International Journal of Scientific & Engineering Research, Volume 3, Issue 5, May-2012
ISSN 2229-5518

Expeditious Scheduling for Precedence Constraint Tasks in Grid

Joshua Samuel Raj, Daphne.S and Dr. V. Vasudevan

Abstract— The tremendous potential of grid computing is efficient scheduling, to exploit the computationally intense problems. In the commonly used decentralized models, large scale scheduling implies time sequence constraints, which makes the models intractable. To resolve this constraint, disintegration and cyclic scheduling are often applied to such scheduling which is time consuming and introducing more complexities. The significance of this paper is marked by speed and efficiency that the task-resource mapping in such a non-deterministic computing environment leads to concerns over scheduling problem to minimize the expected makespan and delay in allocation of tasks thereby reduce the turnaround time for precedence-constraint tasks imposed by application tasks to identify suitable resources. Our rigorous performance evaluation shows that our variant Expeditious Matching algorithm generates schedules with smaller makespan and higher robustness compared with other existing approach gratuitous to the new enhancement.

Index Terms— Grid computing, Job scheduling, Resource Matching, Makespan.

1 INTRODUCTION

Grid computing is a geographically distributed computing that links virtual supercomputers of vast amount of computing capacity through the Internet to solve complex problems from e-Science in less time than known before. In the past few years we have experienced how Grid computing has achieved a breakthrough in physics, meteorology, medicine and other computing fields. Grid computing is a technology that enables resource virtualization, on-demand provisioning and large scale resource sharing. Examples of such large-scale applications are known from optimization, Collaborative/e-Science Computing, Data-Intensive Computing etc.

The purpose of job scheduling in this environment is to balance the entire system load while completing all the jobs at hand as soon as possible according to the environment status [1-3]. In general, the objective of task scheduling is to minimize the completion time of a parallel application by properly mapping the tasks to the processors. This paper describes about application models that are dependent and precedence-constrained. The problem of mapping (including matching and scheduling) tasks and communications is a very important issue since an appropriate mapping can truly exploit the parallelism of the system thus achieving large speedup and high efficiency [4]. It deals with assigning (matching) each task to a machine and ordering (scheduling) the execution of the tasks on each machine in order to minimize some cost function. The most common cost function is the total schedule length, or makespan.

The rest of the paper is structured as follows: related works in section 2 which describes few decentralized based scheduling models. And in section 3 detailed descriptions about the proposed algorithm is given. Then in section 4 the experimental setup is given. And section 5 ends the paper with the conclusion and future work.

2 RELATED WORKS

In the classical approach, which is also called list scheduling [5, 6], the basic idea is to make an ordered list of nodes by assigning them some priorities, and then repeatedly execute the following two steps until a valid schedule is obtained. First select from the list the node with the highest priority for scheduling. Second select a processor to accommodate this node. The priorities are determined statically before the scheduling process begins. In Dynamic Critical Path (DCP) algorithm [7] extends the list scheduling to avoid scheduling less important nodes before the more important ones, node priorities can be determined dynamically during the scheduling process. The priorities of nodes are re-computed after a node has been scheduled in order to capture the changes in the relative importance of nodes. However this can increase the complexity of the algorithm. The HEFT algorithm [8, 9] is an application scheduling algorithm for a bounded number of heterogeneous processors, which has two major phases: a task prioritizing phase for computing the priorities of all tasks and a processor selection phase for selecting the tasks in the order of their priorities and scheduling each selected task on its best processor, which minimizes the task's finish time.

However, the problem is that if the queue is fully

- *Daphne. S is with the Computer Science department, Karunya University, India. E-mail: daphnesam7@gmail.com*
- *R. Joshua Samuel Raj is with the Computer Science department, Karunya University, India. E-mail: joshuasamuelraj@gmail.com*
- *Dr.V. Vasudevan is the Director, Software Technologies Lab, TIFAC Core in Network Engineering, Srivilliputhur, India*

loaded and workload is heavy, tasks might have to wait in the queue for a very long time. In the extreme case, starvation might occur. The Fastest Processor to Largest Task First (FPLTF) [10] algorithm schedules tasks concordant to the workload in the grid system. The algorithm needs the selective information of CPU speed and task workload. FPLTF works in two steps: one, the task scheduler sort's tasks and cut down CPU searching time. Two, the scheduler assigns the largest task in the queue to the fastest feasible resource node in the grid. Dynamic Fastest Processor to Largest Task First (DFPLTF) [10, 11] is extended from FPLTF. WQR (Work Queue with Replication) [12] is extended from the Work Queue (WQ) algorithm [13].

WQR has an attribute such that a faster processor will be assigning more tasks than a slower processor. In the min-min algorithm [14], the minimum completion time for each task is computed based on all the machines characteristics. The task which corresponds to overall minimum completion time is selected and assigned to the respective machine. The newly mapped task is discarded, and the process repeats until all tasks are allocated with the resources. A schedule is considered efficient if the schedule length is short and the number of processors used is reasonable.

In real world problems with precedence constraints, for interpreting communication cost and processing time as random variables, stochastic grid parallel applications are considered by submitting users and generally independent of each other, which request systems services for their execution. The Stochastic Heterogeneous Earliest Finish Time (SHEFT) [15] aims to schedule the tasks by assigning tasks to the machine that minimizes makespan. During scheduling, the unscheduled task in the task sequence is selected and scheduled on a machine that can complete its execution with minimize approximate earliest finish time.

3 Resource Expeditious Matching

Generally, the scheduler assigns tasks to an appropriate resource node for execution, and the resource nodes with better performance would be assigned first. When task loading is heavy and all resource nodes with better performance are assigned, other tasks have to be assigned to the resource nodes with inferior performance. Therefore, if a task is assigned to a resource node without considering the performance factor, the overall execution time will increase [10, 16]. To solve this problem, a scheduling algorithm that searches for the proper resource for task execution based on the processing speed of resources and computation time of tasks is proposed.

In this section we present our resources Expeditious Matching (EM) method. Our proposal has two phases they are initialization and expeditious matching phase. Here we consider the bounded number of tasks which has specific precedence constraint for their execution.

Before presenting the objective function, it's essential to define the expected finish time of task v_i as F_i, CPU utilization U_i and average execution cost as c_i of task v_i on resources r_j.

$$\text{Expected finish time}, F_i = {w(v_i)}/{p_i} \quad (1)$$

where $w(v_i)$ is the workload of the task v_i and p_i is the expected processing speed for the task v_i.

$$\text{Idle time}, F_i = {no_{load(r_j)}}/{load(r_j)} \quad (2)$$

where $no\_load(r_j)$ is the average period of tasks with no load on r_j and $load(r_j)$ is the average period of tasks with load on r_j.

$$\text{CPU utilization}, U_i = total\ execution\ time - F_i \quad (3)$$

$$\text{Average execution cost}, c_i = {\sum F_i * cs_j}/{v} \quad (4)$$

where the computation cost per second on resources r_j is cs_j.

In the initialization phase, the characteristic of the resources like processing speed and computation time are analyzed based on the information stored in the Grid Information Service (GIS). It's a repository that maintains and updates the details about the resources in the grid environment. The table 1 shows an example of the Estimated Resources Specification (ERS) table for resources and their corresponding processing speed, P_j and the average Resource computation Time (RCT) of resource j. The Estimated Resources Specification (ERS) table is updated in regular intervals of time to make sure that the values in the table are not outdated.

In the Expeditious Matching phase, the tasks are mapped to the available resources which are most appropriate for that task. The Expeditious Matching algorithm sorts the tasks based on the precedence constraints assigned.

TABLE 1
Estimated Resources Specification (ERS)

Resource name	Processing speed	RCT (s)
Cluster4	800 MIPS at 850 MHz	250
Cluster5	7500 MIPS at 1.6 GHz	165
Cluster2	30,000MIPS at 2.5 GHz	150
Cluster3	20,054 MIPS at 2.4 GHz	140
Cluster1	147,600 MIPS at 3.3 GHz	120

Initialization phase: Performed to analyze the computa-

tional time of tasks, only when this scheduling algorithm is first executed.

1 For each resource in the grid
2 Compute the average Resource Computation Time (RCT) using (Eq. 5) based on the recent past history of the resource like processing speed, workload etc.
3 Update the repository (ERS) with the computed values.
4 Sort the resources in the decreasing order based on the RCT
5 End for

Expeditious Matching phase

1 Sort the tasks based on the precedence constraint assigned
2 Do until there is any tasks in the queue
3 For each task v_i
4 While v_i is not assigned
5 If P_j has approximately equal processing speed required and available space for v_i
6 If v_i has expected finish time, $\boldsymbol{F_i}$ approximately equal with the Resource Computation Time (RCT)
7 Perform the task and resource matching
8 End if
9 End if
10 If v_i is not assigned to any of the resources
11 Perform the task and resource matching to the resource which has the least RCT value
12 End if
13 End while
14 For regular intervals of time, calculate the Resource Computation Time (RCT) based on the recent past history of the resource. And update the repository (ERS) with the computed values.
15 Sort the resources in the decreasing order based on the RCT
16 End for

Fig. 1. Pseudo-code of Expeditious Matching (EM) algorithm

Consider a task's resource requirement is specified as 1 GHz. Directly assign this task to a node with 1 GHz processing speed is not good as the Resource Computation Time of that resource varies. Therefore, we have to estimate the actual performance of an assignment using RCT (Eq. 5). For a given task v_i, define $\boldsymbol{W_i}$ as the estimated workload, $\boldsymbol{p_i}$ as the processing speed requirement, and U_i and P_j as the CPU utilization and processing speed of resources r_j, respectively.

$$\mathbf{RCT}(\boldsymbol{j},\boldsymbol{i}) = \left(\boldsymbol{W_i}/\boldsymbol{P_j}\right) \Big/ (\mathbf{1}-\mathbf{U_j}) \quad (5)$$

Given that $w(v_i)$ = 20,054 million instructions and P_i = 2.4 GHz then RCT is 140s. Table 1 lists the RCT value for each cluster of resources.

The resources are mapped if and only if a resource has available space, approximately equal processing speed and approximately equal Resource computation Time which is expected for that task to execute.

Given a task with $\boldsymbol{w}(\boldsymbol{v_i})$ = 20,540 million instructions and $\boldsymbol{p_i}$ = 2.4 GHz then expected finish time, $\boldsymbol{F_i}$ as 100s. From the values in table 1, we infer that the requirement of the task do not match with any of the resource list. So if this kind of exceptional condition arises then the matching is done with the resource which has the least RCT value.

The pseudo code of a newly proposed scheduling algorithm is depicted in figure 1. The data in the Estimated Resources Specification (ERS) table is used to match the incoming tasks specification and the resource specification. And for regular intervals of time, calculate the average Resource computation Time (RCT) based on the recent past history of the resource. There by update the repository (ERS) with the computed values and sort the resources in the decreasing order based on the RCT.

4 Performance Evaluation

In this section, we compare the performance of our Expeditious Matching (EM) algorithm with the well-known scheduling SHEFT [15] algorithm in Grid systems.

We implemented our experiments by Grid Simulator [17], a java-based discrete event grid simulation tool kit. In the simulation, there are two phases.

The performances metric are makespan and scheduling time ratio. Both the metric are essential and it's examined in this paper since smaller the schedule more efficient is the Grid system. The scheduling time ratio is computed by dividing the parallel execution time (i.e., the makespan of the output schedule) by the sequential execution time (i.e., cumulative execution time) as shown in Eq. 6.

$$\text{Scheduling time ratio} = \frac{\text{makespan}}{\sum \text{actual execution time } (v_i)} \quad (6)$$

4.1 Experimental Results

In our simulation experiments, the number of tasks ranges between 500 to 4000 and 10 resource nodes are considered. Every set of tasks for the above parameters are generated based on the bounded number of tasks with the precedence constraint. The processing speed of resource node was assigned between the range 4000 to

150,000 MIPS. Other parameter of the model is the failure rates of processors and links it's assumed to be uniformly distributed between 1 * 10^{-3} and 1 * 10^{-4} failures/hour. In addition, the transmission rates of links are assumed to be uniformly distributed between 10 and 100 Mbits/s.

We infer from figure 2 that the EM outperforms SHEFT [15] in terms of the average makespan by 35% for the 4000 number of tasks. Thus the proposed Expeditious Matching algorithm shows efficiency when compared with SHEFT [15].

Our EM algorithm is the fastest than the SHEFT [15] algorithm and HEFT [8] algorithm is the slowest one among the three, as shown in table 2. On average, the EM algorithm is faster than the SHEFT algorithm by 55 percent and the HEFT algorithm by 75 percent.

One of the reasons for efficient makespan is that, to avoid recalculating it for similar kind of recourses, we update the values and store them for further iterations. Then in task assignment, processing speed can be obtained by a simple table look-up instead of recalculating regarding all tasks.

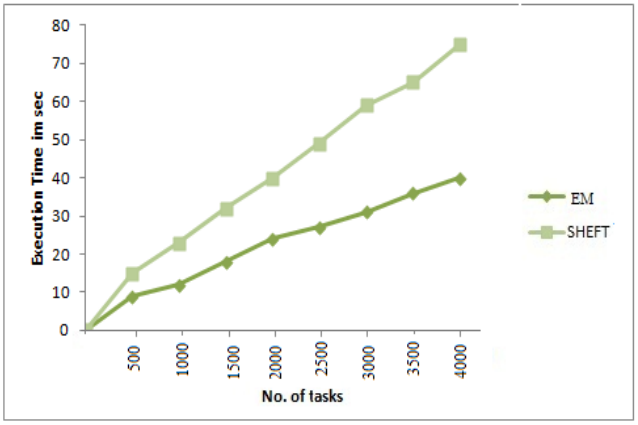

Fig. 2. Performance of execution time ratio for 4000 scheduled jobs

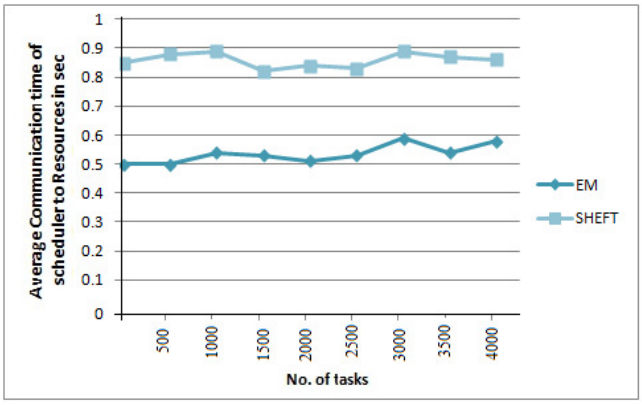

Fig. 3. Average Communication time of scheduler and Resources for 4000 scheduled jobs

TABLE 2

Comparison of Scheduling Techniques

	Makespan	Communication cost	Resource utilization
EM	Low	Low	Good
SHEFT [15]	Medium	Medium	Better
HEFT [8]	High	High	Better

From figure 3, we infer that the scheduling average communication time of scheduler and resources of the jobs in our algorithm have a minimal time when compared with the SHEFT algorithms courtesy to the fact that the recalculation of tasks is reduced drastically.

5 Conclusion and Future Work

In this paper, the major concern in the computing environment over scheduling is to minimize the expected makespan and delays in allocation of tasks thereby reduce the turnaround time. Our algorithm schedules the tasks based on the best match of resource and tasks using the simple ERS table lookup instead of recalculating for all tasks with the precedence constraints. Thus the application tasks identify suitable resources. The updating of the ERS table is expeditious based on recent history of the computation. Also our rigorous performance evaluation shows that our variant Expeditious Matching generates schedules with smaller makespan and higher robustness coupled with smaller scheduling time ratio compared with other existing scheduling approaches. In the near future we plan to combine the intelligence of ant colony for scalability in the existing algorithm. The procedure can also be suitably be modified and applied to any kind of Grid scheduling with different problem environment to optimize any number of objectives concurrently.

References

[1] J Katia Leal., Eduardo Huedo., Ignacio M. Llorente.: "A decentralized model for scheduling independent tasks in Federated Grids". Future Generation Computer Systems 25 (8) (2009) 840–852.

[2] D. A. Reed.: "Grids, the TeraGrid, and Beyond". IEEE Computer 36 (1) (2003) 62–68.

[3] J. Nabrzyski., J. M. Schopf., J. Weglarz.: "Grid Resource Management: State of the Art and Future Trends". Kluwer Academic Publishers, 2003.

[4] J. Breckling., T. D. Braun., H. J. Siegel., N. Beck., L. Boloni., M.Maheswaran., A. I. Reuther., J. P. Robertson., M. D. Theys., B. Yao.:"A taxonomy for describing matching and scheduling heuristics for mixed-machine heterogeneous computing systems. In Reliable Distributed Systems", 1998. Proceedings. Seventeenth IEEE Symposium on, pages 330–335, West Lafayette, IN, 1998.

[5] G. Q. Liu., K. L. Poh., M. Xie.: "Iterative list scheduling for heterogeneous computing". J. Parallel Distrib. Comput., 65(5):654–665, 2005.
[6] H. Topcuoglu., S. Hariri., M. Wu.: "Performance-Effective and Low- Complexity Task Scheduling for Heterogeneous Computing". IEEE Trans. Parallel and Distributed Systems, vol. 13, no. 3, pp. 260–274, March 2002
[7] Y. K. Kwok., I. Ahmad.: "Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors". IEEE Trans. Parallel Distrib. Syst., 7(5):506–521, 1996.
[8] G. C. Sih., E.A.Lee.: "A compile-time scheduling heuristic for interconnection constrained heterogeneous processor architectures". IEEE Trans. Parallel Distrib. Syst., 4(2):175–187, 1993.
[9] Radulescu., A. J. C. Van Gemund.: "Fast and effective task scheduling in heterogeneous systems". In HCW '00: Proceedings of the 9th Heterogeneous Computing Workshop, page 229, Washington, DC, USA, 2000. IEEE Computer Society.
[10] D. Saha., D. Menasce., S. Porto, et al., "Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures", Journal of Parallel and Distributed Computing 28 (1) (1995) 1–18.
[11] D. Silva., W. Cirne., F. Brasileiro.: "Trading cycle for information: using replication to scheduling bag of tasks application on computational grids", in: Proceeding in Ruro-Par, August 2003.
[12] S. Wang., I. Hsu., Z. Huang.: "Dynamic scheduling method for computational grid environments", in: Proceedings of the International Conference on Parallel and Distributed Systems, July 2005, pp. 22–28.
[13] Matei Ripeanu.: "Peer-to-peer architecture case study: Gnutella network", in: Proceedings First International Conference on Peer-to-Peer Computing, August 2001, pp. 99–100.
[14] M. Maheswaran., S. Ali., H. Siegel., D. Hensgen,, Richard F. Freund.: "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems", Journal of Parallel and Distributed Computing 59 (1999) 107–131.
[15] X. Tang., K. Li., G. Liao., R. Li.: "A stochastic scheduling algorithm for precedence constrained tasks on Grid". J.future generation computer sys. 2011.04.007.
[16] Ruay-Shiung Chang., Chun-Fu Lin., Jen-Jom Chen.: "Selecting the most fitting resource for task execution". Future Generation Computer Systems., 27 (2011) 227–231
[17] Anthony Sulistio., Uros Cibej., Srikumar Venugopal., Borut Robic., Rajkumar Buyya.: "A Toolkit for Modelling and Simulating Data Grids: An Extension to GridSim, Concurrency and Computation: Practice and Experience (CCPE)", Wiley Press, New York, USA, Sep.2008.

Name:
R. Joshua Samuel Raj
Affiliation:
Assistant Professor / CSE
Karunya University

Brief Biographical History:
2005 -Graduated in 2005 from the Computer Science and Engineering Department from PETEC under Anna University
2007 -Received M.E Degree in Computer Science and Engineering from Jaya College of Engineering under Anna University
2009 Working towards the Ph.D degree in the area of Grid scheduling under Kalasalingam University

Main Works:
Grid computing, Mobile Adhoc Networking, Multicasting and so forth.

Name:
Daphne.S
Affiliation:
PG Scholar
Karunya University

Brief Biographical History:
2010 -Graduated in Computer Science and Engineering Department from Karunya University
2012 - pursuing her M.Tech in Computer and Communication Engineering from the department of Computer Science in Karunya University

Main Works:
Grid computing, Networking and so forth.

Name:
V. Vasudevan
Affiliation:
Director, Software Technologies Lab, TIFAC Core in Network Engineering,
Srivilliputhur, India

Brief Biographical History:
1984- M.Sc in Mathematics and worked for several areas towards Representation Theory
1992 Received his Ph.D. degree in Madurai Kamaraj University
2008- the Project Director for the Software Technologies Group of TIFAC Core in Network Engineering and Head of the Department for Information Technology in Kalasalingam University, Sirivilliputhur, India

Main Works:
Grid computing, Agent Technology, Intrusion Detection system, Multicasting and so forth.