ISSN 2277-3061

An Improved Min-Min Task Scheduling Algorithm with Grid Utilization and Minimized Makespan

Lalla Singh, Neha Agarwal, Mohammad Manjur Siddiqui
Department of Computer Science & Engineering, Shri Ramswaroop Memorial University
Lucknow, India
lallasingh1991@gmail.com
Department of Computer Science & Engineering, Shri Ramswaroop Memorial University
Lucknow, India
lko.neha@gmail.com
Department of Computer Science & Engineering, Shri Ramswaroop Memorial University
Lucknow, India
manjur.siddiqui1992@gmail.com

ABSTRACT

Grid computing is hardware and software infrastructure which offers a economical, distributable, coordinated and credible access to strong computational abilities [1]. For optimal use of the abilities of large distributed systems, necessitate for successful and proficient scheduling algorithms is enforced. For diminution of total completion time and improvement of load balancing, many algorithms have been executed. In this paper, our goal is to propose new scheduling algorithm based on well known task scheduling algorithm i.e. Min-Min[1]. The proposed algorithm tries to use the advantages of this basic algorithm and excludes its drawbacks with better grid utilization and minimized makespan. In comparison to existing algorithms like Min-Min and improved Min-Min algorithm[1], our proposed algorithm is achieving better results for considered parameters.

Indexing terms/Keywords

Grid resource, task scheduling algorithm, Min-Min, completion time.

Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol.14, No.8
www.ijctonline.com, editorijctonline@gmail.com

INTRODUCTION

Cutback of Makespan is a fundamental Motive of optimizing task scheduling algorithm in distributed systems. In this field, a lot of attempts have been made and Vast projects such as Globus [18] and Condor [2] for the development of computational resources in computer networks is presented. The Grids use of resources of connected- computers to the network and using the outcome of these resources to smoothly do complex reckoning. They do this with fragmenting of resources and allocation of them to a computer in the network. Resource allocation is done in two stages: Resource discovery and resource selection.

Stage 1 (Resource discovery)

In this stage, List of all available resources is prepared. Actually, resource discovery generates a list of potential resources.

Stage 2 (Resource selection)

This stage involves collecting information of resources and selecting the best set to match the application requirements. After this, the task is executed.

To make effective use of the huge capabilities of the computational grids, efficient task scheduling algorithms are required [9]. Many Grid task scheduling algorithms such as [9, 10] have some features in common, that are performed in multiple steps to solve the problem of matching application needs with resource availability and providing quality of service. Also we know that solving the matching problem to find the choice of the best pairs of jobs and resources is NPcomplete problem [17]. The well known example of algorithms is Min-min [17]. This algorithm estimate completion times of each of the tasks on each of the grid resources. Estimating the execution time of each task on different resources, the Min-min algorithm selects the task with minimum completion time and assigns it to the resource on which the minimum execution time is achieved. The algorithm applies a same procedure to the remaining tasks [8]. The Min-Min algorithm seems to do worse operation, whenever the number of small tasks is much more than the large ones. So, proposing a new algorithm to resolve the above mentioned problem is required.

This paper offers a new task scheduling algorithm to resolve this problem with applying the Min-Min or Max-Min algorithms to scheduling. To select the algorithm for first scheduling, we propose new Makespan. The most important of factor that can be improved by our algorithm is total completion time. The remainder of this paper is organized as follows. Related works are presented in section *I*. In section II, existing task scheduling algorithms is presented. In section III, a new scheduling algorithm is proposed and the proposed the algorithm is depicted through an illustrative example. In section IV, the performance and results analysis are presented and discussed. Finally, section V concludes the paper and presents future works.

RELATED WORK

For optimal use of available resources in the network and getting the less execution time, needs to provide a new scheduling algorithm is crucial. These algorithms assign tasks to the resources and provide the best conditions of quality of services.

F.Dong et al. have proposed an algorithm called QoS priority grouping scheduling [8]. This algorithm, considers deadline and acceptation rate of the tasks and the makespan of the wholes system as important factors for task scheduling.

Parsa et al. also have proposed an algorithm called RASA [9]. RASA begins with Min-Min algorithm if the number of available resources is odd and starts with Max-Min algorithm if the number of available resources is even. The remaining tasks are assigned to their appropriate resources by one of the two strategies, alternatively.

K. Etminani et al. have proposed a new algorithm which uses Max-min and Min-min algorithms [10]. The algorithm determines to select one of these two algorithms, dependent on the standard deviation of the expected completion times of the tasks on each of the resources. These algorithms have some advantages and disadvantages.

For example in RASA [9], if number of available resources be odd, the Min-Min strategy is applied to assign the first task, otherwise the Max-Min strategy is applied. The remaining tasks are assigned to their appropriate resources by one of the two strategies, alternatively. Now, if we have odd resources and the Max-Min strategy have better situation than Min-Min, we should select Min-Min instead of Max-Min.

EXISTING TASK SCHEDULING ALGORITHMS

Generally, the scheduling algorithms are divided into two basic categories: immediate mode scheduling and batch mode scheduling. In Immediate mode task is mapped onto a resource as soon as it arrives at the scheduler. For this mode we can mentioned MET and MCT algorithms. The MET (minimum execution time) heuristic assigns each task to the machine that performs that task’s computation in the least amount of execution time [17]. MET deployed in SmartNet [6] and have O(R) time complexity when we have R resources. The MCT (minimum completion time) heuristic assigns each task to the machine so that the task will have the earliest completion time [17]. Also MCT deployed in SmartNet [6] and like the MET have O(R) time complexity when we have R resources. In the batch mode, tasks are not mapped onto the resources as they arrive; instead they are collected into a set that is examined for mapping at prescheduled times called mapping events. The independent set of tasks which is considered for mapping at the mapping events is called a meta-task [14]. Min-Min, Max-Min and Sufferage Algorithm are examples of this type.

A. Min-Min Algorithm

Min-Min algorithm starts with a set of all unmapped tasks. The machine that has the minimum completion time for all jobs is selected. Then the job with the overall minimum completion time is selected and mapped to that resource. The ready time of the resource is updated. This process is repeated until all the unmapped tasks are assigned. Compared to MCT this algorithm considers all jobs at a time. So it produces a better makespan. Time complexity of Min-Min algorithm when we have R resources and T tasks is O(T^2R).

B. Max-Min Algorithm

Max-Min is very similar to Min-Min algorithm. Like the Min-Min, the machine that has the minimum completion time for all jobs is selected. Then unlike the Min-Min, the job with the overall maximum completion time is selected and mapped to that resource. The ready time of the resource is updated. This process is repeated until all the unmapped task are assigned. The idea of this algorithm is to reduce the wait time of the large jobs. This algorithm takes O (T^2R) time, when we have R resources and T tasks.

C. Improved Min-Min Algorithm

It can be seen that depending on the length of unassigned tasks in MT (meta tasks), One of these heuristics has better results than the other one .if there is only one long tasks and too many shorts tasks, Max-Min will execute long tasks to be executed long tasks first and allows shorts tasks to be executed concurrently with the long task, resulting better makespan and even better resources utilization rate and load balancing level, compared to Min-Min that executes all shorts tasks first and then executes the long tasks [1]. Our proposed algorithm is presented in figure 5(a) and 5(b).

THE PROPOSED ALGORITHM

The Min-Min algorithm seems worse in the cases when the number of short tasks is much more than the long ones. For example, if there is only one long task, the Max-Min algorithm executes many short tasks concurrently with the long task. In this case, the makespan of the system is most likely determined by the execution time of the long task. However, since the Min-Min algorithm attempts to assign the short tasks before the long one, the makespan increases compared with the Max-Min. On the other hand,mapping the longest task to the fastest resource provides a better opportunity for concurrent execution of the small tasks on different resources. In this certain situation, the Max-min provides a better mapping which supports load balancing across the grid resources more than the Min-Min [7]. Our proposed scheduling algorithm is presented in Fig.1. Firstly all the tasks should be sorted ascending. It means tasks with minimum completion time are in the front of queue and tasks with maximum completion time are in the rear of queue.

```
(1)  Sort all tasks in MT ascending // MT=meta data tasks
(2)  While there are tasks in MT
(3)  For all tasks $t_i$ in MT
(4)  For all machines $m_j$
(5)  $CT_{ij}$ =$ET_j$ + $r_j$ // $r_j$ =ready time
(6)  For all $t_i$ in MT
(7)  Find the minimum $CT_{ij}$ and resources $m_j$
(8)  If there is more than one resources that obtains it.
(9)  Select resource with least usage so for // for load balancing
(10) Calculate the ACT and SD for all tasks in MT
(11) If ACT >SD
(12)  Take tasks $t_f$ from the front of the queue
(13) For (k=0; k<=j; k++)
(14) $E_{xf}$ = $r_k$ + $ET_f$
(15) Find the minimum $E_{xf}$ and resource $m_j$
(16) Assign $t_f$ to resource $m_j$
(17) Else
(18) Take the $t_f$ from rear of the queue
(19) For (k=0; k<= j; k++)
(20) $E_{xf}$ = = $r_k$ + $ET_f$
(21) Find the minimum $E_{xf}$ and resource $m_j$
(22) Assign $t_f$ to resource $m_j$
(23) End if
(24) Delete assigned tasks from MT
(25) End while.
```

As shown in Fig.1, firstly it computes the amount of task completion time CT_{ij} for all tasks in MT on all resources from the following equation:

$$CT_{ij} = ET_{ij} + r_j \qquad \text{...... (1)}$$

CT_{ij} is completion time and ET_{ij} is expected execution time of task ith on resource jth and r_j is the ready time for resource jth (r_j is the ready time or availability time of resource j after completing previously assigned jobs). After that, the set of minimum expected completion time for each task in MT is found (resource discovery), then the task with the overall minimum expected completion time from MT is selected and assigned to the corresponding resource (resource selection)[1].

Algorithm like the Min-Min algorithm, computes minimum completion time of all tasks on available resources. After that, the resource according to the suitable situation should be chosen. For choosing a task for scheduling, firstly we determine average of completion time and standard deviation of existing tasks. According to [16] average of completion time (ACT) and standard deviation (SD) of tasks can be intended by using the following relations [1]:

$$AC\ T = \frac{\sum_{i=1}^{r} CTij}{r} \qquad \text{..... (2)}$$

$$SD = \sqrt{\frac{\sum_{i=1}^{r} (CTij - ACT)^2}{r}} \qquad \text{... (3)}$$

Figure 2. Mathematical relation of ACT & SD(Where r denotes number of resources).

After, the proposed algorithm compares values of ***ACT*** and ***SD***. By applying this heuristic, two cases might happen [1]:

- If ***ACT*** is less than ***SD***, it means the length of all tasks in *MT* is in a small range, so we will select from front of queue to assign the next task (line 13).
- Consequently, we will select from rear of queue to assign the next task (line 15).

1) Time Complexity of Proposed Algorithm

The order of this algorithm is depending on two for loop that described in line (3) and (4) and also it's should be operable on all tasks (line (2)). In lines 3-5, two nested for loops takes *O(T.R)* time: internal for loop runs *R* times (number of resources) and external for loop runs *T* times (number of tasks). This procedure is done for all tasks in MT and runs *R* times. Therefore, lines 2-17 take $O(T^2R)$ time. So, this algorithm, likes the Min-Min and the Max-Min algorithm takes $O(T^2R)$ time, when we have *R* resources and *T* tasks.

2) An Example

As a simple example, suppose there is a grid habitat with two resources. The completion time of the tasks are depicted in Table 1.

TABLE I. Completion time of the tasks on the resources

	Resources	
Tasks	R1	R_2
T_1	2	4
T_2	4	5
T_3	12	10
T_4	90	45

better completion time, is busy all the time but R2 is free. So here, proposed algorithm has better makespan and load balancing level than Min-Min algorithm.

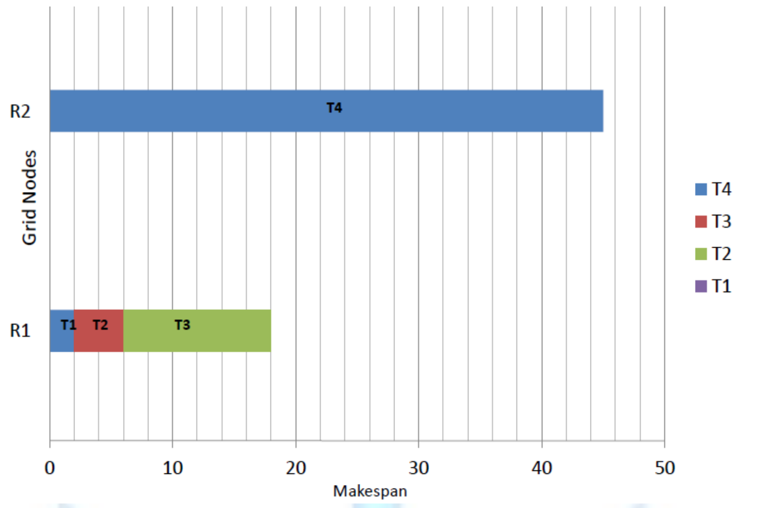
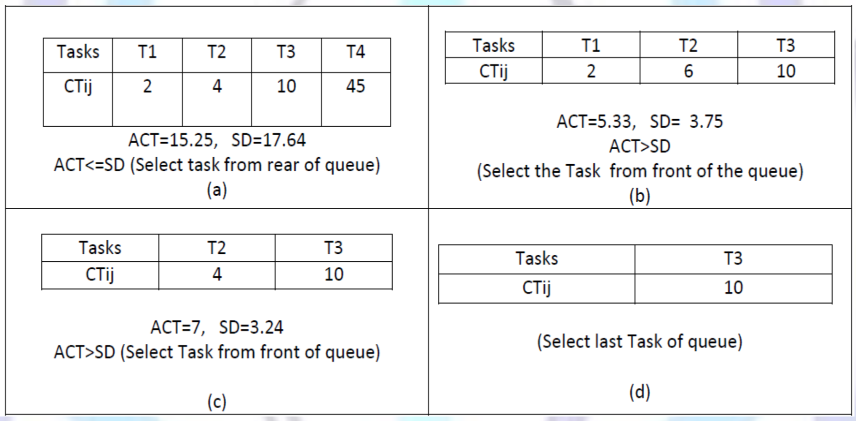


Figure. 3. Makespan of Min-Min algorithm and proposed algorithm

Also Fig.4 shows that how proposed algorithm selects tasks for scheduling, according to the values of completion time that described in Table I.



Tasks	T1	T2	T3	T4
CTij	2	4	10	45

ACT=15.25, SD=17.64
ACT<=SD (Select task from rear of queue)
(a)

Tasks	T1	T2	T3
CTij	2	6	10

ACT=5.33, SD= 3.75
ACT>SD
(Select the Task from front of the queue)
(b)

Tasks	T2	T3
CTij	4	10

ACT=7, SD=3.24
ACT>SD (Select Task from front of queue)
(c)

Tasks	T3
CTij	10

(Select last Task of queue)
(d)

Figure 4. Selection of tasks in proposed algorithm

In figure 4 (a), there exists one long task and three short tasks, the case where proposed algorithm unlike the Min-Min algorithm (that select T_1 for first step), select long task for scheduling (T_4). As it can be seen, the value of ***ACT*** is less than of ***SD***, so we should select task from the rear of queue.

PERFORMANCE AND RESULTS ANALYSIS

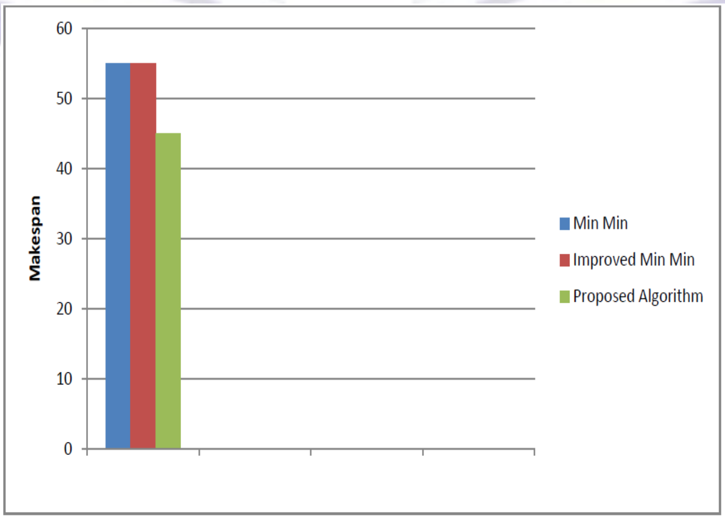


Figure 5(a). Comparison based on the makespan

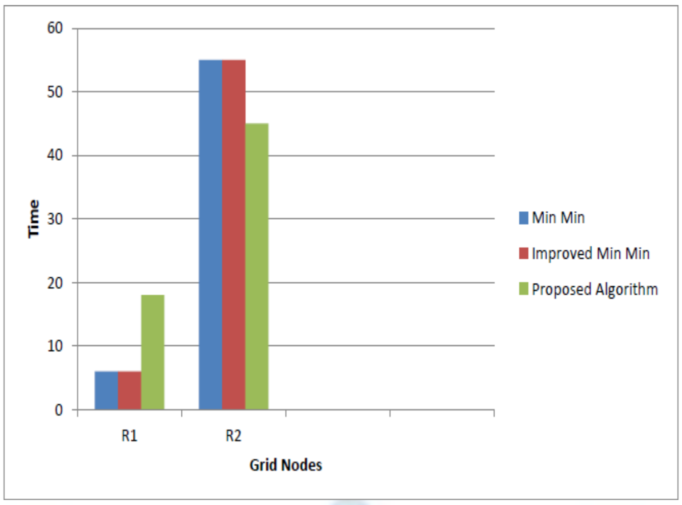


Figure 5(b). Comparison based on grid utilization

In Figure 5(a) we compared makespan of Min-Min, Improved Min-Min and proposed algorithm with tasks, when we have 2,4,6 ,12 and 18 resources. As we see, the proposed algorithm outperforms both Min-Min and improved algorithm and have minimum makespan.

In Figure 5(b) The proposed algorithm perform like the best algorithm in each supposition. It acts better than the both Min-Min and improved algorithm. Average resource utilization rate is one of the metrics that is used in [16] and the most efficient is achieved if average resource utilization equal.

To contrast and evaluate the proposed algorithm with other algorithms such as Max-Min and Min-Min, a simulation habitat known as *GridSim* toolkit [13] has been used.

CONCLUSION AND FUTURE WORK

To overcome the restriction of the Min-Min algorithm, in this paper an elevated task scheduling algorithm based on well-known task scheduling algorithm, Min-Min was presented. This algorithm proposed new situation for selection of the task for scheduling. The proposed Algorithm uses the advantages of Min-Min algorithm and covers their disadvantages. The experimental results acquired by applying our algorithm within the GridSim simulator, displays that the proposed algorithm is submerges better makespan than Min-Min and also helps load balancing. This study worried task execution time and load balancing. For future works, we can apply other issues like time limits on tasks and resources.

REFERENCES

[1] An Improved Min-Min Task Scheduling Algorithm in Grid Computing, Soheil Anousha1,* and Mahmoud Ahmadi2 Litzkow, M., Livny, M., Mutka, M.: Condor - A Hunter of Idle Workstations, In: Proceedings of the 8th International Conference of Distributed Computing Systems, pp. 104–111 (June 1988).

[2] Foster, I., Kesselman, C.: The Grid: Blueprint for a future computing Infrastructure. Morgan Kaufmann Publishers, USA (1999)

[3] Yagoubi, B., Slimani, Y.: Task Load Balancing Strategy for Grid Computing. Journal of Computer Science 3(3), 186–194 (2007).

[4] Maheswaran, M., Ali, S., Jay Siegel, H., Hensgen, D., Freund, R.F.: Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. Journal of Parallel and Distributed Computing 59, 107–131 (1999).

[5] Freund, R.F., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D., Keith, E., Kidd, T., Kussow, M., Lima, J.D., Mirabile, F., Moore, L., Rust, B., Siegel, H.J.: Scheduling Resource in Multi-User, Heterogeneous, Computing Environment with SmartNet. In: The Proceeding of the Seventh Heterogeneous Computing Workshop (1998).

[6] Braun, T.D., Jay Siegel, H., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B.: 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, 810–837 (2001)

[7] Dong, F., Luo, J., Gao, L., Ge, L.: A Grid Task Scheduling Algorithm Based on QoS and Cooperative Computing (GCC 2006). IEEE (2006).

[8] Parsa, S., Entezari-Maleki, R.: RASA: A New Grid Task Scheduling Algorithm. International Journal of Digital Content Technology and its Applications 3(4) (December 2009) .

[9] Etminani, K., Naghibzadeh, M.: A Min-min Max-min Selective Algorithm for Grid Task Scheduling. In: The Third IEEE/IFIP International Conference on Internet, Uzbekistan (2007).Afzal, A., Stephen McGough, A., Darlington, J.:

Capacity planning and scheduling in Grid computing environment. Journal of Future Generation Computer Systems 24, 404–414 (2008).

[10] Afzal, A., Stephen McGough, A., Darlington, J.: Capacity planning and scheduling in Grid computing environment. Journal of Future Generation Computer Systems 24, 404–414 (2008).

[11] Brucker, P.: Scheduling Algorithms, 5th edn. Springer Press (2007).

[12] Buyya, R., Murshed, M.: GridSim: A toolkit for the odeling and simulation of distributed resource management and scheduling for grid computing. Journal of Concurrency and Computation Practice and Experience, 1175–1220 (2002).

[13] Benjamin Khoo, B.T., Veeravalli, B., Hung, T., Simon See, C.W.: A multi-dimensional scheduling scheme in a Grid computing environment. Journal of Parallel and Distributed Computing 67, 659–673 (2007).

[14] Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., Tuecke, S.: A resource management architecture for metacomputing systems. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 1998. LNCS, vol. 1459, pp. 62–82 Springer, Heidelberg (1998).

[15] Cao, J., Spooner, D.P., Jarvis, S.A., Nudd, G.R.: Grid Load Balancing Using Intelligent Agents. Future Generation Computer Systems 21(1), 135–149 (2005).

[16] He, X., Sun, X.-H., Laszewski, G.V.: QoS Guided Min-min Heuristic for Grid Task Scheduling. Journal of Computer Science and Technology 18, 442–451 (2003).

[17] Foster, I: Globus Toolkit Version 4: Software for service-oriented system. In: Jin,H.,Reed, D., Jiang W.(eds) NPC 2005 . LNCS, vol.3779, pp.2-13. Springer, Heidelberg(2005).