Batch Mode Scheduling- Mid_Max Algorithm

Vijay Laxmi
Research Scholar (Ph.D), Department of Computer Science & Engg., Singhania University

Navdeep Kaur
Phd,A.P. & Head of Computer Science & Engg. Deptt, C.E.C. Landran (Mohali)

ABSTRACT
In Desktop grid computing environment, range of computing devices coexists starting from personal computers to supercomputers. These devices are inter-connected to provide a variety of computational capabilities in order to execute applications that have diverse requirements. An important decision for such computing infrastructure is how to optimally allocate computational and communication resources to these applications and to schedule their execution in order to maximize performance benefits.
In order to utilize the power of desktop grid completely, we need an efficient task scheduling algorithm to assign tasks to resources in a desktop grid. In this paper, we propose a Batch Mode Scheduling (Mid_Max algorithm) for the desktop grid environment. Compared to other methods, it performs well.

Keywords
Batch Mode Scheduling, Desktop Grid Computing, Mid_Max algorithm.

1. INTRODUCTION
When human culture advances, current problems in science and engineering become more complicated and need more computing power to tackle and analyze. A supercomputer is not the only choice for complex problems any more as a result of the speed-up of personal computers and networks. Desktop grid technology, which connects a number of personal computers with high speed networks, can achieve the same computing power as a supercomputer does, also with a lower cost. However, desktop grid is a heterogeneous system[1,2,3]. Scheduling independent tasks on it is more complicated. In order to utilize the power of desktop grid completely, we need an efficient task scheduling algorithm to assign tasks to resources in a desktop grid. In this paper, we propose a Mid_Max algorithm for the desktop grid environment.
The remainder of this paper is organized as follows. Existing Batch Mode Scheduling algorithms in Desktop Grid Computing is discussed in Section 2. In Section 3, The Proposed Batch Mode Algorithm is discussed. Section 4 describes the performance of various Batch mode Algorithms. A Conclusion is in Section 5.

2. EXISTING BATCH MODE ALGORITHMS
Min_Min, Max_Min, Sufferage proposed by Maheswaran [4,5,6] are three major heuristics. The performance matrix is given in Table 1.

2.1 Min_Min Algorithm: The Min_Min heuristic begins with all unmapped tasks. Then, the set of minimum completion times, for each task t is found. Next, the task with the overall minimum completion time is selected and assigned to the corresponding machine (hence the name Min_Min).
Last, the newly mapped task is removed from queue, and the process repeats until all tasks are mapped (i.e., U is empty) . Min_Min is based on the minimum completion time, as is MCT. However, Min_Min considers all unmapped tasks during each mapping decision and MCT only considers one task at a time. Min_Min maps the tasks in the order that changes the machine availability status by the least amount that any assignment could. Let ti be the first task mapped by Min_Min onto an empty system. The machine that finishes ti the earliest, say mj ,is also the machine that executes ti the fastest. For every task that Min_Min maps after ti , the Min_Min heuristic changes the availability status of mj by the least possible amount for every assignment. Therefore, the percentage of tasks assigned to their first choice (on the basis of execution time) is likely to be higher for Min_Min than for Max_Min (defined next). The expectation is that a smaller makespan can be obtained if more tasks are assigned to the machines that complete them the earliest and also execute them the fastest[5,6].

Table 1. Performance Matrices

Symbol	Definition
EET(t,r)	Estimated Execution Time: the amount of time the resources r will take to execute the task t, from the time the task starts to execute on the resource.
EAT(t,r)	Estimated Available Time: the time at which the resources r is available to execute task t.
FAT(t,r)	File Available Time: the earliest time by which all the files required by the task t will be available at the resource r.
ECT(t,r)	Estimated Completion Time: the estimated time by which task t will complete execution at resource r.
MCT(t)	Minimum Estimated Completion Time: minimum ECT for task t over all available resources.

2.2 Max_Min Algorithm: The Max_Min heuristic is very similar to Min_Min. The Max_Min heuristic also begins with all unmapped tasks. Then, the set of minimum completion times is found. Next, the task with the overall maximum completion time is selected and assigned to the corresponding machine (hence the name Max_Min). Last, the newly mapped task is removed from queue, and the process repeats until all tasks are mapped [6,7]. Intuitively, Max_Min attempts to minimize the penalties incurred from performing tasks with longer execution times. Assume, for example, that the metatask being mapped has many tasks with very short execution times and one task with a very long execution time. Mapping the task with the longer

execution time to its best machine first allows this task to be executed concurrently with the remaining tasks (with shorter execution times). For this case, this would be a better mapping than a Min_Min mapping, where all of the shorter tasks would execute first, and then the longer running task would execute while several machines sit idle. Thus, in cases similar to this example, the Max_Min heuristic may give a mapping with a more balanced load across machines and a better makespan.

3. PROPOSED BATCH MODE SCHEDULING ALGORITHM-MID_MAX

The Mid_Max heuristic begins with all unmapped tasks. Then the completion time for each task is found. The task with overall midst completion time is selected and assigned to fastest resources. The newly mapped task is removed from the queue and the process repeats until all tasks are mapped. Mid_Max is based on midst completion time as is MCT.

***ALGORITHM:* Mid_Max**

1) Identify the resources and their capabilities using GIS (Grid Information System). Identify the cost of all resources.(Cost expressed in terms of cost per task)
2) Create Users with their proper ID.
3) Create Gridlets or Tasks with different properties.
4) Repeat following steps for each user
a) Sort the resources by decreasing order of there processing speed. If two or more resource have the same speed then select according to First Come to First Serve (FCFS) order
b) Sort the tasks by decreasing order of there required execution time. If two or more task have the same execution time then select according to First Come to First Serve (FCFS) order.
c) Repeat for each unprocessed task depending on the sorted list.

- Apply Binary search algorithms for selecting mid task.
- Assign fastest processor to mid task..
- Calculate execution time and total cost for each task.

[Total cost=∑ (Required execution time for task/Execution speed for resource]
5.) Remove assigned task from unsigned job list.

Figure 1 : Mid_Max Algorithm

4. PERFORMANCE ANALYSIS

We evaluate the performance of our Mid_Max scheduling mechanism through simulation. We implemented our Mid_Max scheduling mechanism in the GridSim Toolkit. We developed our own JAVA program in GridSim toolkit to evaluate the performance of our Mid_Max scheduling algorithm.
We compare the performance of FCFS, Min_Min and Max_Min with our algorithm. Mid_Max gives optimize result in terms of both average cost and average execution time as compare to other algorithms.
The Table-2 shows the summary of the scheduling algorithms performance in terms of average execution time for our running example.

Table 2 : Average Execution time for running tasks

Tasks	FCFS	Min_Min	Max_Min	Mid_Max
7	71	62	87	60
10	98	89	100	80
15	118	101	122	88
20	137	121	130	110
25	154	134	159	122
30	172	159	180	150

The performance of FCFS,Min_Min,Max_Min and Mid_Max algorithms in terms of average execution time are studied in Figure-2 and shows Mid_Max behave same as Max_Min. But when we compare the performance in terms of average cost then Mid_Max gives better performance as compared to other algorithms shown in Figure 3.

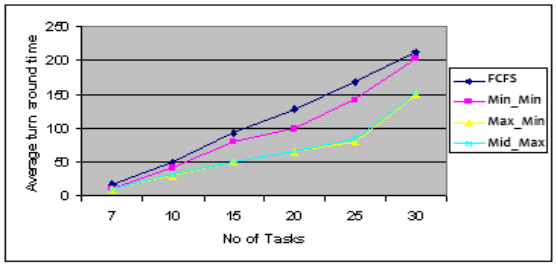

Figure 2: No. of Tasks Vs Average Execution time

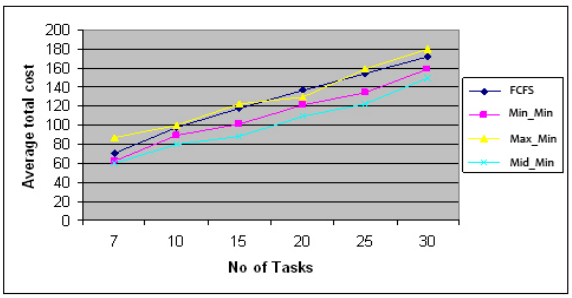

Figure 3: No. of Tasks Vs Average total cost

The Table-3 shows the average total cost for different tasks, for different algorithms like FCFS, Min_Min, Max_Min and Mid_Max.

Table 3: Average total cost for running task

No of Tasks	FCFS	MIN_MIN	MAX_MIN	Mid_Max
7	17	11	10	10
10	50	41	30	29
15	93	80	50	50
20	128	99	65	64
25	167	141	80	74
30	212	201	150	152

5. CONCLUSION

An advantage of FCFS is that it does not require any information about task arrival rates or machine execution rates. FCFS only performs well in the systems with limited task heterogeneity and under moderate system loads. As the application tasks become more heterogeneous and load increases, performance degrades rapidly. On the other hand Max_Min improves the response time but it increases the total cost. Min_Min improves the cost factor but decrease the response time. In order to avoid the limitation done by Max_Min & Min_Min, We propose an algorithm Mid_Max The results show that the proposed algorithm has a better efficiency in comparison with the results obtained from other known algorithms.

We will expand our work by adding replica and adaptive time out technique with our algorithm Mid_Max.

6.REFERENCES

[1] I. Foster, C. kesselman The Grid: Blueprint for a New Computing Infrastructure. 2nd Ed, Morgan Kaufmann, 2004.

[2] S. Ali, T.D. Braun, H.J. Siegel, A.A. Maciejewski, N.Beck, L.Boloni, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys, and B. Yao, "Characterizing Resource Allocation Heuristics for Heterogeneous Computing Systems," in Advances in Computers: Volume 63: Parallel, Distributed, and Pervasive Computing, vol. 63, pp. 93- 129, Elsevier, Apr. 2005.

[3] C. S. Yeo and R. Buyya, "A taxonomy of market-based resource management systems for utility-driven cluster computing," Software:Practice and Experience, vol. 36, issue 13, pp. 1381-1419, Nov. 2006.

[4] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems, The 8th Heterogeneous Computing Workshop , pp. 30-44, Apr. 2001.

[5] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, "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, issue 6, pp. 810-837, Jun. 2001.

[6] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing,"Software: Practice and Experience, vol. 32, issue 2, pp. 135-164, Feb.2002.

[7] J. Yu and R. Buyya, "A Taxonomy of Scientific workflow Systems for Grid Computing," Special Issue on Scientific Workflows, SIMMOD Record, vol. 34, no. 3, pp. 44-49, Sept. 2005.