International Journal of Computer Applications (0975 – 8887)
Volume 102– No.4, September 2014

Dependency Mapper based Efficient Job Scheduling and Load Balancing in Green Clouds

Jaswinder Kaur
Research Scholar, Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India

Supriya Kinger
Asst. Professor, Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India

ABSTRACT

Cloud computing is usually recognized as a technology which has significant impact on IT. However, cloud computing still has many crucial problems. In a cloud computing system, Load balancing is the most central issue in the system i.e. to distribute the load in an efficient manner. It plays a very important role in the realization of efficient and robust cloud computing platform. In this paper, new load balancing mechanisms have proposed based on character/ nature of jobs along with priority consideration. Furthermore Virtualization is considered in more practical way to avoid the wastage of resources over the network. Finally the performance of the proposed algorithm is analyzed and compared with existing Load balancing and Scheduling policies.

Keywords

Cloud Computing, Data centers, Virtualization, Load Balancing, Dependency Mapper.

1. INTRODUCTION

Nowadays the concept of Cloud has become one leading paradigm. The possibility of offering “everything as a service” (platform, infrastructure and services) has allowed companies to move their IT, previously in private, to external hosting. To be more specific, Cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable resources (e.g. Networks, servers storage, applications, and services) that can be rapidly provisioned and released with minimal management efforts or service provider interaction [1].

Fig 1: Cloud computing.

On a cloud computing platform, resources are provided as service and by needs, and it guarantees to the subscribers that it sticks to the Service Level Agreement (SLA). A cloud is constituted by various nodes which perform computation according to the requests of the clients as shown in figure1. With the increasing popularity of cloud computing, the amount of processing that is being done in the clouds is rising rapidly [2].

As the requests of the clients can be random to the nodes and they can also vary in quantity, thus result in load variation on each node. Therefore, every node in a cloud can be unevenly loaded of tasks according to the amount of work requested by the clients. This can extremely reduce the working efficiency of the cloud as some nodes are overloaded while other is under loaded. Therefore, some load balancing mechanism is needed to ensure that every computing resource is distributed efficiently and fairly [3].

The problem of Load Balancing gets further complicated when energy consumption comes in as an additional system objective. The energy consumption also varies depending on the system workload [4]. For instance, the system consumes higher energy if there are numerous newly arriving tasks within a short time interval. However, resource consumption can be kept to a minimum with proper load balancing which not only helps in reducing costs but making enterprises greener [5].

In this paper a new load balancing policy is presented based on dependency mapping with priority concept, furthermore the concept of virtualization is considered in more practical way to avoid wastage of resources along with energy efficiency to the system.

The remainder of this paper is organized as follows. In Section 2 an overview of load balancing concepts and work done by other researchers is presented. Section 3, presents the architecture of proposed algorithm and discusses its working in detail. In Section 4, Experimental result and discussions are summarizing. Finally, the paper concludes in Section 5.

2. RELATED WORK

Load balancing has been an active area of research over the years with an objective to ensure that every computing resource is distributed efficiently and fairly. There are numerous research efforts that are going on in field of load balancing. Researchers have proposed various scheduling algorithms to balance load over the network. Since the performance of load balancing algorithms are greatly influenced by number of factors like dynamicity of workload, nature of jobs, heterogeneous of resources, etc., so there is still a lot which can be improved in field of load balancing.

In traditional computing environments of distributed computing, parallel computing and grid computing, researchers have proposed a series of static and dynamic and mixed scheduling strategies [9]. Static scheduling algorithm, does not take into account the previous state or behavior of a node while distributing the load. This has a major impact on the overall system performance due to the unpredictability of load fluctuation of the distributed system. Dynamic load balancing approach takes into account the current state of the system during load balancing decisions and is more suitable for widely distributed systems such as cloud computing.

H. Mahalle et al. [6][7] discussed Round Robin algorithm in which jobs are divided evenly between all processors in a round robin order without considering the work load. Here the time slicing mechanism is used, which divides the time into multiple slices and each node is given a particular time slice or time interval in which they have to perform their task. Though the work load distributions between processors are equal but the job processing time for different processes are not same. So at any point of time, some nodes may be heavily loaded and others remain idle.

S. Mulay et al. [16] proposed an algorithm which handles the requests with priorities. It is a distributed algorithm by which the load can be distributed not only in balanced manner but also it allocates the load systematically by checking the counter variable of each data center. After checking it transfer the load accordingly i.e. the minimum value of the counter variable will be chosen and the request is handled easily and takes less time and gives maximum throughput.

Central Queue Algorithm [12] works on the principle of dynamic distribution. It stores new activities and unfulfilled requests as a cyclic FIFO queue on the main host. Each new activity arriving at the queue manager is inserted into the queue. Then, whenever a request for an activity is received by the queue manager, it removes the first activity from the queue and sends it to the requester. If there are no ready activities in the queue, the request is buffered, until a new activity is available.

Virtualization provides an effective solution to the management of dynamic resources in cloud data centers. Server virtualization makes it possible to execute concurrently several virtual machines (VM) on top of a single physical machine (PM), each VM hosting a complete software stack (operating system, middleware, applications) and being given a partition of the underlying resource capacity (mainly CPU power and RAM size).

R.Wang et al. [17] proposed an algorithm to addresses the problem with a policy for creating load balancing method in both Physical Machines layer and Virtual Machines layer migration, moreover it also introduced prediction method to ensure the transient spike does not trigger needless migration, and in Virtual Machines layer benefit estimate model is used in order to decide whether the migration of jobs in Virtual Machines of same Physical Machine is benefit for whole system. This algorithm gives better performance and thus ensures higher QoS.
Khanna et al. [8] proposed a model based on the cloud service and VMs, which monitor the resources (CPU and memory) of physical and virtual machines. If a resource exceeds a predefined threshold and some SLA is at risk, then the system migrate a virtual machine to another physical host.

In [10], a new load balancing technique is proposed, which is combination of technologies such as virtualization and consolidation (to move tasks between hosts). Machine learning and data mining is also used to build models from examples of past behaviors. Experimental results show that machine learning algorithms can predict system behavior with acceptable accuracy, and end-users receive quality of service, with least power consumption.

Fei. Ma et al. [11] proposed a new model for distributed load balancing allocation of virtual machine in cloud data center using the TOPSIS method-one of the most efficient Multi Criteria Decision Making (MCDM) techniques. This method can find the most suitable PM in the data center for the migration of problematic VMs. Results show that system can achieve better load balancing in large-scale cloud computing environment with less VM migration.

As technology scales, energy in modern high performance VLSI circuits has moved up dramatically due the burden of maintaining a large computing infrastructure. The problem of Load Balancing gets further complicated when energy consumption comes in as an additional system objective.

A task consolidation technique is proposed in [13] aiming to maximize resource utilization. The approach contributes for promising energy-saving capability as the energy consumption is significantly reduced when the task is consolidated with one or more tasks. These approaches have demonstrated the effectiveness in minimizing energy consumption while still meeting certain performance goals. However, the efficiency of these approaches in dealing with system dynamicity is limited to a certain level.

The scope of energy efficiency also should be stretched further incorporating dynamicity and heterogeneity of both resources and tasks. For the case of large-scale dynamic environment, it is much beneficial for a scheduler to measure its performance and adapt accordingly. The scheduler typically, tries for minimizing response time and ensuring fairness among the running tasks in the system. The impending widespread usage of multiple processor cores appears to be an excellent opportunity for realizing performance and energy benefits [14]. Due to the fact that cores in a processor are in a very close proximity to each other [15], load balancing can be done very effectively reducing idle time of processors

From above description it is clear that above mentioned algorithms or policies lack in consideration of nature of jobs along with energy efficient consideration to system. In most cases, the tasks are related to each other, and the relationships between the tasks are called "dependencies", which determine the order in which activities need to be performed. Dependencies are the relationships of the preceding tasks to the succeeding tasks. Hence, a new scheduling policy which considers nature of jobs must be designed to meet processing requirements while providing energy efficiency to the system. Dynamic scheduling with dependency mapping can be of a very effective approach for proper load balancing while tracking energy consumption. While most previous energy efficiency solutions deal with homogenous resources and/or adopt static scheduling policy, proposed approach explicitly taking into account processing priority with heterogeneous resources.

3. PROPOSED SYSTEM ARCHITECTURE

The target system used in proposed work consists of a number of servers in each of which a set of heterogeneous processors is fully interconnected. The system framework of proposed algorithm is shown in figure 2. User can submit requests to systems, and to be executed by available nodes. Incoming jobs have to wait in waiting queue. After tasks dispatched from waiting queue, dependency mapper module generates the dependency between jobs on basis of relationship between jobs. This result in formation of two queues, one is for dependent tasks and other is for independent tasks. Next, Priority analyzer module considers the priority concept to determine the jobs execution sequence. A job with high priority has to be executed first as compared to job of lower priority. At last executing node executes the jobs and subsequently formed virtual machine by consolidating virtual memory of different servers for execution of unexecuted job/jobs.

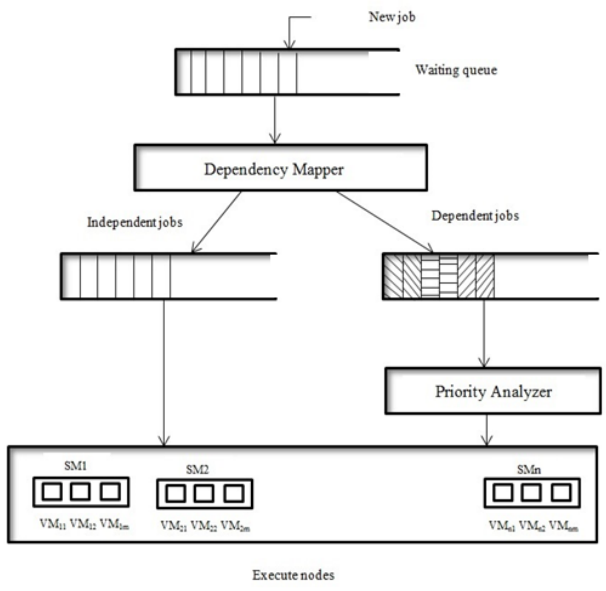

Figure 2: System framework of proposed algorithm.

From the previously defined system design, the proposed algorithm can also be represented in mathematical form as shown in figure 3. Suppose S, J are the sets that represent the set of systems s_1, s_2…, s_n and jobs j_1, j_2..., j_n respectively. Each job from job set J= {j1, j_2..., j_n} is evaluated for dependency existing between jobs, which result in formation of job groups {(j_x, j_y) where j_x represents base jobs and j_y represents dependent jobs. All the job groups are stored in a queue G. Dynamicity allow new jobs arrive at any moment, it absolutely effect dependency, so mapping of jobs is generate again and again. G is further divided into sub-queues G+ and G- to separately represent the base jobs and dependent jobs respectively. G^+ is sorted on decreasing requirement of RAM and is executed. Then all jobs from G- set are executed. Next step is to consider if any tasks remain unexecuted. If it is so, then sorting of unexecution tasks is done and node makes a broadcast of job requirement. If response is positive, job is migrated to that node. Otherwise virtual memory of heterogeneous nodes is consolidated to form virtual machine, which executes the unexcuted job list.

```
PROPOSED ALGORITHM
  1. Generate systems and jobs
     Let S = {$s_1, s_2 \ldots, s_n$}, be set of systems,
         J = {$j_1, j_2 \ldots, j_n$}, be set of jobs
  2. Generate Dependency Mapper
     if (job (i) = = job (j))
       Dependency not possible;
     else randomly generate inter dependency between jobs $j_1, j_2 \ldots, j_n$
     {
       Let G = {($j_x$, $j_y$) | $j_x$, $j_y \in J$, // group jobs on base of dependency,
                                 $j_x$ = base jobs, $j_y$ = dependent jobs
     }
     if ( $New_{jobs}$ = { $k_1, k_2, \ldots, k_n$ })
       Go to step 2
  3. Execution
     Determine the priority of each group in G
     $G^+$ = {($j_x$ | $j_x \in J$ ), where $j_x$ represents base jobs from each group in G
     Sort ($G^+$)          // on decreasing requirement of RAM
     $G^-$ = {($j_y$ | $j_y \in J$ ), where $j_y$ represents dependent job from each group in G
     Sort ($G^-$)
     Execute all jobs in $G^+$   // all base jobs get executed
     Execute all jobs in $G^-$
     if ( $New_{jobs}$ = { $k_1, k_2, \ldots, k_n$ })
       Go to step 2.
  4. For Each Task unexecuted:
     if ($Task_{Unexecuted}$ = = True)
       Sort jobs;
       Broadcast job requirement Request to Network;
       if (Response = = True)
         {Migrate Job to System}
       else
         Broadcast Request for Virtual Memory
         $VMm = \sum_{i=1}^{n} MachineMemory_i$
         Create VM;
  5. Exit
```

Figure 3: Proposed Algorithm.

The working of proposed algorithm is given as follows:
At first, a dependency mapper which considers the type of incoming jobs/tasks is set. Each incoming job has its own nature i.e. a job may be independent or may be chances that it is dependent on another job. In an ideal scenario, a dependent job must execute only after execution of the job on which it depends. So dependency mapper here analyzes the jobs and generates groups of independent and dependent jobs. The proposed cloud computing system offers dynamicity, so jobs can arrive at any moment of time, but it can affect the dependency of existing jobs. So dependency mapping is done again and again. Once the jobs are divided into pairs of (Dep, Indep) jobs, before execution priority of jobs is considered. A job pair with low priority is executed after the execution of job with high priority. Here Virtualization has been considered in more practical way to avoid the wastage of resources over the network. As each heterogeneous server have some virtual memory, so in order to avoid wastage consolidation of virtual memory is done to create virtual machine, which executes the unexecuted jobs.

The flow of proposed algorithm is presented with the help of flowchart shown in figure 4.

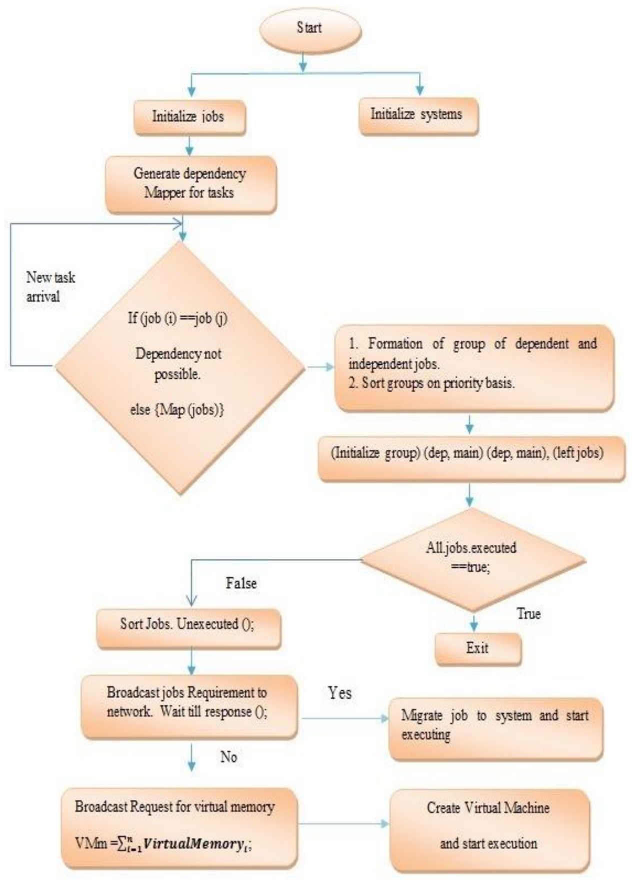
Start
Initialize jobs
Initialize systems
Generate dependency Mapper for tasks
New task arrival
If (job (i) ==job (j)
Dependency not possible.
else {Map (jobs)}
1. Formation of group of dependent and independent jobs.
2. Sort groups on priority basis.
(Initialize group) (dep, main) (dep, main), (left jobs)
All jobs executed ==true;
False
True
Exit
Sort Jobs. Unexecuted ();
Broadcast jobs Requirement to network. Wait till response ();
Yes
Migrate job to system and start executing
No
Broadcast Request for virtual memory
VMm =∑VirtualMemory;
Create Virtual Machine and start execution

Figure 4: Proposed algorithm workflow

4. PERFORMANCE EVALUATION

This section, presents the experiment configuration and present results for the proposed algorithm.

To evaluate the performance of proposed scheduling approach, extensive simulations with real-workload have been conducted. In simulation system, there are 5 to 8 heterogeneous physical nodes, each of which its own specifications. Each node is modelled to have one CPU core with performance equivalent to 1-10 GB of RAM and 1 TB of storage. The arrival times of jobs over network are varied.

The performance of proposed scheduling approach is analyzed for energy efficiency, time consumption and number of overloaded nodes, that is compared with three other existing algorithms, First Come First Serve, Round Robin and Priority scheduling.

In the First Come First Serve, the first data which enters to the queue first, gets executed first. The implementation of FCFS policy is easily managed with FIFO queue. Round Robin scheduling policy is one in which jobs are divided evenly between all processors in a round robin order without considering the work load. Here the time slicing mechanism is used, which divides the time into multiple slices and each node is given a particular time slice or time interval in which they have to perform their task. In Priority scheduling Algorithm**,** priority of each job is decided on the basis of the properties of the tasks. The priority of the task may be judged on the basis of the time consumption of the tasks or CPU scheduling burst time of the tasks or any other parameter.

Performance metrics used for the experiment are Energy consumption, Time consumption and Number of overloaded nodes.

Experimental results are presented by increasing load factor over the network and respectively calculating its impact over all above stated parameters which are considered for evaluation. This is presented in form of three experiments and results obtained have been compared and presented.

Experiment 1: The impact of Increasing load on Time Consumption

As shown in Figure 5, the proposed approach outperforms others in terms of time consumption. Proposed algorithm has achieved better performance because of task mapping which is based on their type, leading to finding the dependent and independent jobs, and then subsequently pairing dependent and independent jobs together.

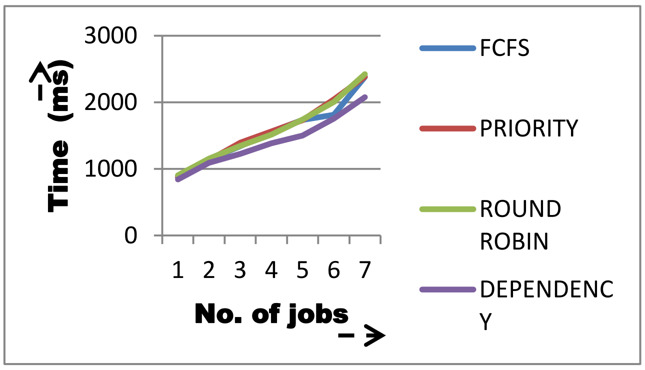

Figure 5: Time consumed by various algorithms.

In above scenario, variation of time consumption during execution is analyzed with varying workload over the network. From Figure 5, it can be seen that when the system load is raised, the timing for execution shows appealing values.

Experiment 2: The impact of Increasing load on Number of Overloaded Nodes.

From Figure 6, it is shown that Number of Overloaded Nodes during execution in proposed algorithm is another compelling strength. Experimental results show that proposed scheme gives high stability in form of overloaded nodes, in comparison to other traditional algorithms.

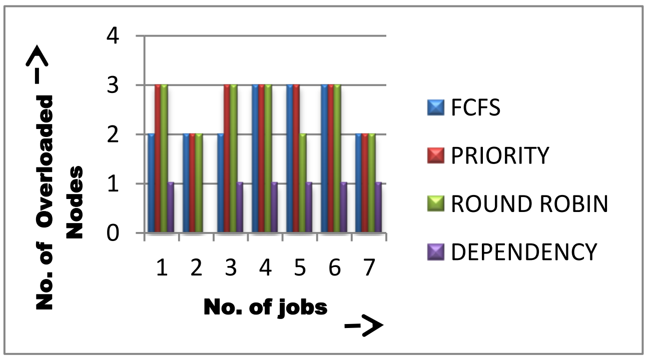

Figure 6: Overloaded Nodes during Execution of tasks.

The Figure 6 describes a situation of overloaded nodes with varying loads over the network. The criteria for overloaded node are calculated with help of written below formula. $N_{overload}$ is the number of overloaded nodes. $N_{systems}$ is the number of nodes.

$$\mathbf{N_{overload} = No.\ of\ Tasks/\ N_{systems}}$$

Experiment 3: The impact of Increasing load on Number of Energy Consumption.

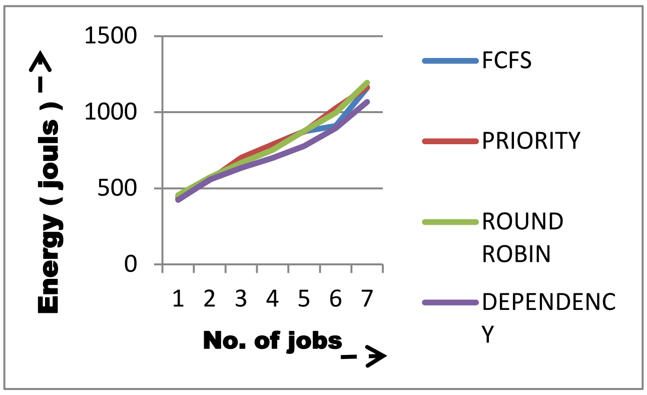

Figure 7: Energy consumption comparison.

Figure 7 shows the impact of increasing workload on the energy consumption. Although the energy consumption by all the four algorithms is almost the same in the beginning, but as more and more jobs arrive with passage of time, energy consumption by proposed algorithm is least. This result gives a strong indication that consideration of dependency analysis and subsequent pairing of jobs has great impact on load balancing.

5. CONCLUSION AND FUTURE SCOPE

Cloud computing system has widely been adopted by the industry, through there are many issues existing like Load Balancing, Migration of Virtual machines, Server Unification which has not been yet fully addressed. Load balancing is one of most central issue. Proposed work is adaptive to the dynamic environments i.e. behavior adaptively according to the workload variation on different nodes results in the change of the dependency mapping of the jobs in the network, this leads to a good performance in load balancing. Any job can randomly add in the job list with little effect on the system performance, and the load balancing can be quickly gain under the abnormal emergence situations of the heavy workload. The results obtained from comparative evaluation study clearly show that proposed algorithm outperforms other schemes in terms of time consumption, energy usage and number of overloaded nodes by a noticeable margin.

Future work can consider different types of application in cloud, and establish a more detailed load assess system. Self-learning, self-healing mechanism can be taken into consideration. Issues like Carbon Emission can also be worked upon.

6. REFERENCES

[1] Qi Zhang, Lu Cheng, R.Boutaba “Cloud Computing: state-of-art and research challenges”, © The Brazilian Computer Society 2010, 8 January 2010/ Accepted: 25 February 2010/ Published online: 20 April2010.

[2] M. Ahmed, A. Chowdhury, M. Ahmed, M. Rafee “An Advanced Survey on Cloud Computing and State-of-the-art Research Issues”, International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.

[3] P Mathur, N Nishchal, “Cloud Computing: New Challenge to entire computer industry”, International conference on parallel, Distributed and Grid computing, IEEE, 2010.

[4] M. Hussin, Y. Choon Lee, A. Zomaya, “Priority-based scheduling for Large-Scale Distribute Systems with Energy Awareness”, Ninth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2011, pp 503.

[5] R. Mata-Toledo, and P. Gupta, “Green data center: how green can we perform”, Journal of Technology Research, Academic and Business Research Institute, Vol. 2, No. 1, May 2010, pages 1-8.

[6] Wei Wang, “A reliable dynamic scheduling algorithm based on Bayes trust model,” Computer Science, 2007.

[7] M.Nikita, “Comparative Analysis of Load Balancing Algorithm in Cloud Computing”, International Journal of Engineering and Science, vol.01.

[8] H.Mahalle, P.Kaveri, V.Chavan, “Load Balancing on Cloud Data Centers”, international Journal of Advance Research in computer Science and Software Engineering, vol. 3, Jan. 2013.

[9] S.Mulay, S.Jain, “Enhanced Equally Distributed Load Balancing Algorithm for Cloud Computing”, International Journal of Engineering and Technology, vol.02, Jun. 2013.

[10] W. Leinberger, G. Karypis, V.Kumar, "Load Balancing Across Near Homogeneous Multi- Resource Servers", 2000, IEEE.

[11] R.Wang, W.Le, X.Zhang, “Design and Implementation of efficient Load Balancing method for virtual machine cluster based on cloud service”, School of Information Science and Engineering, Yunnan University, Kunming, China.

[12] K. Xiong, H. Perros, "Service Performance and Analysis in Cloud Computing," pp. 693-700, 2009.

[13] Josep Ll. Berral, Ricard Gavald`a, Jordi Torres ,”Adaptive Scheduling on Power-Aware Managed Data-Centers using Machine Learning”, Universitat Polit`ecnica de Catalunya and Barcelona Supercomputing Center.

[14] Fei. Ma, Feng Lui, Zhen Lui, “Distributed Load Balancing Allocation of Virtual Machine in Cloud Data Center” , 2012 IEEE

[15] Y. C. Lee, and A. Y. Zomaya, “Energy efficient utilization of resources in cloud computing systems,” Journal of Supercomputing, pp. 1-13, 2010.

[16] I. Ahmad, S. Ranka, and S. U. Khan, “Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy,” in IEEE Int'l Sym. On Parallel and Distributed Processing (IPDPS), Miami, FL, pp. 1-6, 2008.

[17] N. Aggarwal, P. Ranganathan, N. Jouppi, “Configurable Isolation: Building High Availability Systems with Commodity Multi-core Processors,” in Proc. of the 34th Annual Int'l Sym. on Computer Architecture, pp. 470-481, 2007.