RESEARCH ARTICLE OPEN ACCESS

Workflow Scheduling In Grid Environment

Neeraj Mangla[1] and Manpreet Singh[2]
[1]Department of Computer Science & Engineering, M. M. Engineering College, M. M. University, Ambala, Haryana [1]erneerajynr@gmail.com,
[2]Department of Computer Science & Engineering, M. M. University, Sadopur, Ambala, Haryana
[2]dr.manpreet.singh.in@gmail.com

ABSTRACT
Task scheduling in heterogeneous computing environment such as grid computing is a critical and challenging problem. Many parallel applications consist of multiple computational components. While the execution of some of these components or tasks depends on the completion of other tasks, others can be executed at the same time, which increases parallelism of the problem. The task scheduling problem is the problem of assigning the tasks in the system in a manner that will optimize the overall performance of the application, while assuring the correctness of the result. Scientific workflows, usually represented as Directed Acyclic Graphs (DAGs), are an important class of applications that lead to challenging problems in resource management on grid and utility computing systems. In this dissertation, a priority scheduling heuristic is developed which maintains a list of all tasks of a given DAG according to their priorities. It firstly prioritizes all tasks and then selects the best resource for the ready task with highest priority.

Keywords- Grid Computing, Work flow, Scheduling, DAG.

I. INTRODUCTION

Grid Computing can be defined as the seamless provision of access to possibly remote, heterogeneous, untrusting, dynamic computing resources [1][2]. Grid applications are usually divided into many interdependent subtasks in real applications. Every single subtask is processed and the subtasks should process concurrently in order to reduce the task running time, which is one of the most important problems in parallel computing. Workflow applications incorporate multiple dependent modules to be executed in a predefined order and may entail the transfer and storage of a huge amount of data. A very important issue in executing a scientific workflow in computational grids is how to map and schedule workflow tasks onto multiple distributed resources and handle task dependencies in a timely manner to deliver users' expected performance [5] [6]. Directed acyclic graph (DAG) is usually used to illustrate the data dependency among subtasks in workflows [3]. In DAG, workflow structure can be categorized into sequence, parallelism, and choice. Sequence is defined as an ordered series of tasks, with one task starting after a previous task has completed. Parallelism represents tasks which are performed concurrently, rather than serially. In choice control pattern, a task is selected to execute at run-time when its associated conditions are true.

The workflow execution time consists mainly of two parts: the task execution time and data transfer time. The task execution time is not simply the sum of times spent carrying out all tasks because some of them are executed concurrently. For a workflow that can be modeled as a DAG, critical tasks are those that must be started on their earliest start times in order to achieve the best performance of the workflow execution [7]. The sum of the execution times of critical tasks is the time spent for workflow task execution. In a workflow, if two tasks having data dependencies, such as intermediate files are allocated on different resources then intermediate files need to be transferred between the two resources. In a grid environment with slow network, the data transfer time may become a significant part of the total workflow execution time. But not all data transfers impact the workflow performance; only those that delay the launching of critical tasks, directly or indirectly, do so.

A workflow scheduler should have two capabilities: first, resource allocation, which distributes tasks onto multiple resources and second, task execution and coordination, which submits tasks to the resource's local schedulers in the right order, and handles task dependencies [4]. In a DAG workflow, the task dependencies determine the order of task submission and file transfer, which is the topological order of the workflow DAG. In this order, the earliest start-time of each task can be calculated easily, as long as we know when the workflow itself should be started. An allocated resource for a task should be available before its

earliest start-time so that no delay is incurred because of the unavailability of resources [8].
In this paper, the grid workflow scheduling problem is formulated and a priority based solution is discussed. A separate module is developed to generate the DAG topology of a workflow. Given the workflow structure and the number of processors with randomly generated processing power, the mapping scheme using both priority based and round-robin strategy is established separately.

II. RELATED WORK

Finding a single best solution for mapping workflows onto Grid resources for all workflow applications is difficult since applications and Grid environments can have different characteristics [9]. Many researchers have studied scheduling strategies for mapping application workflows onto the grid. In [10], authors developed a framework to schedule a DAG in a Grid environment that makes use of advance reservation of resources and also considers the availability knowledge about task execution time, transfer rates, and available processors to generate a schedule. Their simulation results show advantages of unified scheduling of tasks rather than scheduling each task separately. A static scheduling is applied [11] to ensure that the key computational steps are executed on the right resources and large scale data movement is minimized. Authors use performance estimators to schedule workflow applications. In [12], authors mapped the entire workflow to resources at once or portions of it. This mapping can be done before or during the workflow execution. Their algorithm prefers to schedule computation where data already exist. Additionally, users are able to specify their own scheduling algorithm or to choose between a random and a round robin schedule technique. A new grid scheduling algorithm that minimizes the cost of the execution of workflows while ensuring that their associated QoS constraints are satisfied is proposed [13]. The algorithm views a grid environment as a queuing system and schedules tasks within this system. This algorithm is system oriented and considers the execution cost. Hence, it is suitable for economic grids. Since the algorithm is non-linear, as the size of the problem gets large the time it takes to obtain a suitable scheduling becomes very long and unacceptable.

III. SYSTEM FRAMEWORK & IMPLEMENTATION

3.1 Directed Acyclic Graph (DAG)

Task scheduling problem in computational grid can be represented as DAG, a directed graph with no directed cycles. In a DAG, a node is an individual task and an edge represents the inter-job dependency. A child task cannot be executed before all its parent task finish successfully and its required data inputs in place. Nodes and edges are weighed for computation cost and communication cost respectively.

```
1.  Decide the levels of the graph.
2.  For each level do
3.      Begin
4.          Allocate tasks to the level.
5.      End
6.  For each intermediate level do
7.      Begin
8.          For each task do
9.              Begin
10.                 Make at least one dependency of the task($t_i$) with the tasks of
                    its previous and next level each.
11.                 Assign weight to the dependencies.
12.             End
13. End
```

In above algorithm, there are certain numbers of levels. On each level, some number of tasks is assigned not necessarily to be different. There is at least one dependency of each task with the tasks of its previous and next level each, such that each and every task has its predecessor and successor.

3.2 Round Robin Scheduling Algorithm

It is a static scheduling strategy which maps resources to each individual task before workflow execution. In this approach, resources are assigned in round robin manner. In the round- robin scheduling algorithm, initially processors are allocated to the tasks of workflow application. After allocation of processors, EST(Earliest Start Time) and EFT (Earliest Finish time) for each task is calculated. For EST, communication cost between task and its parent is considered if both are on different processors. There may be more than one parent of a single task. Corresponding to each parent, EST is to be calculated and then maximum value among those will be EST for the task. EFT can be calculated as sum of EST and computation time of task on the allocated processor. Finally the EFT for the last task also known as exit task is makespan for the workflow application. The algorithm is as follows:

```
1.  For each task do
2.      Begin
3.          Allocate processor to the task($t_i$) in the round robin manner.
4.      End
5.  For each task do
6.      Begin
7.          Calculate EST for the task($t_i$) corresponding to each parent and then choose the maximum of those which will be EST($t_i$) for the task($t_i$).
8.          Calculate EFT($t_i$) = EST($t_i$) + Computation Time of task($t_i$) on the allocated Processor.
9.      End
10. Makespan = EFT($t_{exit}$).
```

3.3 Priority Based Scheduling Algorithm

It is a list scheduling strategy, where a resource mapping is done only when a task is ready to execute without requiring any prior application and environment knowledge. Here, tasks are prioritized and executed in the order of their priorities.

In this algorithm, initially priority is assigned to the tasks at each level. Priority can be assigned in the decreasing order of their linkcost which is the sum of the uplink and downlink cost where uplinkcost is the maximum of the communication cost among its successor and downlink cost is the maximum of the communication cost among its predecessors. Higher prioritised task is executed first. Now, EST for the task with respect to each processor corresponding to all parents is calculated and among those maximum value of EST is chosen for that processor. Then EFT is calculated corresponding to each EST and minimum EFT is selected and hence the processor with minimum EFT is allocated to the task. Finally, the EFT for the last task also known as exit task is makespan for the workflow application. Algorithm is as follows:

```
1.  For each level do
2.      Begin
3.          For each task do
4.              Begin
5.                  Calculate Downlinkcost($t_i$) of task($t_i$).
6.                  Calculate Uplinkcost($t_i$) of task($t_i$).
7.                  Calculate Linkcost($t_i$)=Downlinkcost($t_i$)+Uplinkcost($t_i$)+max{linkcost of its Predecessor}.
8.              End
9.          Sort the tasks in decreasing order of their linked cost and assign priority to them.
10.     End
11. For each level do
12.     Begin
13.         For each task according to their priority do
14.             Begin
15.                 For each processor do
16.                     Begin
17.                         Calculate EST for task($t_i$) corresponding to each parent for processor $p_k$ and then choose maximum of those which will be EST($t_{i,k}$).
18.                         Calculate EFT($t_{i,k}$) = EST($t_{i,k}$) + Computation Time of task($t_i$) on processor k.
19.                     End
20.                 Choose the processor with minimum EFT of task($t_i$) and allocate it to task($t_i$).
21.             End
22.     End
23. Makespan = EFT($t_{exit}$).
```

IV. RESULTS & DISCUSSION

This section contains a description of experiments carried out during simulation in Java along with relevant parameters as shown in Table 1.

Table 1: Simulation Parameters

Parameter	Value
Simulation Runs	10
Number of Levels	7-20
Number of Tasks	25-100
Number of Processors	5-30

Table 2: Makespan of Workflow Applications with variation in No. of Tasks

Number of Tasks	Number of Levels	Makespan (Round Robin)	Makespan(Priority)
25	7	520	332
50	10	835	592
75	12	1024	757
100	20	1810	1087

Table 3: Makespan of Workflow Applications with variation in No. of Processors

Number of Processors	Makespan(Round Robin)	Makespan(Priority)
5	1638	1073
10	1793	1030
15	1864	1001
20	1773	1020
25	1771	1048
30	2023	1009

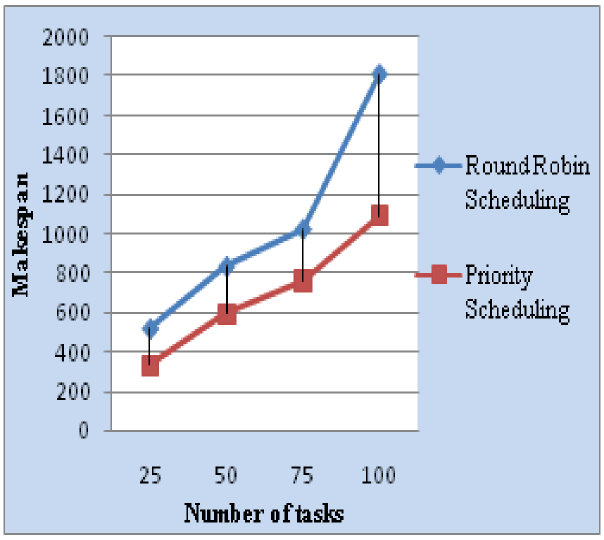

Figure 1: Comparison of Makespan for Different Number of Tasks

The performance of these algorithms is tested under two scenarios. In scenario 1 with a simulation run of 10 times, DAG is randomly generated with 4 different numbers of tasks, i.e. 25, 50, 75 and 100 and the execution environment comprises of 5 processors as shown in Figure 1 and Table 2. In scenario 2 with a simulation run of 10 times, DAG is randomly generated comprising of 100 tasks and executed in a system with processing capability of 5, 10, 15, 20 and 25 processors as shown in Figure 2 and Table 3. Simulation results confirmed that priority based workflow scheduling has significant performance improvement over round-robin approach.

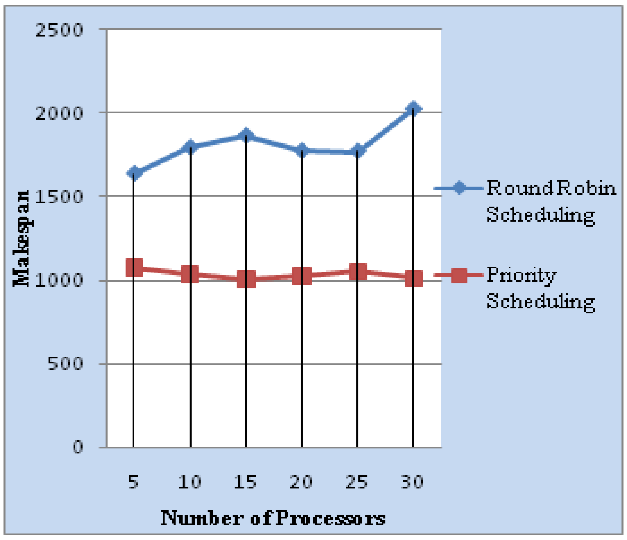

Figure 2: Comparison of Makespan for Different Number of Processors

V. CONCLUSION

Tasks with DAG dependencies are frequent in case of Grid applications and they require advanced scheduling procedures. In this research work, the grid workflow scheduling problem is formulated and a priority based solution is discussed. Simulation results show that priority based workflow scheduling outperforms the traditional round-robin policy commonly used in real systems.
Future work will include, among other things: the analysis of a wider set of scheduling algorithms currently used in Grid systems and the establishment of relevant performance measures.

REFERENCES

[1] I. Foster, C. Kesselman, and S. Tuecke, “The anatomy of the grid: enabling scalable virtual organizations,” International Journal of High Performance Computing Applications, 15 (3), 2001, 200-222.

[2] R. Buyya and S. Venugopal, “A Gentle Introduction to Grid Computing and Technologies” CSI Communication, 2005, 9-19.

[3] Ehsan Ullan Munir and Jian-Zhong Li. Performance Analysis of Task Scheduling Heuristics in grid. Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 2007, 19-22.

[4] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, S. Korranda, “Mapping abstract complex workflows onto Grid environments”, Journal of Grid Computing,1 (1), 2003, 25-39.

[5] E. Ilavarasan and R. Manoharan, “High Performance and Energy Efficient Task Scheduling Algorithm for Heterogeneous Mobile computing System”, International Journal of Computing Science and Engineering, 2 (2), 2010.

[6] J. Yu, R. Buyya and K. Ramamohanarao ,“Workflow Scheduling Algorithms for Grid Computing”, Metaheuristic for scheduling in Distributed Computing Environments, Springer Verlag, 146 (1), 2008, 173-214.

[7] L. Canon, E. Jeannot, R. Sakellariou and W. Zheng, “Comparative Evaluation of the Robustness of DAG Scheduling Heuristics,” Grid Computing, 2008, 73-84.

[8] Marek Mika Wojciech Piątek, Grzegorz Waligóra, Jan Węglarz, “Computational Experiments for Scheduling Workflow Applications in Grid Environment”, Computational Methods in Science and Technology, 17 (1), 2011, 53-62.

[9] J. Yu and R. Buyya, “A Taxonomy of Workflow Management Systems for Grid Computing”, Journal of Grid Computing, 3 (3-4), 2005, 171-200.

[10] Ammar H. Alhusaini, Viktor K. Prasanna, C.S. Raghavendra. "A Unified Resource Scheduling Framework for Heterogeneous Computing Environments," Heterogeneous Computing Workshop, 1999, 156-165.

[11] A. Mandal, K. Kennedy, C. Koelbel, G. Marin, J. Crummey, B. Liu, L. Johnsson, “Scheduling Strategies for Mapping Application Workflows onto the Grid”, The 14th IEEE International Symposium on High-Performance Distributed Computing (HPDC-14), 2005, 125-134.

[12] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman, “Workflow Management in GriPhyn”, Grid Resource Management, Kluwer Academic Publishers, US, 2004.

[13] A. Afzal, A. Stephen McGough and J. Darlington, “Capacity planning and scheduling in Grid computing environment”. Journal of Future Generation Computer Systems, 24 (2), 2008, 404-414.