Max-Average: An Extended Max-Min Scheduling Algorithm for Grid Computing Environment

J.Y Maipan-uku, A. Muhammed, A. Abdullah, M. Hussin
Department of Communication Technology and Networks,
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia, 43400 UPM Serdang, Selangor D.E., Malaysia.
jamilu4yahaya@yahoo.co.uk

***Abstract*—Sharing numerous computational and communication power from connected heterogeneous systems over the world are the two key points of Grid computing. Grid computing can also be referred as a computing platform for users to utilise the remote heterogeneous resources for solving their large scale jobs that require a huge amount of processing power or a huge data storage. Sharing these resources that way effectively requires a very good scheduling strategy, which is the focus of this research. This paper presents a new proposed grid based scheduling algorithm called Max-Average, inspired from Max-Min algorithm. In order to produce good quality solutions, the proposed algorithm is designed in two phases; firstly it uses an initial task queue like the traditional Max-Min for estimating task completion time for each of resources, and in the second phase choose the fitting resource for scheduling according to requirements. The results from our simulation showed that our proposed algorithm is performing better in producing good quality solutions, particularly in executing tasks fast and in balancing the load (resource utilisation) among the resources more effectively when compared to standard Minimum Execution Time (MET), Minimum Completion Time (MCT), Min-Min, and Max-Min heuristic approaches.**

***Index Terms*—Scheduling Algorithm; Grid Computing; Minimum Execution Time (MET); Minimum Completion Time (MCT).**

I. INTRODUCTION

Grid computing provides medium of using oodles of computers worldwide, stretching from simple laptops, to a cluster of computers and supercomputers connected over heterogeneous network in an effective, secure and dependable manner. However, the inspiring problem of heterogeneously and adaptively allocating resources in response to demanding application requests remain unresolved [1]. The performance objective for a grid environment is to efficiently utilise the numerous resources indoors. Most grid systems include some kind of job scheduling algorithm [2].

Scheduling algorithm in [3] is considered as a significant subject in Grid computing, especially in utilising shared resources. The need of having good scheduling algorithms to obtain high performance computing is increase. It is often hard to have an ideal resource scheduler that is able to minimise job completion time (makespan) and to utilise the available resources efficiently. The three key stages [4] for task scheduling in grid environment remain resource collection, resource ability (information) and task execution. Selecting best resources for tasks/jobs execution has remained NP-complete problem.

Grid task scheduler (scheduling algorithm) is responsible for allocating tasks to resources under grid environment for execution [1]. In computational grid, scheduling problem is enhanced by minimising makespan and maximising system utilisation, balancing the loads and fulfils economical system demand and user constraints [5].

The remainder of the paper is organised as follows. Section II presents scientific literature. Section III discusses the two algorithms (Min-Min and Max-Min) and the performance metrics used in this paper. Our proposed approach named Max-Average (An Extended Max-Min) scheduling algorithm is presented in Section IV. Section V describes the methodology used and experimental results. Finally, Section VI summarises the research findings and presents the future work.

II. RELATED WORKS

Many researchers have done research on Max-Min algorithm for heterogeneous computing scheduling due to its capability to produce good quality solutions. These include; Maheswaran et al., in [6] reviewed four heuristics for dynamic mapping of a class of independent tasks onto heterogeneous computing systems include Max-Min, Braun et al., [7] studied eleven heuristics for static scheduling in heterogeneous computing environments, Max-Min inclusive. Also, Fujimota et al., in [8] compared scheduling algorithms for independent coarse-grain tasks; among them is Max-Min. Xhafa et al., [9] assessed numerous static scheduling policies for allocations of jobs on resources using the batch mode method, including Max-Min. Similarly, Luo et al., [10] analysed and relate a set of twenty heuristics under different circumstances.

However, the Max-Min algorithm undergone very few extensions by researchers due to its capability of reducing idle time of resources (utilisation). Ming and Li [11] proposed an improved Max-Min algorithm for cloud task scheduling. Amalarethinam and Kfatheen [12] proposed Max-Min based algorithm. In this algorithm, tasks are assembled like Max-Min at the first phase, for selecting resource, mean of completion time (meanCT) is compared with resource completion time (CTj), and then if CTj is less than or equal to meanCT, task with maximum completion time is scheduled otherwise best

maximum execution time is scheduled. Devipriya and Ramesh [13] proposed an improved Max-Min algorithm for task scheduling in cloud computing. In this algorithm, task with maximum execution time is assigned to resources that produced its minimum completion rather than assigning task with maximum completion time, to the resource which provides minimum task execution time. Similarly, Mao et al. [14] presented Max–Min task scheduling algorithm for load balancing in cloud computing, and other related work is Li et al., as in [15].

Moreover, some researchers hybridised Max-Min with Min-Min that considers the task with minimum execution time for mapping at first will yield a reasonable benefit in overcoming its drawback. Etminani and Naghibzadeh [16] presented selective algorithm, Wenzheng and Wenyue [17] presented filter Min-Min that considers the value of average completion time and standard deviation of all resources and chooses either Min-Min or Max-Min for mapping. Likewise, Parsa et al. [18] introduced Resource Aware Scheduling Algorithm (RASA). In this algorithm Min-Min is applied when the number of available machines is odd, otherwise Max-Min is applied. Gupta and Singh [19] proposed Switcher algorithm that chooses between the two algorithms under a prescribed conditions. Similarly, Anousha et al. [20] improved total completion time (makespan) through his proposed algorithm where the algorithm is designed to select either Max-Min or Min-Min algorithm for the mapping task to available resource after comparing the summation time of all jobs except the maximum value. In another work, Panda et al., in [21] proposed Skewness-based grid task scheduling that chooses among the two algorithm base on prescribe condition.

It is obvious that, task selection is a key challenge to this heuristic. For this reason, a substantial enhancement in the computational efficiency of the algorithm is necessary.

Table 1
Notations and its definition

Notations	Definition
X_{min}	Minimum execution time
X_{max}	Maximum execution time
MaxAverage	Efficient Max-Min
Meanest	Minimum execution, completion time
T_i	Meta-task Id of meta-task i
ET_i	Execution Time
minET	Minimum Execution Time
IR_j	Resource Id of resource j
$C_{i,j}$	Completion time for meta-task i on resource *j*
$X_{i,j}$	Execution time for meta-task i on resource *j*
R_j	Ready time of j
RU	Resource Utilisation
Avgru	Average resource utilisation
MT	Meta-Tasks

III. SCHEDULING ALGORITHMS

A. Min-Min Scheduling Algorithm

Min-Min algorithm discovers task that has a *minET* and allocating it to the machine that produces its minimum *CT* as shown in Figure 1. Then resource ready time is keeping up to date. This procedure is iterated until the whole tasks are mapped [22]. This algorithm has a problem of high makespan production, low resource utilisation, and load imbalanced when numbers of tasks that have minimum time to be executed are much more comparable to the tasks that have maximum time to be executed.

Standard Min-Min Algorithm

Begin with tasks t_i in Mt
Begin with machines r_j in m_j
Calculate Completion Time ($CT_{ij} = ET_{ij} + r_j$)
do till the entire tasks in the meat are allocated
For a separate task t_i in Mt
discover minimum CT_{ij} and machine that acquires it.
discover task t_k with least CT_{ij}.
allot t_k to resource m_l that
delete $t_k \epsilon$ mt
keeps r_l up to date
keeps CT_{ij} up to date for all i

Figure 1: Steps for Min-Min Algorithm

B. Max-Min Scheduling Algorithm

In [19], Max-Min differs from Min-Min in the second phase (line 7), this is where tasks with an overall maximum expected completion time from MT is chosen and assigned to the corresponding machine as shown in Figure 2. Then resource ready time is keeping up to date. This procedure is re-iterated for all available tasks.

Standard Max-Min Algorithm

Begin with tasks t_i in MT
Begin with machines m_j
Calculate Completion Time (CTij = ETij + rj)
do till the entire tasks in the meat are allocated
For a separate task t_i in Mt
discover minimum CT_{ij} and machine that acquires it.
discover the task t_k with the maximum CTij.
allot tk to resource m_l that
delete $t_k \epsilon$ MT
keeps r_l up to date
keeps CTij up to date for all i
end do

Figure 2: Steps for Max-Min Algorithm

C. Performance Metrics

Performance metrics are typically used to determine the effectiveness and efficiency of a scheduler with respect to grid users or service provider requirement. Different number of performance metrics can be used to describe resource organisation and scheduling system's efficiency in computational Grid. In this paper, we used two performance metrics as described below;

i. Makespan

Makespan is the time taken to execute the most recent task. This parameter shows the quality of assignment of resources from the executional time perspectives. Makespan is calculated as in Equation (1).

$if\ T = t_1, t_2, t_3 \ldots, t_n$ *is the set of tasks submitted to the scheduler and*

$R = r_1, r_2, r_3, \ldots, r_n$ *is the set of resources available at the time of task arrival,*

$$Makespan = \max\{completion\ time\ (CT_j)\ \forall\ j\ in\ R\} \quad (1)$$

Also, $CTij = (ET_{ij}) + (R_j)$

where:

ET_{ij} = Expected Execution Time of the task t_i on machine m_j.

R_j = Time when machine m_j is ready to execute t_i.

ii. Resource Utilisation

Minimising resource idle time implies its utilisation rate achieved. This parameter shows the efficiency of an algorithm in keeping the available resources busy throughout the simulation time. In this research, since we are dealing with statics jobs, average resource utilisation is considered. Equations 2 and 3 illustrate the pattern of calculating resource utilisation and average resource utilisation respectively.

$$ru = \sum \forall j, R_{ij=1} \frac{(R_{rt} - R_{it})}{makespan} * 100 \quad (2)$$

where:

R_{rt} = Busy time of resource

R_{it} = Unused time of resource

$$Avgru = \frac{\sum_{i=1}^{n}(ru)}{n} \quad (3)$$

where:

$\{n$ = number of resources$\}$

IV. PROPOSED ALGORITHM (MAX-AVERAGE)

Satisfying shareholders of computational grid hinge on the eminence of schedule produced by scheduling algorithm. Ideologically, it's possible to obtain a reasonable schedule policy if we can allocate tasks for execution to resources that likely produce it minimum processing time. Max-Min algorithm as argued previously do this, but allocating tasks in order of Max-Min result in large value of Makespan, and poor resource utilisation in the computational grid, if longer tasks are much more than the smaller tasks. This is the major drawback of Max-Min algorithm. However, the main contributions of this article are;

- *to minimise total completion time (makespan)*
- *to make sure of uniformity in resource utilisation*

However, we present our proposed algorithm in Figure 3. Firstly, all the tasks will be arranged in increasing order. This means, tasks with a minimum time of execution are in the frontage of the train and task with maximum time of execution at the end of the queue. Secondly, average of completion time for executing the entire tasks on the available resources is computed and appropriate resource is chosen based on defined condition.

Proposed Algorithm (Max-Average)

sort all tasks in MT in non-decreasing order
while there are tasks in MT
for all submitted tasks in the set; T_i
for all resources; R_j
calculate completion time (CT_{ij}) = et_{ij} + rt_j; (for each task in all resources)
discover the minimum CTij and resource Rj
if there is more than one resource that obtains it
select resource with least usage so far // for stability
discover AverageCT // and hold it
compare AverageCT with X_{min}
if AverageCT ≤ X_{min}, assign X_{max} to the resource with MinECT
else assign X_{min} to the resource with MinECT
end if
remove the task from the set
keeps ready time of the selected resource R_j up to date
keeps ct_{ij} up to date for all T_i
end while

Figure 3: Pseudo code of the proposed algorithm (Max-Average)

For selecting a task to schedule, we compute the average completion time of all resources as a display in Equation 4 below:

$$AverageCT = \frac{\sum_{n=1}^{m} C_{ij}}{n} \quad (4)$$

After that, the proposed algorithm compares value of AverageCT with X_{min}, then select task as follows:

1. If *AverageCT* is less than or equal to X_{min}, it means the length of smaller tasks in MT is more than the heavy ones, so we will select from the rear of queue to assign the next task (X_{max}).
2. Otherwise, assigned task with minimum completion time.

A. Proposed Algorithm Time Complexity

From Figure 3, the proposed algorithm order depend on lines (3) and (4) for-loop, this is also applicable for all tasks in line (2). Position (3 – 5) hold two nested for-loop with *0(T.R)* time: The inner for-loop goes *R* times (number of resources) and outer for-loop goes *T* times (number of tasks). This process is carried out for all tasks in *MT*. Therefore, lines (2-17) take $O\ (T^2R)$ time. So, our proposed algorithm takes $O\ (T^2R)$ time.

B. Descriptive Example

We consider a problem with resources R_1 and R_2 along with a set of task t_1, t_2, t_3 and t_4. The algorithm schedules all the tasks based on the existing processor R_1 and R_2 as shown in Table 1.

Table 1
Execution Time

Task/Resource	R_1	R_2
T_1	9	25
T_2	7	18
T_3	10	41
T_4	8	19

From Table 1, Min-Min algorithm assigns all tasks to resource R_1 and leave resource R_2 idle achieving a makespan of 34 seconds. Similarly, Max-Min produces a makespan of 26 seconds with resource utilisation of 100% on R_1 and 70% on R_2. However, our proposed algorithm produces a makespan of 25 seconds with evenly tasks distribution among resources, achieving 100% on R_1 and 100% on R_2. Hence, the proposed algorithm has better makespan and resource utilisation rate when compared with Min-Min and Max-Min algorithms.

V. Methodology, Results And Discussion

Due to the difficulties in implementing, testing and evaluating the performance of the proposed algorithm with an existing one (Min-Min, Max-Min, MCT and MET) on a real system (test-bed), a Java based simulation running on Intel (R) core (TM) i5-3470 CPU @ 3.20GHz, 3.20GHz has been implemented. To compare our proposed method among the heuristic algorithms, we use the Expected Time to Compute (ETC) model of benchmark simulation proposed by Braun et al. [7]. This model is based on Expected Time to Compute (ETC) matrix of *m* tasks and *n* resources. Four different instances are used. These instances are based on task heterogeneity and resource heterogeneity as described below;

h_ih_i : ***heavy*** *set of tasks along with* ***high*** *capacity resources.*
h_il_o ***heavy*** *set of tasks along with* ***low*** *capacity resources.*
l_oh_i ***light*** *set of tasks along with* ***high*** *capacity resources*
l_ol_o ***light*** *set of tasks along with* ***low*** *capaciy resources*

Table 2:
The makespan performance of different algorithms using 512 x 16 (in seconds)

512 x 16	MET	MCT	Min-Min	Max-Min	Max-Average
hihi	89	88	90	85	97
hilo	81	79	83	91	97
lohi	60	80	79	65	95
lolo	47	64	76	80	96

Tables 3: Average resource utilisation comparison with different heuristics (in percentage)

512 x 16	MET	MCT	Min-Min	Max-Min	Max-Average
hihi	151.41	124.95	125.40	133.65	118.48
hilo	207.19	179.93	173.69	162.57	153.47
lohi	59.99	67.99	70.81	84.95	58.76
lolo	56.99	94.71	92.61	90.34	75.48

A. Makespan

Table 2 presents the results of makespan performances, produced by different scheduling algorithms in this research. From the experimental results, it showed that the proposed algorithm is outperforming all the existing algorithms substantially in three scenarios (*hihi, hilo, and lohi*) while approaching the performance of MET with a slight different when the number of tasks are of more light and low capacity resources (*lolo*). Similarly, Max-Min is outperforming Min-Min, MCT and MET in three the same scenarios as our proposed algorithm did.

However, MCT outperforms MET in two scenarios (*hihi & hilo*) while MET performs worst in two scenarios (*hihi*) and better in two cases (*lohi & lolo*) compared to Max-Min, Min-Min and MCT. We depict a pictorial chart of the makespan for Max-Average and Max-Min in Figure 4.

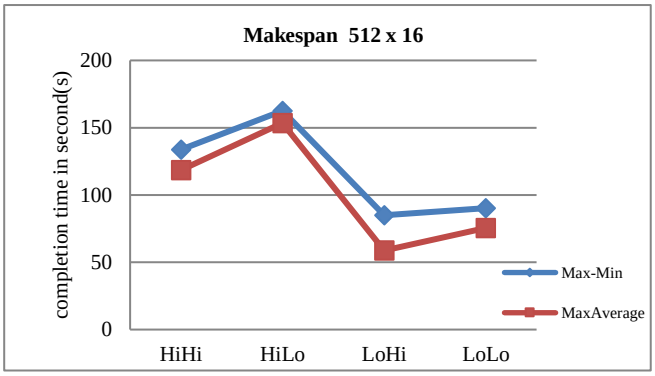

Figure 4: Makespan Comparison of Max-Min and Max-Average for 512 x 16

B. Resource Utilisation

Table 3 presents the values of *Avgru* for the five mentioned algorithms. Max-Average is the top performing, able to produce the maximum resource utilisation for all instances. This is because, in both cases, our proposed algorithm was able to effectively engage with all the available resources. Figure 5 shows the effects of using Max-Average algorithm in comparison with the standard benchmark, Max-Min algorithm. The results showed that the performance produced by our proposed algorithm is far better in all scenarios (all above the value of 94%).

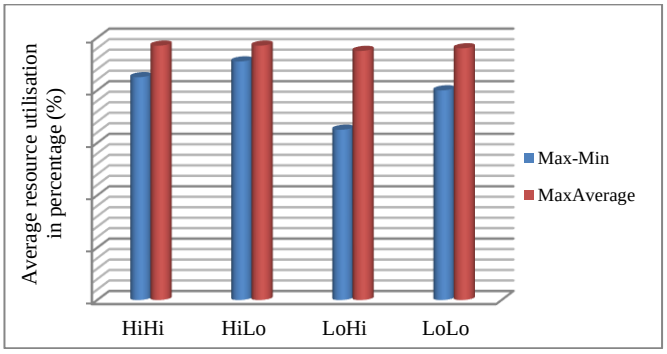

Figure 5: Average resource utilisation of Max-Min and Max-Average using 512 x 16

VI. Conclusion and Future Work

An efficient scheduling algorithm plays a crucial role, particularly in minimising the makespan (job completion time) from the point of view of the grid user and also utilising the grid resources which is important to the grid providers. Thus, our proposed algorithm is designed to address these two problems

and results of the simulation show that the proposed algorithm is outperforming standard Max-Min and other traditional scheduling algorithms. We plan to investigate the effects of applying "deadline" to the tasks and resources and also priority among jobs as our future work.

REFERENCES

[1] Xhafa, F. (2008). Metaheuristics for Scheduling in Distributed Computing Environments. *Springer-Verlag*, pp. 2-9.

[2] Jacob, B. (2005). Introduction to grid computing. *United States: IBM, International Technical Support Organization.* Vol. 1(1), pp. 100.

[3] Kokilavani, T. and Amalarethinam, D.I.G., (2011). Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing. *International Journal of Computer Applications*. Vol. 20(2), pp. 43-47.

[4] Kokilavani, T., and Amalarethinam, D. I. (2010). Applying Nontraditional Optimization Techniques to Task Scheduling In Grid Computing-An Overview. *International Journal of Research & Reviews in Computer Science*. Vol. 1(4), pp. 34-38.

[5] Hemamalini, M., (2012). Review of Grid Task Scheduling in Distributed Heterogeneous Environment. *International Journal of Computer Applications*. Vol. 40 (2), pp. 24 – 26.

[6] Maheswaran, M., Ali, Siegel, H. J., Hensgen, D. and Freund, F. R. (1999). Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems1. *Journal of Parallel and Distributed Computing*. Vol. 59(2), pp.107 – 131.

[7] Braun, T. D., Siegel, H. J. and Beck, N., (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*. Vol. 61, pp. 823 – 831.

[8] Fujimoto, N., and Hagihara, K. (2004). A Comparison among Grid Scheduling Algorithms for Independent Coarse-Grained Tasks. Vol. 2(4), pp. 7-7.

[9] Xhafa, F., Barolli, L., and Durresi, A. (2007). Batch mode scheduling in grid systems. *International Journal of Web and Grid Services*, Vol. 3(1), pp. 19-19.

[10] Luo, P., and Shi, Z. (2007). A revisit of fast greedy heuristics for mapping a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel and Distributed Computing*. Vol. 67(6), pp. 695-714.

[11] Ming, G., and Li, H. (2011). An Improved Algorithm Based on Max-Min for Cloud Task Scheduling. *Recent Advances in Computer Science and Information Engineering Lecture Notes in Electrical Engineering*. Vol. 125, pp. 217-223.

[12] Amalarethinam, G.D.I. and Kfatheen V.S., (2014). Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems. *International Journal of Computer Science and Information Technologies*. Vol. 3, pp. 3659-62.

[13] Devipriya, S., and Ramesh, C. (2013). Improved Max-Min Heuristic Model for Task Scheduling in Cloud. *IEEE*, pp. 883-888.

[14] Mao, Y., Chen, X., and Li, X. (2014). Max–Min Task Scheduling Algorithm for Load Balance in Cloud Computing. *Proceedings of International Conference on Computer Science and Information Technology, Advances in Intelligent Systems and Computing*. Vol. 255, pp. 457-465.

[15] Li, X., Mao, Y., Xiao, X., and Zhuang, Y. (2014). An Improved Max-Min Task-Scheduling Algorithm for Elastic Cloud. *International Symposium on Computer, Consumer and Control*, pp. 340-343.

[16] Etminani, K., Naghibzadeh, M., and Yanehsari, N.R., (2007). A Hybrid Min-Min Max-Min Algorithm with Improved Performance. Department of Computer Engineering, *Ferdowsi University of Mashad, Iran*. Vol.32, pp. 1 – 3.

[17] Li, W., and Zhang, W. (2009). An improved Scheduling Algorithm for Grid Tasks. *International Symposium on Intelligent Ubiquitous Computing and Education*. Vol. 35, pp. 9-12.

[18] Parsa, S and Reza, E. M., (2009). RASA: A New Task Scheduling Algorithm in Grid Environment. *World Applied Sciences Journal*. Vol. 7, pp. 152-155.

[19] Gupta, K., and Singh, M., (2012). Heuristic Based Task Scheduling In Grid. *International Journal of Engineering and Technology (IJET)*. Vol. 4, pp. 254 – 258.

[20] Anousha, S., Shoeib, A., and Ahmadi, M. (2014). A New Heuristic Algorithm for Improving Total Completion Time in Grid Computing. *Springer-Verlag Berlin Heidelberg*, pp. 17-26.

[21] Panda, S., Agrawal, P., Khilar, P., & Mohapatra, D. (2014). Skewness-Based Min-Min Max-Min Heuristic for Grid Task Scheduling. *In 4th IEEE International Conference on Advanced Computing and Communication Technologies.* pp. 282-289.

[22] Vijayalakshmi, R., and Vasudevan, V. (2015). Static Batch Mode Heuristic Algorithm for Mapping Independent Tasks in Computational Grid. *Journal of Computer Science*. Vol. 11(1), pp. 224.