A Toolkit for Modeling and Simulation of Real-Time Virtual Machine Allocation in a Cloud Data Center

Wenhong Tian, Yong Zhao, Minxian Xu, Yuanliang Zhong, and Xiashuang Sun

***Abstract*—Resource scheduling in infrastructure as a service (IaaS) is one of the keys for large-scale Cloud applications. Extensive research on all issues in real environment is extremely difficult because it requires developers to consider network infrastructure and the environment, which may be beyond the control. In addition, the network conditions cannot be predicted or controlled. Therefore, performance evaluation of workload models and Cloud provisioning algorithms in a repeatable manner under different configurations and requirements is difficult. There is still lack of tools that enable developers to compare different resource scheduling algorithms in IaaS regarding both computing servers and user workloads. To fill this gap in tools for evaluation and modeling of Cloud environments and applications, we propose CloudSched. CloudSched can help developers identify and explore appropriate solutions considering different resource scheduling algorithms. Unlike traditional scheduling algorithms considering only one factor such as CPU, which can cause hotspots or bottlenecks in many cases, CloudSched treats multidimensional resource such as CPU, memory and network bandwidth integrated for both physical machines and virtual machines (VMs) for different scheduling objectives (algorithms). In this paper, two existing simulation systems at application level for Cloud computing are studied, a novel lightweight simulation system is proposed for real-time VM scheduling in Cloud data centers, and results by applying the proposed simulation system are analyzed and discussed.**

***Note to Practitioners*—This paper was motivated by the problem of simulating scheduling algorithms in Cloud data centers to evaluate their performance for different metrics. Existing tools such as CloudSim [4] and CloudAnalyst [13], are based on SimJava [8] and GridSim [3], which treat a Cloud data center as a large resource pool and consider only application-level workloads, may not be suitable for IaaS simulation where each VM as a resource is requested and allocated. There is still lack of tools that enable developers to evaluate requirements of large-scale Cloud applications in terms of comparing different resource scheduling algorithms regarding geographic distribution of both computing servers and user workloads. To fill this gap in tools for evaluation and modeling of Cloud environments and applications, in this paper we propose CloudSched, for dynamic resource scheduling in Cloud datacenter. Real-time constraint of both VMs and PMs, which is often neglected in literature, is considered in this paper. The main contributions of this paper are: proposing a simulation system for modeling Cloud computing environments and performance evaluation of different resource scheduling policies and algorithms; focusing on simulation of scheduling in IaaS layer where related tools are still lack; designing and implementing a lightweight simulator combining real-time multidimensional resource information. CloudSched offers the following novel features: (i) modeling and simulation of large scale Cloud computing environments, including data centers, VMs and physical machines; (ii) providing a platform for modeling different resource scheduling policies and algorithms at IaaS layer for Clouds; and (iii) both graphical and textual outputs are supported.**

***Index Terms*—Cloud computing, data centers, dynamic and real-time resource scheduling, lightweight simulation system.**

Manuscript received April 08, 2013; accepted May 28, 2013. Date of publication July 12, 2013; date of current version December 31, 2014. This paper was recommended for publication by Associate Editor L. H. Lee and Editor H. Ding upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 61150110486.

W. Tian and Y. Zhao are with the School of Computer Science, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: tian_wenhong.AT.uestc.edu.cn).

M. Xu, Y. Zhong, and X. Sun are with the University of Electronic Science and Technology of China, Chengdu 610054, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2013.2266338

I. Introduction

CLOUD computing is developed based on various recent advancements in virtualization, Grid computing, Web computing, utility computing and related technologies. Cloud computing provides both platforms and applications on demand through the Internet or intranet [1]. Some of the key benefits of Cloud computing include the hiding and abstraction of complexity, virtualized resources and efficient use of distributed resources. Some examples of emerging Cloud computing platforms are Google App Engine [23], IBM blue Cloud [24], Amazon EC2 [21], and Microsoft Azure [25]. Cloud computing allows the sharing, allocation and aggregation of software, computational and storage network resources on demand. Cloud computing is still considered in its infancy as there are many challenging issues to be resolved [1], [2], [5]. Youseff *et al.* [17] establish a detailed ontology of dissecting Cloud into five main layers from top to down: Cloud application (SaaS), Cloud software environment (PaaS), Cloud software infrastructure (IaaS), software kernel and hardware (HaaS), and illustrates their interrelations as well as their interdependency on preceding technologies. In this paper, we focus on Infrastructure as a service (IaaS) in Cloud data centers. Cloud data center can be a distributed network in structure, which is composed of many computing nodes (such as servers), storage nodes, and network devices. Each node is formed by a series of resources such as CPU, memory, network bandwidth and so on. Each resource has its corresponding properties. There are many different types of resources for Cloud providers. This paper focuses on the IaaS. The definition and model defined by this paper are aimed to be general enough to be used by a variety of Cloud providers.

In a traditional data center, applications are tied to specific physical servers that are often overprovisioned to deal with

workload surges and unexpected failures [10]. Such configuration rigidity makes data centers expensive to maintain with wasted energy and floor space, low resource utilizations and significant management overheads. With virtualization technology, today's Cloud data centers become more flexible, secure and on-demand allocating.

One key technology plays an important role in Cloud datacenter is resource scheduling. One of the challenging scheduling problems in Cloud data center is to consider allocation and migration of reconfigurable virtual machines (VMs) and integrated features of hosting physical machines.

It is extremely difficult to research widely for all these problems in real Internet platform because the application developers can't control and process network environment. What is more, the network conditions cannot be predicted or controlled, but affect the quality evaluation of strategies. The research of dynamic and large-scale distributed environment can be achieved by building data center simulation system, which supports visualized modeling and simulation in large-scale applications in cloud infrastructure. Data center simulation system can describe the application workload statement, which includes user information, data center position, the amount of users and data centers, and the amount of resources in each data center. Using this information, data center simulation system generates response requests and allocates these requests to VMs. By using data center simulation system, application developers can evaluate suitable strategies such as distributing reasonable data center resources, selecting data center to match special requirements, reducing costs and so on.

A. Related Work

There are quite many research conducted in scheduling algorithms. Most of them are for traditional web servers or server farms. Beloglazov *et al.* [2] introduce heuristics algorithms and challenging issues for resource allocation in Cloud computing. Prodan *et al.* [20] model the workflow problem in Grid computing as an extension of the multiple-choice knapsack problem and propose a general bicriteria scheduling heuristic called dynamic constraint algorithm (DCA) based on dynamic programming. Zhu *et al.* [18] treat outpatient scheduling as a collaborative activity and create a qualification matrix to apply the group role assignment algorithm. Cao *et al.* [19] propose dynamic control of resource scheduling and allocation in data streaming to avoid resource shortage and overprovision. Buyya *et al.* introduce GridSim [3] toolkit for modeling and simulation of distributed resource management for grid computing. Dumitrescu and Foster [6] introduce GangSim tool for grid scheduling. Buyya *et al.* [4] introduce modeling and simulations of Cloud computing environments at application level, a few simple scheduling algorithms such as time-shared and space-shared are discussed and compared. CloudSim [4] is one of Cloud computing simulators, which has the following functions: supporting modeling large-scale cloud computing infrastructure, both in a single physical computing node and Java virtual machine data center; modeling for the data center, service agency, scheduling and distributing strategies; providing virtual engines, which is helpful to create and manage several independent and collaborative virtual services in a data center node; be able to switch flexibly between processing cores with space-sharing and time-sharing. CloudAnalyst [6] aims to achieve the optimal scheduling among user groups and data centers based on the current configuration. Both CloudSim and CloudAnalyst are based on SimJava [8] and GridSim [3], which treat a Cloud data center as a large resource pool and considers only application-level workloads, may not be suitable for Infrastructure as a service (IaaS) simulation where each VM as a resource is requested and allocated. Wood *et al.* [14] introduced techniques for VM migration and proposed some migration algorithms. Zhang [15] compared major load balance scheduling algorithms for traditional web servers. Singh *et al.* [10] proposed a novel load balancing algorithm called VectorDot for handling the hierarchical and multidimensional resource constraints by considering both servers and storage in Cloud computing.

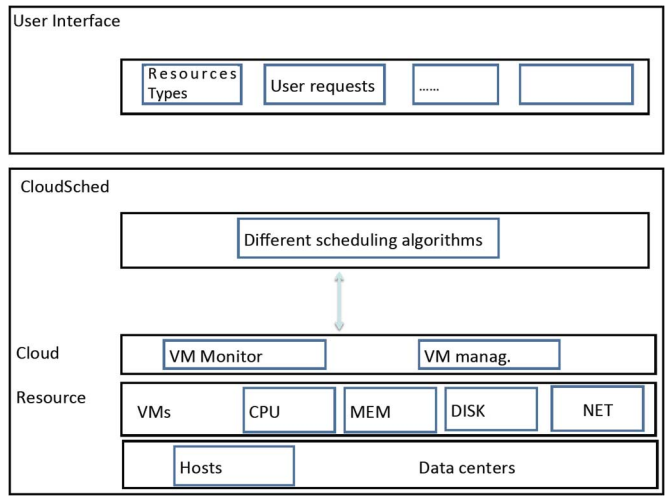

Fig. 1. A simplified layered architecture of CloudSched.

The organization of remaining parts of this paper is as follows. Section II introduces CloudSched architecture and its main features, Section III discusses performance measurement of different scheduling algorithms, Section IV presents design and implementation of CloudSched, Section V discusses the simulation results by comparing a few different scheduling algorithms, and finally a conclusion is provided in Section VI.

II. CloudSched Architecture and Main Features

The simplified layered architecture is shown in Fig. 1.

1) User Interface. At the top layer, there is an interface for a user to select resource and send requests, basically, a few types of VMs are preconfigured for a user to choose.
2) Core layer of Scheduling. Once user requests are initiated, they go to next level—CloudSched scheduling, which is responsible to choose appropriate data centers and physical machines based on user requests. CloudSched provides support for modeling and simulation of Cloud data centers, especially allocating VMs (consisting of CPU, memory, storage and bandwidth, etc.) to suitable physical machines. This layer can manage a large scale of Cloud data centers consisting of thousands of physical machines. Different scheduling algorithms can be applied in different data centers based on customers' characteristics.

3) Cloud Resource. At the lowest layer, there are Cloud resources which include physical machines and VMs, both of them consisting of certain amount of CPU, memory, storage, and bandwidth, etc.

Some other tools such as CloudSim and CloudAnalyst are based on existing simulation tools such as JavaSim and GridSim, which makes the simulation system very large and complicated. In view of these, CloudSched is lightweight design with focus on resource scheduling algorithms. The main features of CloudSched are the following.

- **Focusing on IaaS layer**. Unlike existing tools which focus on application (task) level such as CloudSim and CloudAnalyst, CloudSched focuses on scheduling VMs at IaaS layer, i.e., each request needs one or more VMs, while each request only occupies a portion of the total capacity of a VM in CloudSim and CloudAnalyst.
- **Providing an uniform view of all resources**. Similar to Amazon EC2 [21] real applications, CloudSched provides an uniform view of all physical and virtual resources so that both system management and user selections are simplified. We will explain this in details in the following section.
- **Lightweight design and scalability**. Comparing to other existing simulation tools such as CloudSim and CloudAnalyst which are built on GridSim, CloudSched is focusing on resource scheduling polices and algorithms. CloudSched can simulate tens of thousands of requests in a few minutes.
- **High extensibility**. Modular design is applied in CloudSched. Different resource scheduling policies and algorithms can be compared with each other for performance evaluation.
- **Easy to use and repeatable** . CloudSched enables users to set up simulation easily and quickly with easy to use graphical user interfaces and outputs. It can accept inputs from text files and output to text files. CloudSched can save simulation inputs and outputs so that modelers can repeat experiments. CloudSched ensures that repeated simulation yield identical results. Some GUIs are shown in Figs. 2 and 3 for illustrations.

A. Modeling Cloud Data Centers

The core hardware infrastructure related to the Cloud is modeled in the simulator by a data center component for handling VM requests. A data center is mainly composed by a set of hosts, which are responsible for managing VMs during their life cycles. A host is a component that represents a physical computing node in a Cloud: it is assigned a preconfigured processing capability (expressed in computing power in CPU units), memory, bandwidth, storage, and a scheduling policy for allocating processing cores to VMs. A VM can be represented in a similar way.

B. Modeling VM Allocation

With virtualization technologies, Cloud computing provides flexibility in resource allocation. For example, a PM with two processing cores can host two or more VMs on each core concurrently. Only if the total used amount of processing power by all VMs on a host is not more than available capacity in that host, VMs can be allocated. Taking the widely used example

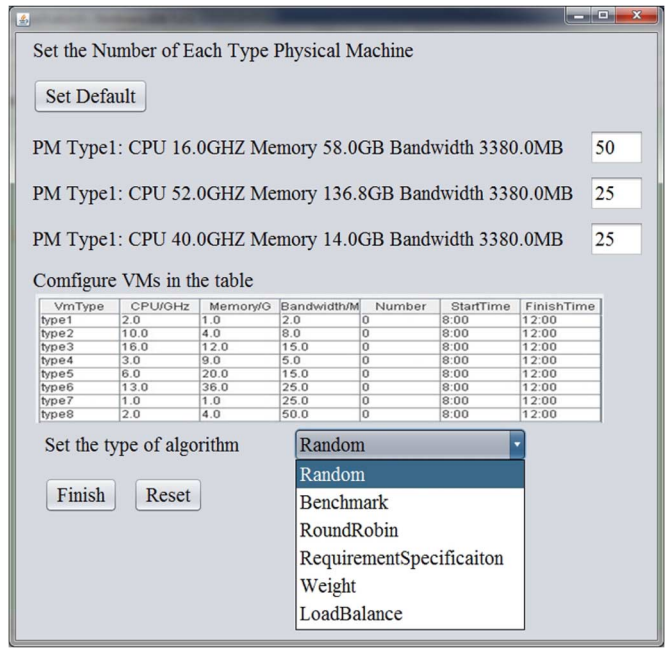

Fig. 2. Main interface of CloudSched (1).

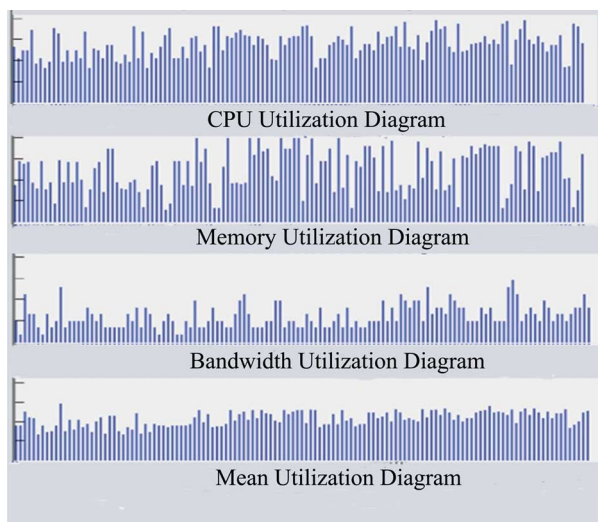

Fig. 3. Main interface of CloudSched (2).

of Amazon EC2 [21], we show that a uniform view of different types of VMs is possible. Table I shows eight types of VMs from Amazon EC2 online information. We can therefore form three types of different PMs (or PM pools) based on compute units. In a real Cloud data center, for example, a physical machine with 2 $\times$ 68.4 GB memory, 16 cores $\times$ 3.25 units, 2 $\times$ 1690 GB storage can be provided. In this or similar way, a uniform view of different types of VMs is possibly formed. This kind of classification provides a uniform view of virtualized resources for heterogeneous virtualization platforms, e.g., Xen, KVM, VMWare, etc., and brings great benefits for VM management and allocation. Customers only need selecting suitable types of VMs based on their requirements. There are eight types of VMs in EC2, as shown in Table I, where MEM is for memory with unit GB, CPU is normalized to unit (each CPU unit is equal to 1 GHz 2007 Intel Pentium processor [21]) and Sto is for hard disk storage

TABLE I
EIGHT TYPES OF VIRTUAL MACHINES (VMS) IN AMAZON EC2

MEM (GB)	CPU (units)	BW (STO)	VM
1.7	1 (1 cores x 1 units)	160	1-1(1)
7.5	4 (2 cores x 2 units)	850	1-2(2)
15.0	8 (4 cores x 2 units)	1690	1-3(3)
17.1	6.5 (2 cores x 3.25 units)	420	2-1(4)
34.2	13 (4 cores x 3.25 units)	850	2-2(5)
68.4	26 (8 cores x 3.25 units)	1690	2-3(6)
1.7	5 (2 cores x 2.5 units)	350	3-1(7)
7.0	20 (8 cores x 2.5 units)	1690	3-2(8)

TABLE II
THREE TYPES OF PHYSICAL MACHINES (PMS) SUGGESTED

CPU (units)	MEM (G)	BW (STO)	P_{min}	P_{max}
16 (4 cores x 4 units)	30	3380G	210W	300W
52 (16 cores x 3.25 units)	136.8	3380G	420W	600W
40 (16 cores x 2.5 units)	14	3380G	350W	500W

with unit GB. Three types of PMs are considered for heterogeneous case, as shown in Table II. Currently, CloudSched implements dynamic load-balancing scheduling algorithms, utilization maximization and energy-efficient scheduling algorithms. Other algorithms such as reliability-oriented and cost-oriented, etc., can be applied as well.

C. Modeling Customer Requirements

CloudSched models customer requirements by randomly generating different types of VMs, and allocates VMs based on appropriate scheduling algorithms in different data centers. The arrival process, service time distribution, and required capacity distribution of requests can be generated according to random processes. The arrival rate of customers' requests can be controlled. Distribution of different types of VM requirements can be set too. A real-time VM request can be represented in an interval vector: vmID(VM typeID, start-time, end-time, requested capacity). For example, vm1(1, 0, 6, 0.25) shows that the request ID is 1, VM is of type 1 (corresponding to integer 1), start-time is 0 and end-time is 6 (here 6 means the end-time is the sixth slot). Other requests can be represented in similar ways. Fig. 4 shows the life cycles of VM allocation in a slotted time window using two PMs, where PM#2 hosts vm4, vm5, and vm6, while PM#1 hosts vm1, vm2, and vm3. Notice that at any slot, the total capacity constraint of a PM has to be met by all VMs allocated on it, and each VM has a start-time, end-time constraint.

III. PERFORMANCE METRICS FOR DIFFERENT SCHEDULING ALGORITHMS

Unlike traditional scheduling algorithms considering only one factor such as CPU, which can cause hotspots or bottlenecks in many cases, CloudSched treats multidimensional resources such as CPU, memory and network bandwidth integrated for both physical machines and VMs. There is still lack of related metrics for scheduling algorithms considering multidimensional resource. For different objectives of scheduling, there are different metrics. In the following, we consider metrics for load-balancing, energy-efficient and utilization maximization. Other metrics for different objectives can be extended easily.

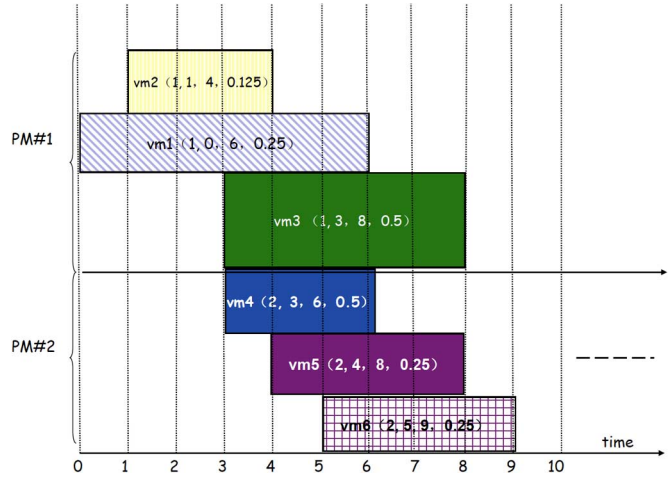

Fig. 4. An example of user requests and allocation.

A. Metrics for Multidimensional Load-Balancing

In the view of advantages and disadvantages of existing metrics for resource scheduling [10], [12], [14], [16], [28], integrated measurement for total imbalance level of Cloud data center as well as average imbalance level of each server are developed for load-balancing strategy. Other metrics for different scheduling strategies can be developed as well. The following parameters are considered.

1) Average CPU utilization (CPU_i^U) of a single CPU i: is averaged CPU utilization during an observed period. For example, if the observing period is one minute and CPU utilization is recorded every 10 s, then CPU_i^u is the average of six recorded values of CPU i.
2) Average utilization of all CPUs in a Cloud datacenter. Let CPU_i^n be the total number of CPUs of server i, then the average utilization of all CPUs on server i is

$$CPU_u^A = \frac{\sum_i^N CPU_i^U CPU_i^n}{\sum_i^N CPU_i^n} \tag{1}$$

where N is the total number of physical servers in a Cloud datacenter. Similarly, average utilization of memory, network bandwidth of server i, all memories and all network bandwidth in a Cloud datacenter can be defined as $MEM_i^U, NET_i^U, MEM_u^A, NET_u^A$, respectively.
3) Integrated load imbalance value (ILB_i) of server i. Variance is widely used as a measure of how far a set of numbers is spread out from each other in statistics. Using variance, an integrated load imbalance value (ILB_i) of server i is defined

$$\frac{\left(Avg_i - CPU_u^A\right)^2 + \left(Avg_i - MEM_u^A\right)^2 + \left(Avg_i - NET_u^A\right)^2}{3} \tag{2}$$

where

$$Avg_i = \frac{\left(CPU_i^U + MEM_i^U + NET_i^U\right)}{3}. \tag{3}$$

(ILB_i) is applied to indicate load imbalance level comparing utilization of CPU, memory and network bandwidth of a single server itself.

4) The imbalance value of all CPUs, memories and network bandwidth. Using variance, the imbalance value of all CPUs in a data center is defined as

$$IBL_{CPU} = \sum_{i}^{N} \left(CPU_i^U - CPU_u^A\right)^2. \quad (4)$$

Similarly, imbalance values of memory (IBL_{mem}) and network bandwidth (IBL_{net}) can be calculated. Then, total imbalance values of all servers in a Cloud datacenter is given by

$$IBL_{tot} = \sum_{i}^{N} ILB_i. \quad (5)$$

5) Average imbalance value of a physical server i. The average imbalance value of a physical server i is defined as

$$IBL_{avg}^{PM} = \frac{IBL_{tot}}{N} \quad (6)$$

where N is the total number of servers. As its name suggests, this value is used to measure average imbalance level of all physical servers.

6) Average imbalance value of a Cloud datacenter (CDC). The average imbalance value of a Cloud datacenter (CDC) is defined as

$$IBL_{avg}^{CDC} = \frac{IBL_{CPU} + IBL_{mem} + IBL_{net}}{N}. \quad (7)$$

7) Average running times. Average running time of proceeding same amount of tasks can be compared for different scheduling algorithms.
8) Makespan. In this paper, it is defined as the maximum load (or average utilization) on any PM.
9) Utilization efficiency. It is defined as (the minimum load on any PM) divides (maximum load on any PM) in this case.

B. Metrics for Energy-Efficiency

1) Energy consumption model.
Most of energy consumption in data centers is from computation processing, disk storage, network, and cooling systems. In [5], authors proposed a power consumption model (P) for blade server

$$14.5 + 0.2U_{CPU} + (4.5e^{-8})U_{mem} + 0.003U_{disk} + (3.1e^{-8})U_{net} \quad (8)$$

where $U_{CPU}, U_{mem}, U_{disk}, U_{net}$ are utilization of CPU, memory, hard disk, and network interface respectively. It can be seen that other factors such as memory, hard disk and network interface have very small impact on total energy consumption. In [3], authors found that CPU utilization is typically proportional to the overall system load, and proposed a power model as follows:

$$P(U) = kP_{\max} + (1-k)P_{\max}U \quad (9)$$

where $P_{\max}$ is the maximum power consumed when the server is fully utilized; k is the fraction of power consumed by the idle server (studies show that on average it is about 0.7); and U is the CPU utilization. In real environment, the utilization of the CPU may change over time due to the workload variability. Thus, the CPU utilization is a function of time and is represented as $u(t)$. Therefore, the total energy consumption by a physical machine (E_i) can be defined as an integral of the energy consumption function over a period of time as:

$$E_i = \int_{t_0}^{t_1} P\left(u(t)\right) dt \quad (10)$$

If $u(t)$ is constant over time (for example, average utilization is adopted, $u(t) = u$), then $E_i = P(u)(t_1 - t_0)$.

2) The total energy consumption of a Cloud data center is computes as

$$E_{cdc} = \sum_{i=1}^{n} E_i. \quad (11)$$

It is the sum of energy consumed by all PMs. Notes that the energy consumption of all VMs on PMs is included.
3) The total number of PMs used. This is the total number of PMs used for the given set of VM requests. It is important for energy-efficiency.
4) The total power-on time of all PMs used. Based on energy consumption equation of each PM, the total power-on time is the key factor.

C. Metric for Maximizing Resource Utilization

1) Average resource utilization. Average utilization of CPU, memory, hard disk, and network bandwidth can be computed and an integration utilization of all these resources can be used too.
2) The total number of PMs used. It is closely related to the average and whole utilization of a Cloud data center.

D. Metrics for Confidence Intervals

Also confidence intervals can be calculated for different metrics as follows: Let $x_1, x_2, x_3, \ldots, x_n$ be the calculated metrics (such as IBL_{tot} and E_{cdc} values, etc.) from n times of repeated simulations. Then, the mean is

$$x_{mean} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (12)$$

and the standard deviation s is

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_{mean} - x_i)^2}{n-1}} \quad (13)$$

and the confidence interval at 95% confidence is given by

$$\left(x_{mean} - 1.96\frac{s}{\sqrt{n}}, x_{mean} + 1.96\frac{s}{\sqrt{n}}\right) \quad (14)$$

IV. Design and Implementation of CloudSched

In this section, we provide details related to the design and implementation of CloudSched. A Java discrete simulator is imple-

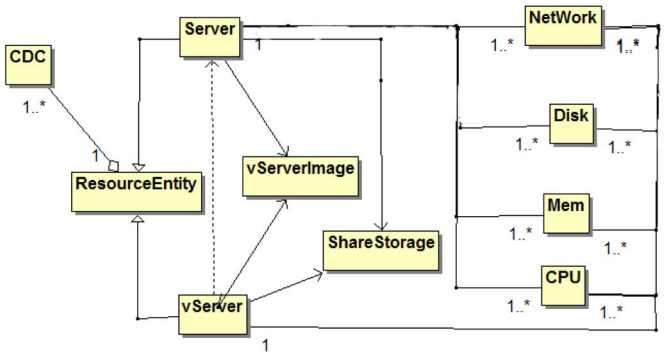

Fig. 5. UML diagram of main resources in Cloud Data Centers.

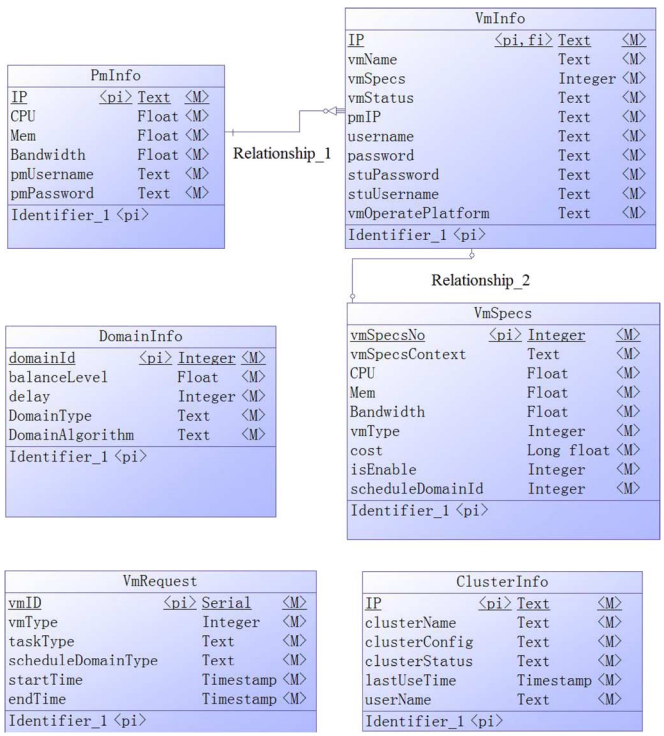

Fig. 6. Detailed UML diagram of main resources in Cloud Data Centers.

mented. In the following, major building blocks of the CloudSched are described briefly.

A. Scheduling Process in Cloud Datacenter

We consider PMs and VMs as IaaS resource, use the same architecture of Cloud data centers, resource scheduling process as proposed in [12]. Figs. 5 and 6 show general and detailed UML diagram of main resources in Cloud data centers, respectively.

B. Scheduling Algorithms—Taking LIF Algorithm as an Example

Fig. 7 shows the pseudocodes of LIF (least imbalance-level first) algorithm [28] for dynamic load-balance of a Cloud data center. Inputs to the algorithm include current VM request r, status of current active tasks and physical machines. For dynamic scheduling, the output is placement scheme for request r. Basically, the algorithm dynamically finds the lowest total

```
Algorithm : Lowest-Average-Value-First (r)
Input: placement request r = (id, t_s, t_e, k);
       status of current active tasks and PMs
Output: placement scheme for r and IBL_tot.
1: initialization: LowestAvg = large number;
2: FOR i=1:N DO
3:   IF request r can be placed on PM (i)
4:   THEN
5:   compute avg(i) utilization value of PM(i) using equations (1)-(3);
6:     IF avg(i)<LowestAvg
7:       THEN
8:       LowestAvg=avg(i);
9:       allocatedPMID=i;
10:      ELSE
11:    ENDIF
12:  ELSE //find next PM
13: ENDFOR
14: IF LowestAvg== large number L // cannot allocate
15:    THEN put r into waiting queue or reject
16:     ELSE place r on PM with allocatedPMID and compute IBL_tot
```

Fig. 7. LIF algorithm.

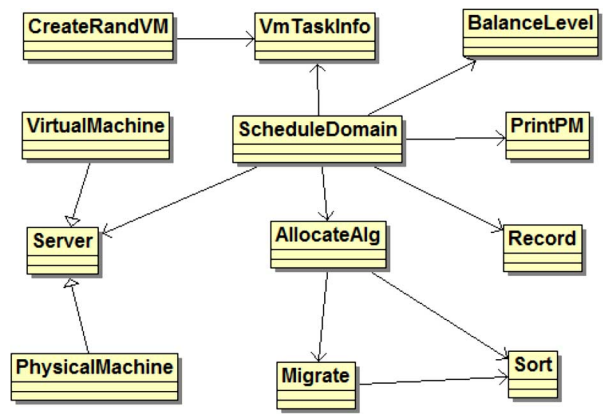

Fig. 8. Main class diagram.

imbalance value of the datacenter when placing a new VM request by comparing different imbalance values if the request is allocated to different physical machines. Actually, the algorithm finds a PM with the lowest integrated-load, this will make the total imbalance-value of all servers in a Cloud data center the lowest. Fig. 8 shows main class diagram of VM allocation of scheduling algorithm. Class ScheduleDomain consists of main methods; it handles tasks in each queue by calling other classes. Class CreateRandVM, and VmTaskInfo, generate task requests. Class Allocate and Sort allocate the requests of VMs; Class Migrate and Allocate_Alg can migrate VMs; Record, PrintPM, and BalanceLevel are responsible for printing and output functions; Server, Physical Machine, and Virtual Machine accomplish functions of physical servers and VMs. Fig. 9 shows one of the interfaces of configuring Cloud data centers in CloudSched. First, a data center is selected (by manager) using different IDs, then the number of and types of PMs are set up. Manager can also add/delete data centers. Fig. 10 shows one of the interfaces of configuring user requests. Probability distribution of each types of VMs, the total number of simulated VMs and preferred data centers can be set up. For simulation, once the probability distribution of each types of VMs and total number of all VMs are given, the number of VMs for each type can be easily computed.

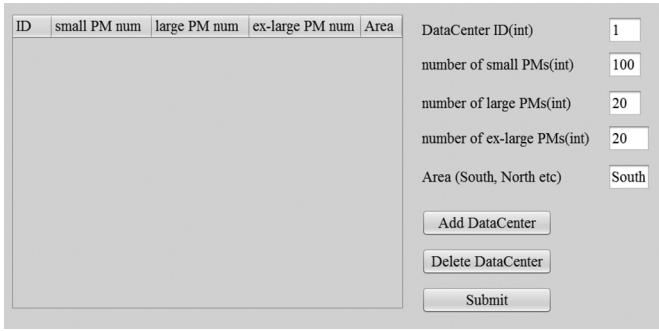

Fig. 9. One interface of configuring Cloud Data Centers.

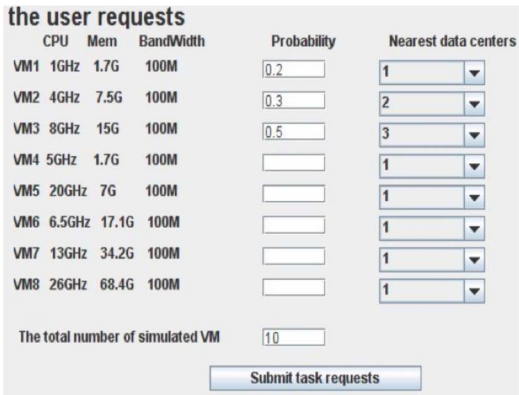

Fig. 10. One interface of configuring user requests

V. Performance Evaluation

We use regular Pentium PC with CPU 2 GHz, memory 2 GB for the simulation.

A. Load-Balance Comparison

In this section, we provide simulation results for comparing five different scheduling algorithms for load-balance. For convenience, short name is given for each algorithm as follows.

1) ZHCJ algorithm: as introduced in [14], the algorithm always chooses physical machines with lowest referred V value and available resource to allocate VMs.
2) ZHJZ algorithm: selects a referring physical machine [16], and calculates the value and chooses physical machines with lowest referred B value and available resource to allocate VMs.
3) LIF algorithm: based on demands characteristics (for example, CPU intensive, high memory, high bandwidth requirements, etc.), always selects physical machines with lowest integrated imbalance value [as defined in (4) and (5)] and available resource to allocate VMs.
4) Rand algorithm: randomly assigns requests (VMs) to physical machines which have available resource.
5) Round-Robin (Round) algorithm: is one of the simplest scheduling algorithms, which assigns tasks to each physical servers in equal portions and in circular order, handling all tasks without priority (also known as cyclic executive). For simulation, three types of heterogeneous physical machines (PMs) are considered; each physical machine pool consists of some amount of physical machines (can be dynamically configured and extended). For simulation of large number of VM requests, both CPU and memory are configured with large size, which can be set dynamically

$$\begin{aligned}&\text{type 1 CPU} = 6\ \text{GHz},\ \text{memory} = 8\ \text{G},\\ &\text{bandwidth} = 1000\ \text{M}\\ &\text{type 2 CPU} = 12\ \text{GHz},\ \text{memory} = 16\ \text{G},\\ &\text{bandwidth} = 1000\ \text{M}\\ &\text{type 3 CPU} = 18\ \text{GHz},\ \text{memory} = 32\ \text{G},\\ &\text{bandwidth} = 1000\ \text{M}.\end{aligned}$$

Similar to Amazon eight EC2 instances with high-CPU, high-memory and standard configuration, eight types of VMs are generated randomly as follows (can be dynamic configured) as in Table I.

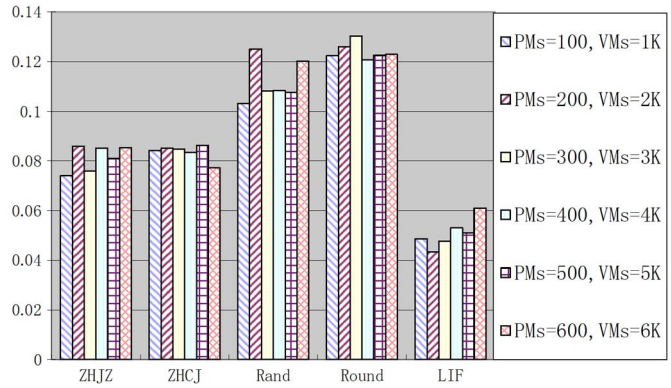

Fig. 11. Average imbalance values of a Cloud Data Center.

For all the simulation in this section, different requests are generated as follows: the total numbers of arrivals (requests) can be randomly set; all requests follow Poisson arrival process and have exponential length distribution; the maximum length of requests can be set; for each set of inputs (requests), simulations are run six times and all the results shown in this paper are the average of the six runs. In this section, the number of PMs is ranging from 100 to 600, the number of requests of VMs is varying from 250 to 1500, a Pentium PC with 2 GHz CPU, 2 G memory is used for all simulation. The input data of user requests is generated using program by considering equal probabilities of above mentioned eight types of VMs. Of course, different (random) probabilities of different types of VMs can be generated. For steady-state analysis, a warm-up period (initial 2000 requests) is used to drop the transient period. Fig. 11. shows average imbalance level (defined in (7)) of a Cloud data center. It can be seen that LIF algorithm has lowest average imbalance level when the total number of VMs and PMs are varying. Through extensive simulation, similar results are observed.

B. Comparing Energy-Efficiency

We considered four algorithms for energy-efficiency.

1) Round Robin (RR): the round-robin is the most commonly used scheduling algorithm (for example by Eucalyptus [26] and Amazon EC2), which allocates VM requests in turn to each PM. The advantage of this algorithm is that it is simple to implement.

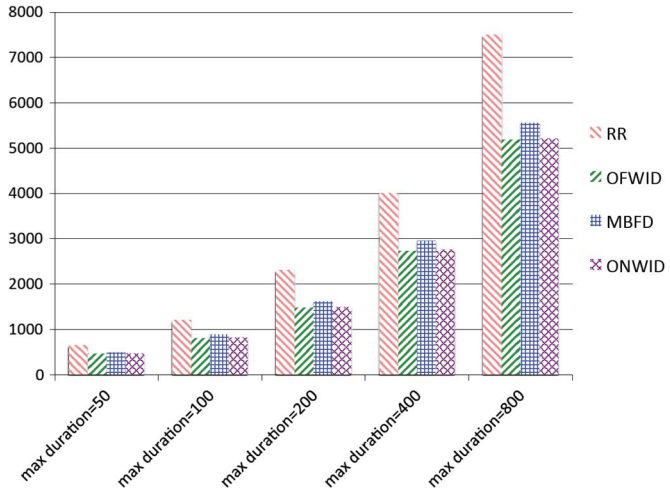

Fig. 12. The total energy consumption (in kw hours) by varying maximum duration of VM requests.

2) Modified Best Fit Decreasing (MBFD): Modified Best Fit Decreasing (MBFD) algorithm is a bin-packing algorithm. Best Fit Decreasing is shown to use no more than 11/9 OPT $+1$ bins (where OPT is the number of bins given by the optimal solution) [2].
3) Offline without delay (OFWID): OFWID knows all requests in advance and follows the requests exactly without delay. It firstly sorts requests in increasing order of their start-times and allocates requests to PMs in increasing order of their IDs. If all running PMs cannot host the request, then a new PM is turned on.
4) Online without delay (ONWID): ONWID knows one request each time. It allocates requests to PMs in increasing order of their IDs. If all running PMs cannot host the request, a new PM is powered on. When the total number of PMs is fixed, if all PMs still cannot host the request, then the request is blocked.

In this case, eight types of VMs are considered, as shown in Table I, which is based on Amazon EC2. The total numbers of arrivals (requests) is 1000 and each type of VMs has equal number, i.e., 125. All requests follow Poisson arrival process and have exponential service time, the mean inter arrival period is set as 5, the maximum intermediate period is set as 50, the maximum duration of requests is set as 50, 100, 200, 400, 800 slots, respectively. Each slot is 5 min. For example, if the requested duration (service time) of a VM is 20 slots, actually its duration is $20 \times 5 = 100$min. For each set of inputs (requests), experiments are run six times and all the results shown in this paper are the average of the six runs. The configuration of physical machines is based on eight types of VMs, as shown in Table II. In this configuration, there are three different types of PMs (heterogeneous case) and the total capacity of a VM is proportional to the total capacity of a PM. For comparison purpose, we assume that all VMs occupy all their requested capacity. Fig. 12 shows the total energy consumption (in kilowatts hours) of the four algorithms as the maximum duration varies from 50 to 800, while all other parameters are the same.

Similar results for load-balance and energy-efficiency are obtained for other cases, we do not present all of them because of page limit.

VI. Conclusion

In this paper, we introduce a lightweight Cloud resources scheduling emulator, CloudSched. Its major features and design and implementation details are presented. Simulation results are discussed for load-balance and energy-efficient algorithms. CloudSched can help developers to identify and explore appropriate solutions considering different resource scheduling policies and algorithms. CloudSched can also import an external benchmark workload (such as LLNL data [27]); currently it needs transferring the format of external benchmark into CloudSched acceptable format. There are quite a few open issues for simulating real-time VM allocation.

- Developing more metrics to measure the quality of related algorithms. For different scheduling strategies such as utilization maximization and maximum profits, etc., of multidimensional resource, we will add more metrics.
- Considering more simulation scenarios. Also some more simulation results such as varying the probability of each VM request, fixing total number of physical servers but varying number of VMs are collecting. Extension to multiple federated data centers can be considered.
- Considering user priority. Currently, we considered that all users have same priority. Different priority policies can be created for users to have different priorities for certain types of VMs.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, Above the Clouds: A Berkeley view of cloud computing Univ. California at Berkley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28, Feb. 10, 2009.

[2] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing future generation computer systems," , vol. 28, no. 5, pp. 755–768, May 2012.

[3] R. Buyya and M. Murshed, *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. Concurrency and Computation: Practice and Experience*. New York, NY, USA: Wiley, Nov.–Dec. 2002, vol. 14, pp. 13–15.

[4] 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, Software: Practice and Experience*. New York, NY, USA: Wiley, Jan. 2011, vol. 41, pp. 23–50, 0038-0644, Number 1.

[5] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, *Cloud Computing and Emerging IT Platforms: Vision, Hype, Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems*. Amsterdam, The Netherlands: Elsevier Science, Jun. 2009, vol. 25, pp. 599–616, (6).

[6] C. L. Dumitrescu and I. Foster, "GangSim: A simulator for grid scheduling studies," in *Proc. IEEE Int. Symp. Cluster Comput. Grid (CCGrid 2005)*, Cardiff, U.K., 2005, pp. 1151–1158.

[7] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments, 2006," in *Stanford Univ./HP Labs Workshop on Modeling, Benchmarking, Simulation (MoBS)*, Jun. 18, 2006, pp. 70–77.

[8] F. Howell and R. Mcnab, "SimJava: A discrete event simulation library for Java," in *Proc. 1st Int. Conf. Web-Based Modeling and Simulation*, 1998.

[9] A. Legrand, L. Marchal, and H. Casanova, "Scheduling distributed applications: The SimGrid simulation framework," in *Proc. 3rd IEEE/ACM Int. Symp. Cluster Comput. Grid*, 2003, pp. 138–145.

[10] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: Integration and load balancing in data centers," in *Proc. ACM/IEEE Conf. Supercomput.*, 2008, pp. 1–12.

[11] W. H. Tian, "Adaptive dimensioning of cloud data centers," in *Proc. IEEE 8th Int. Conf. Dependable, Autonomic Secure Comput. (DASC-09)*, Chengdu, China, Dec. 12–14, 2009.
[12] W. H. Tian, Y. Zhao, Y. L. Zhong, M. X. Xu, and C. Jing, "Dynamic and integrated load-balancing scheduling algorithms for Cloud data centers," *China Commun.*, vol. 8, no. 6, pp. 117–126, 2011.
[13] B. Wickremasinghe *et al.*, "CloudAnalyst: A CloudSim-based tool for modelling and analysis of large scale cloud computing environments," in *Proc. 24th IEEE Int. Conf. Adv. Inform. Netw. Appl. (AINA'10)*, Perth, Australia, Apr. 20–23, 2010.
[14] T. Wood *et al.*, "Black-box and gray-box strategies for virtual machine migration," in *Proc. Symp. Netw. Syst. Design. Implementation (NSDI)*, 2007, pp. 229–242.
[15] W. Zhang, "Research and Implementation of elastic network service," (in Chinese) Ph.D. dissertation, National Univ. Defense Technology, Changsha, China, 2000.
[16] H. Zheng, L. Zhou, and J. Wu, "Design and implementation of load balancing in web server cluster system," *J. Nanjing Univ. Aeronautics Astronautics*, vol. 38, no. 3, pp. 347–351, Jun. 2006.
[17] L. Youseff *et al.*, "2008. Toward a unified ontology of cloud computing," in *Proc. Grid Comput. Environ. Workshop, GCE'08*, 2008, pp. 1–10.
[18] H. Zhu, M. Hou, C. Wang, and M. Zhou, "An efficient outpatient scheduling approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 4, pp. 701–709, Oct. 2012.
[19] J. Cao, W. Zhang, and W. Tan, J. Cao, W. Zhang, and W. Tan, Eds., "Dynamic control of data streaming and processing in a virtualized environment," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 2, pp. 365–376, Apr. 2012.
[20] R. Prodan and M. Wieczorek, "Bi-criteria scheduling of scientific grid workflows," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 2, pp. 364–376, Apr. 2010.
[21] Amazon EC2, 2013. [Online]. Available: http://aws.amazon.com/ec2/
[22] DMTF Cloud Management, 2013. [Online]. Available: http://www.dmtf.org/standards/cloud
[23] Google App Engine, 2013. [Online]. Available: http://code.google.com/intl/zh-CN/appengine/
[24] IBM blue cloud, 2013. [Online]. Available: http://www.ibm.com/grid/
[25] Microsoft Windows Azure, 2013. [Online]. Available: http://www.microsoft.com/windowsazure
[26] Eucalyptus, 2013. [Online]. Available: www.eucalyptus.com
[27] Hebrew University, Experimental Systems Lab, 2012. [Online]. Available: www.cs.huji.ac.il/labs/parallel/workload
[28] W. Tian, X. Liu, C. Jin, and Y. Zhong, "LIF: A dynamic scheduling algorithm for Cloud data centers considering multi-dimensional resources," *J. Inform. Comput. Sci.*, vol. 10, no. 12, 2013, appear in.

Wenhong Tian received the Ph.D. degree from the Computer Science Department, North Carolina State University, Raleigh, NC, USA.

He is now an Associate Professor at the University of Electronic Science and Technology of China, Chengdu. His research interests include dynamic resource scheduling algorithms and management in Cloud data centers, dynamic modeling and performance analysis of communication networks, and biocomputing. He has published about 30 journal and conference papers in related areas.

Yong Zhao received the Ph.D. degree from the Computer Science Department, Chicago University, Chicago, IL, USA.

He is a Professor at the University of Electronic Science and Technology of China, Chengdu. His research interests include Grid computing, large-data process in Cloud computing, etc. He has published about 30 journal and conference papers in related areas.

Minxian Xu is working towards the Ph.D. degree at the School of Computer Science, University of Electronic Science and Technology of China, Chengdu, under the supervision of Dr. Tian and Dr. Zhao.

Yuanliang Zhong is a graduate student at the School of Computer Science, University of Electronic Science and Technology of China, Chengdu, under the supervision of Dr. Tian and Dr. Zhao.

Xiashuang Sun is a graduate student at the School of Computer Science, University of Electronic Science and Technology of China, Chengdu, under the supervision of Dr. Tian and Dr. Zhao.