An Improved Grid Scheduling based on SLA for Workflow Application

Animesh Kuity
Electronics & Computer Engineering Department
Indian Institute of Technology Roorkee
Roorkee-247667

Sateesh Kumar Peddoju
Electronics & Computer Engineering Department
Indian Institute of Technology Roorke
Roorkee-247667

ABSTRACT
In grid environment multiple resource providers work together in order to accomplish a complex job. The service level agreement is negotiated between client and a provider for executing the job on high performance computing resources. Most of today grid applications consist of highly correlated tasks. The performance of the workflow application containing highly correlated tasks mainly depends on scheduling.
This paper proposes an improved SLA based Grid scheduling for workflow application using improved heterogeneous earliest finish time algorithm to select appropriate resource(s) to execute each task of workflow application and advance reservation based resource negotiation to get commitment of the resources towards the task. The proposed model to schedule each task of workflow application not only keeps in mind the earliest finish time but also requirement of the user's job. The resultant effect of the proposed model decreases both execution times of the workflow applications as well as rescheduling needed for each task.

Keywords
Grid Computing, SLA, DRS, HEFT, Advance reservation based resource negotiation, QoS, DAG.

1. INTRODUCTION
Grid [1] is defined as "a dynamic collaborative computing environment which allows sharing, aggregation and allocation of resources among multiple administrative domains". Shared resources include computation power, storage, memory, data, and special instruments such as telescope. Grid is a highly decentralized environment involving multiple autonomous administrative domains with its own policies for scheduling, access cost, security, etc. The aim of Grid is to utilize the idle CPU cycles of different contributing organizations for running complex scientific applications requiring days of computation. Efficient utilization of resources is primary goal of Grid environment.
A workflow application is defined as "a set of tasks which is coordinated by control and data dependency" (e.g. weather forecasting workflow, bioinformatics workflow etc.)[2]. Now a days, most of the users want their jobs to finish execution within their specified time and they pay for that. A workflow application is represented by a directed acyclic graph where each node of the graph represents execution time of the task and each edge represents the data transfer time between the tasks. The direction of the edge represents data transfer direction and parent-child relationship between the tasks. For efficient scheduling of workflow application many static and dynamic scheduling algorithms have been proposed (e.g. ADOS [3], MAHEFT [4], LAR [5] etc.). How to schedule the workflow application based on user requirement such as QoS and timing constrain has not been address in those papers. For higher QoS, resource broker demands higher prices. So in most of the cases, user wants cheaper service with lower QoS that are sufficient to meet their requirements [6].
Service level agreement [7] is a contract between user and resource providers which state QoS required by the job, restriction on utilization of resources and penalties during violation of objective. Users need some commitment and assurance on the top of allocated resources to accomplish a job and for dealing with erroneous condition. These terms need to be agreed upon before use and manifested in form of SLA.
When a job is submitted by a user to resource broker, a SLA workflow is created between user and resource broker by describing all the terms and conditions. Resource broker works as a middle man. Each workflow consists of more than one task. Resource broker creates a SubSLA for each task of the workflow application with the service provider so that overall execution time of the job is within user's specified time.
Most of the current research [8, 9] address that task is submitted to the resource provider which has lowest execution time. But if resource broker schedule the task depending on only execution time, it may happen that at later time rescheduling is needed for that task. Due to rescheduling of the task, job may not be able to complete their execution within the specified time. The main focus of this paper is to address these issues. Also advance reservation based resource negotiation is used to get commitment of the service provider to execute the task. A simple architecture of an improved SLA based job scheduling for workflow application is shown in Fig. 1.

2. RESOURCE SELECTION FOR EXECUTING THE JOB
Whenever a user submits a job, resource broker does the entire task on the behalf of user that is selection of suitable resources to execute the job and negotiation with the resource providers. The payment for the job will be the sum of cost incurred due to resource broker and service provider. According to the improved deviation based HEFT [10], tasks of the workflow application are scheduled to the resource provider.

2.1 Improved deviation based HEFT
This proposed model schedules the job according to the requirements of the job rather than resources' own scheduling metric. The workflow application containing more than one task is represented by directed acyclic graph. The proposed improved deviation based HEFT consists of two phase. Task prioritize phase in which the priority of each task is calculated.

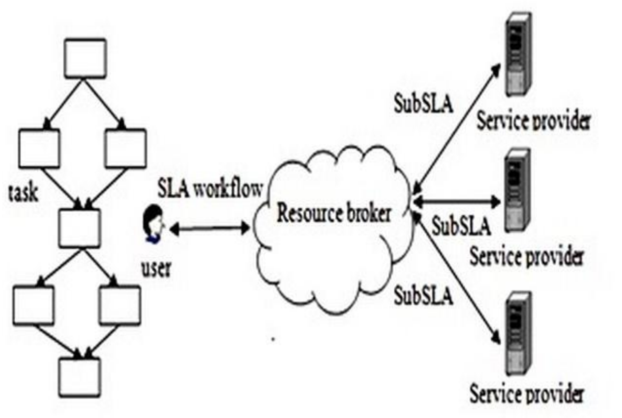

Fig 1: A SLA based job scheduling for workflow application.

To assign the priority of the tasks n_i, improved deviation based HEFT uses task's upward rank (u_i). It is the longest path from the task node n_i to the exit node. Upward rank (u_i) is calculated from (1).

$$Rank_u(n_i) = w_i + \max_{n_j \in succ(n_j)} \left(c_{i,j} + rank_u(n_j)\right) \quad (1)$$

Where w_i is the average computation cost of the task at all available resource providers or processors. $C_{i,j}$ is the average communication cost of edge (i,j) and succ(n_i) is the immediate successor of task n_i .
After calculating the rank of all tasks of the workflow application, tasks are arranged in non-decreasing order of rank. If tie occurs, then it is resolved by taking any task randomly. In resource selection phase suitable resources are selected to execute each task. Resources are selected not only depending on earliest finish time but also which meets the user requirements exactly. It consists of two phases: approximate earliest execution finish time calculation. In this phase, approximate earliest execution finish time of every task to every resource is calculated from (2). In this phase, search for required time slot to execute task n_i on processor p_j starts from ready time that is the time when all input data of task n_i that were sent by all predecessor task of n_i have arrived at processor p_j.

$$EST(n_i, p_j) = \max\{T\_Available[j], \max_{n_m \in pred(n_i)}(EFT(n_m, p_k), c_{m,j})\}$$

$$EFT(n_i, p_j) = w_{i,j} + EST(n_i, p_j) \quad (2)$$

Where EST(n_i, p_j) is the earliest start time of task n_i on processor p_j and EFT(n_i, p_j) is the earliest finish time of task n_i on processor p_j. T_available[j] is the earliest time at which processor p_j will be available for task execution and pred(n_i) is the set of immediate predecessor tasks of task n_i. And $w_{i,j}$ is the execution cost of task n_i.
In the second phase deviation is calculated using the deviation based resource scheduling algorithm against the requirements of the task and the amount of resources that resource provider have. Resources are ordered according to the deviation coefficient. Then, it is selected to fulfil the task requirements.
In deviation based resource scheduling algorithm (DRS) [11], the percentage of deviation D_{ij} is calculated against jth parameter of the task requirement and the corresponding parameter of the ith available resources using (3).

For every available resources from i=1 to N

Percentage of deviation

$$D_{ij} = \begin{cases} \frac{A_v(t)-R_j(t)}{A_v(t)} \times 100 & if\ A_v(t) > R_j(t) \\ \frac{A_v(t)-R_j(t)}{R_j} \times 100 & if\ R_j(t) > A_v(t) \end{cases} \quad (3)$$

Where j=number of resource parameter from 1 to m
$A_v(t)$ is the jth parameter of the ith available resources at time t
$R_j(t)$ is the jth parameter of the request at time t

After calculating percentage deviation D_{ij} for every available resource, in order to scale down percentage of deviation between -1 to +1, percentage of deviation is divided by maximum percentage deviation if it is positive otherwise it is divided by minimum percentage of deviation. Resource with zero deviation value is the resource that exactly meets the task's requirements. Resource with positive deviation value is the resource that has greater capability than what task requires. Resource with negative deviation value is the resource that has lesser capability than what task requires.
Resources are arranged according to non-decreasing order of earliest finish time. Our proposed model first selects the resource that has earliest finish time as well as satisfies all the requirements of the task. Then, it moves towards the increase earliest finish time that satisfies the task requirement until one found. If no resource is available with zero or positive deviation value then it selects the resource of negative deviation value.
ALGORITHM-1

```
1. Set the computation costs of the tasks and communication
   costs of edges with mean value
2. Compute rank of all tasks by traversing graph upward,
   starting from exit node using (1).
3. Sort the tasks in a scheduling list by non-increasing order of
   rank value
4. While there is unscheduled task $t_i$ in the list
      a. Select the first task $n_i$ from the list of scheduling
      b. For each processor $p_j$ in the processor set
            1. Calculate EFT for task $t_i$ using (2)
      c. For each resource from the resource set
            1. Calculate percentage deviation of each
               resource against task requirements using (3)
      d. Scale down the calculated percentage deviation
         between -1 to +1
      e. Sort the resources according to non-decreasing
         order of EFT
      f. Select the resource(s) for execution of task $n_i$ which
         has minimum EFT and also satisfy all the
         requirement of task $n_i$
      g. If it does not satisfy the requirement
            1. Select the resource from increasing order of
            minimum EFT that satisfy all the requirements of
            task $n_i$
      h. If no such resource is available that alone satisfy
         task requirements
            1. select more than one resource that satisfy all
               the requirements of the task
```

2.1.1 *Improved deviation based HEFT by Example*

Fig. 2 shows a simple task graph of a workflow application containing 10 tasks. The number on edges represents communication cost between tasks. Table 1 shows the computation cost of those 10 tasks on processor P_i (i=1 to 5) of resources A_j (j=1 to 5). Rank is calculated based on mean communication cost and mean computation cost as shown in Table 2. Earliest finish time of every task on every processor is calculated using (2). Table 3 shows the earliest finish time calculation of task 1 on every processor. After calculating EFT of task 1, deviation value is calculated against the requirement of the task and amount of resources that resource provider has, using (3). Table 4 shows the deviation value calculation for task 1 using deviation based resource scheduling algorithm. R represents requirement (e.g. CPU_Count, RAM in MB, CPU %) of task 1 and A_i (i=1 to 5) represents the capability of resources. Using Table 3 and Table 4, resource that has minimum earliest finish time and also satisfy all the requirements of task is selected for execution of task 1. In this case, A_2 will be chosen for execution of task 1. And availability of resource 2 (i.e. A_2) will be the execution time of task 1. If no such resource is available, then our proposed model proceeds according to algorithm 1. In this process, our proposed model not only minimizes execution time but also satisfy the user requirement.

Table 1. Computation Costs

Task	P_1	P_2	P_3	P_4	P_5
1	7	15	16	22	15
2	18	8	6	17	13
3	15	16	9	11	18
4	12	11	14	10	9
5	9	14	19	13	11
6	14	20	22	9	12
7	21	10	13	13	8

Table 2. Calculated Rank of Task

Task	Rank
1	133.8
2	104.8
3	100.2
4	73.4
5	47.2
6	45.4
7	13

Table 3. EFT Calculation of Task 1 on every processor

Processor	EFT of task 1
P_1	7
P_2	15
P_3	16
P_4	22
P_5	15

Table 4. Deviation Value calculation for task 1

	CPU_Count	RAM in MB	CPU %
R	6	2727	66
A_1	9	343	71
A_2	8	2823	7
A_3	6	1758	42
A_4	6	402	23
A_5	8	1318	26

$\Downarrow$

	CPU_Count	RAM in MB	CPU
A_1	50	-695.04	-500
A_2	33.33	3.52	6.49
A_3	0.0	-55.12	-57.17
A_4	0.0	-578.36	-186.96
A_5	33.33	-123.89	-158.85

$\Downarrow$

	CPU_Count	RAM in MB	CPU
A_1	0.0	-1.0	-1.0
A_2	0.67	1.0	1.0
A_3	0.0	-0.079	-0.114
A_4	0.0	-0.83	-0.373
A_5	0.67	0.173	-0.318

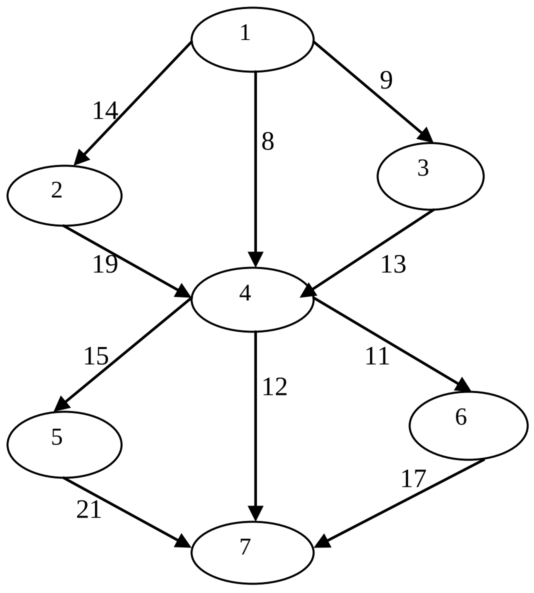

Fig 2: A simple task graph of a workflow application containing 8 tasks

2.2 Advance Reservation based Resource Negotiation

After selecting the resources using improved deviation based HEFT algorithm, the advance reservation [12] based resource negotiation is used to get the commitment of the resources towards the completion of the job. If resource with earliest finish time that satisfy task's requirement (resource with zero or positive deviation value) is available then negotiation will be done with that resource using advance reservation by setting the reservation time as earliest start time.

If earliest start time slot is not available then resource providers will send their free time slot. Resource broker will then negotiate for one of those free time slots as per user requirements to execute the job. If there does not exist any zero or positive deviation value resource that means task's requirements are greater than availability of the resources. In this case two or more resource providers are needed to execute the task. For that the proposed model uses advance reservation based resource negotiation thread. The job of the advance reservation based resource negotiation thread is to first select the resources which are required to execute the task. Then it will create as many threads as the selected number of resources and request them simultaneously to send their free time slot. A common free time slot is chosen from those free time slots and resource broker negotiates with all the selected resources using advance reservation with time as the common free slot time. If any of the resources disagree then resource broker selects another resource from the remaining resources. After successful completion of negotiation, the task is submitted to the selected resource(s). Then, the entire process is monitored by a monitoring engine that monitors the proper consumption of the resource(s). If any kind of violation occurs then an event will be generated and appropriate action is taken. In this process, our proposed model not only minimizes the execution cost but also meet the time constraints imposed by the user.

ALGORITHM-2

```
1. Identify the minimum number of resource(s) required to
   execute the task selected using improved deviation based
   HEFT
   M=minimum number of resource(s)
2. If resource with earliest finish time that satisfy all the
   requirement of task is available
   //set reservation time to earliest start time
      a. Set time=earliest start time
      b. CreateReservation with the resource
      c. If createReservation is successful
            1. Then commitReservation with that
               resource
      d. Else
            1. Send request with QueryFreeTime for free
               time slot
            2. Receive free time slot and send
               commitment for one of the free time slot
3. Else
      a. For 1 to M
      b. Create resource negotiation thread
      c. For each resource negotiation thread
            1. Send request with QueryFreeTime for free
               time slot
      d. Find a common free time slot that is suitable
         to execute the job
      e. For 1 to M
      f. Set time = common free time slot
      g. CreateReservation with the resource
      h. If createReservation is successful
            1. Then commitReservation with that
               resource
      i. Else
            1. Choose another resource for negotiation
            2. Repeat the process
```

3. CONCLUSION

Grid computing environment can be highly heterogeneous. Different hardware characteristics such as CPU speed, cache and interconnect can impact the time that a job takes to execute. Different sites use different local scheduling policies. Some site might favor parallel job or may restrict on finish time of the job. So SLA can be used to remove all these difficulties. A SLA is a contract between participating parties which acts as a legal document for a set of guarantee and QoS metrics. This paper proposes an efficient service level agreement based grid scheduling for workflow application which improves not only required execution time of SLA-based workflow application in grid environment but also satisfy all the requirement of the job. In this process, we reduce rescheduling needed for tasks. In future, the implementation of whole proposed model will be done in Globus environment.

4. REFERENCES

[1] I. Foster, C. Kesselman, S.Tuecke, 2001. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputer Applications, 15(3).

[2] Zhifeng Yu and Weisong Shi, 2007. "An Adaptive Rescheduling Strategy for Grid Workflow Applications", Proc. 21st Int'l Parallel and Distributed Processing Symp. (IPDPS), pp. 1-8.

[3] Young Choon Lee, Riky Subrata, and Albert Y. Zomaya, 2009. “On the Performance of a Dual-Objective Optimization Model for Workflow Applications on Grid Platforms”, IEEE Transactions on Parallel and Distributed Systems, vol. 20(9), pp. 1273 - 1284.

[4] Ahmed Mohamed A. Ghanem, Ahmed I. Saleh, Hesham Aarafat Ali, 2010. “High Performance Adaptive Framework for Scheduling Grid Workflow Applications”, International Conference on Computer Engineering and Systems (ICCES), pp. 52-57.

[5] Fang Dong, Junzhou Luo, Aibo Song, Jiuxin Cao, 2011. “Load-aware based Adaptive Rescheduling Mechanism for Workflow Application”, 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp.400-407.

[6] Jia Yu, Rajkumar Buyya and Chen Khong Tham,2005. “QoS-based Scheduling of Workflow Applications on Service Grids”, Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing (e-Science 2005, IEEE CS Press, Los Alamitos, CA, USA), Dec. 5-8, Melbourne, Australia.

[7] P.Balakrishnan, S. Thamarai Selvi and G.Rajesh Britto, 2008.“GSMA based Automated Negotiation Model for Grid Scheduling”, publish in SCC 2008WIP, IEEE Congress on Services, SERVICES 2008, July 8-11, Honolulu, Hawaii, USA, pp. 569-570.

[8] D.I.George Amalarethinam, F.Kurus Malai Selvi,2011. “An Efficient Dual Objective Grid Workflow Scheduling Algorithm”, International Journal of Computer Applications (0975 – 8887), vol. 33(1).

[9] Dang Minh Quan & J¨orn Altmann, Laurence T. Yang, 2008. “Optimizing the execution time of the SLA-based workflow in the Grid with parallel processing technology”, Asia-Pacific Services Computing Conference (APSCC '08. IEEE), pp. 15-20

[10] Haluk Topcuoglu, Salim Hariri, Min-You Wu,2002. “Task Scheduling Algorithms for Heterogeneous Processors” Eighth Heterogeneous Computing Workshop(HCW '99) Proceedings, pp.3-14

[11] P.Balakrishnan, S. Thamarai Selvi and G.Rajesh Britto, 2008. “Service Level Agreement based Grid Scheduling” International Conference on Web Service publish in SCC 2008WIP, Honolulu, Hawaii, USA, pp. 203-210.

[12] Anthony Sulistio , Rajkumar Buyya, Rao Kotagiri, 2008. “Advance Reservation and Revenue-based Resource Management for Grid System”, Doctor of Philosophy thesis, University of Melbourn, Australia.