IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-5, May 2019 E-ISSN: 2347-2693

UMAX Meta Task Scheduling Algorithm in Grid Computing

K. Padma Priya[1], M. Hemamalini[2*]

[1,2]Department of Computer Science, A.V.C College (Autonomous), Mayiladuthurai, Mannampandal, Nagapattinaam District, TamilNadu, India 609305

[*]*Corresponding Author: maliniavcce@gmail.com, Tel.: +91 9442389661*

DOI: https://doi.org/10.26438/ijcse/v7i5.15881592 | Available online at: www.ijcseonline.org

Accepted: 21/May/2019, Published: 31/May/2019

***Abstract*—**Grid computing technology can be seen as a positive alternative for implementing high-performance distributed computing. The goal of Grid computing is to create the illusion of virtual computer out of a large collection of connected heterogeneous nodes. Scheduling jobs on computational grids is identified as NP-hard problem due to the heterogeneity of resources; the resources belong to different administrative domains and apply different management policies. Today a highly secure or virtual grid is very demanding in which you can share any resource from any cluster even with existence of fault in system. In this paper, an algorithm named as UMAX is proposed. This method aims to improve the resource utilization with maximum efficiency and throughput

***Keywords*—** Meta task; Scheduling; Resource utilization; Grid task scheduling

I. INTRODUCTION

The major goal of distributed computing research was to give users an easy, simple and transparent method of access to a vast set of heterogeneous resources. This is generally known as metacomputing. Metacomputing done on local area networks is typically known as Cluster Computing Environments and those that are done on wide area networks are known as Grid Computing Environments. This paper deals with the later one Grid Computing. A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to computational capabilities [1]. Grid computing concepts were first studied and explored in the 1995 I-WAY experiment, in which high-speed networks were used to connect, for a short time, high-end resources at 17 sites throughout the United States. From this experiment, a number of Grid research projects emerged that developed the core basic technologies for Grids in various communities and scientific disciplines. For example, the US National Science Foundation's National Technology Grid and NASA's Information Power Grid are both creating Grid infrastructures to serve university and NASA researchers, respectively. Across Europe and the United States, the closely related European DataGrid, Particle Physics Data Grid and Grid Physics Network (GriPhyN) projects plan to analyze data from frontier physics experiments [2,3].

A. Characteristics of a computational grid

There are many desirable properties and features that are required by a grid to provide users with a computing environment [4, 5]. They are as follows:

- **Heterogeneity:** The grid involves a number of resources that are varied in nature and can encompass a large geographical distance through various domains.
- **Scalability:** The grid should be tolerant to handle a large number of nodes without any performance degradation.
- **Adaptability or Fault Tolerant:** In a grid, unexpected computational aborts, hardware or software faults etc. are high. These faults are generally handled by resource managers.

The major Grid components [6,7] that are necessary to form a grid are as follows:

- **User Level:** this layer houses the application and high-level interfaces. Applications can be varied and encompass a vast variety of problems from chemistry to nuclear engineering. The high-level interfaces implement an interface and protocols allowing the applications and users to access the middleware services.
- **Middleware Level:** the major functionalities of grid systems normally occur in this layer. This layer provides many services such as resource discovery, resource scheduling and allocation, fault tolerance,

security mechanisms and load balancing. It should provide the users a transparent view of the resources available.

- **Resource Level:** This layer typically provides local services that render computational resources such as CPU cycles, storage, computers, network infrastructure, software, etc.

II. RELATED WORK

In a study by [8] Phillipa Sessini, statistical analysis of data based on arrival rate and interarrival times was proposed. In a study by Carsten Franke et al. [9], a summarization of several different approaches for scheduling in Grid environments was proposed. Two Grid Computing Scenarios discussed are HPC Grids and Global Grid. HPC Grids define cooperation between a limited or moderate number of computing sites with high-performance computer systems and a dedicated user community. Global Grid defines large-scale Grids with a diverse number of resources and independent users.

Hemamalini and Srinath [10] proposed the method memory constrained load shared minimum execution Time Grid Meta task scheduling algorithm is used to reduce the makespan and balance the load based on memory requirement of the task. Resource utilization can be calculated and is used to analyze the performance with MET Meta scheduling algorithm.

In the study by Wisnesky [11], the results of a simulation study of a heterogeneous computational grid using different scheduling algorithms were presented. The definition of robustness based on the concept of work completion latency is discussed; a method to simulate grids based on estimated time to compute matrices is presented. The rapid growth in communication capacity among machines today makes grid computing practical, compared with the limited bandwidth available when distributed computing was first emerging.

Therefore, it should not be a surprise that another important resource of a grid is data communication capacity. This includes communications within the grid and external to the grid. Communications within the grid are important for sending jobs and their required data to points within the grid. Some jobs require a large amount of data to be processed and it may not always reside on the machine running the job [11]. The bandwidth available for such communications can often be a critical resource that can limit utilization of the grid.

In a study by Jairam Naik et al. [12], the load is balanced in the grid by grouping the resources into levels as per their success rate and speed. Where, the main consideration was the success rate history of each resource to determine a suitable resource for the job, and to take scheduling decision thereafter. Based on the resource speed requirement, the job is scheduled to the resource of a specific level. Once the job is scheduled on the selected resource, it is removed from the available resource set. When the resource finishes the assigned job then it moves to the set of available resources. If this resource completes the allotted job in estimated time, then its success rate value is incremented by 1, otherwise –1, this value is considered for determining successful resource in the future. With this approach no resource is overloaded.
Load balancing algorithm is an attempt to share the load among all the available resources [13]. The algorithm used to improve the utilization of resources with light load and freeing the resources with heavy load. The two qualities of service factors such as execution time and memory requirement are used for load balancing and effective utilization of resources [14].

The response time minimization algorithm is used to share the load to the available virtual machine efficiently. It minimizes the delay time and improves the response time. Hence the load balancing is achieved. Several performance metrics such as utilization, availability, and responsiveness are used to investigate the impact of different strategies on both provider and user point of views [15].

Balasangameshwara and Raju [16] proposed a fault tolerant hybrid load balancing strategy namely AlgHybrid_LB, which takes into account grid architecture, computer heterogeneity, communication delay, network bandwidth, resource availability, resource unpredictability and job characteristics. AlgHybrid_LB juxtaposes the strong points of neighbour-based and cluster-based load balancing algorithms.

They tackle fault tolerant load balancing by taking into account all the factors pertaining to the characteristics of the grid computing environment mentioned above. This approach eliminates the complexity of a site to gather current state information of the whole grid since real-time monitoring of whole grid will cause system overhead and is completely unrealistic in large-scale grid environment. A well-designed information exchange scheduling scheme is adopted to enhance the efficiency of the load balancing model [16].

A. UMAX Grid Task scheduling Algorithm

The proposed UMAX algorithm concentrates to minimize the overall execution time of given task and to maximize the utilization of all possible resources. So as to improve the utilization, the tasks are separated into three categories.

The tasks that have more load than any other available resources, tasks that have less load among available resources, and the tasks that have equal load among available

resources. The scheduling will be done first to the list that contains more number of tasks. The UMAX algorithm is explained below.

ETC has been generated for the given Tasks (T) and for the given Resources (R). The overall ETC is segregated into three etc[] at line 5. The segregated etc[] is ordered to descending so as to keep the etc collection that has more task in top. Then the threshold value of every etc in etc[] collection is calculated at the beginning of scheduling of corresponding etc.

The task of current etc in etc[] collection is further divided into two with the threshold value and in algorithm it is named aetc. The aetc, which is lesser to t threshold value is applied with MinMin algorithm and the aetc which is higher to t threshold value is applied with MaxMin algorithm.

UMAX Algorithm

1. Number of tasks (*n*), Number of machines (*m*),
2. Grid $G = \{M1, M2, \ldots, Mm\}$, Tasks $T = \{T1, T2, \ldots, Tn\}$,
3. Machines availability *R*; Estimated time of computation ETC.
4. *t*-Threshould value
5. etc[]<=Segregate(ETC)
6. OrderDescend(etc)
7. $i = 0$
8. do
9. aetc<=etc(*i*)
10. $t = \sum \frac{E(t)}{n}$ where $t \in T$
11. $j = 0$
12. do
13. if ($E(t)$<=t)
14. FS<=MinMin(aetc)
15. Else
16. FS<=MaxMin(aetc)
17. $j = j+1$
18. do step 12
19. $i = i+1$
20. do step 8

where,
etc[] is a collection of ETC
aetc an temporary etc
$E(t)$ means Execution time of T

FS means Final Schedule

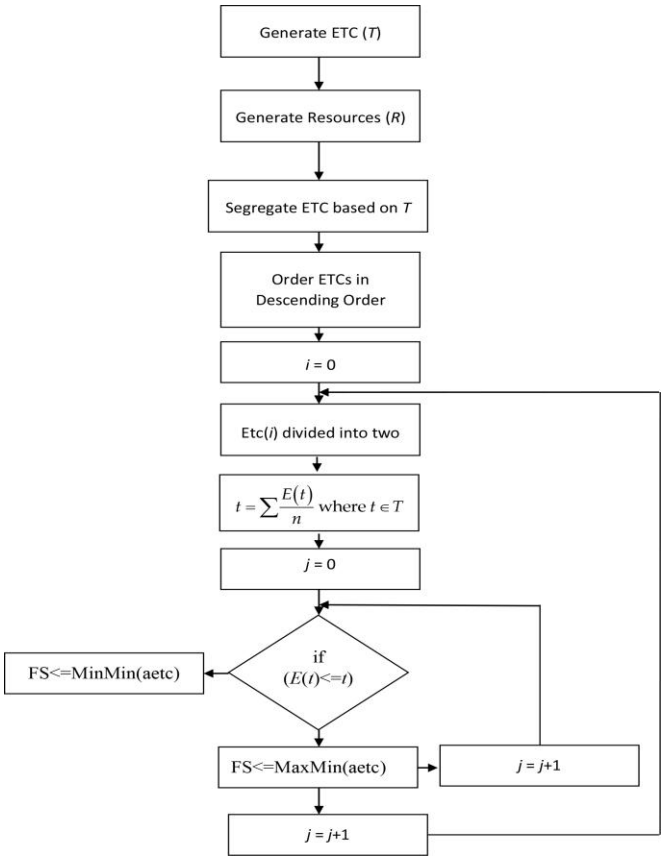

Figure 1. Flow chart of UMAX Grid Task scheduling Algorithm.

To prove the accuracy of the proposed algorithm, a java application is developed with Grid Simulation framework; the results are discussed here in next chapter.

III. EVALUATION AND RESULTS

To show the accuracy of proposed algorithm, the sample input and results are given below. Table 1 shows the task table which has 10 tasks and 3 resources.

\

Table 1. Task table

Number of tasks		10
Number of resource		3
Task table		
ID	Index	Load
T1	1	493
T2	2	5,930
T3	3	7,985
T4	4	7,433
T5	5	6,170

T6	6	2,180
T7	7	9,989
T8	8	8,245
T9	9	8,678
T10	10	5,911

Table 2. Resource table

Resource table		
ID	Index	Load
R1	1	958
R2	2	232
R3	3	260

Table 3. Generated ETC matrix from given inputs

Generated ETC from given inputs			
Task	Resource R1	Resource R2	Resource R3
T1	294.4542852	934.7201808	1,022.437468
T2	1,215.573243	1,748.851374	1,903.542347
T3	1,110.858255	1,364.217349	1,440.423005
T4	674.0213821	1,262.708893	1,648.849057
T5	690.8878372	1,111.471234	1,135.652767
T6	1,120.56217	1,151.878974	1,714.133885
T7	1,048.29944	1,352.554074	1,581.813914
T8	615.3903528	662.526237	828.622274
T9	920.5175737	1,368.532292	1,562.029257
T10	1,061.809359	1,327.687129	1,555.508831

For the above inputs, the MaxMin scheduling is given below:

Table 4. Result of MaxMin Grid task scheduling algorithm

MaxMin			
Task	Resource	Start time (in milliseconds)	End time time (in milliseconds)
T8	R3	0	828.62227
T1	R3	828.622274	1,851.0597
T5	R3	1,851.059742	2,986.7125
T3	R3	2,986.712509	4,427.1355
T10	R3	4,427.135514	5,982.6443
T9	R3	5,982.644345	7,544.6736
T7	R3	7,544.673602	9,126.4875
T4	R3	9,126.487515	10,775.337
T6	R3	10,775.33657	12,489.47
T2	R3	12,489.47046	14,393.013
MakeSpan: 14,393.012805359922			

For the above inputs the UMAX scheduling is given below:

Table 5. Results of UMAX Grid task scheduling algorithm

UMAX			
Task	Resource	Start time	End time
T2	R3	0	1,440.423
T3	R3	1,440.423005	2,995.9318
T4	R3	2,995.931836	4,557.9611
T6	R3	4,557.961093	6,139.775
T7	R3	6,139.775007	7,788.6241
T9	R3	7,788.624064	9,502.7579
T10	R3	9,502.75795	11,406.3
T1	R1	0	294.45429
T5	R1	294.4542852	909.84464
T8	R1	909.844638	1,600.7325
MakeSpan: 11,406.300296780904			

The results given by the tool developed for various inputs is given in Table 1. Makespan produced by MaxMin and UMAX algorithm is given in milliseconds. Even the numbers shows that the proposed algorithm yields best result than the traditional MaxMin algorithm.

Table 6. Results of MaxMin and UMAX algorithm for various inputs

No. of tasks & resource	MaxMin (in milliseconds)	UMAX (in milliseconds)
50 × 5	68,910.2891	44,152.20595
100 × 10	142,593.911	85,314.16829
500 × 50	730,342.207	428,401.9082
1,000 × 50	1,500,993.91	878,847.0089

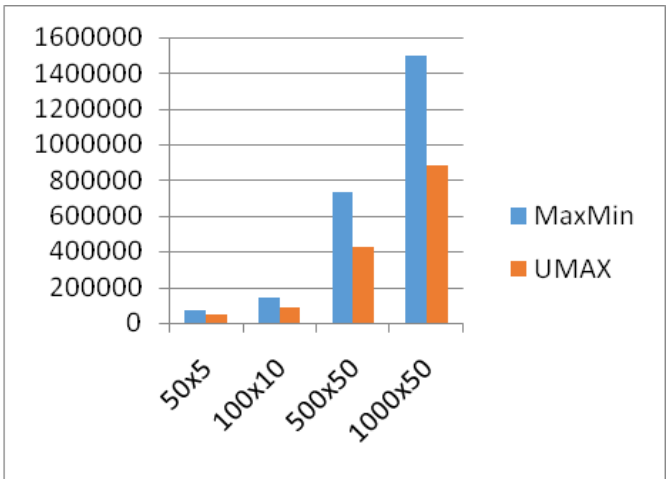

Figure 2. Comparison of MaxMin and UMAX Grid task scheduling algorithm. In nature, everything has its own negatives. Similar to that, the proposed UMAX algorithm generates good result for high number of tasks but in case of minimum number of tasks, the proposed algorithm has yielded high makespan than MaxMin.

In nature, everything has its own negatives. Similar to that, the proposed UMAX algorithm generates good result for high number of tasks but in case of minimum number of tasks, the proposed algorithm has yielded high makespan than MaxMin.

IV. CONCLUSION

One of the most important issues in grid computing is scheduling, which is an allocation of jobs on the available network resources in order to accomplish the tasks in the shortest possible time. It is one of the most difficult tasks in the grid computing. This paper proposed a new job scheduling algorithm UMAX. This algorithm is able to provide good results in terms of makespan and resource utilization.

REFERENCES

[1] W. Gentzsch, "DOT-COMing the GRID: Using Grids for Business", Sun Microsystems Inc, Palo Alto, California, USA, pp. 1–3, 2002.
[2] A News Article Web Page in "Nature", Available at: http://www.nature.com/nature/webmatters/grid/ grid.html.
[3] M. Baker, R. Buyya, D. Laforenza, "The Grid: International Efforts in Global Computing", SSGRR 2000 The Computer & eBusiness Conference, l`Aquila, Italy, July 31 2000 – August 6 2000.
[4] C. Germain, V. Néri, G. Fedak, F. Cappello, "XtremWeb: Building an Experimental Platform for Global Computing", Grid2000, IEEE Press, December 2000.
[5] I. Foster, C. Kesselman, S. Tuecke, *"The Anatomy of the Grid: Enabling Scalable Virtual Organizations"*, The International Journal of High Performance Computing Applications, Vol. 15, Issue. 3, pp. 200–222, 2001.
[6] I. Foster, C. Kesselman, Eds., "Computational Grids", In: "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 1998
[7] D.C. Arnold, S.S. Vahdiyar, J. Dongarra, "*On the Convergence of Computational and Data Grids*", Parallel Processing Letters, Vol. 11, pp. 187–202, 2001, ISSN 0129-6264.
[8] P. Sessini, "Scheduling in Grid Computing Systems", University of Calgary, Alberta, Canada, 2015.
[9] C. Franke, U. Schwiegelshohn, R. Yahyapour, "Job Scheduling for Computational Grids", University of Dortmund, Germany.
[10] M. Hemamalini, M.V. Srinath, "*Memory Constrained Load Shared Minimum Execution Time Grid Task Scheduling Algorithm in a Heterogeneous Environment*", Indian Journal of Science & Technology, Vol. 8, 2015, ISSN (Print): 0974–6846, ISSN (Online): 0974–5645.
[11] R.J. Wisnesky, "Evaluating Scheduling Algorithms on Distributed Computational Grids", 2002.
[12] K. Jairam Naik, K. Vijaya Kumar, N. Satyanarayana, "*Scheduling Tasks on Most Suitable Fault Tolerant Resource for Execution in Computing Grid*", IJGDC, Vol. 5, pp. 121–132, 2012.
[13] M. Hemamalini, "*Review on Grid Task Scheduling Algorithm in a Distributed Heterogeneous Environment*", International Journal of Computer Applications, Vol. 40, Issue. 2, pp. 24–30, 2012.
[14] M. Hemamalini, Dr. M.V. Srinath, "*State of the Art: Task Scheduling Algorithms in Heterogeneous Grid 0 50 100 150 200 250 Time in Milliseconds Response Time Minimization*", International Journal of Computer Applications, Vol. 145 p. 14, 2016, "*Computing Environment*", Elysium Journal of Engineering Research Management, Vol. 1, August 2014.
[15] M. Hemamalini, M.V. Srinath, "*Response Time Minimization Task Scheduling Algorithm*", International Journal of Computers and Applications, Vol. 145, pp. 9–14, 2016.
[16] J. Balasangameshwara, N. Raju, "*A Hybrid Policy for Fault Tolerant Load Balancing in Grid Computing Environments*", Journal of Network and Computer Applications, Vol. 35, Issue. 1, pp. 412–422, 2012.

Authors Profile

Dr. M. Hemamalini pursued Masters Degree in Computer Applications from A.V.C College (Autonomous), Mannampandal, Bharathidasan Univresity in the year 1999 and obtained Ph.d in computer science from STET Women's college, sundarakkottai, Mannargudi, Bharathidasan University in the year 2018. She currently working as Assistant Professor in Department of computer science, A.V.C College (Autonomous), Mannampandal, Mayiladuthurai, Tamil Nadu, India. She has published papers in National and International journals and she also presented papers in National and International conferences. Her research interests include Grid Computing, Mobile Computing, Cloud Computing and Data Mining. She has 17+ years of teaching experience.

Ms. K. Padma Priya pursued Masters Degree in Computer Science from Bharathidasan University, Trichy in the year 2017 and Bachelor of Education degree from Indira Gandhi National Open University, New Delhi in the year 2005. She is currently pursuing M.Phil (Part time) degree in A.V.C. college (Autonomous), Mayiladuthurai, Bharathidasan University and currently working as Computer Instructor in Government Higher Secondary School, Andarmullipallam, Cuddalore District, Tamil Nadu, India. Her research interest focuses on grid computing. She has 20 + years of teaching experience.

