FlexCloud: A Flexible and Extensible Simulator for Performance Evaluation of Virtual Machine Allocation

Minxian Xu
Department of Computing and Information System,
School of Engineering, University of Melbourne
Email: xmxyt900@gmail.com,

Guozhong Li, Wutong Yang, Wenhong Tian
School of Information and Software Engineering,
University of Electronic Science and Technology of China
Email: GuozhongLi@hotmail.com, uestcywt@foxmail.com,
tian_wenhong@uestc.edu.cn,

***Abstract*—Cloud Data centers aim to provide reliable, sustainable and scalable services for all kinds of applications. Resource scheduling is one of keys to cloud services. To model and evaluate different scheduling policies and algorithms, we propose FlexCloud, a flexible and scalable simulator that enables users to simulate the process of initializing cloud data centers, allocating virtual machine requests and providing performance evaluation for various scheduling algorithms. FlexCloud can be run on a single computer with JVM to simulate large scale cloud environments with focus on infrastructure as a service; adopts agile design patterns to assure the flexibility and extensibility; models virtual machine migrations which is lack in the existing tools; provides user-friendly interfaces for customized configurations and replaying. Comparing to existing simulators, FlexCloud has combined features for supporting public cloud providers, load-balance and energy-efficiency scheduling. FlexCloud has advantage in computing time and memory consumption to support large-scale simulations. The detailed design of FlexCloud is introduced and performance evaluation is provided.**

***Index Terms*—Cloud Data Centers; Resource Scheduling Algorithms; Virtual Machine Allocation; Performance Evaluation; Flexibility and Extensibility**

I. Introduction

With various recent advancements in virtualization, like Grid computing, Web computing, utility computing and related technologies, Cloud computing obtains great development. Cloud computing aims to provide both infrastructure and services on demand through the Internet or intranet [3], and its benefits can be concluded as hiding and abstraction of complexity, virtualized resources and efficient use of distributed resources. Cloud computing allows the sharing, allocation and aggregation of software, computational and storage network resources on demand. Currently, quite a few IT enterprises products, like Amazon EC2, Google App Engine, IBM blue Cloud and Microsoft Azure have shown their practice of emerging Cloud computing platforms [14]. Whereas there are many challenging issues to be resolved [3], Cloud computing is still considered in its infancy.

An essential technology in Cloud datacenter is resource scheduling. One challenge problem related to scheduling in Cloud data center is to consider allocation and migration of reconfigurable virtual machines and integrated features of hosting physical machines. Different from existing load-balancing scheduling algorithms that consider only physical servers with one factor such as CPU, the new algorithms treat CPU, memory and network bandwidth integrated for both physical machines (PMs) and virtual machines (VMs). Besides that, real-time virtual machine allocation for multiple parallel jobs and physical machines is taken into consideration. With the development of cloud computing, the size and density of the cloud data center become huge, and problems which need to be solved therewith.

Because of the uncertainty of network environments, it is extremely hard to do an in-depth research for all these problems in real Internet platform. In addition, the network conditions cannot be predicted or controlled accurately, but affect the validation of strategies. A considerate way in research is developing a 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. By using data center simulation system, researchers can evaluate

978-1-5090-1893-2/15 $31.00 © 2015 IEEE
DOI 10.1109/SmartCity.2015.143

suitable strategies such as distributing reasonable data center resources, selecting data center to match special requirements, reducing costs, finding efficient scheduling algorithms and so on.

The major contributions of this paper is the proposal of a new cloud simulator, FlexCloud, with light weight design to simulate cloud environment. FlexCloud has following features:

- FlexCloud is built on Java platform and can be run on a single computer installed JVM to simulate large scale cloud infrastructure as a service (IaaS). A computer with 4 GB memory can simulate large scale applications. With a 4 GB memory computer, experiments can simulate scheduling process of more than 100,000 requests. We have extended our tests with computers with 2GB memory, that configuration can simulate requests ranging from 25,000 to 50,000.
- A user-friendly GUI is provided and lots of customized configurations can be set to satisfy various simulation assumptions. The basic operations like: select algorithm type, set VM numbers, set average duration, set start time, set total number of PMs, select comparison algorithms and indices are included.
- A scheduling process framework is defined, each step of process can be extended easily in agile style.
- New scheduling algorithms and performance metrics are flexible and extensible to be added in; currently load-balancing and energy-efficiency scheduling algorithms are considered.
- Virtual machine migration is modeled, though this is still lack in current simulation tools.

II. Related works

Some research has been conducted in cloud simulation systems. Buyya et al. introduce GridSim [12] toolkit for modeling and simulation of distributed resource management for grid computing. Dumitrescu and Foster [7] introduce GangSim tool for grid scheduling. Buyya et al. [13] 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 [13] is one of Cloud computing simulators, which provides: modeling large-scale cloud computing infrastructure; models for the data center, service agency, scheduling and distributing strategies; virtual engines, which is helpful to create and manage several independent and collaborative virtual services in a data center node; switching 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 [10] and GridSim [12]. Also CloudSim and CloudAnalyst treat a Cloud data center as a large resource pool and consider only application-level workloads, may not be suitable for Infrastructure as a service (IaaS) simulation where each virtual machine as resource is considered to be requested and allocated. A CloudSim-based simulation tool considering DVFS energy model is proposed in [16]. Kliazovich et al. propose an energy-aware simulation environment named GreenCloud for Cloud datacenters [8]. Nunez et al. [5] introduce a new simulator of cloud infrastructure named iCanCloud using C++ and compare the performance with CloudSim. Di et al. [15] design a cloud simulation system, GloudSim, which is based on a one-month Google trace produced with large scale applications and jobs on hosts.

In our teaching practice in our university, we have adopted CloudSim, a mature simulator, as a teaching tool assisted, but according to the students' feedback, CloudSim is a bit complex to use and heavy to execute. That complexity is also a feature of iCanCloud. As for MDCSim, a commercial tool, is not appropriate for researching. Apart from that, it's not easy to use several languages together in GreenCloud, since it is implemented with C++ and OTcl.

The main contribution of FlexCloud lies in that it is implemented with light weight design, flexible to extend as well as easy to start. Besides the benefits for teaching, we also cooperate with a company researching in resource scheduling to boost the functions of FlexCloud under multi-datacenter environment. They would use FlexCloud to explore suitable algorithms for their company applications. FlexCloud is an open source tool that can be fetched from [9].

III. The Architectural Model of FlexCloud

Fig.1 shows the overview architecture of FlexCloud with layered components. The top layer is Client Layer that provides the interface for user to configure requests properties and have results feedbacks from lower layers. At this layer, a GUI implemented with Java Swing supports user to configure algorithm types, set PM and VM specifications and select scheduling algorithms. After all settings are completed, the defined configurations would be submitted to lower layer and a sequence of scheduling steps would be processed. Comparison diagrams as well as result outputs would be sent as feedback to Client Layer. At lower layer, a Requests Broker is implemented

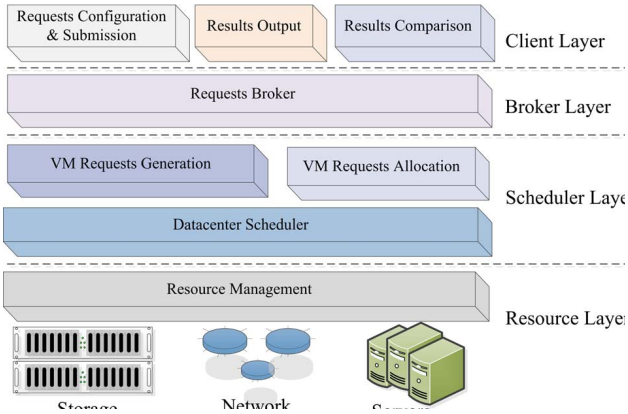

Fig. 1. Layered FlexCloud architecture

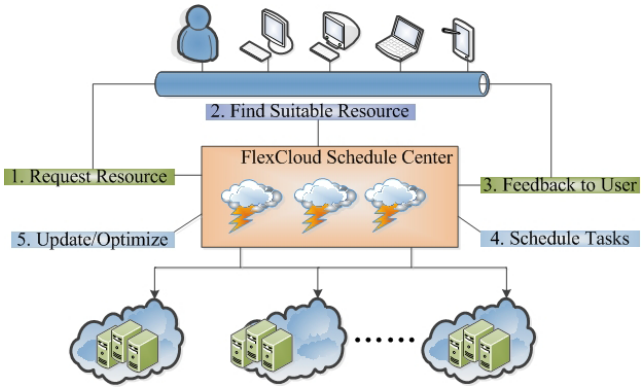

Fig. 2. A scenario architecture with FlexCloud

at Broker Layer acting as a mediator between Client Layer and Scheduler Layer. This Layer is responsible for verifying the inputs from Client Layer and transforming the settings into recognized commands at Scheduler Layer. For instance, the number of VM requests submitted from Client Layer would be written into a configuration file, which could be read in the process of scheduling at Scheduler Layer. Scheduler Layer implements the core functions for FlexCloud system. At this layer, the scheduling process is defined: VM Requests Generation component generates the VM requests with configured properties on user interface; Datacenter Scheduler component schedules the particular algorithms to allocate VMs to corresponding PM according to algorithms; VM Requests Allocation component manages the allocated VMs, including checking the allocation conditions and removing VMs at the end of their lifecycles. At bottom layer, Resource Layer contains a Resource Management component providing resource that VM requests require and supporting services for higher levels. Besides the component, the physical resource, such as servers, network and storage are resources of the whole system.

Fig.2 shows an application scenario with FlexCloud. This figure shows three main components: user, FlexCloud scheduler center and other computing centers. The FlexCloud scheduler center is responsible for the following main tasks: (1) accepting the VM requests sent by users; (2) managing computing centers that in service; (3) finding available computing unit to allocate requests; (4) sending feedback information to users. Computing centers represent a pool of Physical Machines (PMs) or Virtual Machines (VMs), each one configured with a predefined specification such as CPU, memory and storage.

TABLE I
3 TYPES OF PHYSICAL MACHINES (PMS) SUGGESTED [1]

PM Pool Type	Compute Units	Memory	Storage
Type 1	16 units	30GB	3380GB
Type 2	52 units	136GB	3380GB
Type 3	40 units	14GB	3380GB

Users are represented as component that submits a set of jobs to be allocated to specific PM in computing center. These submissions are submitted directly to the FlexCloud scheduler center. Then, the requests are managed by this module to be allocated to specific PM in the corresponding data center. After all requests have been processed, a feedback report would be sent back to the user.

A. Modeling the datacenter in FlexCloud

From computing resource point of view, a data center consists of a number of physical servers (PMs), network devices, storages and other related equipment. A PM contains several kinds of resources, like CPU, memory, storage and bandwidth, etc. Before VM requests are coming, the PMs are at the state of turned-on, which means the class of Physical Machine is instantiated in FlexCloud. The number of instances depends on the number of PMs would provide services.

In TABLE I, the 3 suggested types of heterogeneous PMs in FlexCloud are listed, and the configuration can be dynamically set. The type and property values, like CPU, memory, storage and power, are recorded in a configuration XML file, which would be loaded into system. Besides the load balance algorithms, we also implement energy-saving algorithms that contain a new property named power consumption. This property is

added in the configuration XML file and corresponding methods are added in class PhysicalMachine.

In datacenter model, the left resource capacity decides whether a VM request can be allocated to that PM. At initialization stage, PM has a full capacity resource to offer services. Either the allocation or remove operation would update the available capacity value and influence the later requests allocation.

B. Modeling VM requests in FlexCloud

We use a simple example to show how VM requests are modeled in FlexCloud in Fig. 3. Slots $\#1, \#2, \ldots, \#6$ represent the time slots in discrete time, which can be treated as a second or a minute that requests are in. For instance, VM2 occupies time slot 3 to 5, so lifecycle of VM2 is 3 slots. The value 0.0625 is proportion of resource occupation, meaning that VM2 would occupy 6.25% resource of the PM that it would be allocated to, during time slot 3 to 5. In our model, several VM requests can share the capacity of the same PM at the same time slot only if the capacity is enough.

VM# slot	#1	#2	#3	#4	#5	#6
VM1	0.25	0.25	0.25	0.25	0.25	0.25
VM2			0.0625	0.0625	0.0625	
VM3				0.5	0.5	0.5
VM4				0.5	0.5	0.5
VM5				0.5	0.5	0.5
VM6					0.25	0.25

Fig. 3. an example of VM requests

TABLE II shows the corresponding CPU, memory, storage values for different VMs. Also for extensible reason, these property values are also recorded into a configuration XML file. Once a VM request is allocated to a PM, the left resource capacity would be decreased by the value of that request, and the capacity is increased back when request is released.

In FlexCloud, several VM requests generation approaches have been implemented, in which requests can be generated in Poisson, Normal and Random distributions. When the specific distribution is selected, the start time or duration of the generated requests would follow the distribution. Moreover, it's available for FlexCloud to import requests data from file in the *Generate VMs* step, which means it can be tested under realistic data.

C. Modeling Scheduling Algorithms in FlexCloud

Four kinds of scheduling algorithms are provided in FlexCloud based on scheduling goals and request types. For request types, scheduling algorithms can be divided into online algorithms and offline algorithms, the difference lies in whether the requests information is all known before scheduling. Requests would come and be operated one by one in online algorithm, while requests sequence can be adjusted by processing time or end time because all requests information have been collected before scheduling in offline algorithms. Another division principle is via goal: we consider load balancing and energy saving in FlexCloud.

TABLE II
8 TYPES OF VIRTUAL MACHINES (VMS) IN AMAZON EC2 [1]

Compute Units	Memory	Storage	VM Type
1 units	1.7GB	160GB	1-1(1)
4 units	7.5GB	850GB	1-2(2)
8 units	15GB	1690GB	1-3(3)
6.5 units	17.1GB	420GB	2-1(4)
13 units	34.2GB	850GB	2-2(5)
26 units	68.4GB	1690GB	2-3(6)
5 units	1.7GB	350GB	3-1(7)
20 units	7GB	1690GB	3-2(8)

When comparing the effects of different algorithms, the scheduling process would be same except that the scheduling algorithms are different. For online load balancing comparison, Random, Round-Robin (Round), List Scheduling (LS) algorithms have been implemented. Under the layered architectural model and related design pattern (introduced in later section), new created algorithms can be added to scheduling algorithm library, without influencing other existed algorithms.

D. Performance Metrics in FlexCloud

In this section, we introduce the major performance metrics we used in FlexCloud:

For load balancing algorithms:

Average utilization: Each PM would have the utilization value in scheduling process, and average utilization is the arithmetic average value of all PMs in the data center;

PM resource: $PM_i(i, PCPU_i, PMem_i, PStorage_i)$, i is the index number of PM, $PCPU_i$, $PMem_i$, $PStorage_i$ are the CPU, memory, storage capacity of that a PM can provide.

VM resource:
$VM_j(j, VCPU_j, VMem_j, VStorage_j, T_j^{start}, T_j^{end})$, j is the VM type ID, $VCPU_j, VMem_j, VStorage_j$ are the CPU, memory, storage requirements of VM_j, T_j^{start}, T_j^{end} are the start time and end time, which are used to represent the life cycle of a VM.

Time slot: we consider a time span from 0 to T be

divided into parts with same length. Then n parts can be defined as $[(t_1 - t_0), (t_2 - t_1), \ldots, (t_n - t_{n-1})]$, each time slot T_k means the time span $(t_k - t_{k-1})$.
Average CPU utilization of PM_i during slot 0 and T_n:

$$PCPU_i^U = \frac{\sum_{k=0}^{n}(PCPU_i^{T_k} \times T_k)}{\sum_{k=0}^{n} T_k} \quad (1)$$

And memory $PMem_i^U$ and storage $PStorage_i^U$ utilization of both PMs and VMs can be computed in the same way. Similarly, average CPU utilization of a VM can be computed.
Integrated load imbalance value ILB_i of PM_i: The variance is widely used as a measure of how far a set of values is spread out from each other in statistics. Using variance, an integrated load imbalancing value ILB_i of server i is defined

$$ILB_i = \frac{(Avg_i - CPU_u^A)^2}{3} + \frac{(Avg_i - Mem_u^A)^2}{3} + \frac{(Avg_i - Storage_u^A)^2}{3} \quad (2)$$

where

$$Avg_i = \frac{PCPU_i^U + PMem_i^U + PStoarge_i^U}{3} \quad (3)$$

and $CPU_u^A, Mem_u^A, Storage_u^A$ are respectively the average utilization of CPU, memory and storage in a Cloud data center.
ILB_i is applied to indicate load imbalance level comparing utilization of CPU, memory and network bandwidth of a single server itself.
Makespan: is as same as traditional definition, and therefore the capacity_makespan of all PMs can be formulated as below:

$$capacity\_makespan = \max_i (L_i) \quad (4)$$

Load efficiency (skew of makespan): is defined as the (minimal average load divided by maximal average load) on all machines:

$$skew(makespan) = \frac{\min_i(L_i)}{\max_i(L_i)} \quad (5)$$

where L_i is the load of PM i. Skew shows the load balancing efficiency to some degree.
Capacity_makespan: In any allocation of VM requests to PMs, we can let $A(i)$ denote the set of VM requests allocated to machine PM_i, under this allocation, machine PM_i will have total loads,

$$L_i = \sum_{j \in A(i)} c_j t_j \quad (6)$$

TABLE III
THEORETICAL AND SIMULATION RESULTS COMPARISON OF LS ALGORITHM

LS Indices	Theoretical	Simulation
Average Utilization	0.5	0.5
Imbalance Degree	0.0	0.0
Makespan	0.5	0.5
Skew(makespan)	1	1
Capacity_makespan	50	50
Skew(capacity_makespan)	1	1

TABLE IV
THEORETICAL AND SIMULATION RESULTS COMPARISON OF LPT ALGORITHM

LPT Indices	Theoretical	Simulation
Average Utilization	0.505	0.505
Imbalance Degree	0.0	0.0
Makespan	1	1
Skew(makespan)	1	1
Capacity_makespan	50.5	50.5
Skew(capacity_makespan)	1	1

For energy saving algorithms, some indices are also provided, like energy consumption, numer of PM turned on and etc, because of page limitation, we omit the detailed introduction here. Other metrics could also be included for further research.

IV. VALIDATION OF FLEXCLOUD

To validate the accuracy of FlexCloud, we have designed some test cases to compare the theoretical results and simulation results. In this section, we use LS (List Scheduling), LPT(Longest Processing Time First), EDF(End-time Decreasing First) algorithms to compare theoretical and simulation results.

The test cases we designed are easily theoretically calculated and can reflect some general situations. For LS algorithm that always allocate a VM to the PM with the lowest load, we set that there are 100 PMs and 100 VMs requests both in the same types, the start-time of requests are ordered in increasing sequence, $1, 2, 3, \ldots, 100$, and all requests duration are 100 and require capacity is 0.5 of a PM. Since PMs number and VMs number are same in this case, LS algorithm works as Round-Robin algorithm, that means each PM would undertake a VM task. Then we calculate the values in theoretical way and simulation, same results have been observed and shown in TABLE III.

We also design a test case for LPT algorithm, an offline algorithm that VM requests can be reordered by processing time before they are allocated. In this case,

TABLE V
THEORETICAL AND SIMULATION RESULTS COMPARISON OF EDF ALGORITHM

EDF Indices	Theoretical	Simulation
Power Consumption	250000	250000
Rejected Number	10	10
Turned on PMs	20	20

there are 50 PMs and 100 VMs both in the same types, each request requires 0.5 capacity of a PM and starts at $1, 2, 3, \ldots, 100$, and the durations of VMs are ordered in decrease order from 100 to 1 as $100, 99, 98, \ldots, 1$. Same results have been observed and collected in TABLE IV.

For energy saving algorithm EDF, it should be noticed that comparison indices are different and requests are ordered by end-time. In this case, we set that there are 20 PMs and 50 VMs both in the same types, the start-times of VM requests are ordered in increasing as $1, 2, 3, \ldots, 50$ and end-times are decreasing as $100, 99, 98, \ldots, 51$. Each VM requires 0.5 capacity of a PM. We adopt the energy saving model referred to [2] and assume host minimum power $P_{min} = 300$, host maximum power $P_{max} = 500$. Same theoretical and simulation values have been collected in TABLE V. Referring to the collected data in TABLE IV and V, the results show the correctness of FlexCloud.

V. EVALUATIONS

To extend performance evaluations, we also compare scheduling algorithms performance with advanced settings and collect the comparison data. The related settings are as following:

1) Algorithm type is offline load balancing;

2) PM specifications: using suggested specifications in Amazon EC2 shown in Table I. PMs with different numbers are considered. PMs numbers are varying from 15, 30, 60 to 240 and each type of PMs occupies about 1/3 of total PMs numbers;

3) VM requests: using suggested specifications in Amazon EC2 shown in Table II. We adopt the log data at Experimental System Lab (ESL) [11] data, which has been used in hundreds of studies, to reflect realistic data generation. The log contains months of records collected by a large Linux cluster and has characteristics consistent with our problem model. Each line of data in that log file includes 18 elements, while we only need the request-ID, start-time, duration and number of processors (capacity demands) in our simulation. We convert the units from seconds in ESL log file into minutes, as we design 5 minutes to be a time slot length;

4) Algorithms for comparison: RoundRobin (R-R), Longest Processing Time first (LPT, referring to section IV), Post Migration Algorithm (MIG, virtual machine migration is implemented) [4], Capacity_makespan Prepartition Algorithm (CMP, partiton the requests before allocation) [17];

5) Indices for comparison: average utilization, imbalance degree, longest process time and capacity_makespan.

Fig.4 and Fig.5 show the average utilization, imbalance degree, makespan and capacity_makespan comparison for different algorithms with ESL data trace. From these figures, we can notice that CMP algorithm has better performance than other algorithms in average utilization, imbalance degree, makespan, capacity_makespan. CMP algorithm has 10%-20% higher average utilization than MIG and LPT, and 40%-50% higher average utilization than Random-Robin (R-R). Prepartition algorithm has 10%-20% lower average makespan and capacity_makespan than MIG and LPT, and 40%-50% lower average makespan and capacity_makespan than R-R. The results lie in that R-R is the simplest algorithm with quite limited load balancing effects, the MIG balances better than LPT as it has migration process to reallocate loads after the LPT allocation process, as for CMP, it works in a much more refined and desired scale by prepartion based on reservation data while MIG is a best-effort trial by migration.

VI. CONCLUSIONS AND FURTHER WORK

In this paper, we introduce the FlexCloud, a novel simulator for performance evaluation of virtual machine allocation in Cloud data centers. It is flexible, scalable to simulate resource scheduling in cloud data centers. A complete simulation framework has been built and introduced.

There are a few research directions for extending the simulator:

- Considering more scheduling algorithms. In FlexCloud, we already implemented load-balancing and energy-efficiency, other scheduling algorithms such as cost-oriented or reliability-oriented algorithms can be added in easily.
- Providing more visual outputs such as dashboards and logical view of different data centers and their resource usages. This information is very important for managers and operators to have.
- Considering more infrastructures, such as networking devices. Currently FlexCloud considers bandwidth requests and allocations. The network devices such as three-tire switches and routers distributed in different data centers are under consideration.

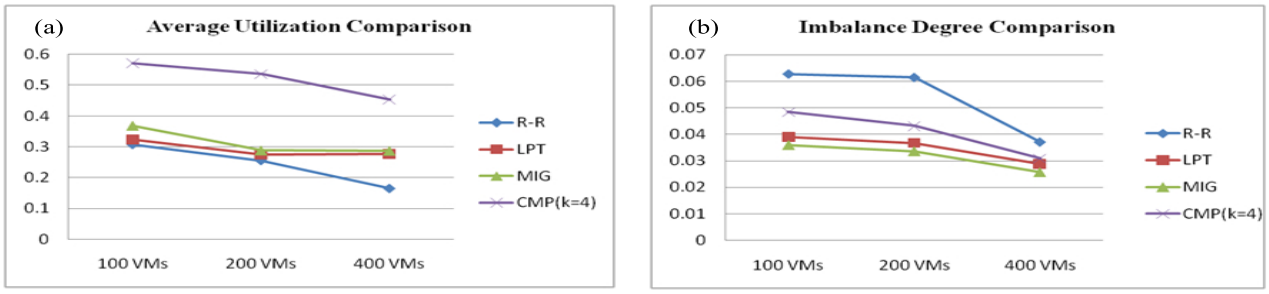

Fig. 4. The offline algorithm comparison of average utilization (a) and imbalance degree (b) with ESL trace

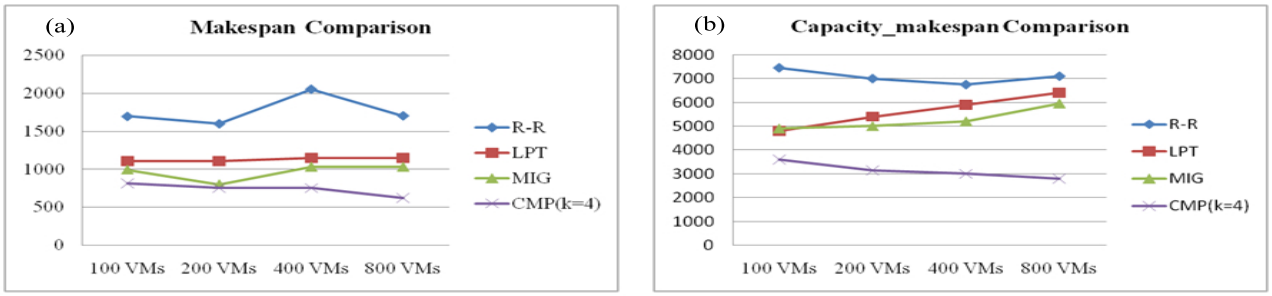

Fig. 5. The offline algorithm comparison of makespan (a) and capacity_makespan (b) with ESL trace

ACKNOWLEDGMENT

This research is supported by China National Science Foundation (CNSF) with project ID 61450110440 and China Scholarship Council.

REFERENCES

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

[2] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing[J]. Future generation computer systems, 2012, 28(5), 755-768.

[3] Fox A, Griffith R, Joseph A, et al. Above the clouds: A Berkeley view of cloud computing[J]. Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, 2009, 28: 13.

[4] Gulati A, Shanmuganathan G, Holler A, et al. Cloud-scale resource management: challenges and techniques[C]//Proceedings of the 3rd USENIX conference on Hot topics in cloud computing. USENIX Association, 2011: 3-3.

[5] Nuenz A, Vazquez-Poletti J L, Caminero A C, et al. iCanCloud: A flexible and scalable cloud infrastructure simulator[J]. Journal of Grid Computing, 2012, 10(1): 185-209.

[6] Wickremasinghe B. Cloudanalyst: A cloudsim-based tool for modelling and analysis of large scale cloud computing environments[J]. MEDC project report, 2009, 22(6): 433-659.

[7] Dumitrescu C L, Foster I. GangSim: a simulator for grid scheduling studies[C]//Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on. IEEE, 2005, 2: 1151-1158.

[8] Kliazovich D, Bouvry P, Khan S U. GreenCloud: a packet-level simulator of energy-aware cloud computing data centers[J]. The Journal of Supercomputing, 2012, 62(3): 1263-1283.

[9] FlexCloud Project, http://sourceforge.net/projects/flexcloud/, 2014.

[10] Howell F, McNab R. SimJava: A discrete event simulation library for java[J]. Simulation Series, 1998, 30: 51-56.

[11] Hebrew University, Experimental Systems Lab, www.cs.huji.ac.il/labs/parallel/workload, 2007

[12] Buyya R, Murshed M. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing[J]. Concurrency and computation: practice and experience, 2002, 14(13-15): 1175-1220.

[13] 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 and Simulation, 2009. HPCS'09. International Conference on. IEEE, 2009: 1-11.

[14] Buyya R, Yeo C S, Venugopal S. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities[C]//High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on. Ieee, 2008: 5-13.

[15] Di S, Cappello F. GloudSim: Google trace based cloud simulator with virtual machines[J]. Software: Practice and Experience, 2014.

[16] Gurout T, Monteil T, Da Costa G, et al. Energy-aware simulation with DVFS[J]. Simulation Modelling Practice and Theory, 2013, 39: 76-91.

[17] Tian W, Xu M, Chen Y, et al. Prepartition: A new paradigm for the load balance of virtual machine reservations in data centers[C]//Communications (ICC), 2014 IEEE International Conference on. IEEE, 2014: 4017-4022.