International Journal of Grid Distribution Computing
Vol.7, no.3 (2014), pp.71-88
http://dx.doi.org/10.14257/ijgdc.2014.7.3.08

Three Levels Load Balancing on Cloudsim

Yongsheng Hao[1], Guanfeng Liu[2] and Junwen Lu[3]

[1]*Network Center, Nanjing University of Information Science & Technology, Nanjing, 210044, China*
[2] *Advanced Data Analytics Laboratory of Soochow University, Suzhou, 215006, China*
[3]*Xiamen University of technology, Xiamen, 361024, China*
hyslove@163.com

Abstract

Cloud balancing provides an organization with the ability to distribute application requests across any number of application deployments located in different data centers and through Cloud-computing providers. In this paper, we propose a load balancing method-Minsd (Minimize standard deviation of Cloud load method) and apply it on three levels control: PEs (Processing Elements), Hosts and Data Centers. Simulations on CloudSim are used to check its performance and its influence on makespan, communication overhead and throughput. A true log of a cluster also is used to test our method. Results indicate that `our method` *not only gives good Cloud balancing but also ensures reducing makespan and communication overhead and enhancing throughput of the whole the system.*

Keywords: *Load Balancing, Cloudlet, Standard Deviation, CloudSim*

1. Introduction

"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (*e.g.*, networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models." Details can be found in [5, 46]. A lot of research, including resource scheduling [39, 40], heterogeneous language supporting [42], IO data management [38] and load balancing [23, 31, 34, 35, 36, 41], have been widely studied in the Cloud.

A growing number of organizations are turning to Cloud computing to meet the demands including advertising services, viral marketing activities, online games, retail sites and so on. Cloud computing meets many challenges [8] including Cloud balancing [25]. Cloud balancing extends the architectural deployment model used in conjunction with global server load balancing to the Cloud [1]. It increases the choices for organizations when they determine from the place where a given application should be delivered.

One aim of the Cloud is to provide adequate resources to meet current and expected data loads [2]. However, even the most state-of-the-art Cloud environment will be of little use without a highly robust, automated load balancing component. Many of the top public Cloud services have made the load balancing a priority, primarily as a means to distinguish themselves as full-service solution providers rather than mere adjuncts to internal data center infrastructure. Amazon recently incorporated the Zeus [3] Simple Load Balancer into the EC2

ISSN: 2005-4262 IJGDC
Copyright ⓒ 2014 SERSC

[4] platform with an aim at drawing in smaller and mid-sized customers. Hosting.com is pursuing a similar method with the addition of Coyote Point's Dedicated Load Balancing solution [6], which is expected to improve application response time and availability as well. Rackspace is also testing load balancing method [7].

Technical goals and business goals for Cloud balancing are given in [1]. Cloud balancing uses a global application delivery solution to determine, on a per user/customer basis, the best location from which to deliver an application. Cloud balancing should include traditional global server load balancing parameters such as:

(1) Application response time.
(2) Location of the user.
(3) Availability of the application at a given implementation location.
(4) Time of day.
(5) Current and total capacity of the data center/Cloud computing environment in which an application is deployed.

Additionally, Cloud users must consider more variables including:

(1) Cost to execute the request at a given location.
(2) Total cost to deliver the request to a user/customer.
(3) Regulatory compliance and/or legal restrictions.
(4) Services required by the user/customer to fulfill the request based on contractual obligations.

The paper pays attention to Cloud balancing methods. We select Cloudsim [9, 14] as our research platform because Cloudsim provides load balancing parameters and the parameters of Cloud users that have been introduced above. Simulations on Cloudsim are executed to test our methods. A true log from [45] also is used to test the performance of our method.

The rest of the paper is organized as follows. In the next section, we introduce some proposed load balancing schemes of the distributed computing systems and especially on Grid and Cloud. Section 3 introduces simulation on Cloudsim. In Section 4, we present our Cloud balancing scheme. Section 5 gives the simulation and the comparison to Min-min [29], Max-min [29], FPLTF (Fastest Processor to largest Task First) [30] and LBEGS [28]. In section 5, we also test our method on a true log of [45]. Finally, section 6 concludes the paper

2. Related Work

Cloud Balancing is the new coming issue, so we give an introduction to tradition load balancing in distributed system and Grid which has been discussed in traditional distributed systems literature for more than three decades and relates to Cloud computing, and then some methods of load balancing in Cloud are discussed in detail.

Various methods and algorithms of load balancing have been proposed, implemented and classified in a number of studies [10-11] in traditional distributed systems. A load balancing algorithm attempts to improve the response time of user's submitted applications by ensuring maximal utilization of available resources. The main goal is to prevent the condition where some processors are overloaded with a set of tasks while others are lightly loaded or even idle [12]. Load balancing can be defined by their implementation of the following policies [13]: (1) Information policy specifies what workload information to be collected, when it is to be collected and from where; (2) Triggering policy determines the appropriate period to start a load balancing operation; (3) Resource type policy classifies a resource as a server or receiver of tasks according to its availability status; (4) Location policy uses the results of the resource type policy to find a suitable partner for a server or receiver; (5) The selection policy defines

 Copyright ⓒ 2014 SERSC

the tasks that should be migrated from overloaded resources (source) to most idle resources (receiver).

Some load balancing methods in traditional distributed systems also can be used on Grid and Cloud computing through there is a difference between traditions distributed computing, Grid and Cloud [5]. The structure of the hierarchical portion of the taxonomy about load balancing in general purpose distributed computing system is given in [10]. Then, let us introduce some methods for load balancing, especially for Grid and Cloud computing:

People have proposed some load balancing methods in Grid. Several papers discuss the application of the DLT (Divisible Load Theory) to Grid computing [15~18]. In [19], C. Ruay-Shiung, *et al.*, proposes a balanced Ant Colony Optimization (BACO) algorithm for job scheduling in the Grid environment. The main contributions of their work are to balance the entire system load while trying to minimize the makespan of a given set of jobs. In [20] each job submitted to the Grid invokes an ant and the ant searches through the network to find the best node to deliver the job to. Ants leave information related to the nodes they have seen as a pheromone in each node which helps other ants to find lighter resources more easily. The Routing Tables method, previously examined in [21], suggests a way of directing resource requests in a Grid network. Based on a Grid-router model, each router maintains routing tables in order to direct requests for certain resources in a Grid system. I. Konstantinos, *et al.*, [22] extend the framework, creating the Re-routing Tables method in a de-centralized manner, focusing on its effectiveness in a dynamical Grid, where resources are not permanently online.

Especially, LBEGS (Enhanced GridSim architecture with load balancing) was proposed by K. Qureshi, *et al.*, [28] which is based on Gridsim. It gives the load level calculation algorithms of PE, machine, resource. At the same time, it also gives the algorithms of balancing between PEs, machines and resources. The proposed scheme not only reduces the communication overhead of Grid resources but also cuts down the idle time of the resources during the process of load balancing based on GridSim. They gauge the effectiveness in terms of communication overhead and response time reduction and simulations show that significant savings are delivered by their proposed technique compared to other approaches such as centralized load balancing and no load balancing. We find it also can be used on Cloudsim and in the paper we will use it to compare with our Cloud balancing algorithm.

Cloud balancing also brings new views on balancing because Cloud is virtualized [35]. Some organizations have provided support for Cloud which we have discussed in Section 1 [3, 4, 7]. Some load methods in traditional distributed computing and Grid also can be used in Cloud computing. Researchers also propose new methods in Cloud computing.

F. Yoke, *et al.*, [23] propose a two levels task scheduling method based on load balancing in Cloud computing and it also has been discussed in [24]. The first level scheduling is from the users' application to the virtual machine, and the second level is from the virtual machine to host resources. In these two levels scheduling model, the first scheduler creates the task description of a virtual machine, including the task of computing resources, network resources, storage resources, and other configuration information, according to resource demand of tasks. Then the second scheduler find appropriate resources for the virtual machine in the host resources under certain rules, based on each task description of the virtual machine. Via the two levels scheduling, the task can obtain the required resources, and it would not lead to the resource allocated to some tasks is less than the requirement and increase the implemental time while others are more than their requirements and lead to the waste of resources.

Copyright ⓒ 2014 SERSC

Martin, *et al.,* investigates three possible distributed solutions proposed for load balancing in Cloud [25]: approaches inspired by Honeybee Foraging Behavior [31, 34], Biased Random Sampling [32] and Active Clustering [33].

One of the key benefits is a better load balancing by using of VM migration between hosts. To migrate, we must know which virtual machine needs to be migrated and when this relocation has to be done and, moreover, which host must be destined. In [34], S. Jing, *et al.,* proposed a novel model for this problem based on fuzzy TOPSIS to detect the hotspots and balance load.

W. Kleiminger, *et al.,* [36] present a combined stream processing system that, as the input stream rate varies, adaptively balances workload between a dedicated local stream processor and a cloud stream processor. It is possible to construct a combined stream processing system that uses the resources of a cloud infrastructure to assist a local stream processor. The combined approach scales well with increasing input rates by using cloud resources and achieves increased throughput. This approach only utilizes cloud machines when the local stream processor becomes over loaded. They have evaluated a prototype system with financial trading data. Their simulation results show that it can adapt effectively to workload variations, while only discarding a small percentage of input data.

In [41], S. Wang, *et al.,* proposed a Three-level Cloud Computing Network. The three-level are request manager, service manager and service node. The goal of this study is to reach load balancing by OLB (Opportunistic Load Balancing) scheduling algorithm, which makes every node in working state. Besides, in their research, the LBMM (Load Balance Min-Min) scheduling algorithm is also utilized to make the minimum execution time on the node of each task and the minimum whole completion time is obtained. However, the load balancing of three-level cloud computing network is utilized; all calculating results could be integrated first by the second level node before sending back to the management. Thus, the goal of loading balance and better resource manipulation could be achieved.

In this paper, we pay attention to the load balancing on Cloudsim and propose a three levels control methods based on Cloudsim. The three levels include PEs, Hosts, and Datacenter. Every level checks their state periodically and scheduling between PEs ultimately to form load balancing in the resource entity and ultimately form load balancing of the whole system (From Physical Structure of Cloud, Figure 1).

3. Simulation of Scheduling on Cloudsim

The Cloudsim toolkit supports both system and behavior modeling of Cloud system components such as data centers, virtual machines (VMs), Processing Elements (PEs) and resource provisioning policies. Figure 1 is physical structure of Cloud. Cloudsim supports modeling and simulation of Cloud computing environments with different parameters. User can set different parameters in Figure 1. Every Host has many PEs and a data center has many hosts. Datacenter broker is in charge of the resource scheduling of Cloud.

Figure 2 is the user view of Cloud. For a Cloud user, he only pays attention the VM(s) which is assigned to him. A Datacenter can manage several hosts who in turn manage VMs during their life cycles. The host is a Cloudsim component that represents a physical computing server in a Cloud: it has parameters including speed (expressed in millions of instructions per second-MIPS), memory, storage, and a provisioning policy for allocating processing cores (or PEs) to VMs. The Host component implements interfaces that support modeling and simulation of different parameters (memory, hard disk, CPU, *etc.*,). Each host either implements the space-shared or the time-shared policy for allocating cores to VMs [9].

 Copyright ⓒ 2014 SERSC

In this paper, assumptions are given as follows: (1) One host can only be assigned to one VM; (2) We do not pay attention the memory capacity and storage capacity of every Cloudlet, only care about the instruction length of every Cloudlet.

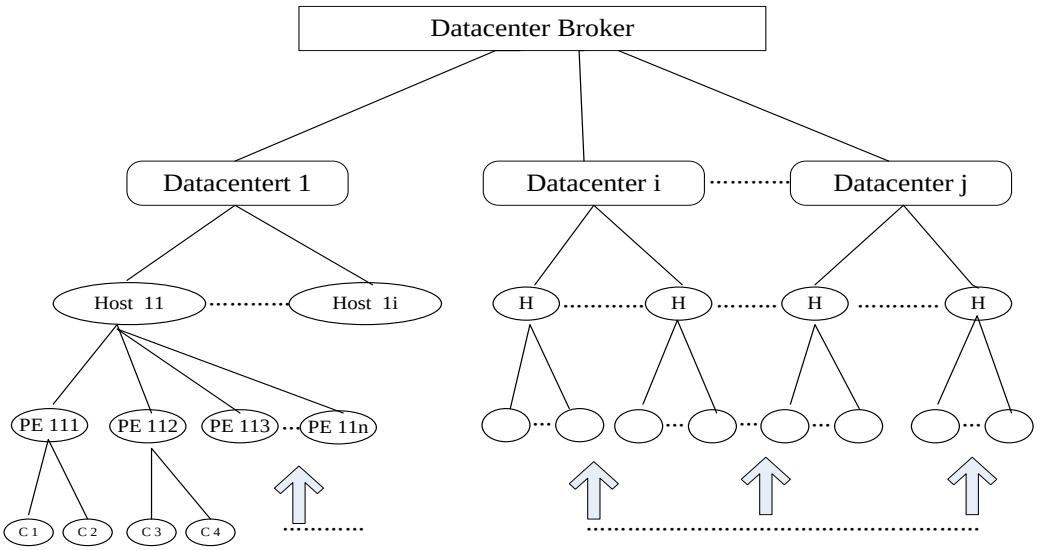

Figure 1. Physical Structure of Cloud

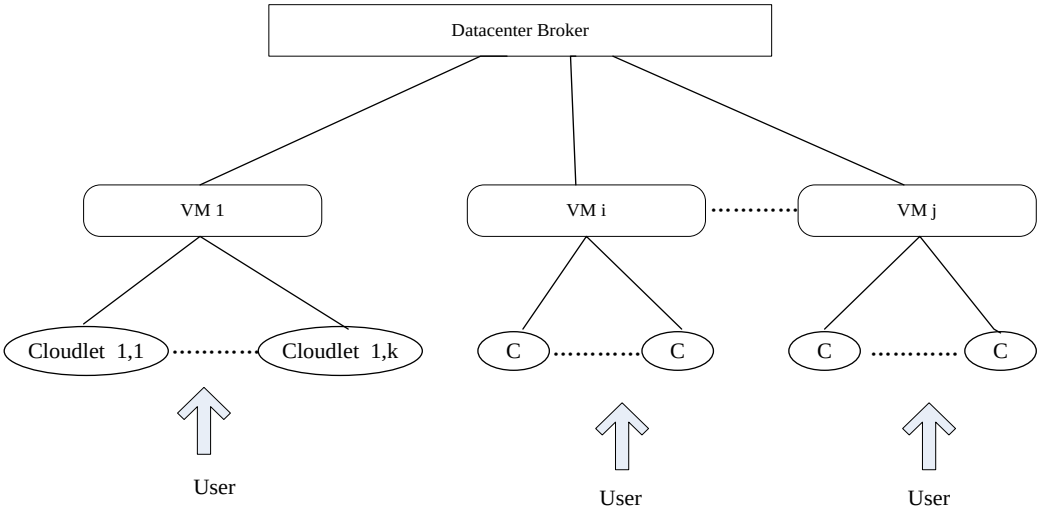

Figure 2. User View of Cloud

4. Min Standard Deviation of Cloud Load Based on Three Levels Control on Cloudsim (Minsd)

Our proposed load balancing scheme works at three levels: Datacenter, Host, PE. When a new Cloudlet comes from a user, it is submitted to a VM which is under loaded. From Figure 2, we can find that Cloudlet has been assigned to PEs ultimately. So Cloud balancing is control by the physical resource entity including Datacenters, Hosts and PEs (Figure 1). After a specific interval of time, the resource entity checks its load and classifies them into three

Copyright ⓒ 2014 SERSC

categories: "under loaded", "over loaded", and "normally loaded". The classifications of Datacenter, Host and PE according Load are listed on Table 1.

For the easy of analysis, class definitions of resource entity are given as Table 2.

d_b, h_b , p_b are the lower limit of the load of the datacenter, host and PE respectively. d_t, h_t , p_t are the upper limit of the load of the datacenter, host and PE respectively. If the load of PE/Host/Datacenter is more than $p_t/h_t/d_t$, the state of the PE/Host/Datacenter is "over loaded"; if the loaded of PE/Host/Datacenter is less than $p_b/h_b/d_b$, the state of the PE/Host/Datacenter is "under loaded"; otherwise, the state of the PE/Host/Datacenter is "normally loaded".

Table 1. The Classification of Datacenter, Host, PE and VM According to Load

Resource entity	Requirement	Under loaded	Normally loaded	Over loaded	ParentResourceEntity
Datacenter	$[d_b,d_t]$($d_b<d_t$)	$< d_b$	$[d_b,d_t]$	$> d_t$	*DataCenter Broker*
Host	$[h_b,h_t]$($h_b<h_t$)	$< h_b$	$[h_b,h_t]$	$> h_t$	$DataCenter_i$
PE	$[p_b,p_t]$($p_b<p_t$)	$< p_b$	$[p_b,p_t]$	$> p_t$	$Host_j$

Table 2. The Variables of Datacenter, Host, PE and VM According to Load

ResourceType	LowerLimit	UpperLimit	Load	ParentResource Entity
Datacenter	d_b	d_t	d	*DataCenter Broker*
Host	h_b	h_t	h	$DataCenter_i$
PE	p_b	p_t	p	$Host_j$

The states of a PE/Host/Datacenter are categorized into three kinds based on the load: under loaded, normally loaded, and over loaded and they are denoted by 0, 1 and 2 separately. We can forecast the CPU load [26] on Windows or Unix OS [27]. In [44], Z. Yuanyuan, *et al.*, has presented a running time prediction method for grid tasks based on their previous work. So it is not difficult to forecast the load of resource entity (PE/Host/Datacenter) by the resource entity itself, and then it adds the load from others Cloudlets which is assigned to the resource entity; we get the new load of the resource. We get his new state of the resource entity by checking the value of the load (As algorithm 1.). When we find the state of the resource entity changed, we post a request to its parent resource entity to change the state of the resource record. Algorithm 1 is the detail:

Algorithm 1: Check state(Resource entity *r*)

```
1   loadtemp=Get the load of r ;
2   statetemp=r.state;
3   If (loadtemp<r. LowerLimit)
4     newstate=0;
5   Else
6     If (loadtemp>r.UpperLimit)
7         newstate=2;
8     Else
9       newsate=1;
10    Endif
11  Endif
12  If(statetemp!=newstate)
```

 Copyright ⓒ 2014 SERSC

```
13    r.state=newstate;
14   If(newstate==0)
15      Trigger r. ParentResourceEntity that r is "under loaded";
16   Endif
17      if(newstate==1)
18        Trigger r. ParentResourceEntity that r is "normally loaded";
19      Endif
20      if(newstate==2)
21        Trigger r. ParentResourceEntity that r is "over loaded";
22      Endif
23  Endif
```

There are three lists of every resource entity (Host, datacenter, Datacenter broker) which record different state of the resource entity of next level (Such as in Figure 1. The next level of the datacenter broker includes datacenter *1* to datacenter *j*. The next level of datacenter *1* includes host *11* to host *1i*. The next level of Host *1* includes PE *111* to PE *11n*). *Under loaded list*/*Normally loaded list*/*Over loaded list* are used to record the resource entity whose state is under loaded/normally loaded/over loaded. When a resource entity *parent* gets "*under loaded/ normally loaded/ over loaded*" that which is triggered from resource entity *r* (Algorithm 2, lines 2-3), the resource leaves the list which it belongs to (Algorithm 2, line 4) and inserts into the list that the resource should belong to (Algorithm 2, line 5). Scheduling is as follows:

Algorithm 2: resource entity *parent* scheduling resource algorithm

```
1…. Begin
2     If Resource Entity parent receives "under loaded/normally loaded/overloaded" from a Resource
Entity r  then
3        Deletes r from the list which it belongs to in parent;
4        Inserts the r into Under loaded list/ Normally edlist/ Over loaded list of Resource Entity parent;
5     End if
6     If parent receives "over loaded " message
7          Load Balancing parent）;
8     End if
9   End
```

"Load Balancing (*parent*)" is used to balance that PEs in the same Host, Datacenter or Datacenter broker and this happens only when the Host (or Dacenter or Dater center broker) is over loaded (line 7-9).

Suppose that resource entity (Host/Datacenter/Data center broker) r_p has hosts as {$Host_{p,1}$,$Host_{p,2}$,…,$Host_{p,temp}$,… $Host_{p,m}$}.

$Host_{p,temp}$ has $numPEt_{p,temp}$ PEs and each with *rating* $PEt_{p,temp}$ processing ability. So the processing capacity of r_p can be calculated as follows:

$$r_p.cp = \sum_{temp=1}^{m} numPE_{p,temp} \times ratingPE_{p,temp} \quad (1)$$

Suppose that the load of resource entity *p* is l_p. In this paper, we only pay attention the instruction length （i.e. computational Cloud） and we call it Cloud let Length in Cloudsim. When a new Cloudlet *g* is assigned to *l*. The new loaded of *l* can be calculated as:

$$l_p=l_p+load \ (g.\ Cloud\ Length,\ resource\ entity_{\ p}.\ capacity) \quad (2)$$

Formula 2 is used to calculate the loaded of PE/Host/Datacenter. Load (*g.Cloudletlength*, r_p.*cp*) is inversely proportional to r_p.*cp* and is proportional to *g. Cloudletlength*.

Copyright ⓒ 2014 SERSC

When we calculate standard deviation of load, we pay attention to PE which it belongs to the Host/Data center/Data center broker. Suppose that every PE shared the same burden of instructions number of an assigned Cloudlet. PEs belong to the Host/Datacenter/Data center broker are { PE_1, $PE_2 \ldots PE_{temp} \ldots PE_{end}$}.

The standard deviation of l_p is given by:

$$\sigma_r = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(l_i - \bar{l})^2} \quad (3)$$

Where σ is the standard deviation of loaded (%), *end* is the number of PEs, l_i is the loaded of resource *i*, and $\bar{l}$ is the average load of each PE. If the standard deviation value of a method is small, it means that the difference of each load is small. The small standard deviation tells that the load of the entire system is balanced. The lower value the standard deviation has, the more load balanced the system has.

Our load balancing method is: if we can find a scheduling that makes the standard deviation the minimum value, we schedule it. The standard deviation load may be more or less than before scheduling. We select the scheduling which makes the selected resource state is normally loaded or under loader and insures the standard deviation reduce most; otherwise, we select the scheduling whose standard deviation increases the least and the scheduling insures the state of the resource is "normally loaded" or "under loaded". Algorithm 3 gives the detail.

Algorithm 3: Load Balancing Resource Entity *parent*)

```
1… While-(under loaded list is not empty and unassigned cloud list is not empty)
2    σ = Get standard deviation load of parent
3    minσ1 = +∞ , minσ2 = σ , statechange1=0, statechange2=0;
4    For every Resource Entity r in under loaded list
5      For every Cloudlet g in unassigned grid list l
6        Calculate newσ = suppose Cloudlet g assigned to resource entity r
7        If( σ > newσ )
8          statechange1=1;
9          If( minσ1 > newσ )
10           If the state of r is "normally loaded" under condition of g is assigned to r
11             statechange11=1;
12             minσ1 = newσ ;
13             Assignedresource1=r;
14             Assignedgridlet1=g;
15           Else
16             If( statechange11==0)
17               minσ1 = newσ ;
18               Assignedresource2=r;
19               Assignedgridlet2=g;
20             End if
21           End if
22         End if
23       End if
24     If( σ < newσ &statechange1==0 )
25       statechanage2=1;
26       If( minσ2 > newσ )
27         If the state of r is "normally loaded" under condition of g is assigned to r
28           statechanage21=1;
29           minσ2 = newσ ;
```

 Copyright ⓒ 2014 SERSC

```
30              Assignedresource3=r;
31             Assignedgridlet3=g;
32           Else
33              If( statechange21==0)
34                 min σ1 = new σ ;
35                 Assignedresource4=r;
36                 Assignedgridlet4=g;
37              End if
38            End if
39           End if
40        End if
41       End for
42    End for
43    If (statechanage1==1 & statechanage1==1)
44      Assigning Assignedgridlet1 to Assignedresource1 ;
45    End if
46    If (statechanage1==1 & statechanage1==0)
47      Assigning Assignedgridlet2 to Assignedresource2 ;
48    End if
49    If(statechanage2==1  & statechanage1==1)
50     Assigning Assignedgridlet3 to Assignedresource3 ;
51    End if
52    If(statechanage2==1  & statechanage1==0)
53     Assigning Assignedgridlet4 to Assignedresource4;
54    End if
55 End while
```

Lines 4~5 are the initialization. $min\sigma 1$ and $min\sigma 2$ record the minimum when all kinds of scheduling makes the standard deviation smaller or not. statechange1 (==1) records the condition that some scheduling makes the standard deviation become smaller. On the contrary, statechange2 (==1) records the condition that there are not any scheduling can make the standard deviation become smaller. Lines 10-28 are the solution for the first condition and lines 28-49 for the second condition. On the first condition, if some scheduling make one resource change from under loaded to over loaded, we select the scheduling (lines 13-20); otherwise, we select the scheduling (line 13-20) which keeps the standard deviation as minimum (lines 22-26). For the second condition, if some scheduling make one resource change from under loaded to normally loaded, we also select the scheduling (lines 33-40); otherwise, we select the scheduling (lines 13-20) which keeps the standard deviation as minimum (lines 42-46). Lines 52~63 is the assignment details.

From the analysis, we can find that although the Cloud balancing can happen in PEs, Hosts and Datacenters, the balancing happens in PEs level ultimately. Every VM, in fact, is consisting of some PEs.

This is actually a centralized resource management, but whose scalability is not a problem when there are large-scale computer systems including data centers for the cloud. The three levels bring convenience for management. For every Host, only need to know the load information of PEs which belongs to the Host. For a Datacenter, it only needs the summary load information of Hosts which belongs to the Datacenter and it does not need the details of load information. The same method is taken by the Datacenter broker who only needs summary load information of the datacenter. The three level controls also bring easiness for the management of resource entity.

Copyright ⓒ 2014 SERSC

In fact, considering the standard deviation of load distributions and searching all possible task assignments, this may be costly. But depending on the workload at any specific moment, the system has the ability to lease new VMs up to a total number of 120. This is a limitation posed by Amazon EC2 which allows up to 20 "Regular" and up to 100 "Spot" VMs which can be leased under certain conditions [43], hence virtually up to 120 VMs. For this limit, to get the standard deviation is not a problem. In addition, the load balance scheduling happens only when there are resource entities in over loaded state and it does not happen anytime. We take max-min method when the resource is under loaded.

5. Simulations

One of the most common measures in evaluating the performance of a Grid is measuring the makespan. The makespan is the "total application execution time" [19]. The total application execution time is measured from the time that the job is sent to the Cloud to the time that the job comes out of the Cloud. The small standard deviation tells that the load of the entire system is balanced and it also shows that the balancing level of the Cloud has. The lower value of the standard deviation has, the more load balanced the system is [10]. The communication overheads are calculated by counting the number of messages over PEs, Hosts, Datacenters and we calculated by counting the number of messages over the different resource entity (Datacenters/Hosts/PEs) [28]. Job throughput which is the number of completed jobs per elapsed time and it has been widely used [25] to test the performance of load balancing methods. So, makespan, standard deviation of loaded, the communication overheads and job throughput are selected as four standards for our simulations.

There are many load balancing methods in the distributed computing system and Grid. Some of them can be used in Cloud balancing. We select Min-min [29], Max-min [29], FPLTF (Fastest Processor to largest Task First) [30] and LBEGS [28] to compare with our method. Min-min [29] set the tasks which can be completed earliest with the highest priority. The main idea of Min-min is that it assigns tasks to resources which can execute tasks the fastest. LBEGS has been introduced in Section 3.

Suppose that every PE has a processing power with a random number between 1 and 10 (one million instructions per second is assigned to Grade 1, two million instructions per second is assigned to Grade 2, *etc.*,) Without loss of generality, we set the local load factor for resources to be zero; this does not affect the performance measure of the algorithms. We set it to zero to help us better analyze the effect and behavior of the algorithms. Resource load threshold parameters and Cloud parameters are listed on Table 3.We can express the Cloudlet length in a random number between 1 and 10 (1 Million Instructions is assigned to Grade 1, 2 Million Instructions is assigned to Grade 2, *etc.*,).

Table 3. The Parameters of the Simulation

Cloudlet parameters	Load Level		
Cloudlet length	Resource	Machine	PEs
[1,10] PEs	0.6-0.65	0.75-0.80	0.8-0.85

In addition, as mention in formula (2), we calculate the load when Cloudlet g is assigned to resource entity p as follows:

$$load(g.CloudletLength, r_p.cp) = \frac{g.CloudletLength}{r_p.capacity \times r_{p,g}.t} \quad (4)$$

$r_{p,g}.t$ is the time that from the time that g is executed on p to the time that g is finished .

 Copyright ⓒ 2014 SERSC

Section 5.1 and Section 5.2 give a simulation with constant Cloudlets and with a constant PEs. Section 5.3 gives a simulation with a true log [45].

5.1 Simulations with Constant Cloudlets

The readings are taken by varying the number of PEs starting at 200 and ending at 1200 with a step of 200, *i.e.*, the number of datacenter starting at 10 to 60 with a step of 10. The numbers of Cloudlets is kept constant at 200000. The communication overheads are calculated by counting the number of messages ("over loaded") between resource entities.

Makespan of different methods are shown on Figure 3. In general, before the number of PEs reaches 400, Min-min and Minsd perform well and the makespan is the least, LBEGS is the worst; when the number of PEs is more than 400, LEBGS is the best method and FPLTF is the worst, the others are middle and get the same performs basically. From Figure 3, we know that LBEGS performs well when the resource is enough. But when the resource is not enough, it doesn't perform well as wishes. Minsd performs well on any condition relatively.

The times of communication over head of different methods are shown on Figure 4. Generally, for FPLTF, the overhead times drop dramatically when the number of PEs between 200 and 600 and then keep dropping slowly; for others, the value always keeps at a low value between 0 and 10.

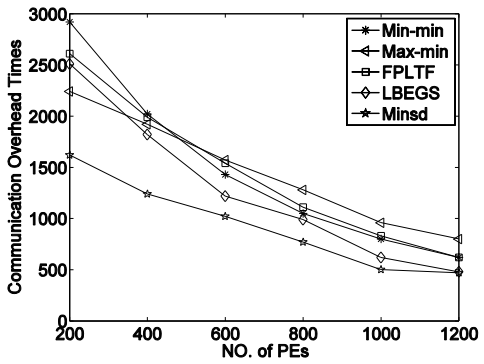

Figure 3. Makespan

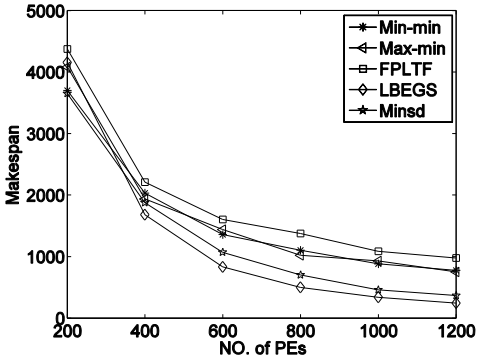

Figure 4. Communication Overhead Times

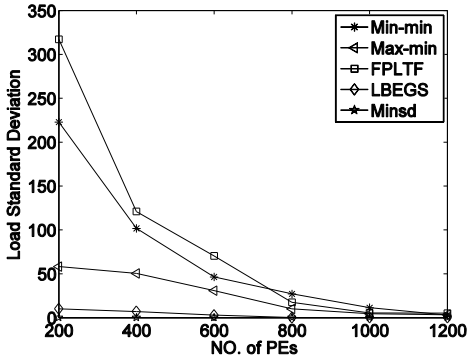

Figure 5. SDL

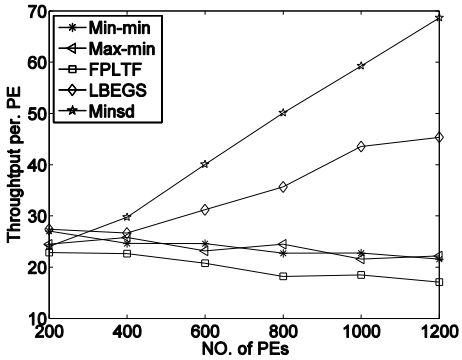

Figure 6. Throughput Per. PE

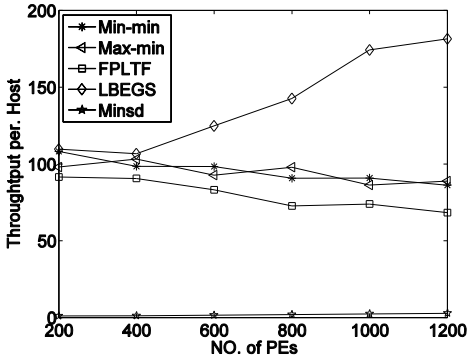

Figure 7. Throughput Per Host

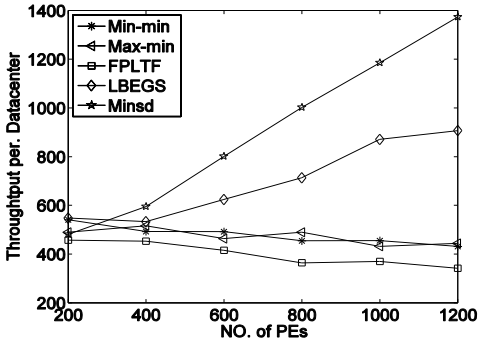

Figure 8. Throughput Per. Datacenter

Standard deviations of loaded (SDL) of different methods are shown on Figure 5. In general, for FPLTF, Min-Min, Max-Min, the values of standard deviation drop dramatically when the number of PEs between 200 and 600 and then keep dropping slowly; for LEBGS and Minsd, the value always keeps at a low value between 0 and 100. Figure 5 shows Min-Min and FPLTF perform well only when there are enough resources. If there are not enough resources, they cannot provide balancing of the Cloud. LEBGS and Minsd perform well on any conditions and Minsd performs better when the system is busy (such as the number of PEs is 200). Figure 5 is the standard deviation of loaded of different algorithms. In general, the values of all methods drop with the increasing of PEs. LBEGS and Minsd have the best performance in all of them.

In conclusion, when the number of Cloudlets is a constant, Minsd provides good Cloud balancing, and at the same time, it reduces makespan and overhead communication.

Figure 6-8 are the job throughput of PE, Host and Datacenter. The throughput of Min-min, Max-min and FPLTF are kept as a constant basically. The value of PE/Host/Datacenter of them is about 20, 100, and 500 separately. The throughput of LBEGS and Minsd increases with the increasing of PEs. Minsd increases more quickly than LBEGS.

From Figure 3-8, it is not difficult to find that LEBGS although can provide balancing at most of time, but the overhead times is too many especially when the resource is not enough (before the number of PEs reaches 600). LEBGS provides stricter rules for loaded balancing and avoids this before the resource is "over loaded".

5.2 Simulations with Constant PEs

The results are taken by varying the number of Cloudlets starting at 40000 and ending at 140000 with a step of 20000, and the numbers of PEs are kept constant at 100, *i.e.* , there are 5 datacenters in the Cloud. The communication overheads are calculated by counting the number of messages ("over loaded") between the resource entities.

Makespans of different methods are shown on Figure 9. Generally, the makespan increases with the increasing of the number of Cloudlets. Minsd performs the best and LBEGS is the better.

The times of communication overhead of different methods are shown on Figure 10. In general, for Min-min, Max-min and FPLTF, the overhead times increase dramatically with the number of Cloudlets increasing; the value of Minsd and LBEGS always keep at a low level between 0 and 1500 and increases slowly with Cloutlet increasing.

Standard deviations of loaded of different methods are shown on Figure 11. The value of standard deviation of Max-min increases dramatically with the number of Cloudlet

 Copyright ⓒ 2014 SERSC

increasing. The values of all the methods keep at a level between 0 and 100. LBEGS and Minsd always provide the best balancing on any conditions.

Figure 12-14 are the job throughput of PE, Host and Datacenter. The throughput of Min-min, Max-min and FPLTF are increasing slowly and even keeping as a constant. The value of PE/Host/Datacenter of them is about 60, 280, and 1200 separately. The throughputs of LBEGS and Minsd increase with the increasing of PEs and Minsd increases more quickly than LBEGS before Cloudlets is more than 80000. After the number of Cloudlets comes to 80000, the throughput of Minsd keeps as a constant and LBEGS increases and the value is less than Minsd.

In conclusion, Minsd provides Cloud balancing on most of the time and it also gives support to minimize the overhead, makespan, standard deviation of loaded and enhances job throughput. FPLTF also gives good performs as Minsd and its balancing level is less than Minsd relatively.

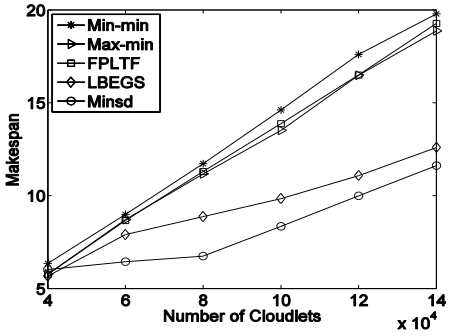

Figure 9. Makespan

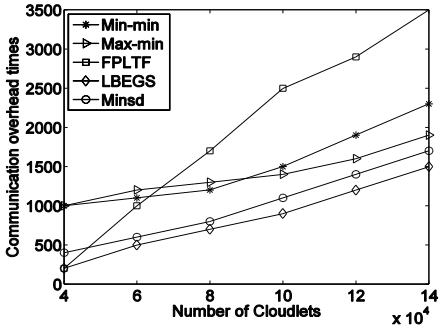

Figure 10. Communication Overhead Times

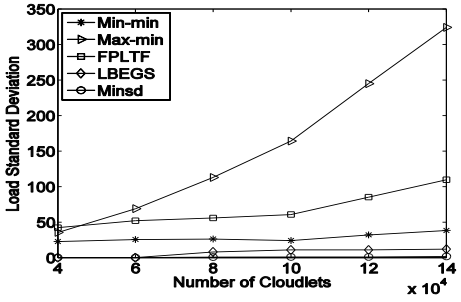

Figure 11. Standard Deviation of Load

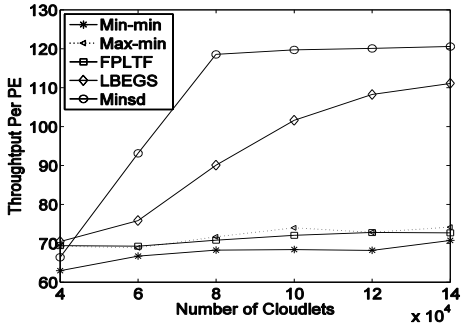

Figure 12. Throughput Per. PE

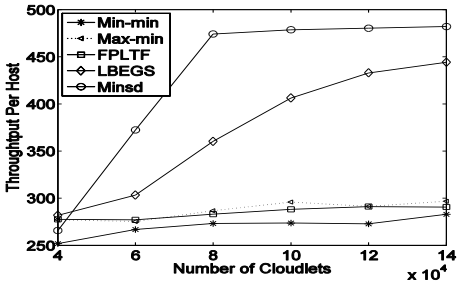

Figure 13. Throughput Per Host

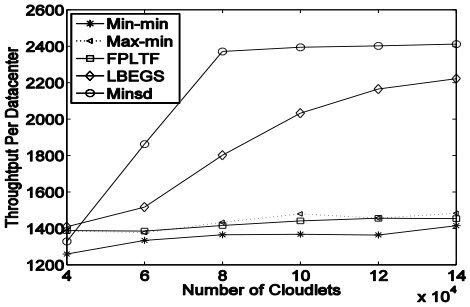

Figure 14. Throughput Per. Datacenter

Copyright ⓒ 2014 SERSC

5.3. Simulations on a True Log

This log contains several months accounting records from the RICC installation in Japan. RICC (RIKEN Integrated Cluster of Clusters) is composed of 4 clusters, and was put into operation in August 2009. The data provided here is from the " massively parallel cluster", which has 1024 nodes, each with 12 GB of memory and two 4-core CPUs, for a total of 12 TB memory and 8192 cores [45]. We suppose that the resources belong to space-shared. The comparisons only are taken between LBEGS and Minsd. The results are listed on Table 2. Variance of datacenter (varDC), variance of host (varhost), variance of PE (varPE), makespan and average waiting time of every job (Avgwtime) are shown in Table 4. In the log, some jobs fail and we delete those jobs first. The log does not give detailed information of job assignment so we cannot get the variance of datacenters, hosts and PES. We use "-"to express the result.

Table 4 shows that the variance of Datacenters of Minsd is less than LBEGS. At the same time, Minsd also has a lower value in makespan though it has a higher value in average waiting time (Avgwtime). In fact, we find the jobs in the log are big job which need much time and more PEs. We also test the load of every datacenter. Table 5 shows that the capacity of jobs of every datacenter and it tells the reason why the variance of datacenter of Minsd is the minimum. (The capacity of a job=Number of Allocated Processors* Average CPU Time Used. See [45]).

Table 4. Simulation of LBEGS and Minsd on Log

	varDC	Makespan(s)	Avgwtime (s)
LBEGS	4.4756e+12	6.3298e+06	4.7896e+06
Minsd	2.1196e+12	6.3171e+06	4.7995e+06
log	-	8.4600e+006	1.6155e+06

From the analysis of the true log of RICC, Minsd has relative better performance than LBEGS.

Table 5. Simulation results of every datacenter

		Cluster 1	Cluster 2	Cluster 3	Cluster 4
Load	LBEGS	2.1412e+10	2.1309e+10	2.1479e+10	2.1396e+10
	Minsd	2.1399e+10	2.1400e+10	2.1399e+10	2.1398e+10
Var	LBEGS	2.5727e+8	6.7427e+7	2.5146e+7	2.1279 e+7
	Minsd	2.2336e+2	5.7426 e+1	1.9727e+1	1.3317e+2

6. Conclusions and Future Work

This paper proposes Minsd for Cloud balancing and used it on PES/Hosts/Datacenters. Simulations indicate that it has relative good performance in makespan, communication overhead and standard deviation. Future work includes taking more parameters into account such as memory capacity, bandwidth. Router table [21, 22] considers the route selection and bandwidth also brings a new view of load balancing in Cloud. We hope to extend our load method on the router Table. The ultimate goal of Cloud balancing is to deliver an application to a user as quickly as possible with the least amount of resources and for the lowest cost [1], so we need to pay attention to the cost of customers and load balancing in the future. The major benefit of VM migration is to avoid hotspots [8] and we also can apply Minsd to VM migration but need more attentions for more parameters. The proposed load level detection and resource allocation algorithms do not share the assumption that the global information

 Copyright ⓒ 2014 SERSC

about available resources from all nodes in the cloud is available and accurate. Some nodes with wrong data do not influence the result too much. How to check node whether valid is a problem for our solution in the future.

References

[1] http://www.f5.com/pdf/white-papers/cloud-balancing-wp.pdf [Visited: 7-11-2012].
[2] http://www.itbusinessedge.com/cm/blogs/cole/load-balancing-in-the-cloud/?cs=45781 [Visited: 12-11-2012].
[3] http://www.zeus.com/solutions/cloud-balancing [Visited:9-11-2012].
[4] http://aws.amazon.com/ec2 [Visited: 2-11-2013].
[5] P. Mell and T. Grance, "NIST definition of cloud computing", National Institute of Standards and Technology, **(2009)** October 7.
[6] http://www.sfgate.com/cgi-bin/article.cgi?f=/g/a/2011/02/22/prweb5097954.dtl [Visted:2-2-2013].
[7] http://www.infoq.com/news/2011/02/rackspace-loadbalancers-beta [Visited: 2-11-2012].
[8] Q. Zhang, L. Cheng and R. Boutaba, "Cloud computing: state-of-the-art and research challenges", Journal of Internet Services and Applications, vol. 1, no. 1, **(2010)** May, pp. 7-18.
[9] N. C. Rodrigo, R. Rajiv, B. Anton, A. F. D. R. Cesar and B. Rajkumar, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", Software: Practice and Experience (SPE), vol. 41, no. 1, **(2011)** January, pp. 23-50.
[10] T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling in general purpose distributed computing systems", IEEE Transactions on Software Engineering, vol. 14, no. 2, **(1994),** pp. 141–153.
[11] M. J. Zaki, W. Li and S. Parthasarathy, "Customized dynamic load balancing for a network of workstations", In Proc. of the 5th IEEE Int. Symp. HDPC, **(1996),** pp. 282–291.
[12] X. Cheng-Zhong and C. L. Francis, "Load Balancing in Parallel Computers: Theory and Practice", Kluwer, Boston, MA, **(1997).**
[13] H. D. Karatza, "Job scheduling in heterogeneous distributed systems", Journal of Systems and Software, vol. 56, **(1994),** pp. 203–212.
[14] W. Bhathiya, N. C. Rodrigo and B. Rajkumar, "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications", aina, 2010 24th IEEE International Conference on Advanced Information Networking and Applications, **(2010),** pp. 446-452.
[15] K. Seonho and B. W. Jon, "A Genetic Algorithm Based Approach for Scheduling Decomposable Data Grid Applications", icpp, 2004 International Conference on Parallel Processing (ICPP'04), **(2004),** pp. 406-413.
[16] Z. Tao, W. Yongwei and Y. Guangwen, "Scheduling divisible loads in the dynamic heterogeneous Grid environment", In: Proceedings of the 1st International Conference on Scalable Information Systems, Hong Kong, **(2006).**
[17] Y. Chen, C. M. Dan, "Algorithms for Divisible Load Scheduling of Data-intensive Applications", J Grid Computing, vol. 8, **(2010)**, pp. 133–155.
[18] S. Bataineh and T. G. Robertazzi, "Distributed computation for a bus network with communication delays", In: Proc. Conf. Information Sciences and Systems, Baltimore, MD., **(1991)**
[19] C. Ruay-Shiung, C. Jih-Sheng, L. Po-Sheng, C. Ruay-Shiung, C. Jih-Sheng and L. Po-Sheng, "An ant algorithm for balanced job scheduling in grids", Future Generation Computer Systems, vol. 25, **(2009)**, pp. 20–27.
[20] A, L. Simone and M. Azin, "Swarm Intelligence Approaches for Grid Load Balancing", Journal of Grid Computing, vol. 9, Issue 3, **(2011),** pp. 279-301.
[21] L. Wei, X. Zhiwei, D. Fangpeng and Z. Jun, "Grid Resource Discovery based on a Routing-Transferring Model", Grid 2002, LNCS, vol. 2536, **(2002),** pp. 145–156.
[22] I. Konstantinos, Karaoglanoglou and D. K. Helen, "Resource Discovery in a dynamical grid based on Re-routing Tables", Simulation Modelling Practice and Theory, vol. 16, Issue 6, **(2008)**, pp. 704-720.
[23] F. Yiqiu, W. Fei and G. Junwei, "A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing", WISM 2010, LNCS, vol. 6318, **(2010)**, pp. 271–277.
[24] S. Sadhasivam, N. Nagaveni, R. Jayarani and R. V. Ram, "Design and Implementation of an efficient Two-level Scheduler for Cloud Computing Environment", In: International Conference on Advances in Recent Technologies in Communication and Computing, vol. 148, **(2009)**, pp. 884–886.
[25] R. Martin, L. David, A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", waina, 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, **(2010)**, pp. 551-556.
[26] Z. Yuanyuan, S. Wei and I. Yasushi, "CPU Load Predictions on the Computational Grid", ccgrid, Sixth IEEE International Symposium on Cluster Computing and the Grid, **(2006)**, pp. 321-326.
[27] R. Wolski, N. Spring and J. Hayes, "Predicting the CPU availability of time-shared Unix systems on the

Copyright ⓒ 2014 SERSC

computational grid", Cluster Computing, vol. 3, no. 4, **(2000),** pp. 293-301.
[28] K. Qureshi, A. Rehman and P. Manuel, "Enhanced GridSim architecture with load balancing", The Journal of Supercomputing, vol. 57, no. 3, **(2010),** pp. 265-275.
[29] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen and R. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing system", Journal of Parallel and Distributed Computing, vol. 59, **(1999)**, pp. 107–131.
[30] D. Saha, D. Menasce and S. Porto, "Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures", Journal of Parallel and Distributed Computing, vol. 28, no. 1, **(1995),** pp. 1–18.
[31] R. Martin, A. Taleb-Bendiab and L. David, "Scalable Self-Governance Using Service Communities as Ambients", In Proceedings of the IEEE Workshop on Software and Services Maintenance and Management (SSMM 2009) within the 4th IEEE Congress on Services, IEEE SERVICES-I, **(2009)** July 6-10, Los Angeles, CA.
[32] O. Abu-Rahmeh, P. Johnson and A. Taleb-Bendiab, "A Dynamic Biased Random Sampling Scheme for Scalable and Reliable Grid Networks", INFOCOMP - Journal of Computer Science, ISSN 1807-4545, vol. 7, no. 4, **(2008),** pp. 01-10.
[33] F. Saffre, R. Tateson, J. Halloy, M. Shackleton and J. L. Deneubourg, "Aggregation Dynamics in Overlay Networks and Their Implications for Self-Organized Distributed Applications". The Computer Journal, **(2008)** March 31,
[34] S. Nakrani and C. Tovey, "On Honey Bees and Dynamic Server Allocation in Internet Hosting Centers", Adaptive Behavior 12, **(2004),** pp. 223-240.
[35] S. Jing, K. She, "A Novel Model for Load Balancing in Cloud Data Center", Journal of Convergence Information Technology, vol. 6, no. 4, **(2011)** April.
[36] W. Kleiminger, E. Kalyvianaki and P. Pietzuch, "Balancing load in stream processing with the cloud", In Proceedings of ICDE Workshops, **(2011),** pp.16~21.
[37] W. Wang, G. Zeng, D. Tang and J. Yao, "Cloud-DLS: Dynamic trusted scheduling for Cloud computing", Expert Systems with Applications, ISSN 0957-4174, vol. 39, Issue 3, **(2012)** February 15, pp. 2321-2329.
[38] K. Ericson and S. Pallickara, "Efcient data IO for a Parallel Global Cloud Resolving Model", Environmental Modelling & Software, vol. 26, **(2011),** pp.1725-1735.
[39] A. Nathani, S. Chaudhary and G. Somani, "Policy based resource allocation in IaaS cloud", Future Generation Computer Systems, vol. 28, **(2012)**, pp. 94–103.
[40] C.-Z. Xu, J. Rao and X. Bu, "URL: A unified reinforcement learning approach for autonomic cloud management ", Parallel Distrib. Comput. In press.
[41] S.-C. Wang, K.-Q. Yan, W.-P. Liao and S.-S. Wang, "Towards a Load Balancing in a three-level cloud computing network," Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on , vol. 1, **(2010)** July 9-11, pp. 108-113.
[42] K. Ericson, S. Pallickara, "Adaptive heterogeneous language support within a cloud runtime", Future Generation Computer Systems, vol. 28, **(2012)**, pp.128–135.
[43] "Amazon Web Services LLC (2009) Amazon elastic compute cloud (EC2)", http://aws.amazon.com/ec2/
[44] Y. Zhanga, W. Suna and Y. Inoguchib," Predict task running time in grid environments based on CPU load predictions", Future Generation Computer Systems, vol. 24, **(2008)**, pp. 489–49.
[45] http://www.cs.huji.ac.il/labs/parallel/workload/l_ricc/index.html [Visited:2-11-2013].
[46] R. Buyya, C. Y. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future generations computer systems, vol. 25, no. 6, **(2009)**, pp. 599-616.

Author

Yongsheng Hao, he received his MS Degree of Engineering from Qingdao University in 2008. Now, he is a laboratory technician of Network Center, Nanjing University of Information Science & Technology. His current research interests include distributed and parallel computing, mobile computing, Grid computing, web Service, particle swarm optimization algorithm and genetic algorithm. He has published 8 papers in international conferences and journals.

 Copyright ⓒ 2014 SERSC

Guanfeng Liu, he is a Ph.D. candidate in computer science at Macquarie University, Australia. He received his BEng degree in computer science and technology from QingdaoUniversity of Science and Technology (QUST), P.R. China in 2005, and MEng degree in computer software and theory from Qingdao University (QDU), P.R. China in 2008. His current research focuses on trust management in online social networks.

Junwen Lu, he received his MS Degree of Engineering from Qingdao University in 2008. Now, he is a teacher of Xiamen University of technology. His current research interests include distributed and parallel computing, mobile computing, and embedded system.

Copyright ⓒ 2014 SERSC

Copyright ⓒ 2014 SERSC