A Hybrid Load Balancing Policy underlying Cloud Computing Environment

S.C. WANG, S.C. TSENG, S.S. WANG*, K.Q. YAN*
Chaoyang University of Technology
168, Jifeng E. Rd., Wufeng District, Taichung 41349
TAIWAN, R.O.C.
{scwang; s10314903; sswang*; kqyan*}@cyut.edu.tw
*: Corresponding author

Abstract: - Network bandwidth and hardware technology are developing rapidly, resulting in the vigorous development of the Internet. A concept, cloud computing, uses low-power hosts to achieve high usability. The cloud computing refers to a class of systems and applications that employ distributed resources to perform a function in a decentralized manner. Cloud computing is to utilize the computing resources (service nodes) on the network to facilitate the execution of complicated tasks that require large-scale computation. Thus, the selecting nodes for executing a task in the cloud computing must be considered. However, in this study, a hybrid load balancing policy, which integrated static, and dynamic load balancing technologies to assist in the selection for effective nodes. In addition, if any selected node can no longer provide resources, it can be promptly identified and replaced with a substitutive node to maintain the execution performance and the load balancing of the system.

Key-Words: - Cloud computing, Distributed computing, Load balancing

1 Introduction

Cloud computing is a concept in distributed systems [1,6,8]. It is currently used mainly in business applications in which computers cooperate to perform a specific service together. As network bandwidth and quality outstrip computer performance, various communication and computing technologies previously regarded as being of different domains can now be integrated [6,12]. In a cloud-computing environment, users have access to faster operational capability on the Internet, and the computer systems must have high stability to keep pace with this level of activity [8].

However, cloud computing aggregates the rich computing resources to form a powerful computing capability and assist in the computing of large amounts of complicated tasks. In the cloud-computing environment, there are numerous nodes capable of providing computing resources; some of them are frequently available and able to provide sharing of computing resources constantly, but some are not [4]. Therefore, when an inefficient node is chosen for distributing tasks, re-distribution and re-execution of tasks may frequently occur, lowering the execution performance of the system. How to select efficient nodes is one of the issues worth of our further investigation. Thus, this study proposed a hybrid load balancing policy, which first selects effective node sets in the stage of static load balancing to lower the odds of selecting ineffective nodes and makes use of the stage of dynamic load balancing. When the status of a node is changed, a new substitute can be located in the shortest time to maintain the execution performance of the system.

The rest of this paper is organized as follows. The literature review is discussed in Section 2. The proposed policy is discussed in Section 3. The simulation results are shown in Section 4. Finally, the conclusions and the future works are discussed in Section 5.

2 Related Works

Cloud computing is a distributed computing system in which nodes are interconnected with the internet. However, previous studies of load balance [3,9,11] are not specifically address cloud computing to order the application of internet. As cloud computing is a distributed system utilizing the nodes scattered, the most critical issue pertaining to distributed systems is how to integrate and apply every node into a distributed system, so as to achieve the goals of enhancing performance, resource sharing, extensibility, and increase availability [12]. “Load Balancing” is very important in a distributed environment. In distributed systems, every node has different processing speed and system resources, so in order to enhance the utilization of each node and shorten the consumption of time, “Load Balancing” will play a critical role. On the other hand, in distributed systems, the policies and methods for keeping a "Load Balancing" will directly affect the performance of the system [2,5]. In addition, the load balancing policies for distributed systems can be generally categorized into static load balancing policies and dynamic load balancing policies [7,11].

Static load balancing policies use some system information, such as the various information related to average operation, operation cycle, etc. According to these data, tasks are distributed through mathematic formulas or other adjustment methods, so that every node in the distributed system can process the assigned tasks until completed. The merit of this method is that system information is not required to be collected at all time, and through a simple process, the system can run with simple analyses [7,11].

Dynamic load balancing policies refer to the current state of the system or the most recent state at the system time, to decide how to assign tasks to each node in a distributed system. If any node in the system is over-loaded, the over-loading task will be transferred to other nodes and processed, in order to achieve the goal of a dynamic balance [2,5]. However, the migration of tasks will incur extra overhead to the system [2,5]. It is because the system has to reserve some resources for collecting and maintaining the information of system states. If this overhead can be controlled and limited to a certain acceptable range, in most conditions, dynamic load balancing polices outperform than static load balancing policies [9,10].

3. Research Methods

The proposed system consists of a dispatcher that manages system related tasks and nodes that join the cloud computing and provide resources. A dispatcher's role in the system is the management of tasks, including maintenance of the load-balancing, monitor on the status of each node in the cloud computing, selection for nodes for task execution, and assignment and adjustment of tasks for each node. To make a dispatcher accomplish its mission in the most efficient way, a dispatcher has the mechanisms of an agent, node selection, and calculation of effective node.

In this system environment, an agent mainly collects related information of each node participating in this cloud computing, such as CPU utilization, remaining CPU capability, remaining memory, the execution condition of the assigned task, and etc. After all these data are collected, they will be provided to the dispatcher and assist it in maintaining the load balancing of the system.

To make the dispatcher select appropriate nodes effectively, all of the nodes in the system will be evaluated by the threshold that is derived from the demand for resource needed to execute the task. Nodes that pass the threshold are considered effective, and will be organized into a table of effective nodes. Through the mechanism of value functions, the value of each effective node can be derived, and will be listed in ascending order. Besides, to get hold of the status of task distribution, all the assigned tasks and executing tasks will be incorporated into an execution aggregate, and the time required by each node to process the task will be predicted too. In addition, the unassigned nodes will be incorporated into a waiting aggregate. When any node in the execution aggregate accomplishes its assigned task, it will be transferred to the waiting aggregate, waiting for another assignment.

In a cloud computing environment, the composition of nodes is dynamic, every node is likely to enter a busy state at any time and thus lower its performance, so when selecting nodes, CPU utilization cannot be the sole factor of consideration. Other factors affecting the effectiveness of nodes are the past level of contribution, past completion rate, possibility of continuous provision of the resource, etc. Thus, a value function is proposed to evaluate the value of each effective node, and provide reference for selecting effective nodes.

In this value function, the properties of the task to be processed is the decision variable in the value function, and the relative values of each decision variable can be derived with equations or figure correspondence tables. In addition, to search for a node that meets the demand most, different weight values will be given to the nodes in accordance with the level of preference for the task, so as to select the nodes most suitable for the execution of the task. Therefore, the value function is shown as Eq. (1).

$$V_j = \sum_i^n w_i f(x_{i,j}) \tag{1}$$

Where

$$\sum_i^n w_i = w_1 + w_2 + \ldots + w_n = 1,\ 1 \le j \le N;\ 0 \le f(x_{i,j}) \le 1$$

And, $f(x_{i,j})$ is the score of the decision variable i in node j; V_j is the estimated value of node j; i is the decision variable adopted in this value function, and there are totally n decision variables; and there are totally N nodes; w_i is the weight value of each decision variable.

Nodes play the role of assisting in the execution of tasks in this system. When any node is in an available state and can provide its resource, a “join” message and related hardware information will be transmitted to the dispatcher; when it can no longer provide resource, it has to transmit an “exit” message to the dispatcher.

To maintain the load balancing of the system, this study proposed a hybrid load balancing policy, consisting of static and dynamic load balancing stages. In the stage of static load balancing, when request of executing a task is made, the task will be assigned to an appropriate node to achieve the goal of load balancing. In addition, in the stage of dynamic load balancing, the system will be adjusted dynamically according to the load balancing until a balance is reached.

3.1 The static load balancing stage

Because cloud resources are provided by available nodes, when a request for executing a task is proposed, the task has to be divided into several subtasks. Based on the proposed task, a table of effective nodes for the task will be built, so that the needed nodes can be selected from the table of effective nodes. Suppose in executing a certain task and k nodes are required to provide assistance, the top k nodes with the highest value will be selected from the table of effective nodes to assist the execution of the task. If the total amount of nodes in the table is smaller than the required amount k, a portion of subtasks will be first assigned to effective nodes and the remaining subtasks will be processed when new nodes are incorporated in the table of effective nodes. The relationship between different roles in static load balancing stage is described in Fig. 1.

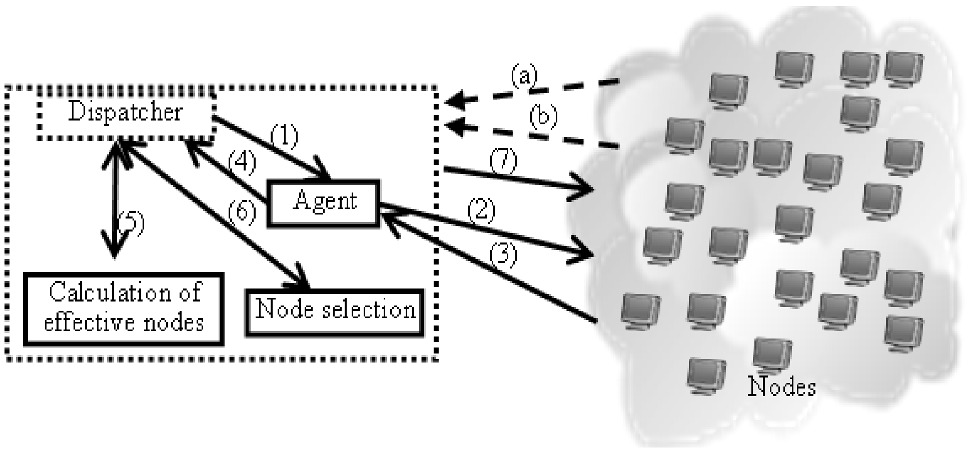

Fig. 1. The interaction of different roles at static load balancing stage

In Fig. 1, node sends message to dispatcher:

(a) When any node is in a ready state and can provide its resource, a “join” message and related hardware information will be transmitted to the dispatcher.

(b) When node can no longer provide resource, it has to transmit an “exit” message to the dispatcher.

The processes of static load balancing stage are shown as follow:

(1) When a request for executing a task is proposed, dispatcher dispatches agent to collect related information of each node participating in this Cloud computing environment.

(2) Agent collects the related information of node, such as remaining CPU capability, remaining memory ...etc.

(3) Nodes sent its own information to agent.

(4) Agent provides all node related information to dispatcher.

(5) Dispatcher builds a table of effective nodes by mechanism of calculation of effective nodes.

(6) Dispatcher selects effective nodes set from a table of effective nodes by mechanism of node selection.

(7) Dispatcher assigns subtasks to selected nodes.

3.2 The dynamic load balancing stage

In a cloud-computing environment, the assignment of tasks has to be dynamically adjusted in accordance with the variation of node status. The variation of the node status can be identified in two conditions; firstly, when the dispatcher receives the message that a certain node can no longer provide resources, and secondly, when the execution of a certain node exceeds the expected time. When any of the above situations are occurred and is detected by the dispatcher, the dispatcher will launch the agent mechanism to collect the related data of all the nodes in the table of effective nodes and compare the collected data with historic ones, in order to confirm if the node is still effective. If the node remains effective, the distribution of the task will not be re-adjusted, but the time required for the node's execution of the task will be estimated again. If the node is confirmed ineffective, a node with the highest value will be selected from the waiting aggregate, and the existing task will be transferred from the ineffective node to the new effective one. The relationship between different roles in dynamic load balancing stage is described in Fig. 2.

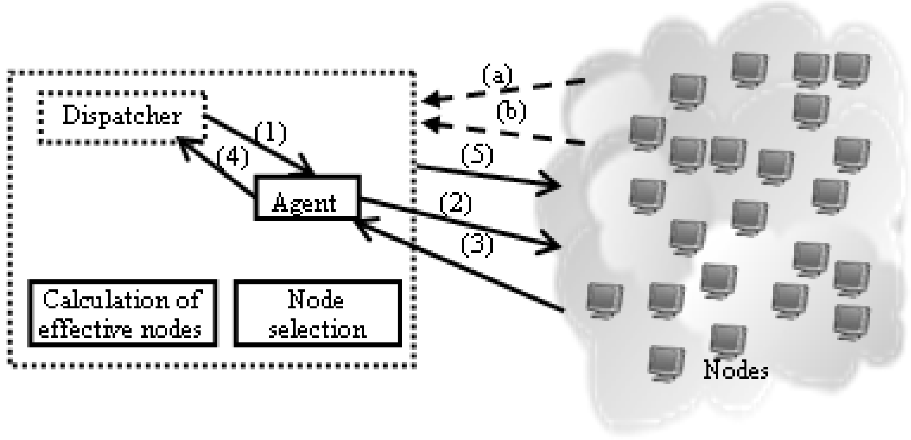

Fig. 2. The interaction of different roles at dynamic load balancing stage

In Fig. 2, node sends message to dispatcher:

(a) When any node is in a ready state and can provide its resource, a "join" message and related hardware information will be transmitted to the dispatcher.
(b) When node can no longer provide resource, it has to transmit an "exit" message to the dispatcher.

The processes of dynamic load balancing stage are shown as follow:

(1) When state of node changes, dispatcher dispatches agent to collect related data of each node in the table of effective nodes.
(2) Agent collects related information of node in the table of effective nodes.
(3) Nodes sent its own information to agent.
(4) Agent provides collected data to dispatcher. Dispatcher compares the collected data with historic ones in order to confirm if the node is still effective.
(5) If the node is confirmed ineffective, a node with the highest value will be selected from the wait aggregate, and the subtask is re-executed by new effective node.

4. Experiment

In this section, simulation experiments are conducted according to the proposed method. Based on the demands for computing resource for the execution of the task and transmission rate, simulation tests were conducted on the four different conditions and compared to other node-selection mechanisms in terms of performance, in order to verify if the mechanism proposed by this study had a better performance.

The experiment verification is conducted on the nodes selected to execute tasks by the value function of this study. The value of each node is estimated with the value function and serves as the basis for task assignment. This method first divides the task into several independent subtasks and takes minimum resource demand of each node as the threshold value. After the value of each node is estimated with the value function, the nodes for the execution of the task are selected by the order of their values. In the value function, decision variables can be given a different setting according to the factor focused in the actual application. In the experiment, the available CPU capacity, size of available memory, transmission rate, and the past completion rate were the four factors regarded as the threshold for the value function to select nodes and the decision variables for estimating node values.

In the cloud computing environment, the computing capability provided by the CPU and the available size of memory of each node are different. In addition, cloud computing utilizes idle resources of each node, so the available resource of each node may vary in a busy condition. From the perspective of task completion time, the available CPU capacity and size of available memory are the two decisive factors for the duration of execution. Thus, in this experiment, the available CPU capacity and the size of available memory were taken as the threshold for value function and decision variables for estimating node values.

In cloud computing environment, nodes may not provide as much transmission rate as the bandwidth due to ongoing execution of tasks. In this experiment, the transmission rate of each node was taken as the threshold for the value function to select nodes and a parameter for estimating node values. Furthermore, the effectiveness of the nodes in the cloud-computing environment may vary with time, and the status of the nodes in the next moment may not be completely predicted. However, the time that nodes can provide resources can be predicted through historic records. Given that the nodes capable of completing the execution of the assigned task as expected, and the time of providing resources for cloud computing is longer, in this study, the node’s past completion rate was also taken as the threshold for the value function to select nodes and one of the variables for estimating node values.

After the decision variables for the value function are determined, to make every decision variable comparable, each of the variables has to be quantified. In this experiment, the available CPU capacity and the size of available memory are quantified. In addition, in the aspect of transmission rate, because the transmission rate between dispatcher and each node is limited to their network bandwidth, the network bandwidth of the dispatcher is taken as the denominator to quantify the transmission rate of each node. To verify that the nodes selected by the value function of this study were able to effectively reduce the times of re-assigning tasks in a cloud-computing environment and achieve the goal of enhancing system performance, it was assumed in the experiment that the time for every node to provide the resource was limited.

According to the aforementioned assumption, this experiment was carried out in two stages. In the first, the network simulator – Network Simulation Version 2 (NS-2) [13] was adopted to create a cloud-computing environment dynamically. In the second, Java was adopted to simulate various task-scheduling algorithms. In the first stage, cloud computing environments with 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000 nodes were dynamically created with NS-2. Data packages were generated at random sizes, and transmitted in CBR (constant bit rate). The transmission rates between the dispatcher and every node were tested. To simulate the heterogeneity of nodes in the cloud computing environments, the CPU capability, memory size, CPU usage and memory usage, past task completion rate of each node are generated at random. In addition, the effective time of each node was generated at random and then multiplied by the past task completion rate, so as to reflect the relation between the past task completion rate and effective time of the node.

Because the composition of resources is dynamic and varies with time, the state of node is difficult to forecast. However, in our experiment, the effective time of each node is generated randomly. According to the computing resource and amount of data transmission required to execute the task, the experiment was conducted in the following four conditions.

Condition 1:	Demand for computing resource is large (25,000GHz), and amount of data transmission is small (100MB).
Condition 2:	Demand for computing resource is small (1,000GHz), and amount of data transmission is large (300,000MB).
Condition 3:	Demand for computing resource is large (25,000GHz), and amount of data transmission is large (300,000MB).
Condition 4:	Demand for computing resource is small (1,000GHz), and amount of data transmission is small (100MB)

In the experiment, the four different task scheduling algorithms, including First Come First Served (FCFS), Later In First Out (LIFO), available CPU usage (CPU-based), and value function (VF) was employed to select 10 nodes for task assignment. The task was divided into 10 subtasks and distributed to 10 nodes. If any of the nodes could not complete the assigned task, new nodes would be selected and substituted for according to various task scheduling algorithms, and the task would be re-distributed and executed. Based on the above-mentioned four conditions, the simulation experiments comparison of the task completion time are shown in Fig. 3-6.

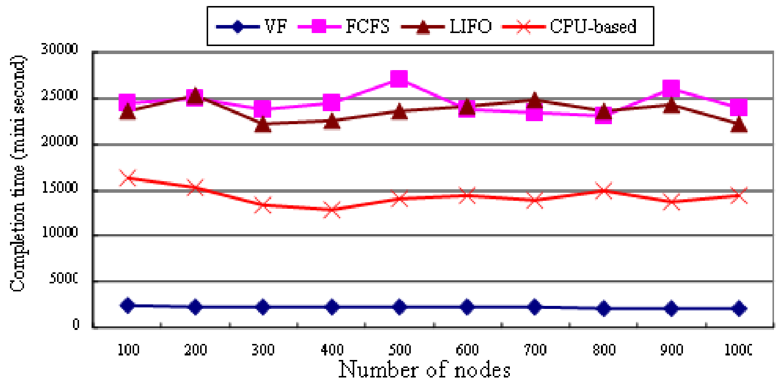

Fig. 3. Condition 1:
(Demand for computing resource is large, and amount of data transmission is small)

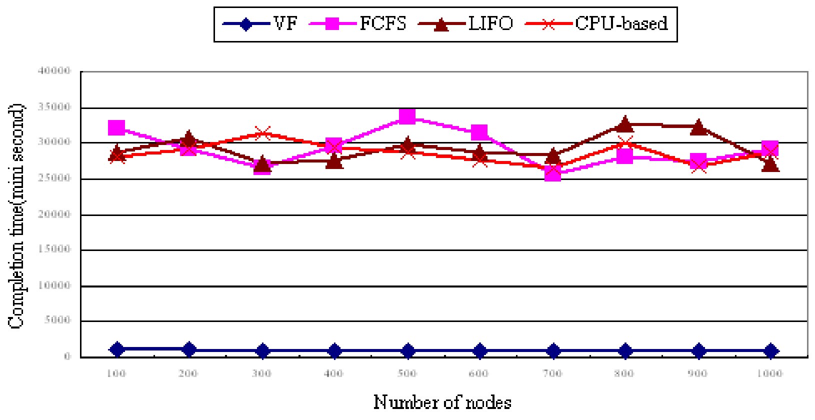

Fig. 4. Condition 2:
(Demand for computing resource is small, and amount of data transmission is large)

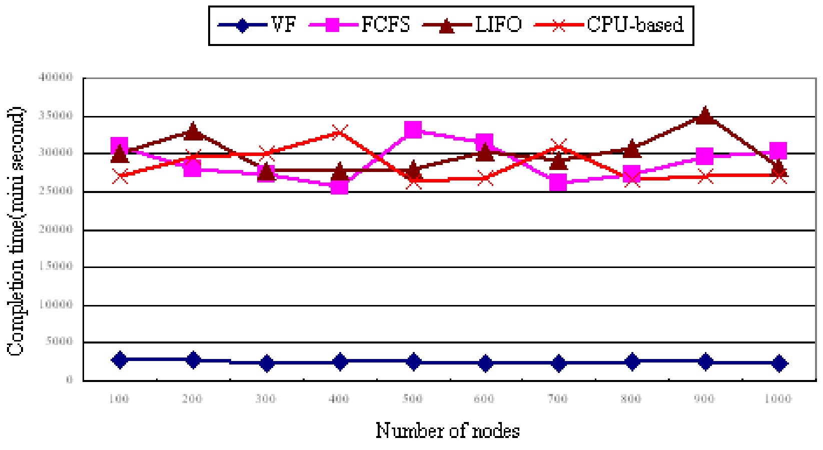

Fig. 5. Condition 3:
(Demand for computing resource is large, and amount of data transmission is large)

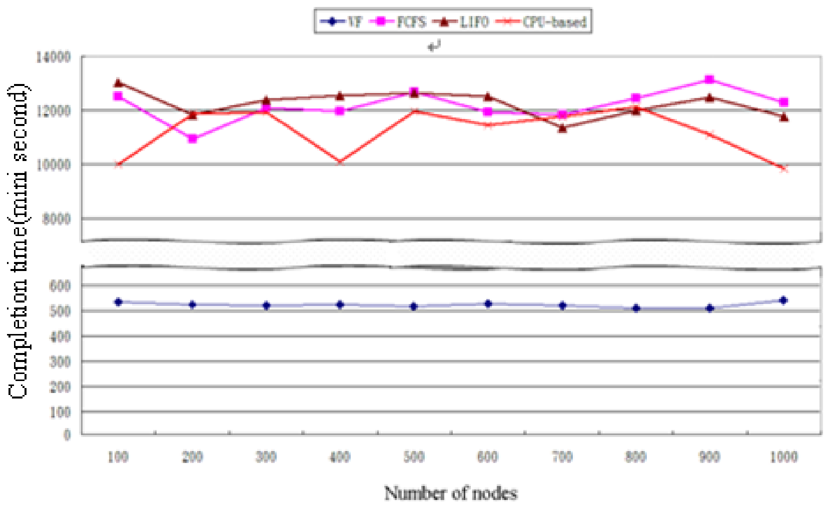

Fig. 6. Condition 4:
(Demand for computing resource is small, and amount of data transmission is small)

5. Conclusion

The resources in a cloud-computing environment are composed of a lot of resources on the network, and nodes that provide these resources will dynamically change with time. Therefore, a new node has to be located to take over the already assigned task, and task re-execution and re-distribution will occur. Therefore, node selection cannot simply depend on the resources that a node can provide (such as the available CPU capacity), other factors of the node has to be considered too, so as to find suitable and effective nodes for assisting the execution of tasks. To maintain the load balancing of the system in a cloud-computing environment, this study proposed a hybrid load balancing policy, which integrates static and dynamic load balancing techniques, to locate effective nodes, identify system imbalance in the shortest time when any node becomes ineffective, and fill in with a new node. The experiment on the above-mentioned four conditions has proven that the proposed method is far more effective than the other methods in selecting nodes with better effectiveness and performance for task execution, reducing task completion time, and avoiding the occurrence of task re-distribution and re-execution. Thus, it can effectively maintain the load balancing and enhance the performance of the system. The proposed method has the main advantage of locating proper resources according to the task properties. It can reduce the drop of system performance resulted from miss-selection of ineffective nodes, and maintain the load balancing of the system.

In this paper, a hybrid load balancing policy is proposed for the cloud computing environment. Since the cloud computing environment is more complicated than other distributed system, if the policy can achieve the load balancing in the cloud computing environment, the policy will also be able to solve the load balancing in other distributed systems.

References:

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, “A View of Cloud Computing,” Communications of the ACM, Vol. 53 No. 4, 2010, pp. 50-58.

[2] H. Chang, X. Tang, “A Load-balance based Resource-scheduling Algorithm under Cloud Computing Environment,” Lecture Notes in Computer Science, Vol. 6537, 2011, pp. 85-90.

[3] R Kaur, S Kinger, “Analysis of Job Scheduling Algorithms in Cloud Computing,” International Journal of Computer Trends and Technology, Vol. 9, No. 7, 2014, pp. 379-386.

[4] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, A. Ghalsasi, “Cloud Computing -The Business Perspective,” Decision Support Systems, Vol. 51, Issue 1, 2011, pp. 176-189.

[5] T. Mathew, K.C. Sekaran, J. Jose, “Study and Analysis of Various Task Scheduling Algorithms in the Cloud Computing Environment,” in Proceedings of International Conference on Advances in Computing, Communications and Informatics, 2014, pp. 658-664.

[6] P. Mell, T. Grance, The NIST Definition of Cloud Computing, National Institute of Standards and Technology (NIST), 2011, pp. 1-7.

[7] X. Ren, R. Lin, H. Zou, “A Dynamic Load

Balancing Strategy for Cloud Computing Platform based on Exponential Smoothing Forecast," in Proceedings of 2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), 2011, pp. 220-224.

[8] L.H. Wang, J. Tao, M. Kunze, "Scientific Cloud Computing: Early Definition and Experience," in Proceedings of 10th IEEE International Conference on High Performance Computing and Communications, 2008, pp. 825-830.

[9] X. Wang, B. Xu, S. Zhong, "A Study on Optimized Method of Task Scheduling Oriented Cloud Computing Environment," Applied Mechanics and Materials, Vol. 143, 2012, pp. 245-249.

[10] P. Yang, P. Chavali, E. Gilboa, A. Nehorai, "Parallel Load Schedule Optimization with Renewable Distributed Generators in Smart Grids," IEEE Transactions on Smart Grid, Vol. 4, Issue 3, Sept. 2013, pp. 1431-1441.

[11] Y. Yu, Y. Wang, H. Guo, X He, "Optimisation schemes to improve hybrid co-scheduling for concurrent virtual machines," International Journal of Parallel, Emergent and Distributed Systems, Vol. 28, Issue 1, 2013, pp. 46-66.

[12] A. Vouk, "Cloud Computing- Issues, Research and Implementations," Information Technology Interfaces, 2008, pp. 31-40.

[13] The Network Simulator Version 2 (NS-2), http://www-isi.edu/nsnam/ns/ns-documentation, 2005.