The 11th IEEE International Conference on Computational Science and Engineering - Workshops

An Economic Approach for Scheduling Dependent Tasks in Grid Computing

Hamid Mohammadi Fard
hamid_m_f@yahoo.com
Iran Telecommunication Research Center

Hossein Deldari
hdeldari@yahoo.com
Ferdowsi university of Mashhad-Iran

Abstract

Grids are emerging as a promising solution for resource and computation demanding applications. Here, task scheduling is one of the most important subjects. In traditional approaches for grid scheduling, makespan or completion time of scheduling is the most important parameter for optimization. But in these algorithms, some parameters like user requirements aren't considered. Over the last few years, grid technologies have progressed towards a service-oriented paradigm that enables users to consume these services, based on their QoS (Quality of Service) requirements. However, the heterogeneity of resources in grid computing complicates resource management and scheduling of applications. In addition, the commercialization of the grid requires policies that can take into account user requirements, and budget considerations, in particular. In this paper a new list heuristic algorithm for workflow applications modeled as Directed Acyclic Graphs (DAGs), is developed to solve the scheduling problem, considering time and cost parameters. In this algorithm, the objective function is optimizing both time and cost using a greedy approach.

***Index Terms*—DAG, Economic Grid, Grid Scheduling, Workflow**

1. Introduction[*]

In the context of Grid computing, a wide range of applications can be represented as workflows, many of which can be modeled as Directed Acyclic Graphs (DAGs) [8], [2], [6]. A DAG represents a model that helps build a schedule of the tasks onto resources in a way that precedence constraints are respected and the schedule is optimized. Virtually all existing work in the literature [7], [10] aims to minimize the total execution time (length or makespan) of the schedule.

In the economic Grids, users must pay the cost of using services to owner of those services. However, to impose a workflow paradigm on the economic Grids, execution cost must also be considered when scheduling tasks on resources. The price of a utility service is mainly determined by its QoS level such as the processing speed of the service. Users may not always need to complete workflows earlier than they require. They sometimes may prefer to use cheaper services with a lower QoS that is sufficient to meet their requirements.

There has been some work examining issues related to economic scheduling in a Grid context. The most relevant work is available in [3], [4], [9], [11], [12] these scheduling algorithms can allocate jobs to machines in a way that satisfies constraints of deadline and budget at the same time. In most of these approaches the users must determine a constraint such as deadline or cost, but normally determining these constraints depends on the grid conditions. This means the user must be aware of some grid status, so in most of these algorithms [9] there is a rescheduling phase for covering improper user defined constraints and also in none of these approaches there is a method for decreasing both time and cost without involving the users in grid details.

In order to solve the problem of scheduling optimally under both cost and makespan, we propose an algorithm denoted as TCI (Time and Cost Improvement) which is evaluated in the paper.

The idea in this algorithm is a greedy approach which level sorts the nodes and then the nodes are ranked in each level and are assigned to machines with the proper short makespan and short cost.

The rest of the paper is organized as the following: Section II gives some background information about DAGs. In Section III we present the proposed algorithm along with a sample. In Section IV, we present experimental results that evaluate the approach and compare it with some common algorithms. Finally, Section V concludes the paper.

2. Problem Overview

[*]This work was supported in part by Eqbal Lahoori Institute of Higher Education and Azad University of Mashhad

A workflow application is modeled as a DAG, $G\langle V,E\rangle$, where $V=\{V_i;i=1..n\}$ is the set of n tasks. E is the set of directed edges. $Data$ is a $n\times n$ matrix of communication data, where $Data_{i,k}$ is the amount of data required to be transmitted from task V_i to task V_k. In a given task graph, a task without any parent is called an entry task and a task without any child is called exit task. Following similar studies ([2], [8]), we adopt the following assumptions in DAGs model. It is assumed that there is one entry task to the DAG and one exit task from the DAG. In an actual implementation, we can create a pseudo entry task and pseudo exit task with zero computation time and communication time.

A heterogeneous computing system consists of a set $M=\{M_j;j=1..m\}$ of m independent different types of machines fully interconnected by a high speed arbitrary network. The bandwidth (data transfer rate) of the links between different machines in a heterogeneous system may be different depending on the kind of the network. The data transfer rate is represented by an $m\times m$ matrix, $R_{m\times m}$. T is a $n\times m$ computation time matrix in which each $T_{i,j}$ gives the Estimated Computation Time (ECT) to complete task V_i on machine M_j where $1\le i\le n$ and $1\le j\le m$. The ECT value of a task may be different on different machines depending on the machines computational capability.

The task executions of a given application are assumed to be non-preemptive. The communication cost between two machines M_x and M_y, depends on the channel initialization at both sender machine M_x and receiver machine M_y, in addition to the communication time on the channel. This is a dominant factor and can be assumed to be independent of the source and destination machines. In this study, the channel initialization time is assumed to be negligible.

D is a $n\times n$ matrix of communication times .The communication time of the $edge(i,k)$, which is for transferring data from task V_i (scheduled on machine M_x) to task V_k (scheduled on machine M_y) is defined by

$$D_{i,k}=Data_{i,k}/R_{x,y} \tag{1}$$

Otherwise, $C_{i,k}=0$ when both the tasks V_i and V_k are scheduled on the same machine. Further, for illustration, we assumed that the data transfer rate for each link is 1.0 and hence communication time and amount of data to be transferred will be the same. Consider a task graph with 11 tasks given in Fig. 1 and its computation time matrix in Table 1.

Let $EST(V_i,M_j)$ and $EFT(V_i,M_j)$ are the Earliest Start Time and Earliest Finish Time of task V_i on M_j, respectively. For the entry task V_{entry}, $EST(V_{entry},M_j)=0$ and for the other tasks in the graph, the EST and EFT values are computed recursively, starting from the entry task, as shown in (2) and (3). In order to compute the EFT of a task V_i, all immediate predecessor tasks of V_i must have been scheduled.

$$EST(V_i,M_j)= \max\{Available[j], \max\{AFT(V_t)+D_{ij};V_t\in \Pr ed(V_i)\}\} \tag{2}$$

$$EFT(V_i,M_j)=T_{ij}+EST(V_i,M_j) \tag{3}$$

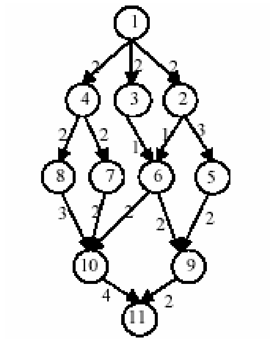

Fig. 1. Task Graph given in [1]

Table 1.Computation time matrix (T) given in [1]

Task Vi	M1	M2	M3
1	4	4	4
2	5	5	5
3	4	6	4
4	3	3	3
5	3	5	3
6	3	7	2
7	5	8	5
8	2	4	5
9	5	6	7
10	3	7	5
11	5	6	7

Where $\Pr ed(V_i)$ is the set of immediate predecessor tasks of task V_i and $Available[j]$ is the earliest time at which machine M_j is ready for task execution. If V_K is the last assigned task on machine M_j, then $Available[j]$ is the time that machine M_j completed the execution of the task V_K and it is ready to execute another task when we have a non insertion-based scheduling policy. The inner max block in (2) returns the ready time, i.e., the time when all the data needed by V_i has arrived at M_j. After a task V_t is scheduled on a M_j, the earliest start time and the earliest finish time of V_t on machine M_j is equal to the actual start time $AST(V_t)$ and the actual finish time $AFT(V_t)$ of task V_t, respectively. After all tasks in a graph are scheduled, the schedule length (i.e. the overall completion time) will be the actual finish time of the exit task V_{exit}. Finally the schedule length is defined as

$$ScheduleLenght = \max\{AFT(V_{exit})\} \quad (4)$$

The contribution of this paper relates to the extension of the traditional DAG model with one extra condition: the usage of each available machine costs some money. As a result, an additional constraint needs to be satisfied when scheduling the DAG, which is the overall financial cost of the schedule. We define the overall (total) cost as the sum of the costs of executing each task in the DAG onto a machine, that is,

$$ScheduleCost = \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij} \text{ (} V_i \text{ is assigned to } M_j \text{)} \quad (5)$$

C is a $n \times m$ computation cost matrix in which each $C_{i,j}$ gives the Computation Cost (CC) to completion task V_i on machine M_j. For the DAG in Fig. 1 matrix C is computed by $C_{i,j} = T_{i,j} \times Cost_j$; which computation cost of machines are shown in Table 2.

The objective functions of the task-scheduling problem are to schedule the tasks of an application to machines such that its both schedule length (given by (4)) and schedule cost (given by (5)) is optimized.

3. Proposed Approach; TCI (Time And Cost Improvement) Algorithm

The proposed algorithm is an extended version for PETS [5]. PETS is an DAG-based algorithm for optimization of makespan. We extend this algorithm by considering cost of machine usage. TCI presents a new list heuristic with an greedy approach. In this algorithm we consider optimization of both makespan and cost. This algorithm consists of three phases: level sorting, task prioritization and machine selection. The main idea in the proposed approach is processor selection which is given in section C.

The explanation of each phase is detailed below:

3.1. Level sorting phase

In the level sorting phase, the given DAG is traversed in a top-down fashion to sort task at each level in order to group the tasks that are independent of each other. As a result, tasks in the same level can be executed in parallel. Given a DAG $G\langle V,E\rangle$, level 0 contain entry task. Level i consist of all tasks V_k such that, for all $edge(V_j,V_k)$, task V_i is in a level less than i and there exists at least one $edge(V_j,V_k)$ such that V_j is in level $i-1$. The last level comprises of some of the exit tasks.

3.2. Task prioritization phase

In the task prioritization phase, priority is computed and assigned to each task. For assigning priority to a task, we have defined three attributes namely, Average Computation Cost (ACC), Data Transfer Cost (DTC) and the Rank of Predecessor Task (RPT). The ACC of a task is the average computation cost on all the m processors and it is computed by using (6)

Table 2. Computation cost of machines for Fig. 1

Mj	Costj
1	180$
2	120$
3	100$

$$ACC(V_i) = \sum_{j=1}^{m} T_{ij} / m \quad (6)$$

The DTC of a task V_i is the amount of communication cost incurred to transfer the data from task V_i to all its immediate successor tasks and it is computed at each level l using (7)

$DTC(V_i) = \sum_{j=1}^{n} D_{ij}; i < j$, where n is the number of nodes in the next level ; $DTC(V_{exit}) = 0$ (7)

The RPT of a task V_i is the highest rank of all its immediate predecessor tasks and it computed using (8)

$$RPT(V_i) = \max\{rank(V_1), rank(V_2),..., rank(V_h)\}$$

Where $V_1, V_2,..., V_h$ are the immediate predecessors of V_i ; $RPT(V_{entry}) = 0$ (8)

Rank is computed for each task V_i based on its ACC, DTC and RPT values. We have used the maximum rank of predecessor tasks of task V_i as one of the parameter to calculate the rank of the task V_i and the rank computation is given in (9).

$$rank(V_i) = round(ACC(V_i) + DTC(V_i) + RPT(V_i)) \quad (9)$$

Priority is assigned to all the tasks at each level l, based on its rank value. At each level, the task with highest rank value receives the highest priority followed by task with next highest rank value and so on. Tie, if any, is broken using ACC value. The task with minimum ACC value receives higher priority. For example, for the task graph given in Fig. 1, the ACC, DTC, RPT, rank and priority values are computed as in Table 3.

Table 3. The DTC, ACC, RPT, rank and priority values for the tasks in Fig. 1

Level	Task	DTC	ACC	RPT	rank	Priority
1	1	6	4	0	10	1
2	2	4	5	10	19	1
2	3	1	5.25	10	16	3
2	4	4	3	10	17	2
3	5	2	3.75	19	25	2
3	6	4	3.5	19	27	1
3	7	2	5.75	17	25	3
3	8	3	3.5	17	24	4
4	9	2	5.75	26.5	34	2
4	10	4	4.25	26.5	35	1
5	11	0	6.5	34.75	41	1

3.3. Machine selection phase

In the machine selection phase we want to select some machines which give us an optimized scheduling for time and cost.

At first some parameters must be defined. C_{avg} is the approximate average of executing cost, for all of tasks on all machines which computed by (10)

$$C_{avg} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} T_{ij}} \quad (10)$$

Then EST and EFT values for all the tasks in the graph are computed. The tasks are selected for execution based on their priority values. For each V_i, consider the smallest EFT which happen for M_a. For each V_i, the difference between execution time for M_j and M_a is computed by (11)

$$EFT\_Diff(i, j, a) = EFT(i, j) - EFT(i, a) \quad (11)$$

And the difference between execution cost for M_j and M_a is computed by (12)

$$Cost\_Diff(i, j, a) = C_{i,a} - C_{i,j} \quad (12)$$

Now by considering these differences, select the best choice by using the following :

Each machine with a negative $Cost\_Diff$ is a bad choice and is put away, because this means that for M_j we have paid more cost, but our task lasts longer. In the remaining machines (machines with positive $Cost\_Diff$) we compute the difference between cost and gain, using 13, for executing V_i on M_j comparing V_i on M_a. Here machine with smaller $Main\_Diff$ is selected.

$$Main\_Diff(i, j, a) = \lfloor C_{i,j} - C_{i,a} \rfloor - [EFT(i, j) - EFT(i, a)] * C_{avg} \quad (13)$$

We named the whole process of machine selection for V_i by $MachineSelection(V_i)$.Then TCI algorithm is listed in Fig.2 .

For task graph in Fig.1 scheduling and selection machine are shown in Fig. 3 and Table 4.

```
Step 1: read the DAG G<V,E>, and M,C,T,D Matrixes;
Step 2:Leveling the DAG;
Step 3: for all tasks $V_k$ at each level $L_i$ do
        begin
            compute $ACC(V_k)$,$DTC(V_k)$ and $RPT(V_k)$;
            $rank(V_k) = ACC(V_k) + DTC(V_k) + RPT(V_k)$;
//tie, if any, is broken based on ACC value, the task with minimum
ACC value receives the higher priority followed by the task with
//next minimum ACC value and so on;
        end;
Step 4: construct a priority queue using ranks;
Step 5: while there are unscheduled task $V_k$ in the queue do
            MachineSelection($V_k$)
```

Fig. 2. Proposed TCI algorithm

Table 4.The Computed EST,EFT on Machines by TCI algorithm

	M1		M2		M3		
Task	EST	EFT	EST	EFT	EST	EFT	Selected Processor
1	0	4	0	4	0	4	3
2	6	11	6	11	4	9	3
4	6	9	6	9	9	12	2
3	6	10	9	15	9	13	3
6	14	17	14	21	13	15	3
5	12	15	12	17	15	18	1
7	15	20	9	17	15	20	3
8	15	17	9	13	20	25	2
10	22	25	22	29	20	25	3
9	17	22	17	23	25	32	2
11	29	34	29	35	25	32	3

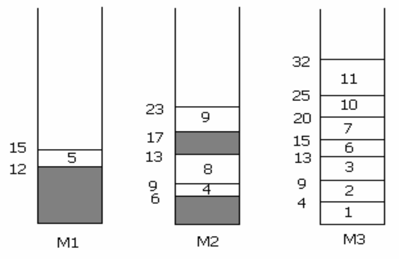

Fig. 3. Machine Selection by TCI algorithm

4. Comparison and Performance Evaluation

We use GridSim to simulate a Grid environment for our experiments. We compare our proposed algorithm denoted as TCI with the smallest loss money in terms of schedule length (LOSS) [9] (an economic scheduling algorithm) and Heterogeneous Earliest Finish Time (HEFT) [10] (a classical scheduling algorithm).

The LOSS algorithm starts with an assignment of tasks onto machines that is optimized for makespan (using a standard algorithm for DAG scheduling onto heterogeneous resources, such as HEFT). As long as the budget is exceeded, the idea is to keep swapping tasks between machines by choosing first those tasks where the largest savings in terms of money will result in the smallest loss in terms of schedule length. In this experiment Budget for LOSS is limited by cost of scheduling by TCI (because of ability of comparison in equal conditions).

HEFT is a two-phase task scheduling algorithm for a bounded number of heterogeneous processors. The first phase namely, task-prioritizing phase is to assign the priority to all tasks. To assign priority, the upward rank of each task is computed. The upward rank of a task is the critical path of that task, which is the highest sum of communication time and average execution time starting from that task to exit task. Based on upward rank, priority will be assigned to each task. The second phase (processor selection phase) is to schedule the tasks onto the processors that give the earliest finish time for the task. It uses an insertion based policy which considers the possible insertion of a task in an earliest idle time slot between two already scheduled tasks on a processor, should be at least capable of computation cost of the task to be scheduled and also scheduling on this idle time slot should preserve precedence constraints.

We created two entities that simulate the users and the brokers by extending the GridSim class. In our simulation, we assume each machine has one PE(Processing Element). The simulation is repeated 200 times for 100 jobs and 15 machines with speed between 1000 to 4000 MIPS. Each gridlet has minimum 10000 MI with 10% growing rate (a set of Gridlets that represent application jobs with different processing).

In the simulation, we use MI (million instructions) to represent the length of tasks and use MIPS (Million Instructions per Second) to represent the processing capability of machines. Network communication assumes a mesh topology. The available network bandwidths between machines are between 16 Mbps to 64 Mbps. We define cost by a function like $Cost = K \times Speed$ which K is selected randomly (in each iteration K is unique).

The two metrics used to evaluate the scheduling approaches are execution time and execution cost.

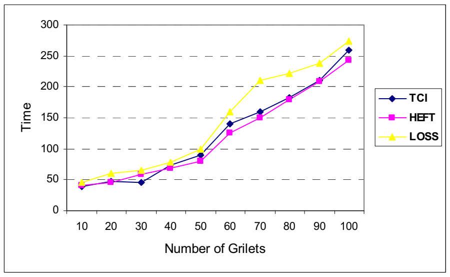

Fig. 4. Makespan Comparison for TCI, LOSS, HEFT

In the first experiment, Fig. 4, makespan of TCI is normally equal to HEFT and it is better than LOSS, specially in more gridlets. It is because in TCI, time is considered in both level sorting and task prioritization phases.

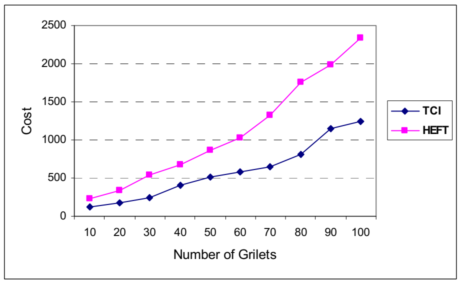

Fig. 5. Cost Comparison for TCI, HEFT

In the next experiment, we compare the algorithms in two charts, because HEFT has a classical approach and it does not have an economic approach. As we can see in Fig. 5 and Fig. 6, the cost of TCI is better than both LOSS and HEFT.

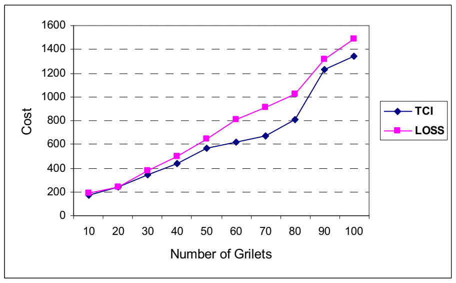

Fig. 6. Cost Comparison for TCI, LOSS

For evaluating the effect of cost function, we consider different cost functions like these:

$Cost = \sqrt{Speed}$ Function1

$Cost = Speed^2$ Function2

$Cost = e^{speed}$ Function3

At first we compare makespan of three algorithms in term of different functions. As we can see in Fig. 7, the effect of different functions in TCI is lower than LOSS.

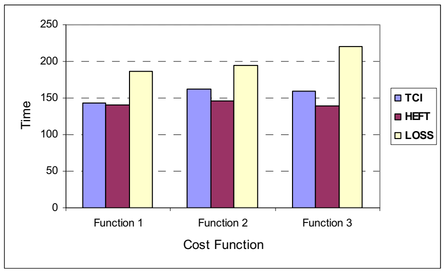

Fig. 7. Makespan Comparison for TCI, LOSS, HEFT with 3 Functions

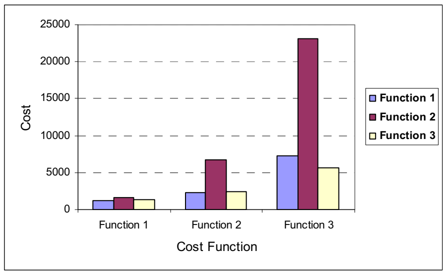

Fig. 8. Cost Comparison for TCI, LOSS, HEFT with 3 Functions

In comparing costs of three algorithms in term of different cost functions, as we can see in Fig. 8, costs of TCI in Function1 and Function2 are better than the others, but in Function3 cost of LOSS is better than TCI. But Function3 has an exponential relation between cost and speed and we could ignore it, because in real world it happens rarely.

The above experiments indicate that the algorithm proposed in this paper is able to find affordable assignments with better cost in comparison HEFT and LOSS algorithms and better makespan in comparison LOSS. That's because TCI ranks task by EFT at first and then select machines by considering costs. Also it can be viewed TCI has low dependency on Cost function in term of machine's speed.

5. Conclusion and Future Work

Economic scheduling in Grids enables users to execute theirs applications with QoS consideration on world-wide network environment such as grid. In this paper, we proposed a DAG-based scheduling algorithm in order to optimizing cost and time of user's workflow executions. We have used a list heuristic algorithm with greedy approach to schedule workflow task execution, such that it can find the optimal scheduling to execute tasks in a user's DAG. The experimental results demonstrate that the proposed scheduling approach can meet low time and low cost well. In future, we will further enhance our scheduling method to support some dynamism parameters such as resource failure in run time; also it could be extend by considering other user requirement.

6. References

[1] R. Bajaj, D.P. Agrawal, 2004. Improving scheduling of tasks in a heterogeneous environments. IEEE Trans. on Parallel and Distributed Systems, 15: 107-118.

[2] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy. Resource Allocation Strategies forWorkflows in Grids In IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005).

[3] R. Buyya. Economic-based Distributed Resource Management and Scheduling for Grid Computing. PhD thesis, Monash University, Melbourne, Australia, http://www.buyya.com/thesis, April 12 2002.

[4] R. Buyya, D. Abramson, and J. Giddy. An economy grid architecture for service-oriented grid computing. In 10th IEEE Heterogeneous ComputingWorkshop (HCW'01), San Fransisco, 2001.

[5] E. Ilavarasan and P. Thambidurai. Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments; Journal of Computer Sciences 3 (2): 94-103, 2007

[6] A. Mandal, K. Kennedy, C. Koelbel, G. Marin, J. Mellor-Crummey, B. Liu and L. Johnsson. Scheduling Strategies for Mapping Application Workflows onto the Grid. In IEEE International Symposium on High Performance Distributed Computing (HPDC 2005), 2005.

[7] R. Sakellariou and H. Zhao. A hybrid heuristic for DAG scheduling on heterogeneous systems. In 13th IEEE Heterogeneous Computing Workshop (HCW'04), Santa Fe, New Mexico, USA, April 2004.

[8] R. Sakellariou and H. Zhao. A low-cost rescheduling policy for efficient mapping of workflows on grid systems. In Scientific Programming, volume 12(4), pages 253–262, December 2004.

[9] R. Sakellariou, H. Zhao, E. Tsiakkouri, M. D. Dikaiakos. "Scheduling Workflows with Budget Constraints". In S.Gorlatch, M.Danelutto (Eds.), Integrated Research in Grid Computing, CoreGrid series, Springer-Verlag, to appear,2005

[10] H. Topcuoglu, S. Hariri, and M. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. In IEEE Transactions on Parallel and Distributed Systems, volume 13(3), pages 260–274, March 2002.

[11] J. Yu,R. Buyya, C. Khong Tham . Cost-based scheduling of scientific workflow applications on utility grids , Vic., Australia;,e-Science and Grid Computing, 2005. First International Conference on, Dec. 2005.

[12] Yu et al. A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms, Workshop on Workflows in Support of Large-Scale Science, Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing (HPDC 2006, IEEE CS Press, Los Alamitos, CA, USA), June 19-23, 2006, Paris, France.