A Framework for User Priority Guidance based Scheduling for Load Balancing in Cloud Computing

Venkateshwarlu Velde *, B.Rama

Department of Computer Science, Kakatiya University, Warangal,T.S.
* Corresponding author E-mail: veldevenkat@gmail.com, rama.abbidi@gmail.com

***Abstract* - Cloud computing has emerged as a new model of computing based on Internet. A cloud with large shared pool of resources provides computing resources on demand in pay as you go. wo important aspects must be considered: i) from the user perspective latency must be decreased and tasks to be completed as expected, ii) from the service provider perspective, effective utilization of computing resources is vital. Many existing solutions that focus on optimized scheduling for improving load balancing do not consider user priority guidance. When user-guided priority is considered, the scheduling follows well informed approach and it can lead to more efficient load balancing. Therefore it is important to have scheduling algorithms that are guided by user priority for better optimization of load balancing. In this paper we propose an algorithm known as User Priority based Scheduling for Load Balancing (UPS-LB) which divides user tasks into elastic and inelastic groups and schedule them for enhanced load balancing. In fact, the algorithm makes rescheduling decisions for heavy loaded resources to improve efficiency of load balancing in cloud computing. We built a prototype application to demonstrate proof of the concept. Our empirical study revealed that the proposed UPS-LB algorithm has comparable performance improvement over its predecessors such as Min-Min, LBIMM and PA-LBIMM.**

Keywords – Cloud computing, task scheduling, load balancing, user-priority based scheduling

I. INTRODUCTION

As of now, cloud computing has emerged as a very potential technology that has huge impact on the society. It is being used by individuals and organizations in one way or other. When number of users of the cloud increase, it is important to cloud service provider (CSP) to ensure the resource are optimally utilized. The efficient resource utilization has its bearing on the service quality and customer satisfaction. Users of public cloud and CSP may have Service Level Agreements (SLAs). Each SLA is the contract between the user and CSP. For instance, the response time needs to be less than a second is one of the SLAs that can be used. In presence of such constraints it is very important for CPS to optimize cloud resource utilization. Resources are to be dynamically allocated instead of reserving resources as it leads to either wastage of resources and scarcity of resources. Before describing the problem solved in this paper, it is important to understand the context in which the problem is considered. Figure 1 shows the need for load balancing and the overview of the general load balancing concept.

As shown in Figure 1, load balancing is illustrated with conceptual approach. People and organizations can use cloud services thorough World Wide Web (WWW). The traffic which reaches the cloud needs to be balanced as there are number of cloud servers involved.

In case of virtualized environments that are used generally in cloud computing, it is essential to ensure that the load is balanced across the servers for efficient completion of jobs and efficient resource allocation as well.

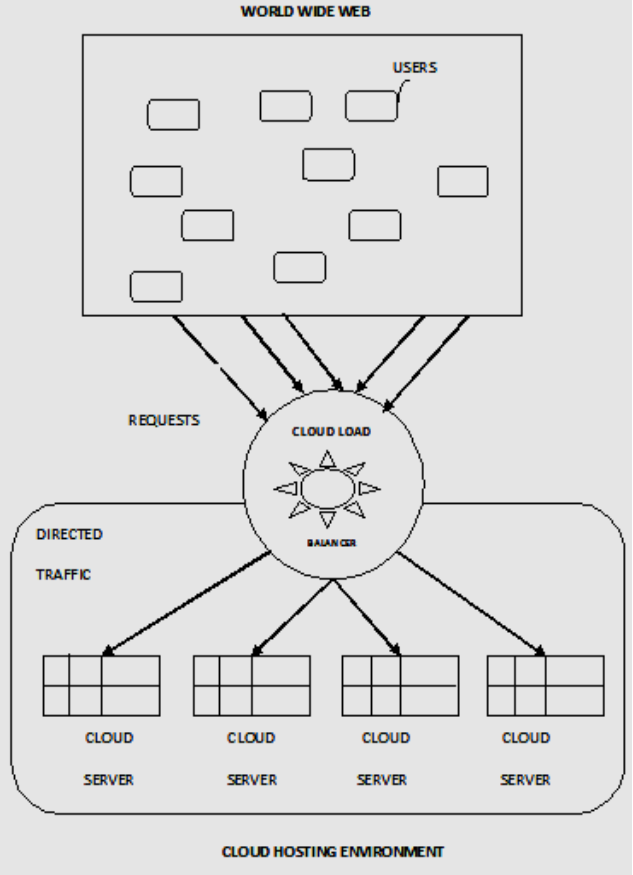

Figure 1: Overview of load balancing in cloud computing [1]

Therefore scheduling in a better way is the problem to be addressed. Load balancing is the process of balancing load across different server computing in cloud hosting environment. It is realized by ideal scheduling algorithms and resource allocation algorithms.

The existing literature revealed many approaches for load balancing in cloud computing. Min-Min algorithm and its variants are studied in [1], [10], [12] and [14] while max-min algorithm is explored in [15]. Round robin task scheduling is and its variants are studied in [20] and [21]. ACO algorithm for load balancing is explored in [23] while Bee colony algorithm is used in [25]. Load balancing with fault tolerance is investigated in [24]. A priority based scheduling algorithm is found in [20]. From the literature it is understood that different approaches for load balancing are used. However, the tasks submitted by users may have priorities. In this paper we proposed an algorithm based on user-guided priority. It is known as User Priority based Scheduling for Load Balancing (UPS-LB) which divides the tasks into elastic and inelastic so as to have better scheduling and achieve intended load balancing. Our contributions in this paper are as follows.

We propose a framework for efficient load balancing through user-guided priority based task scheduling. An algorithm named User Priority based Scheduling for Load Balancing (UPS-LB) is proposed and implemented for effective balance of load in cloud computing. A prototype application is built using cloud simulation framework to demonstrate proof of the concept. The remainder of the paper is structured as follows. Section 2 reviews literature on the existing work done related to load balancing in cloud. Section 3 presents the proposed methodology for enhancing load balancing in cloud. Section 4 presents experimental results. Section 5 provides conclusions and directions for future work.

II. RELATED WORK

This section reviews literature on load balancing in cloud computing. Scheduling also has its bearing on load balancing. Hirales-Carbajal et al. [2] explored scheduling strategies for multiple workflows in grid computing environment based on the user runtime estimates. Similar kind of work is carried out in [3] with discussion on open problems. Job scheduling with adaptive approach for optimization is studied by Chang et al. [4]. In the context of distributed computing, independent tasks are mapped to heterogeneous tasks by using different heuristic approaches as explored Braun et al. [5]. With respect to cloud computing online scheduling algorithm is proposed by Schwiegelshohn and Tchernykh [6] in presence of different service levels. The concept of QoS priority grouping is studied by Dong et al. [7] using grid task scheduling algorithm. Wang et al. [8] on the other hand studied cloud computing and load balancing at three different levels for better balance of load. Chauhan et al. [9] proposed algorithms that are guided by QoS for the purpose of grid task scheduling. In [10] a double min-min algorithm is proposed for scheduling in grid based systems. It was meta-scheduler for better performance. Kong et al. [11] proposed a fuzzy approach for prediction in virtualized data centers in order to achieve dynamic and efficient scheduling. Later on Liu et al. [12] used QoS constraints along with min-min scheduling algorithm for task scheduling in distributed environments.

Similar kind of work is carried out in [13]. Static meta-task scheduling is explored in [14] by using min-min algorithm with load balancing where as in [15] another algorithm known as improved max-min algorithm for load balancing in cloud computing.

In [16] check point based load balancing is studied based on cloud service ranking for real time scheduling of tasks. On the other hand Bhole et al. [17] investigated the usage of autonomous agents for load balancing activities in cloud computing. Different load balancing algorithm used in cloud computing and their performance evaluation is carried out in [18]. Similarity load balancing in terms of virtual machine placement is explored in [19]. Task scheduling in round robin approach with guidance from user priority is explored for load balancing in cloud [20]. In the same fashion, weighted round robin method is explored in [21] for improving efficiency of load balancing.

An efficient task scheduling algorithm is studied in [22] for a cloud data centre. It distributes tasks among all virtual machines in such a way that no VM is kept idle. A hybrid load balancing approach is studied in [23] by combining ACO and honey bee approaches with dynamic feedback. It achieved less task migration time. In [24], a load balancing algorithm is proposed with fault tolerance to make it robust against faults. Scheduling and load balancing are studied in [25] and bee colony algorithm is defined to improve performance of load balancing in cloud. In the literature it is found that load balancing algorithms are used based on scheduling. However, it is important to consider user guided priority to tasks in order to have better approach in load balancing. In this paper we call them as elastic (low priority) and inelastic (high priority) tasks. This paper proposes an algorithm for scheduling based load balancing with user-guided priority.

III. PROPOSED METHODOLOGY

We proposed a methodology to have an efficient load balancing mechanisms in cloud computing. It is based on the concept of user-guided priority which will have more fine-grained possibilities in making scheduling decisions.

A. Problem Definition

Cloud computing is widely used for storage and computations. With respect to computations when jobs are given to a cloud based application, they are to be scheduled

and completed. Moreover there might be number of constraints or SLAs that are to be taken care of. The problem considered is the user-guidance based priority that can leverage performance of scheduling and thus lead to effective load balancing. The existing approaches found in the literature have different mechanisms and algorithms for scheduling. A new algorithm is needed to handle scheduling based on user-guided priorities. This is the problem addressed in this paper.

B. Framework

A framework is proposed to handle the proposed scheduling so as to improve load balancing in cloud computing. The framework takes set of tasks as input. Then the tasks are partitioned into two groups. This grouping is made using user-guided priority. The groups are known as elastic group (EG) and non-elastic group (NEG). EG is also known as the group where tasks are associated with low priority while the NEG tasks do have higher priority. Resource analysis and computation of expected execution time for both categories of tasks are carried out and finally better approach is chosen for rescheduling tasks. This is achieved by proposing an algorithm known as User Priority based Scheduling for Load Balancing (UPS-LB) discussed in Section III.C.

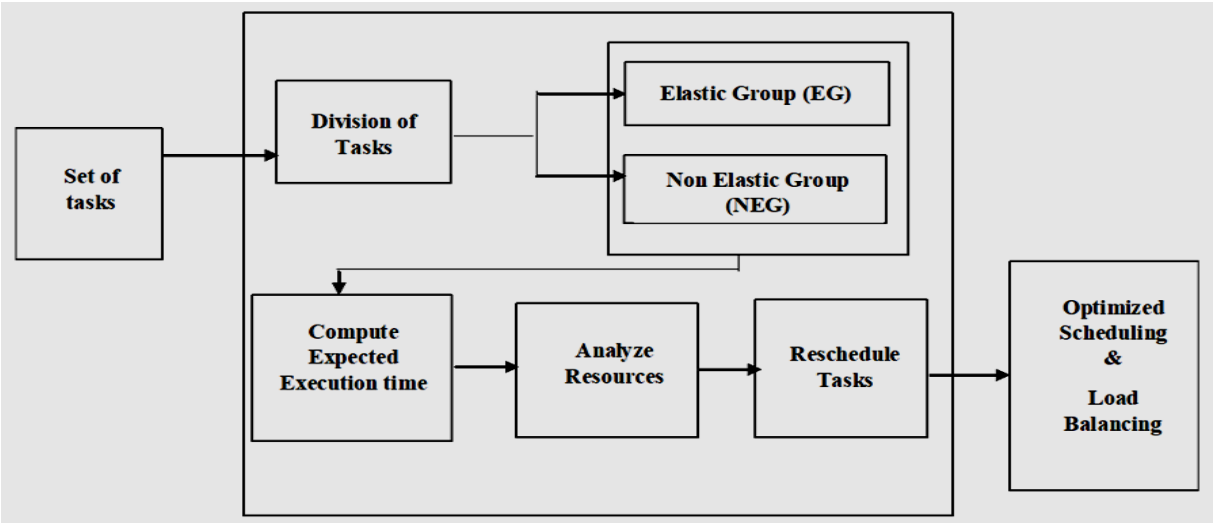

Figure 2: Proposed framework for user priority guidance based scheduling for load balancing.

As presented in Figure 2, the proposed framework reschedules tasks based on the runtime estimation of expected execution time and resource availability. This will lead to optimized scheduling and load balancing. Table 1 show notations used in the proposed algorithm. More details of the functionality are encapsulated in the proposed scheduling algorithm that is based on user-guidance based priority.

TABLE I. NOTATIONS USED IN THE PAPER

Notation	Description
$\forall$	For all
t	task
NEG	non elastic group
$\wedge$	AND operator
R	resource
Et	Expected time
met	Min execution time
$r(t)$	remove t
$ur(t)$	Update ready time of t
EG	elastic group
hr	heavy load resource
NER	non elastic resource
ER	elastic resource

Algorithm: User Priority based Scheduling for Load Balancing (UPS-LB)
Inputs: set of tasks
Output: Optimized scheduling and load balancing

initialize elastic group EG, non elastic group NEG , elastic resource ER, non elastic resource NER
for each task t in NEG
for each resource r in R
compute expected time of t
endfor
endfor

(($(t\ NEG)$ t_n NEG ∧ $(r\ R)$ r_n R ⇒ et(t)

for each task t in NEG
find t that needs min execution time
assign t to NER that provides min expected completion time
remove t from NEG
Update ready time of t
endfor

($(t\ NEG)$ t_n NEG met(t) NEG t (r(t) NEG) $ur(t)$

(($(t\ EG)$ t_n EG ∧ $(r\ R)$ r_n R ⇒ et(t)

for each task t in EG
for each resource r in R
compute expected time of t
endfor
endfor

5. ($(t\ NEG)$ t_n NEG met(t) NEG t (r(t) EG) $ur(t)$

for each task t in NEG
find t that needs min execution time
assign t to NER that provides min expected completion time
remove t from EG
Update ready time of t
End
for
6. Rescheduling for load balancing

$(hr\ R)$ hr_n R met t mct t R $\begin{cases} \text{tmin completion time} & makespan \\ ur(R) & otherwise \end{cases}$

for each heavy load resource r in R
find t that costs minimum execution time
compute minimum completion time
if min completion time < makespan **then**
reassign t to R
update ready time of R
endif
endfor

The notations shown in the table are used in the proposed algorithm. The algorithm is meant for optimized scheduling and load balancing.

C. User Priority Based Scheduling for Load Balancing

Algorithm 1: User Priority based Scheduling for Load Balancing

As provided in Algorithm 1, there are different phases in the algorithm. The initialization phase takes care of different vectors used for processing. Then the estimation of time and resource analysis is made for both elastic and non elastic task groups. Afterwards, the algorithm enters into rescheduling and load balancing phase. The experimental results are presented in the ensuing section.

IV. EXPERIMENTAL RESULTS

Experiments are made with the prototype simulation application to demonstrate the utility of the proposed algorithm. The results of the algorithm are compared with other state of the algorithms like Min-Min, LBIMM and PA-LBIMM. The observations are made in terms of makespan, average resource utilization ratio, average task completion time for elastic and inelastic tasks. Makespan is a measure to know maximum completion time of all tasks that is equal to the completion time of most of the heavy load tasks.

TABLE II: MAKESPAN AGAINST DIFFERENT NUMBER OF TASKS

Number of Tasks	Makespan (sec)			
	Min-Min	LBIMM	PA-LBIMM	Proposed
200	23.6	19	19.1	18.31
400	42.4	37.7	41.9	39.9
600	56.1	51.1	51.1	48.78
800	65.4	59.6	59.8	57.48
1000	90	83	83.2	82.20

As shown in Table II, the number of tasks and Makespan for different algorithms is presented. Number of tasks considered is 200, 400, 600, 800 and 1000.

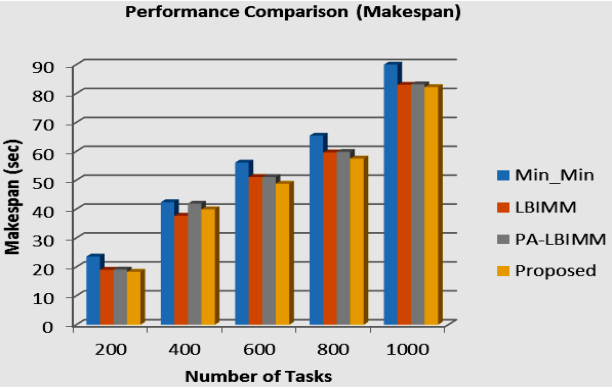

Figure 3. Number of tasks vs. makespan

As shown in Figure 3, the number of tasks is presented in horizontal axis while the vertical axis shows makespan in seconds. There are two observations with respect to makespan performance of the algorithms. The number of tasks has its impact on the makespan in linear fashion. The proposed algorithm showed better performance over other algorithms consistently with all number of tasks.

TABLE III: AVERAGE RESOURCE UTILIZATION RATIO AGAINST NUMBER OF TASKS

Number of Tasks	Average Resource Utilization Ratio			
	Min-Min	LBIMM	PA-LBIMM	Proposed
200	0.523	0.737	0.719	0.786
400	0.746	0.909	0.782	0.854
600	0.814	0.936	0.928	0.995
800	0.825	0.963	0.959	0.997
1000	0.872	0.974	0.972	0.999

As shown in Table III, the results are presented in terms of number of tasks and the average resource utilization ratio for Min-Min, LBIMM, PA-LBIMM and proposed algorithms.

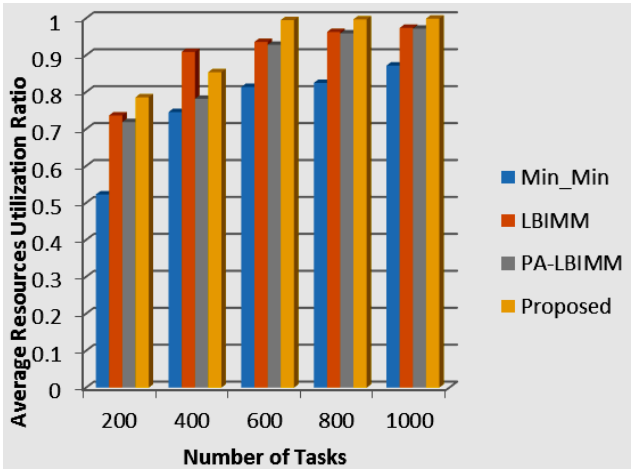

Figure 4. Number of tasks vs. average resource utilization ratio

As shown in Figure 4, the number of tasks is presented in horizontal axis while the vertical axis shows average resource utilization ratio in seconds. There are two observations with respect to the performance of the algorithms.

TABLE IV. AVERAGE COMPLETION TIME OF INELASTIC TASKS

Number of Tasks	Average Inelastic Task Completion Time			
	Min-Min	LBIMM	PA-LBIMM	Proposed
200	9.88	9.96	8.51	9.51
400	16.5	17.3	14.4	15.5
600	18.8	19.7	16.5	17.5
800	21.9	23.2	16.2	17.2
1000	32.2	31.9	21	22

The number of tasks has its impact on the average resource utilization ratio in linear fashion. The proposed algorithm showed better performance over other algorithms

consistently with all number of tasks. As shown in Table IV, the results are presented in terms of number of tasks and the average inelastic task completion time for Min-Min, LBIMM, PA-LBIMM and proposed algorithms.

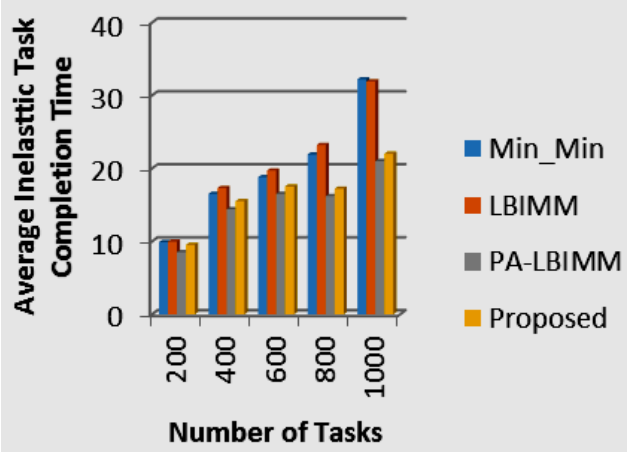

Figure 5. Number of tasks vs. average inelastic tack completion time

As shown in Figure 5, the number of tasks is presented in horizontal axis while the vertical axis shows average inelastic task completion time. There are two observations with respect to the performance of the algorithms. The number of tasks has its impact on the average inelastic task completion time in linear fashion. The proposed algorithm showed better performance over other algorithms consistently with all number of tasks.

TABLE V. AVERAGE COMPLETION TIME OF ELASTIC TASKS

Number of Tasks	Average Elastic Task Completion Time			
	Min-Min	LBIMM	PA-LBIMM	Proposed
200	8.86	9.07	9.78	10.78
400	16.7	17.1	23.9	24.8
600	21.9	22.3	25.1	26.5
800	23.4	24	28.5	29.8
1000	32	32.3	35.3	36

As shown in Table V, the results are presented in terms of number of tasks and the average elastic task completion time for Min-Min, LBIMM, PA-LBIMM and proposed algorithms.

As shown in Figure 6, the number of tasks is presented in horizontal axis while the vertical axis shows average elastic task completion time. There are two observations with respect to the performance of the algorithms. The number of tasks has its impact on the average elastic task completion time in linear fashion. The proposed algorithm showed better performance over other algorithms consistently with all number of tasks.

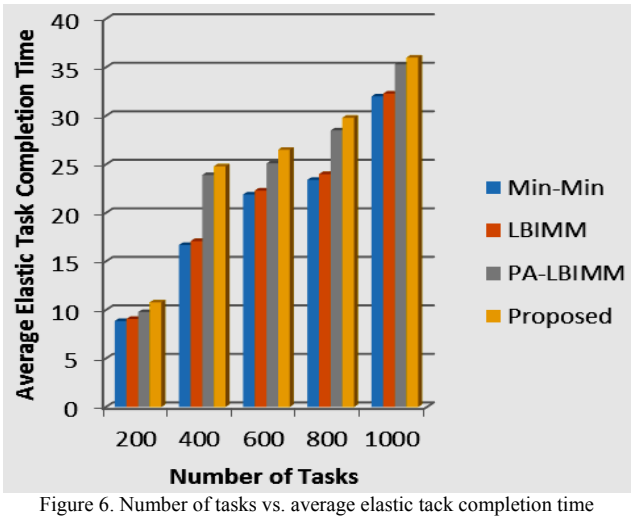

Figure 6. Number of tasks vs. average elastic tack completion time

V. CONCLUSIONS AND FUTURE WORK

In this paper, we studied load balancing approaches through scheduling of tasks in cloud computing environment. When cloud computing is used to outsource computations, users given jobs to cloud. In this context, user tasks are scheduled with different algorithms. However, when scheduling is improved properly it also leads to load balancing. Load balancing is essential for successful utilization of resources in cloud computing and ensure that there is equilibrium between customer satisfaction and business growth of CSP. Many scheduling algorithms came into existence such as min-min, max-min, ACO and their variants. However, it is understood that user-guidance based priority can have performance improvement over scheduling and load balancing. Based on this hypothesis, in this paper, we proposed an algorithm known as User Priority based Scheduling for Load Balancing (UPS-LB). This algorithm is used along with the proposed framework. It divides the tasks into elastic and non-elastic groups so as to have better performance in scheduling based on resource analysis and task expected completion time. We built a prototype application with simulation study. The results revealed that the proposed algorithm shows better performance over other state of the art algorithms like Min-Min, LBIMM and PA-LBIMM. In future we intend to continue our research on scheduling for load balancing in the presence of non-pre-emptive dependent tasks.

REFERENCES

[1] Load Balancing (2018). Retrieved from https://www.quora.com/Which-algorithm-is-easy-to-implement-in-a-project-on-load-balancing-in-cloud-computing-What-are-some-new-ideas-How-can-I-start-with-my-project. Accessed on 10 April 2018.

[2] Hirales-Carbajal, Adán, Et Al. (2012). Multiple Workflow Scheduling Strategies With User Run Time Estimates On A Grid, Journal Of Grid Computing, 10(2), P325-346.

[3] F Dong And Sg Akl. (2006). Scheduling Algorithms For Grid Computing: State Of The Art And Open Problems, Technical Report No. 2006-504, School Of Computing, Queen's University, Kingston, Ontario, Canada,P1-20.

[4] Rs Chang, Cy Lin, And Cf Lin. (2012). An Adaptive Scoring Job Scheduling Algorithm For Grid Computing, Information Sciences – Elsevier, 207, P79-89.

[5] Braun, Tracy D, Et Al. (2001). A Comparison Of Eleven Static Heuristics For Mapping A Class Of Independent Tasks Onto Heterogeneous Distributed Computing Systems, Journal Of Parallel And Distributed Computing , 61(6), P810-837.

[6] U Schwiegelshohn And A Tchernykh. (2012). Online Scheduling For Cloud Computing And Different Service Levels, Parallel And Distributed Processing Symposium Workshops & Phd Forum (Ipdpsw), 2012 IEEE 26th International, P1067-1074.

[7] Dong, Fang, Et Al. (2006). A Grid Task Scheduling Algorithm Based On Qos Priority Grouping, Grid And Cooperative Computing, Gcc 2006 IEEE Fifth International Conference, P58-61.

[8] Wang, Shu-Ching, Et Al. (2010). Towards A Load Balancing In A Three-Level Cloud Computing Network, Computer Science And Information Technology (ICCSIT), 2010 3rd IEEE International Conference On Computer Science and Information Technology, 1, P108-113.

[9] Surendar, A., & Sadulla, S. K. (2018). A time oriented flow inference model based on low rate DDoS attack detection for improved network security. International Journal of Simulation: Systems, Science and Technology, 19(4), 7.1-7.5.

[10] Ddh Miriam And Ks Easwarakumar. (2010). A Double Min Min Algorithm For Task Metascheduler On Hypercubic P2p Grid Systems, International Journal Of Computer Science , 7(4), P8-18.

[11] Kong, Xiangzhen, Et Al. (2011). Efficient Dynamic Task Scheduling In Virtualized Data Centers With Fuzzy Prediction, Journal Of Network And Computer Applications, 34(4), P1068-1077.

[12] J Liu And G Li. (2010). An Improved Min-Min Grid Tasks Scheduling Algorithm Based On Qos Constraints, 2010 International Conference on Optics, Photonics And Energy Engineering (OPEE),1,P281-283.

[13] He, Xiaoshan, Xianhe Sun, And Gregor Von Laszewski. (2003). Qos Guided Min-Min Heuristic For Grid Task Scheduling , Journal Of Computer Science And Technology, 18(4), P442-451.

[14] T Kokilavani And Ga Di. (2011) . Load Balanced Min-Min Algorithm For Static Meta-Task Scheduling In Grid Computing, International Journal Of Computer Applications, 7, P1-12.

[15] Elzeki, O. M., M. Z. Reshad, And M. A. Elsoud. (2012). Improved Max-Min Algorithm In Cloud Computing, International Journal Of Computer Applications, 50(12), P22-27.

[16] Mohammad Riyaz Belgaum, Safeeullah Soomro, Zainab Alansari And Muhammad Alam. (2016). Cloud Service Ranking Using Checkpoint Based Load Balancing In Real Time Scheduling Of Cloud Computing. Progress In Advanced Computing And Intelligent Engineering, P1-10.

[17] Prof. Rahul Bhole, Hoda Jagmeet Singh, Pushkaraj Khamkar, Pranav Joshi And Roshni Bendbhar. (2017). Load Balancing In Cloud Computing Using Autonomous Agents. Imperial Journal Of Interdisciplinary Research (IJIR). 3 (3), P1-5.

[18] Srivastava, K. K., Tripathi, A., & Tiwari, A. K. (2013). Secure Data Transmission In AODV Routing Protocol. Computer Science & Engineering Institute of Technology & Management GIDA, Gorakhpur, 4.

[19] Minxian Xu, Wenhong Tian And Rajkumar Buyya. (2017). A Survey On Load Balancing Algorithms For Virtual Machines Placement In Cloud Computing. Concurrency And Computation: Practice And Experience, P1-22.

[20] Abhijeet Malik And Prabhat Chandra. (2017). Priority Based Round Robin Task Scheduling Algorithm For Load Balancing In Cloud Computing. Journal Of Network Communications And Emerging Technologies. 7 (12), P1-5.

[21] D. Chitra Devi And V. Rhymend Uthariaraj. (2016). Load Balancing In Cloud Computing Environment

[22] Using Improved Weighted Round Robin Algorithm For Nonpreemptive Dependent Tasks. Hindawi Publishing Corporation ・ E Scientific World Journal, P1-14.

[23] Subhadra Bose Shaw. (2017). Balancing Load Of Cloud Data Center Using Efficient Task Scheduling Algorithm. International Journal Of Computer Applications. 159 (5), P1-5.

[24] Nikhit Pawar, Prof. Umesh Kumar Lilhore And Prof. Nitin Agrawal. (2017). A Hybrid Achbdf Load Balancing Method For Optimum Resource Utilization In Cloud Computing. International Journal Of Scientific Research In Computer Science, Engineering And Information Technology. 2 (6), P1-7.

[25] G.Gayathri And R.Latha. (2017). Implementing A Fault Tolerance Enabled Load Balancing Algorithm In The Cloud Computing Environment. International Journal Of Engineering Development And Research. 5 (1), P1-8.

[26] K R Remesh Babu And Philip Samuel. (2016). Enhanced Bee Colony Algorithm For Efficient Load Balancing And Scheduling In Cloud. Journal Of Network And Innovative Computing. 4 , P 135-142.

[27] Yu, Xiaogao and Xiaopeng Yu. (2009). A New Grid Computation-Based Min-Min Algorithm, Fuzzy Systems And Knowledge Discovery, Fskd'09 IEEE Sixth International Conference On Fuzzy Systems and Knowledge Discovery, 1, P43 – 45.