPriority(T);
    a = num(H(T));
}
```

Fig. 1. Dynamic VM Creation Model

C. Analysis of the Model

With the dynamic creation model, the cloud computing system presents new characteristics, which are discussed in this section, such as the maximum number of VMs to be created, the makespan of the scheduling problem, and the user-friendly objectives, etc.

For simplicity, currently the cloud computing system discussed here only supports simple task which will be executed in a single VM, then the maximum number of dynamically created VMs is described as Theory 1.

Theory 1. The maximum number of the dynamically created VMs is no more than the number of tasks, e.g., $M \le N$.

Proof. If each task is VM-exclusive, it will be provided with one VM. Because the number of the tasks is N, the number of the dynamically created VMs is N too, e.g., M = N.

If at least two tasks can share resources and be clustered in one VM, then the necessary VMs is less than N, e.g., M < N.

In a word, $M \le N$. □

Then the makespan has the characteristic described in Theory 2.

Theory 2. The makespan of the submitted tasks is independent of the VM creation model.

Proof. For each task T_i, there are three time points related to the makespan of the submitted tasks, including its submitted time point t_{si}, its dispatched time point t_{di}, and its finished time point t_{fi}. Then the waiting time t_{wi}, the execution time t_{ei} and the makespan t_i of the task T_i is shown as formula (1)-(3). Furthermore, the makespan *t* for all the submitted tasks in the set *T* is the maximum value of the set $\{t_i \mid 0 < i < N\}$, as shown in formula (4).

$$t_{wi} = t_{di} - t_{si} \tag{1}$$

$$t_{ei} = t_{fi} - t_{di} \quad (2)$$

$$t_i = t_{ei} + t_{wi} = t_{fi} - t_{si} \quad (3)$$

$$t = \max\{t_i \mid 0 < i < N\} \quad (4)$$

Combining formula (1)—(4), the makespan t only depends on the task's submitted time t_{si} and the finished time t_{fi}, which is not related to the creation model of VMs. □

The user-friendly objective can also be embodied in the cloud with the dynamic VM creation model. On the one hand, users can just forward fuzzy computing requests for their tasks. Based on the model in Figure 1, users can just focus on the necessary computing resources of the tasks, while the concrete necessary VM configuration for the task is a decision made by the function *match()*.

On the other hand, users can pursue more cost-effective configurations of VMs. It can be explained as follows.

Normally, the unit cost (e.g., cost per hour) of each VM used by the task T_i is represented by c_i, then the total cost c for all the tasks is represented in formula (5). It is assumed that there are K types of VM configurations. Then the unit costs are in the set $Y = \{y_i \mid 0 < i < K, y_{i-1} \le y_i\}$, where the smaller subscript of y_i indicates the lower VM configuration. Therefore, each c_i is variable and belongs to Y. Then c is the summation of terms of variable c_i and t_{ei}. It means that c can be reduced by decreasing the value of t_{ei}, but also by reducing the value of c_i. Specially, when t_{ei} is constant, a more cost-effective configurations of VMs can be achieved with different cost of c_i.

$$c = \sum_{i=0}^{N-1} c_i t_{ei} \quad (5)$$

Furthermore, because the tasks can be dynamically scheduled based on their priorities, the workflow applications can be applied in the systems with the dynamic VM creation mechanism. In mean time, the objectives of the minimum makespan and the minimum cost for all the tasks can be pursued, which further indicates that multi-objective applications can be applied in the system.

IV. Implementation in CloudSim

The dynamic VM creation model is implemented in the current most popular cloud simulator, CloudSim, where the key technologies are presented in this section.

A. Flowchart of the Implementation

The current simulation lifecycle in CloudSim is shown in Figure 2, where the static VM creation model is utilized. The CloudSim package is first initialized, and then the necessary Datacenter and DatacenterBroker are created. Afterwards, the VMs are statically configured, including the number and concrete computer resources, such as the CPU cores, the memory and the storage capacity. The list of those VMs' configuration is submitted to the DatacenterBroker. Then the tasks, represented as cloudlets, are created and also submitted to the DatacenterBroker. Afterwards, the simulation is started, where the scheduling process is launched. The predefined VMs are booted and the submitted cloudlets are dispatched to them with further local scheduling. When all the cloudlets are processed, the simulation is terminated.

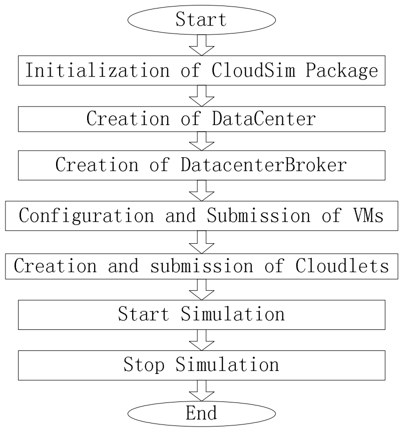
Start
Initialization of CloudSim Package
Creation of DataCenter
Creation of DatacenterBroker
Configuration and Submission of VMs
Creation and submission of Cloudlets
Start Simulation
Stop Simulation
End

Fig. 2. *Simulation Lifecycle in CLousSim*

With the static VM creation model, users should be familiar with the tasks and the concrete computing requirements. And then the requirements of VMs are statically configured and submitted to the DatacenterBroker by the user in the CloudSim simulation. Because the required VMs are normally created before the cloudlets' dispatching, the further dispatching process is just like a static mapping between the cloudlets and the VMs. So the cost-effective scheduling in formula (5) cannot be achieved and the applications with priorities cannot be simulated in CloudSim.

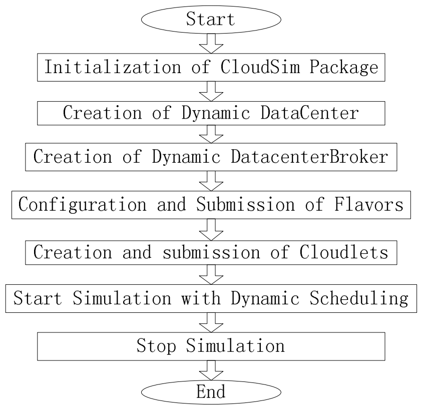
Start
Initialization of CloudSim Package
Creation of Dynamic DataCenter
Creation of Dynamic DatacenterBroker
Configuration and Submission of Flavors
Creation and submission of Cloudlets
Start Simulation with Dynamic Scheduling
Stop Simulation
End

Fig. 3. *Simulation Lifecycle in DCloudSim*

Therefore, CloudSim is implemented with the dynamic VM creation model described in section III, named as DCloudSim, and the simulation process in DCloudSim is presented in Figure 3.

In DCloudSim, users are not necessary to configure the concrete the VMs. In contrast, they submit fuzzy computing requirements for their Cloudlets, and then the scheduling system automatically matches their requirements. In the implementation, there are some key technologies presented in the subsequent sections, such as the introduced concepts of *priority*, *flavor*, *the dynamic datacenter* and the *dynamic datacenter broker*.

B. Introducion of new objects

First, each task, i.e. Cloudlet, can be assigned with a priority in DCloudSim. So the Cloudlet in DCloudSim is extended with a member *mPriority* and the corresponding attribute operations. Smaller the value of *mPriority* is, higher priority the Cloudlet has. Each Cloudlet has a default priority of 0, which means that without priority assignment, all the Cloudlets have the same priority.

Then in order to support the standardized configuration to VMs offered by CP, the object *Flavor* is introduced in DCloudSim. One *Flavor* represents a configuration of a type of VMs, where the properties of the VM include the identification of the type, the number of virtual CPUs, the memory and disk capacity, the unit price for the type of VMs, and so on. Flavors are provided when the CP creates the cloud system. So cost-effective combinations of VMs are dynamically created for users' fuzzy requests. The dynamical process is resolved by the Dynamic Datacenter and the Dynamic Datacenter Broker.

In CloudSim, the cloud resources are clustered in *Datacenter* class where VMs queries are processed and virtualized, and a class of *DatacentreBroker* is used to represent a broker which hide VM management, as creation and destruction of VMs. Currently they support the VM queries with the static VM creation model. To further support the queries with the dynamic VM creation model, a dynamic Datacenter, *DynamicDatacenter (DD)*, and a dynamic Datacenter Broker, *DynamicDatacenterBorker (DDR),* are devised. They will dynamically create VMs according to both the information of the current available resources in the datacenter and the requests from users.

C. Event Processing Flow

To support the dynamic VM creation model, *DDB* and *DD* collaborate with the event-based processing mechanism, as shown in Figure 4, which is depicted in the following.

After the simulation is started in the control flowchart in Figure 3, the system controller sends a message (RESOURCE_CHARACTERISTICS_REQUEST) to DDB for the system check. Then DDB and DB collaborate to call *processResourceCharacteristics()* to setup the system's resource characteristics. Afterwards, DDB tries to prepare VMs for the submitted cloudlets by sending VM_CREATE_REQ to DB. Subsequently the available resources are checked by *checkAvailableRsc*() in DB, which are returned to DDB. Then based on the current resources and the cloudlets' priorities, DDB makes a decision by *matchDecision* () about the information of the current batch of tasks to be scheduled, including the number of the tasks and the necessary VMs based on the current available resources. Afterwards the necessary VMs are dynamically requested by *createVMsInDatacenter*(), and the requests are further forwarded to DB through *VM_CREATE_ACK*. After the VM is created, the acknowledgement is returned to DDB, which schedules the tasks by *submitCloudlets*(). Here each cloudlet is dispatched to a prepared VM, where the cloudlet is scheduled by the local scheduler. Then the message of CLOUDLET_SUBMIT is forwarded to DB to check the execution process of the cloudlet.

After the completion of the cloudlet, the message CLOUDLET_RETURN is dispatched to DDB, where *processCloudletReturn*() is called, whose process is described as follows. (1) The VM for the current task is informed to be destroyed by the message VM_DESTROY. (2) The number of unscheduled tasks are checked. If the number is bigger than 0, then another turn of dynamic scheduling for the remaining tasks is launched by calling the function of *matchDecision*() in DDB. If all the tasks have been scheduled, i.e., the number of remaining tasks is 0, the simulation is finished. (3) END_OF_SIMULATION is forwarded to the system controller to stop the simulation.

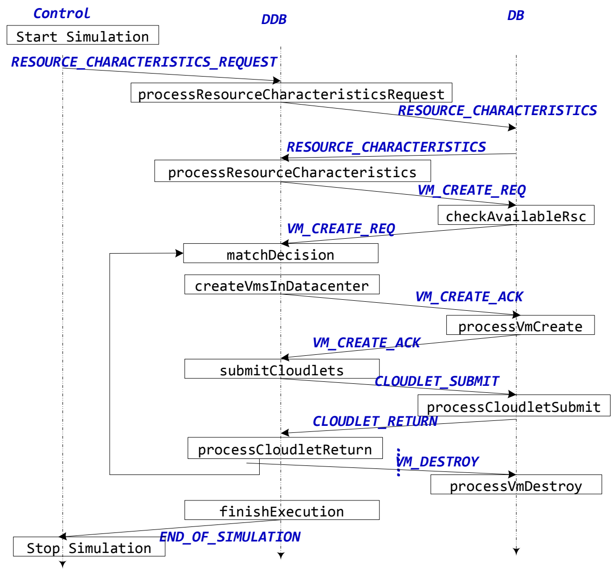

Fig. 4. *Event Processing Flow in DCloudSim*

V. Experiments and Evaluation

Based on the design from section IV, the dynamic VM creation model was implemented in DCloudSim. The experiments are presented in this section to verify the effect of the model.

A. Experimental Scenarios

As the description in section III.A, the scheduling in the cloud is a mapping process between the cloudlets and the VMs.

Therefore, the main factors affecting the scheduling are the properties of the application, the characteristics of the resources, as well as the relationship between the tasks and the resources.

As for the relationship between tasks and resources, the resource-exclusive characteristic is considered. Normally if the cloudlets can share VMs and they can be statically created, the applications can be simulated in the original CloudSim. Therefore, this paper only considers the simulations of those applications whose cloudlets are resource-exclusive, while the cloudlets sharing VMs can be clustered into a resource-exclusive VM.

The key consideration of the application's properties is whether the task has a priority and the characteristics of the resources, including the limitation of the resources and the number of the Flavors in the cloud. Then the scenarios to be simulated in the table 1 will be simulated.

Table 1. Scenarios in the experiments

	Cloudlets' Property	Resources' Characteristics
1	Cloudlets without priorities	Limited resources with one Flavor
2	Cloudlets without priorities	Unlimited resources with one Flavor
3	Cloudlets with priorities	Limited resources with one Flavor
4	Cloudlets with priorities	Unlimited resources with one Flavor
5	Cloudlets with priorities	Limited resources with multiple Flavors
6	Cloudlets with priorities	Unlimited resources with multiple Flavors
7	Cloudlets without priorities	Limited resources with multiple Flavors
8	Cloudlets without priorities	Unlimited resources with Multiple Flavors

B. Results

In order to facilitate the presentation of the experimental results, the scale of the experiments is small. Then the number of the cloudlets is set as 10. Accordingly, the ID of the Cloudlets is increased from 0 to 9, represented as ID_i ($0 \le i \le 9$). The number of necessary processing elements for each Cloudlet, $CPes_i$, is set as 1 or variable with its ID, as shown in formula (6), where "%" is a mod operation. For simple comparison in the experiments, the execution time of each Cloudlet is the same. When the cloudlet has variable priority, its value $CPRI_i$ is shown in formula (7).

$$CPes_i = ID_i \% 4 + 1 \tag{6}$$

$$CPRI_i = ID_i \% 4 \tag{7}$$

To match the scale of the application, the maximum number of the hosts in the cloud datacenter is set as 2. Then the hosts ID is H_i (i =0, 1). For the limitation characteristics of the resources, the number of physical cores in each host is also small, represented as h_i (i =0, 1). The host utilizes time-shared virtual machine monitor (i.e., *VmSchedulerTimeShared* in DCloudSim) in the datacenter, while the VM utilizes the time-shared scheduling policy for the Cloudlet.

To accommodate different cloudlets, the cloud provides different flavors. Especially, there are four flavors in the experiments, represented as F_i ($1 \le i \le 4$). The computing resource in each flavor is shown in formula (8), where RF_i is the computing resource in the flavor F_i, and RF_0 is the basic resource in the flavor, including the number of processing elements, the capacity of the RAM and the disk for the VM.

$$RF_i = RF_0 \times i \tag{8}$$

The results simulating different scenarios are shown in Figures 5-10. Therein, Figures 5-8 correspond to Scenario 1, 3, 5 and 7, respectively, where the computing resources are not sufficient for the current cloudlets. It is seen that the cloudlets are arranged to be executed according to the current resources or their priorities.

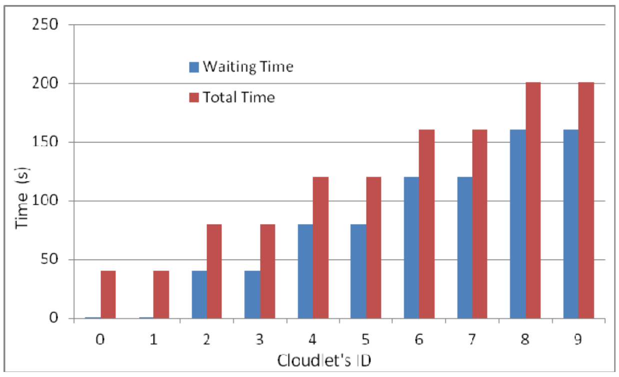

Fig. 5. Time cost when $CPes_i$ =1, h_1 =2, h_2 =0 and $CPRI_i$ =0.

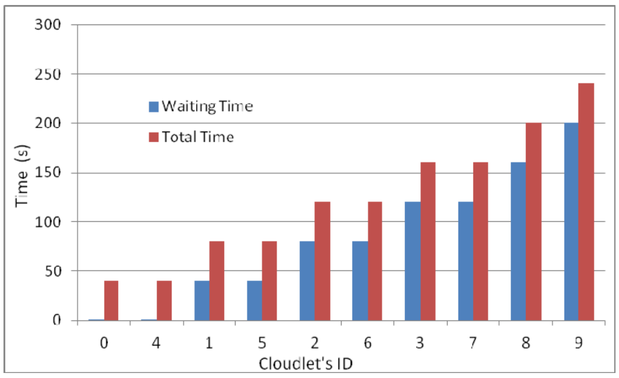

Fig. 6. Time cost when $CPes_i$ =1, h_1 =2, h_2 =0 and $CPRI_i$ is variable.

Figure 9 corresponds to Scenario 2 and 8, and Figure 10 corresponds to Scenario 4 and 6. It means that the cloudlets' priorities have a great influence to their execution process. When the cloudlets have no priorities, they can be executed in one round. When they have priorities, they are executed in several rounds, where those with the same priorities can be executed in the same round.

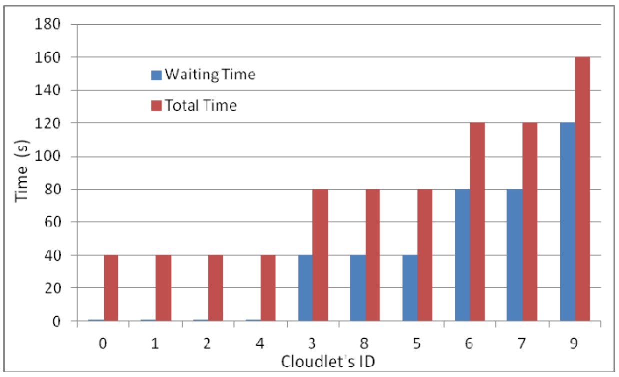

Fig. 7. Time cost when $CPes_i$ is variable, h_1 =4, h_2 =2 and $CPRI_i$ =0.

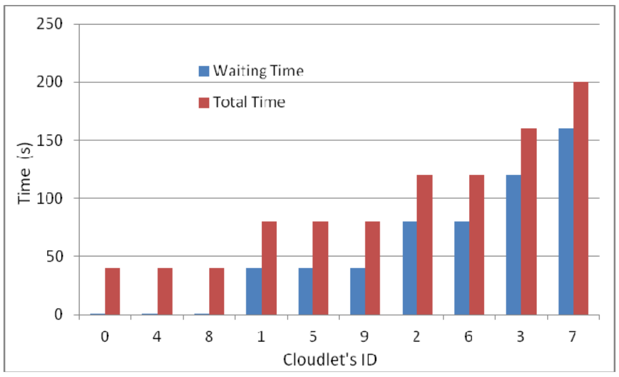

Fig. 8. Time cost when $CPes_i$ is variable, h_1 =4, h_2 =2 and $CPRI_i$ is variable.

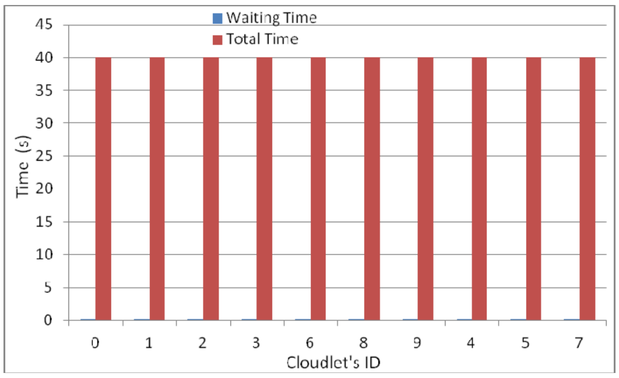

Fig. 9. Time cost when computer resources are sufficient and $CPRI_i$ =0.

The results also verify that with the dynamic VM creation model, the DCloudSim supports applications with priorities, and it can dynamically schedule the cloudlets based on the current resources. With multiple flavors in the cloud, VMs can be automatically configured for the cloudlet, which means that DCloudSim supports users' fuzzy requests for computer resources.

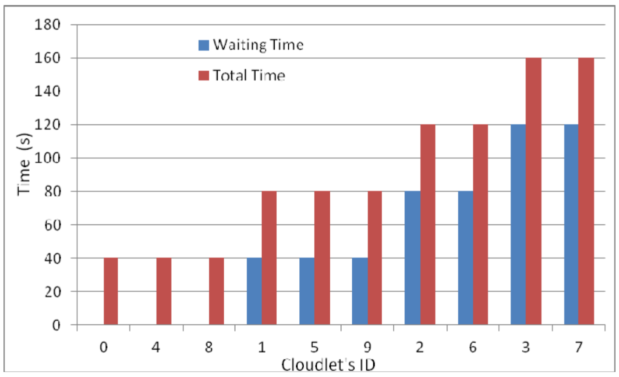

Fig. 10. Time cost when computer resources are sufficient and $CPRI_i$ is variable.

VI. CONCLUSIONS

As a key procedure of resource scheduling in the cloud computing system, the VM's creation model has been researched. Specially a dynamic VM creation model is devised, which extends the range of applications to those with priorities and those for overall optimization. It also provides users with fuzzy requests for computer resources. The model is implemented in CloudSim, named as DCloudSim, where the key technologies are studied, such as the introduction of new objects and the event processing flow. Experiments shows that new applications with priorities can be supported and adaptive configurations for applications can be automatically matched.

ACKNOWLEDGMENTS

This work was done while the author is on current leave at University of California, Irvine (UCI). It was partially supported by the Online Education Fund (pervasive education) of Online Education Research Center in Chinese Ministry of Education (No. 2017YB122), the Nature Science Fund of China (NSFC) (No. 61472139).

REFERENCES

[1] Google App Engine, 2013. http://code.google.com/intl/zh-CN/appengine/

[2] IBM blue cloud, 2013. http://www.ibm.com/grid/

[3] Amazon EC2, 2015. http://aws.amazon.com/ec2/

[4] Antonio Solano, Raquel Dormido, Natividad Duro and Juan Miguel Sánchez. A Self-Provisioning Mechanism in OpenStack for IoT Devices, Sensors, 2016, 16(8): 1306-1324

[5] Wenhong Tian, Minxian Xu, Aiguo Chen, Guozhong Li, Xinyang Wang, Yu Chen. Open-source simulators for Cloud computing: Comparative study and challenging issues, Simulation Modelling Practice and Theory, 2015, 58: 239-254

[6] Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li. Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Comput. Surv., 2015, 47(4): 1-33

[7] J. L. Xu, J. Tang, K. Kwiat, W. Y. Zhang, and G. L. Xue. 2013. Enhancing survivability in virtualized data centers: A service-aware approach. IEEE Journal on Selected Areas in Communications, 2013, 31(12): 2610–2619

[8] A. K. Bardsiri and S. M. Hashemi. A review of workflow scheduling in cloud computing environment. International Journal of Computer Science and Management Research, 2012, 1(3): 348–351

[9] E. Barrett, E. Howley, and J. Duggan. A learning architecture for scheduling workflow applications in the cloud. In Proceedings of the 9th IEEE European Conference on Web Services, 2012, pp. 83–90.

[10] S. Chaisiri, B. Lee, and D. Niyato. Optimization of resource provisioning cost in cloud computing. IEEE Transactions on Services Computing, 2012, 5 (2): 164–177

[11] Y. Chawla and M. Bhonsle. A study on scheduling methods in cloud computing. International Journal of Emerging Trends & Technology in Computer Science, 2012, 1 (3): 12–17

[12] N. Chen, W. N. Chen, Y. J. Gong, Z. H. Zhan, J. Zhang, Y. Li, and Y. S. Tan. An evolutionary algorithm with double-level archives for multiobjective optimization. IEEE Transactions on Cybernetics, 2014, 45 (9): 1851-1863

[13] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. a F. De Rose, and R. Buyya. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. - Pract. Exp., 2011, 41(1): 23–50

[14] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical report no. UCB/EECS-2009-28, University of California at Berkley, USA, February 10, 2009.

[15] Sukhpal Singh, Inderveer Chana1. Resource provisioning and scheduling in clouds: QoS perspective, J Supercomput, 2016, 72:926–960

[16] Pearce O, Gamblin T, de Supinski BR, Schulz M, Amato NM, Quantifying the effectiveness of load balance algorithms. In Proceedings of the 26th ACM international conference on Supercomputing, 2012, pp 185–194.

[17] Lucas Simarro JL, Moreno-Vozmediano R, Montero RS, Llorente IM. Dynamic placement of virtual machines for cost optimization in multi-cloud environments. In Proceedings of 2011 International Conference on High Performance Computing and Simulation (HPCS), 2011, pp 1–7.

[18] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener Comput Syst, 2012, 28(5):755–768

[19] Tang Z, Zhou J, Li K, Li R. MTSD: a task scheduling algorithm for MapReduce base on deadline constraints. In Proceedings of IEEE 26th International Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW), 2012

[20] Zhuo Tang, Yanqing Mo, Kenli Li, Keqin Li. Dynamic forecast scheduling algorithm for virtual machine placement in cloud computing environment, J Supercomput, 2014, 70:1279–1296

[21] Zhen Xiao, Weijia Song, Qi Chen. Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment, IEEE Transactions on Parallel and Distributed Systems, 2013, 24 (6): 1107-1117

[22] Sobir Bazarbayev, Matti Hiltune , Kaustubh Joshi, William H. Sanders, Richard Schlichting, Content-Based Scheduling of Virtual Machines (VMs) in the Cloud. In Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems, 2014, pp.93-101

[23] Inkwon Hwang, Massoud Pedram. Hierarchical Virtual Machine Consolidation in a Cloud Computing System. In Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, 2013, pp.196-203

[24] Youwei Ding, Xiaolin Qin, Liang Liu, Taochun Wang. Energy efficient scheduling of virtual machines in cloud with deadline constraint, Future Generation Computer Systems, 2015, 50: 62–74

[25] Pericherla S Suryateja. A Comparative Analysis of Cloud Simulators, I.J. Modern Education and Computer Science, 2016, 4: 64-71

[26] H. Marwaha and R. Singh. A Comprehensive Review of Cloud Computing Simulators, J. Inf. Sci. Comput. Technol., 2015, 4 (1): 281–286