iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** VI **Month of publication:** June 2017

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Static Load Balancing Using ASA Max-Min Algorithm

Shubham Mathur[1], Ali Abbas Larji[2], Anubhav Goyal[3]
[1,2,3] School of Information Technology, Vellore Institute of Technology Vellore, India

Abstract: ***In recent times, a huge demand for computational resources has led to the development of large network known as a Grid [1]. A grid allows resources to be acquired in real time on an on-demand basis making sophisticated technology available at extremely low prices. However, the availability of such resources to a huge audience also invites a huge number of concurrent requests for same resources. Thus, it is required that such load of requests is efficiently distributed across all the resources with heterogeneous capabilities to keep the network function adequately. A lot of research has been conducted on Load Balancing of distributed systems and several algorithms have been devised. Load Balancing is generally categorized in two ways: a). Static load Balancing – Where all information pertaining to the resources like their number, processing power and amount of memory etc. is known in advance, b). Dynamic load Balancing – Here the network of resources is dynamic in nature i.e. the number as well as capabilities may change over the course of time. In this paper, our focus is on Static load balancing using Max-Min algorithm which performs most efficiently among all the available static algorithms. Max-Min however works only when the tasks to be assigned are clearly heterogeneous in terms of their execution time. In this paper, we propose ASA Max-Min Algorithm that overcomes the above-mentioned problem.***
Keywords—Max-min; makespan; completion time; execution time; task scheduling;

I. INTRODUCTION

Economics and time are important factors when there is a requirement of different distributed services spread across different networks or over grid of resources i.e. the amount of time one uses a service determines the cost that needs to be paid for it and hence, it is very much required that the execution of our tasks on the resources we acquire is fast so that it incurs as low cost as possible on us. For this reason, the study of how to select the best possible resource for a particular task such that it takes the least possible time for executing that task has become a major part of research in distributed systems today. Scheduling is the term used to define the way in which we schedule several tasks on the available pool of resources [2]. A scheduler is a component in a distributed environment that runs a particular algorithm to decide which task to schedule and how to schedule it. Several algorithms [3,4] have been proposed in order to schedule tasks in distributed environment.

The main aim of scheduling process is to achieve the maximum utilization of the available resources and at the very same time provide a fast and cost-effective service to its users. It is important to note that one service cannot come at the expense of other and we must ensure high speed of execution to ensure low expenditure. Scheduling algorithms are broadly categorized into two types. Static Scheduling and Dynamic Scheduling [5]. Static Scheduling is used in system environments wherein we have profound know how of the resource parameters like the number of resources available, their processing power, memory size etc. This prior knowledge of resource nodes makes the scheduling process easier for us. On the other hand, Dynamic Scheduling [6] by its name implies that the distributed environment remains dynamic at all times i.e. the no. of resource nodes, there processing power, memory capacity and other performance parameters may change over the course of time. Dynamicity obviously makes the situation very unpredictable and requires different approaches to be devised for the sake of effective scheduling.

Further classification of scheduling algorithms is done on the basis of how the tasks are assigned to the scheduler. This mainly includes two categories online mode heuristic scheduling and Batch mode heuristic scheduling [7]. In online mode scheduling the tasks are sent to the scheduler immediately as they arrive, however in batch mode scheduling tasks are collected in groups or chunks or batches and then assigned to the scheduler. In our discussions, we stick to the batch mode scheduling method.

The Ali Shubham Anubhav (ASA) Max-min algorithm that we proposed, is based on batch mode scheduling method. It uses traditional max-min method in first phase and in second phase it optimizes its result. Thus, directly overcoming some of the shortcomings of traditional Max-min algorithm with very less overhead involved.

II. RELATED WORK

Conceptually scheduling and Load Balancing are interrelated areas. Scheduling is a process that is aimed at assignment of incoming tasks into a distributed environment to resources in an efficient way, where by efficiency we broadly imply that the computational load introduced to the system is distributed within the system such that no particular resource is overloaded and we achieve the optimum makespan. Several algorithms [6,7] have been developed for scheduling tasks but long research has proved that finding an optimal scheduling algorithm is a NP problem. Hence, heuristic based improvements have been devised in order to achieve a sub-optimal solution.

A. The following are the foundational terms that are related to the most fundamental approach of task scheduling:

1) *Execution time:* The exact amount of time it takes for a task to be executed on a particular resource is called Execution time. The lower the execution time the better that resource is for that task. Scheduling tasks based on minimum execution time may assign the task irrespective of whether the resource is available or not. This can lead to severe overloading of the resources [8].
2) *Completion time:* It is the time taken by a resource to complete the task i.e. it involves the time elapsed while executing other tasks in addition to the execution time of the task in consideration. If scheduling is based on minimum completion time, then tasks may be assigned to resources that do not give minimum execution time for that task [8].

Since, our area of study is batch mode heuristic scheduling, we define the set of tasks to be scheduled as meta-tasks. The following are the basic batch mode scheduling algorithms.

1. For all tasks t in meta-task set
2. For all resources m available in grid
3. Calculate execution time ET.
4. Until all tasks are assigned to resources
5. Find the task t with minimum CT.
6. Find the resource with minimum ET
7. Assign t to this resource.
8. Delete t from meta-task set.
9. Update CT of all tasks for that resource.
10. End

Fig. 1. Min-Min Algorithm

3) *Min-Min Scheduling [9]:* It involves scheduling resources with minimum execution time (ET) within the meta-task to the corresponding resource. First minimum execution time for each task is computed on each resource (r) (1) (2) (3). Thereafter, the task with least completion time is selected (5) and assigned to that resource which have minimum execution time (7). Then remove that task from the meta-task (8) and update completion time of all remaining tasks on that resource (9). This process is repeated until all tasks are executed. Fig. 1 shows the steps followed in the algorithm.

4) *Max-Min Scheduling [9]:* The Max-Min algorithm is quite similar to the Min-Min algorithm since it also initially computes the minimum execution time of each task on every resource (1) (2) (3), then identifies the resource with the maximum completion time and assigns it to the corresponding resource which have minimum execution time (4) (5) (6) (7). The task is then removed from the meta-task set and completion time is updated (8) (9). This process is looped until all the tasks are assigned to the resources for execution. Fig. 2 shows the steps followed in the algorithm.

5) *Min-Min and Max-Min comparison:* Both the algorithms have shown their supremacy with respect to another in different situations. Detailed analysis has shown that while min-min is efficient when all the resources require less execution time, on the

other hand, max-min is better when some tasks have higher time requirements than others because it allows shorter tasks to be executed concurrently when the bigger tasks has occupied the other resource. This gives better makespan than what min-min would have given [10].

Some of the other notable works done are: QoS directed Min-Min [12, 13] has been proposed that considers the QoS requirements while scheduling tasks using min-min algorithm. The principle behind this heuristic is that some tasks may have greater bandwidth requirements in the network grid while some may have lower requirements. Thus, QoS guided min-min assigns tasks with greater bandwidth requirements first via Min-Min heuristic. An improved QoS guided algorithm named QoS-Suffrage [11, 13] has been proposed by O. M. Elzeki. It also considers a task's bandwidth requirement and then schedules it with the min-min heuristic. However, it gives a better makespan in comparison to the Min-Min, Max-Min and QoS directed Min-Min algorithms.

A Selective Min-Min algorithm is proposed by [14]. According to this heuristic, tasks are scheduled based upon the decision that which algorithm out of Min-Min or Max-Min is best suitable for that set of meta-task. It has shown some improvements in some situations.

Several other heuristic based algorithms have been proposed by researchers in order to achieve some improvement upon the already devised methods; however it must be observed that since only a sub-optimal solution is possible hence every new heuristic that can be proposed will be show improvements only in certain conditions.

III. ALGORITHM

Fig. 2. Max-Min Algorithm

```
1.  For all tasks t in meta-task set
2.        For all resources m available in grid
3.              Calculate execution time ET
4.  Until all tasks are assigned to resources
5.        Find the task t with maximum CT.
6.        Find the resource with minimum ET
7.         Assign t to this resource.
8.         Delete t from meta-task set.
9.         Update CT of all tasks for that resource.
10.  End
```

The proposed static load balancing algorithm is an improvement in the existing max-min load balancing algorithm. As mentioned earlier, max-min is a NP- hard problem, so there is not a single solution which could give best results in all situations. As discussed in previous section about the various algorithms proposed by other authors, each and every one was trying to handle certain scenarios in which max-min fails miserably. In max- min algorithm higher priority is given to bigger task, while smaller ones have lower priority [15]. In each iteration, largest available task is selected, and a resource or machine was allocated to that task. The machine was chosen on the basis of minimum execution time. Usually in max min few long tasks run on few machines while other machines handle large number of small tasks, so that overall makespan was as optimized as possible. Thus, in max-min execution of large tasks dominates the total makespan, but makespan cannot be determined if there is a difference in completion and execution times. Max-min fails to balance load properly among all the available resources, usually it just overloads certain machines while others remain idle [16].

Thus, to overcome this scenario we proposed ASA max-min algorithm. This algorithm is divided into two phases. In first phase, it will allocate resources to tasks according to traditional max-min algorithm. In second phase, i.e. optimization phase, we search for that machine which has largest makespan. Then that task is selected which was scheduled at last, then we try to calculate its completion time if it was executed by any other available machines, if that time is less than current makespan, then we de allocate

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

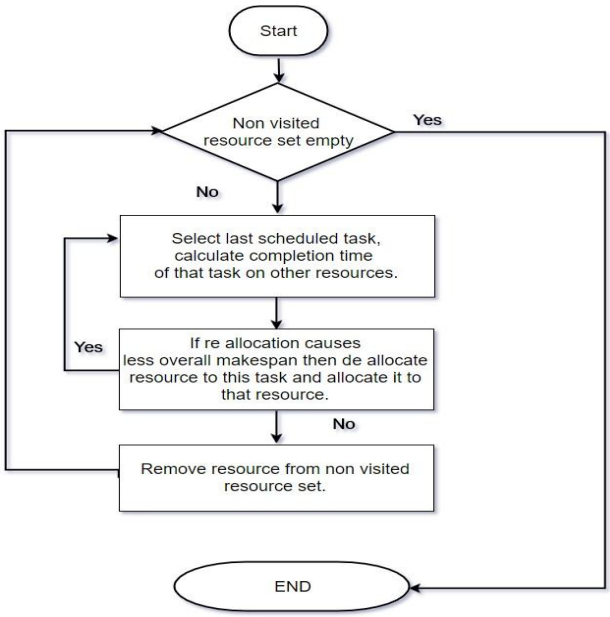


Fig. 3. ASA Max-Min Algorithm Phase 2

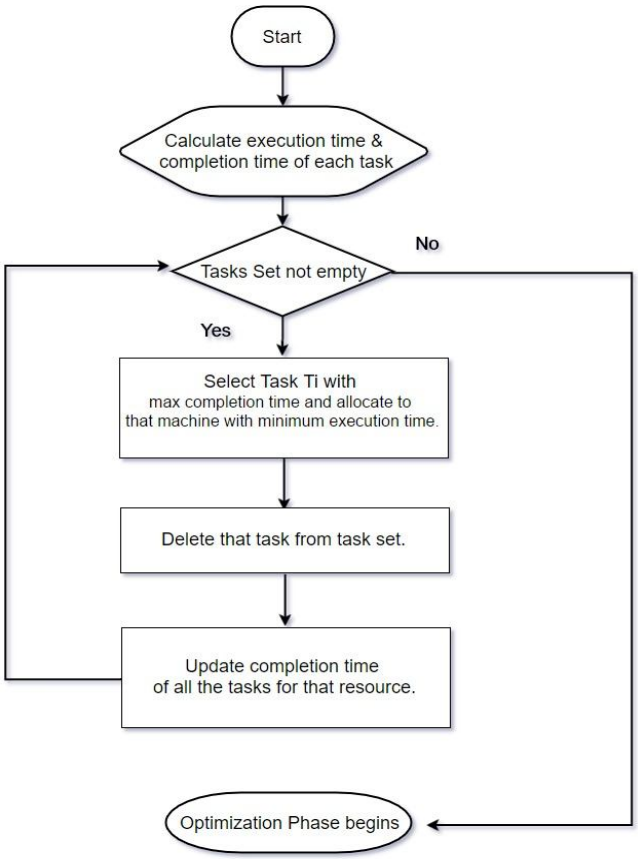


Fig. 4. ASA Max-Min Algorithm Phase 2

current machine to that task, and re allocate to newer machine we found. Same process is repeated for all the tasks for that machine and then for other machines. Fig. 3 shows overall flow chart of the basic max-min algorithm, i.e. first phase. While Fig. 4 shows optimization phase flow chart.

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Our algorithm is based on Max-min so its time complexity for phase one is O (mn^2) [17], where m is the number of resources and n is the number of tasks. Phase two, time complexity in worst case is O (mn) when for each resource it is re allocating resources for n-1 tasks. While in average cases its O (n). So, overall time complexity will be max (O (mn^2), O (mn)), i.e. O (mn^2). It may seem that in terms of time complexity it is not an improvement but in terms of resultant makespan, ASA Max-min outperforms traditional Max-min. The following section describe about such scenarios and compare results of both algorithms.

IV. EXPERIMENT AND RESULTS

We have taken two test cases and implemented them in C++ and generated Gantt chart for each case. First test case with 4 tasks and 2 machines. Table I shows estimated execution time that each machine is going to take for each task. At time t=0 their completion time is equal to their execution time. Periodically as each task is scheduled, completion time of each remaining task of that machine is updated. Fig. 5 shows Gantt chart of max-min algorithm. While Fig. 6 shows Gantt chart of ASA Max-min algorithm. These two figures are developed to show the makespan of both methods.

TABLE I. Test Case 1

Tasks	Machine 1 (m0)	Machine 2 (m1)
T0	2	3
T1	3	5
T2	4	7
T3	10	17

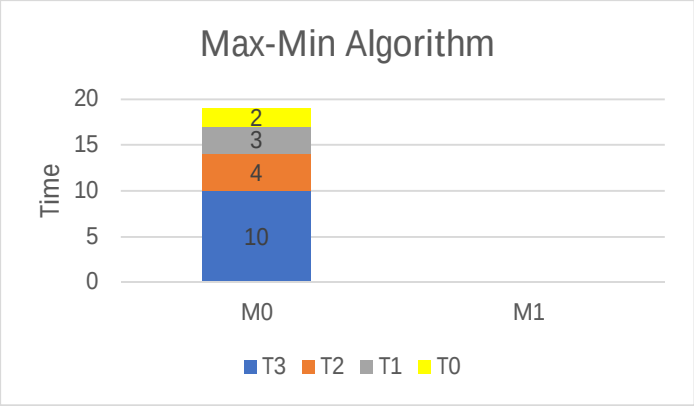


Fig. 5. Gantt Chart of Max-Min Algorithm for Test Case 1

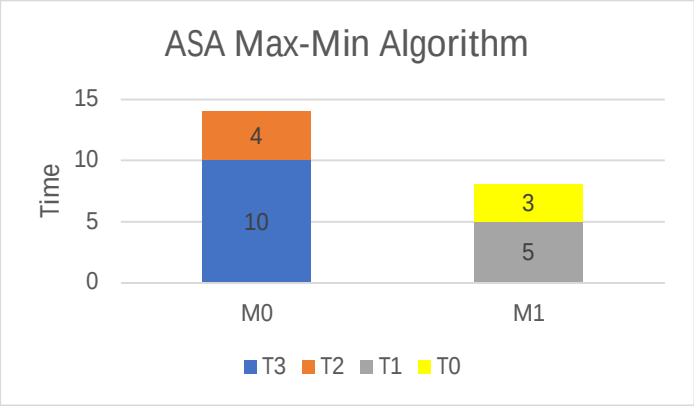


Fig. 6. Gantt Chart of ASA Max-Min Algorithm for Test Case 1

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Second test case contains 8 tasks and 3 machines. Table II shows estimated execution time that each machine is going to take for each task. At time t=0 their completion time is equal to their execution time. Periodically as each task is scheduled, completion time of each remaining task of that machine is updated. Fig. 7 shows Gantt chart of max-min algorithm. While Fig. 8 shows Gantt chart of ASA Max-min algorithm. These two figures are developed to show the makespan of both methods.

TABLE II. Test Case 2

Tasks	Machine 1 (m0)	Machine 2 (m1)	Machine 3 (m2)
T0	2	1	7
T1	11	3	5
T2	18	9	11
T3	5	8	10
T4	10	12	15
T5	7	8	11
T6	3	5	6
T7	20	30	31

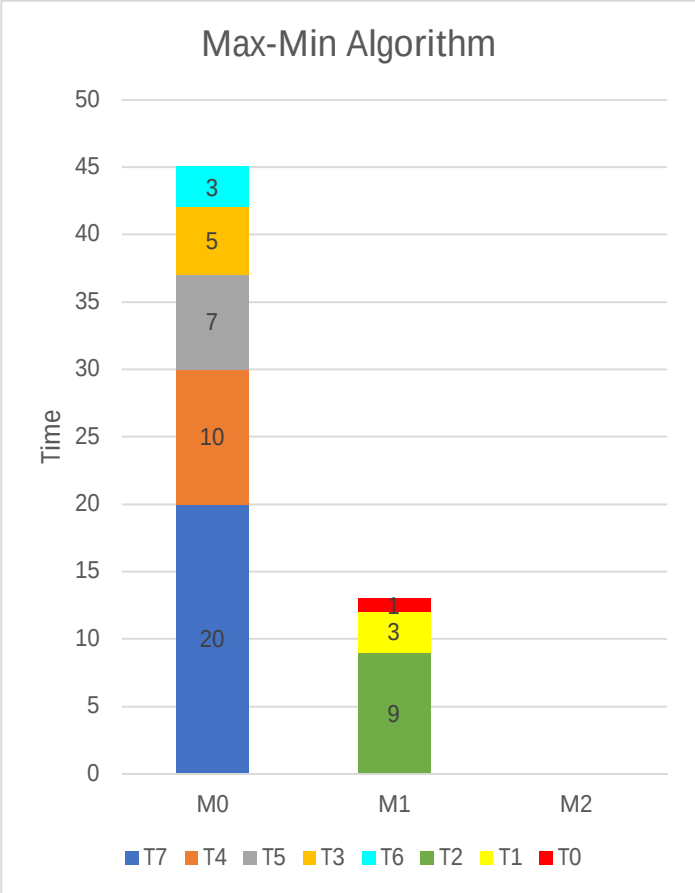


Fig. 7. Gantt Chart of Max-Min Algorithm for Test Case 2

In first test case, due to shorter execution time of tasks by machine m_0 relatively to m_1, all the load gets transferred to m_0 by Max-min algorithm. While on the other hand m_1 remains idle. This causes unbalanced load distribution with high makespan of 19 units. While in ASA Max-min, it optimizes the result by comparing the resultant makespan after reallocation of resources for

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

T0, T1. As its resultant makespan comes low, it finally does the reallocation. As more reallocation of resources in that order doesn't going to give any better makespan, further reallocation was stopped.

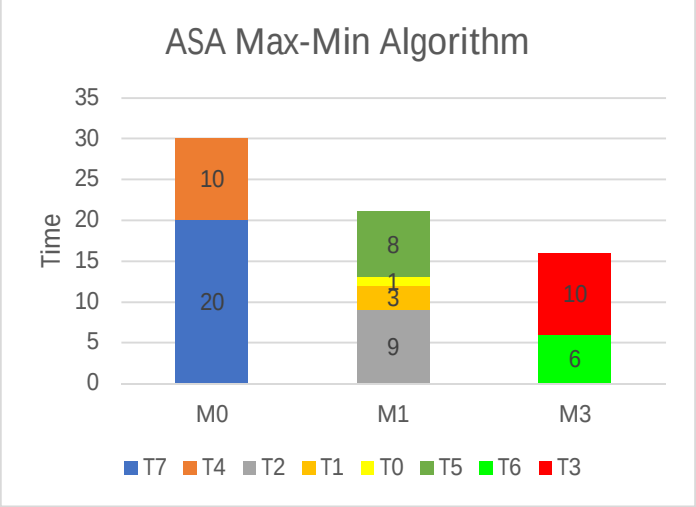


Fig. 8. Gantt Chart of ASA Max-Min Algorithm for Test Case 2

In second test case, there are three machines, and more tasks, but for all those tasks third machine have high execution time, thus Max-min doesn't allocate any task to that machine and it becomes idle while other two machines are overloaded. Among two machines m0 have lower execution time for most tasks than m1 thus among them also m0 is loaded more. Thus, resultant makespan reaches 45 units. While in ASA Max-min, in optimization phase T6 is selected first and checked for lower makespan on re allocation, as m2 was idle it is now allocated to T6. Then T3 is chosen and same procedure is repeated, consequently T5 is chosen, checked and re allocated. After that as no re allocation is possible optimization ends. After optimization, we got more evenly distributed load among machines. The makespan also got reduced from 45 to 30 units.

V. CONCLUSION

To overcome the situation specific optimality of the Max-Min scheduling algorithm the ASA Max-Min algorithm was proposed. The proposed algorithm has been able to demonstrate considerable improvement upon the original Max-Min algorithm due to the fact that it decides the allocation of tasks to machines in a way that the best possible makespan can be achieve. This overcomes the problem of the algorithms optimality only in only specific meta-task set. The test cases taken in the above section shows how the proposed ASA Max-Min algorithm outperforms the Basic Max-Min algorithm in our results. However, many issues regarding other system parameters still persists. Future research can be focused upon taking these constraints into account as well.

REFERENCES

[1] M. Baker, R. Buyya, D. Laforenza, "Grids and Grid Technologies for Wide-area Distributed Computing", Journal of Software-Practice & Experience.

[2] Nayandeep Sran, Navdeep Kaur, "Comparative Analysis of Existing Load Balancing Techniques in Cloud Computing ", vol 2, jan 2013.

[3] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, R. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems", 8th IEEE Heterogeneous Computing Workshop (HCW '99), San Juan, Puerto Rico.

[4] F. Dong, S. G. Akl, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems", Technical Report of the Open Issues in Grid Scheduling Workshop, School of Computing, University Kingston, Ontario.

[5] Karanpreet Kaur, Ashima Narang, Kuldeep Kaur, "Load Balancing Techniques of Cloud Computing", International Journal of Mathematics and Computer Research, April 2013.

[6] Rahmeh OA, Johnson P, Taleb-Bendiab A.,‖A Dynamic Biased Random Sampling Scheme for scalable and reliable Grid Networks‖, The INFOCOMP Journal of Computer Science, vol. 7, 1-10.

[7] Etminani .K, and Naghibzadeh. M, "A Min-min Max-min Selective Algorithm for Grid Task Scheduling," The Third IEEE/IFIP International Conference on Internet, Uzbekistan, 2007.

[8] R. F. Freund, M. Gherrity, S. Ambrosius, M. Camp-bell, M. Halderman, D. Hensgen, E. Keith,T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L.Moore, B. Rust, H. J. Siegel, "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet", 7th IEEE Heterogeneous Computing Workshop

(HCW'98), pp. 184–199 (1998).

[9] T. Kokilavani, Dr. D.I. George Amalarethinam, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing" .

[10] Z. Zhang and Xu. Zhang "A Load balancing mechanism based on ant colony and complex network theory in open cloud computing federation", 2nd International Conference on Industrial Mechatronics and Automation (ICIMA), Wuhan, China, vol. 2, pp.240-243, May 2010.

[11] E. Ullah Munir, J. Li, and Sh. Shi, 2007. "QoS Sufferage Heuristic for Independent Task Scheduling in Grid". Information Technology Journal.

[12] X. He, X-He Sun, and G. V. Laszewski, "QoS Guided Min-min Heuristic for Grid Task Scheduling," Journal of Computer Science and Technology.

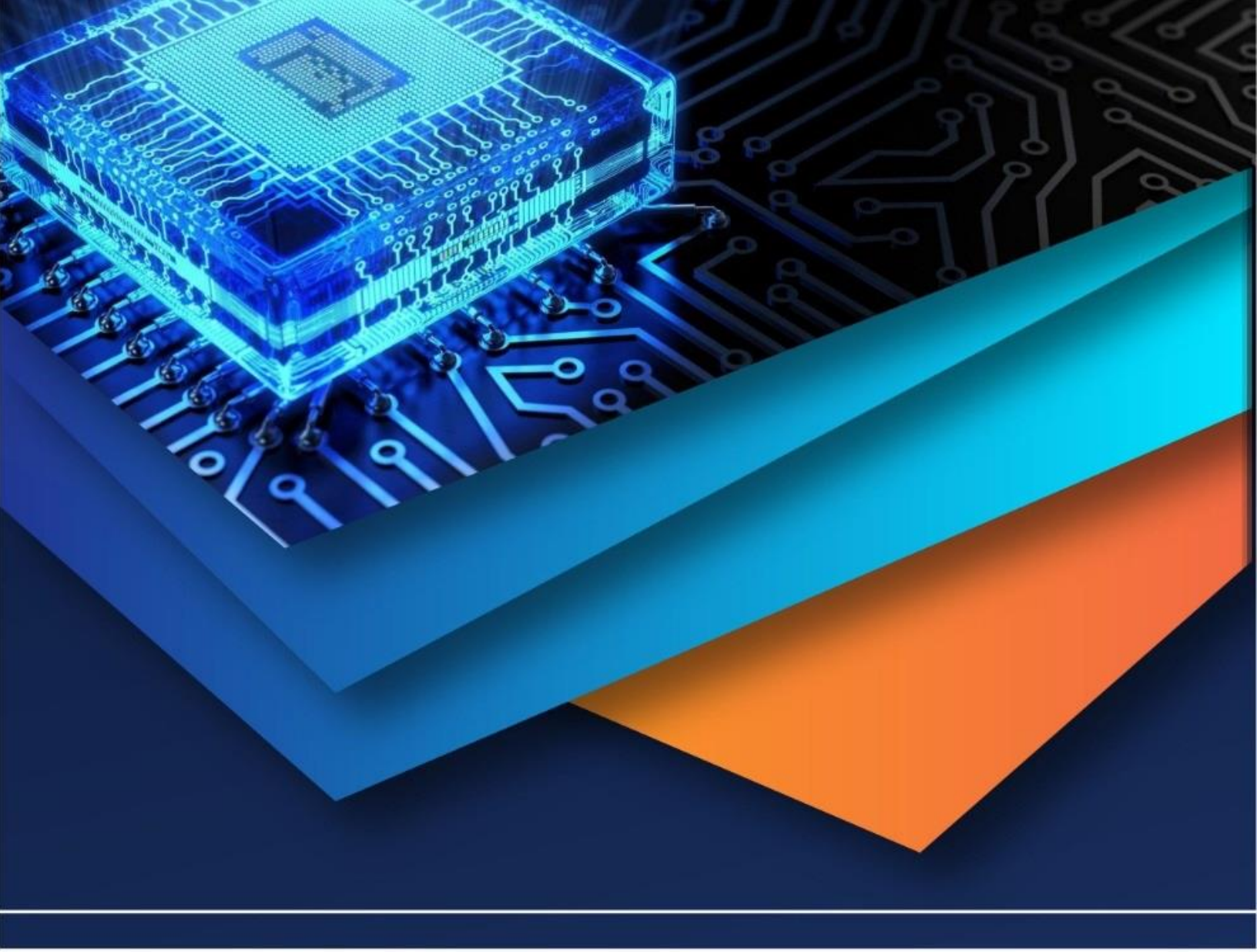
[13] O. M. Elzeki, M. Z. Reshad, M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing", International Journal of Computer Applications.

[14] Kobra Etminani, Mahmoud Naghibzadeh, Noorali Raeeji Yanehsari, "A Hybrid Min-Min Max-Min Algorithm With Improved Performance".

[15] Martin Randles, David Lamb, A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing" in 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops.

[16] George Amalarethinam. D.I, VaaheedhaKfatheen .S, "Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems", International Journal of Computer Science and Information Technologies, Vol. 3 (2) , 2012 ,3659-3663

[17] D.I. George Amalarethinam and P. Muthulakshmi, "An Overview of the scheduling policies and algorithms in Grid Computing ", International Journal of Research and Reviews in Computer Science, Vol. 2, No. 2, pp. 280294, 2011.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY