CONSTRUCTING DIRECTED ACYCLIC GRAPH OF INDEPENDENT TASKS USING DEADLINES FOR SCHEDULING

K. Srikala [1], S. Ramachandram [2]

[1] Offshore Faculty, TutorVista Global Pvt. Ltd, Hyderabad, Andhra Pradesh 500 067, India
[2] University College of Engineering, Computer Science Department, Osmania University, Hyderabad, Andhra Pradesh 500 007, India

ABSTRACT: The need for the scheduling algorithms arises from the requirement to perform multitasking, essentially for present computing systems. Scheduling is completely the resource management. Many of the applications executed on grids are interdependent tasks that is workflows where as some applications are independent tasks. In practice different scheduling algorithms are used for allocation of resources to interdependent and independent tasks. If we can create some sort of dependency between independent jobs then we can use the workflow scheduling algorithm for scheduling independent tasks as well.
In this paper, we are going to present an algorithm that creates a DAG for the given set of interdependent tasks based on the resources available and deadlines of the tasks. Workflow based algorithms search for an efficient allocation for the entire workflow, So, If we can use the same workflow scheduling algorithm for independent tasks scheduling will be more effective. This is the main aim of creating DAG for independent tasks.
KEYWORDS: Grid workflow, workflow scheduling, inter dependent tasks, independent tasks, directed acyclic graph.

1. INTRODUCTION

Grids have emerged as global cyberinfrastructure for the next-generation of e-Science and e-Business applications, by integrating distributed, heterogeneous, large-scale resources. The applications executed on grids are independent tasks and workflows. A workflow is a set of linked inter dependent tasks. A workflow management system is responsible to define, manage and execute workflow applications on grid resources. A workflow management system may use a specific scheduling strategy for allocating the tasks in a workflow to suitable grid resources in order to meet the user requirements where as Task based algorithms allocate resources to tasks using a Greedy approach. As a result there is more than one algorithm for scheduling interdependent and independent tasks.
Some workflow scheduling strategies have been proposed in literature. Mustafizur Rahman et al [RVB07] developed a DCP-G (Dynamic Critical path for grids) algorithm this is designed for mapping tasks on to homogenous processors. Jia Yu et al [YB06a] developed a genetic algorithm for optimized scheduling. K. Srikala et al [KS13] developed and investigated that Pre-emptive resource allocation in grid computing is an efficient algorithm for scheduling independent tasks. These are the algorithms developed for scheduling independent tasks. Jia Yu et al [YBR08] developed workflow scheduling algorithms for Grid computing. Jim Blythe et al [B+05] developed Task scheduling strategies for workflow based applications in grids. J. Yu et al [YB06b] developed A budget constrained scheduling of workflow applications on utility grids using genetic algorithms. A. Mandal et al [M+05] developed scheduling strategies for mapping application workflows on to the grids. These algorithms are developed for scheduling workflow of tasks. These are some of the observations that bring new challenges in designing an algorithm for constructing Directed Acyclic Graph from the set of independent jobs so that we can use the same algorithm for scheduling the independent tasks and workflow of jobs in the future. However this paper deals only with the generation of directed acyclic graph for the set of independent jobs and then it compares the scheduling of jobs with workflow and independent task scheduling strategies.
Constructing Directed acyclic graph of independent tasks using deadlines for Scheduling addresses issues mentioned above.
The rest of the paper is organized as follows.
In section 2 we describe related work. Section 3 describes the Activity involved in Constructing directed acyclic graph for the set of independent jobs. Section 4 explains the process with an example. In section 5 we describe the algorithm for Constructing directed acyclic graph for independent jobs. Simulation environment and results are discussed in section 6. Finally, we conclude the paper in section 7.

2. RELATED WORK

Review of literature reveals that grid environments are more failure prone as the resources may come and go at any time. Most of the time we get data intensive applications or computational applications to be scheduled on grid. We use workflow – based algorithm for scheduling this type of tasks. The tasks may be independent or interdependent. We would like to use the same algorithm for both the tasks.
In general, a workflow application is represented as a directed acyclic graph (DAG) in which the nodes of the graph represent data dependencies among the tasks. The independent tasks are also represented by the directed acyclic graph in which we create a dependency based on the deadlines given for the set of tasks. Since scheduling is a special case of a DAG it is NP-Complete. Some of the well-known DAG generators are DIGITIZER and CCF. These algorithms generate a Directed acyclic graph for a set of independent tasks. We are going to introduce the algorithm that creates a deadline dependency using earliest deadline first among independent jobs.
Constructing directed acyclic graph for independent tasks is different from other algorithms that can generate directed acyclic graphs. Other algorithms generate Directed Acyclic Graphs for interdependent tasks where as we are creating a deadline dependency between independent tasks and generate DAG for independent tasks.
Many applications require some assurances of quality of services (QoS) like results to be delivered before the deadline. Therefore, scheduling is required to be able to analyze QoS requirements specified by the user and map tasks on available resources such that the execution of independent tasks or workflow can be completed to meet the QoS constraints specified by the user. Most QoS constraint based workflow scheduling is based on either cost or time constraints. Constructing directed acyclic graphs of independent tasks using deadlines for scheduling is basically deals with Construction of directed acyclic graph for independent tasks by making the deadline as the dependency between the tasks. This algorithm uses earliest deadline first for creating tasks at each level that is the task with smallest deadline is placed at the top of the directed acyclic graph. Our intention in creating DAG for independent tasks is to use the same algorithm for independent and interdependent tasks so that a workflow scheduling algorithm can be used for scheduling of independent tasks as well.
The main objectives of generating DAG for independent tasks are

i. Same workflow scheduling algorithm is used to schedule independent and interdependent tasks by completing the tasks within the deadline.
ii. Workflow scheduling algorithms improve resource owner benefits by increasing the resource utilization rate and user benefit by completing their task within the deadline.

3. ACTIVITY FOR CONSTRUCTING DIRECTED ACYCLIC GRAPH OF INDEPENDENT TASKS USING DEADLINES FOR SCHEDULING

The grids are heterogeneous and dynamic environments consisting of network, storage and computing resources. The resources are sorted based on the processing power. Resources that are able to complete the given set of jobs are identified. If there are n resources n directed acyclic graphs are built for the given set of jobs. Job, task and gridlet notation is used interchangeably. Gridlet that has shortest deadline will be the root node of the graph. Thereafter, we use the same condition for adding child nodes. Figure1 show the activity involved in constructing Directed acyclic graph for the independent jobs.

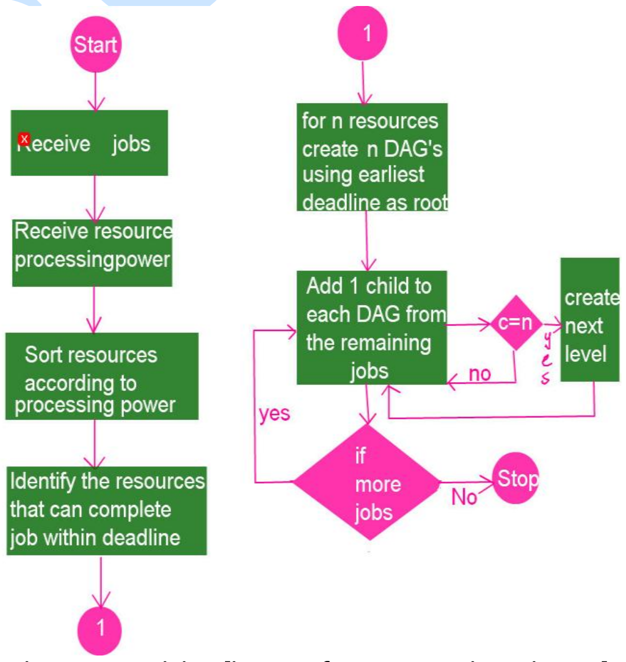

Start
Receive jobs
Receive resource processingpower
Sort resources according to processing power
Identify the resources that can complete job within deadline
1
1
for n resources create n DAG's using earliest deadline as root
Add 1 child to each DAG from the remaining jobs
c=n
yes
create next level
no
yes
if more jobs
No
Stop

Figure 1: Activity diagram for constructing Directed Acyclic Graph of the given independent set of jobs

In the diagram c stands for number of child nodes of a node of DAG and n is the number of resources that can complete job within deadline.
From the given set of jobs first add a root node to each DAG. Thus, there are n Directed Acyclic Graphs for n number of resources. Then from the remaining set of jobs add one child to each root of DAG and then if there are more jobs keep adding to one job for each DAG in the same level. Each time

while adding a child node count the number of children to each node. If children are equal to resources start adding to the next node, when the next node is not available create next level. This is the procedure used to create a DAG of the independent jobs. In this way we can model the DAG for the given set of independent jobs.

4. CONSTRUCTING DIRECTED ACYCLIC GRAPH OF INDEPENDENT TASKS USING DEADLINE FOR SCHEDULING WITH AN EXAMPLE

Job	Size (Millions of instructions)	Deadline (sec)	MIPS /Sec
J1	12	9	1.33
J2	8	10	0.8
J3	6	8	0.75
J4	9	5	1.8
J5	3	8	0.375
J6	9	6	1.5
J7	14	3	4.66
J8	11	2	5.5
J9	8	4	2
J10	7	5	1.4
J11	3	9	0.33
J12	6	3	2
J13	4	1	4

Figure 2a: Given set of jobs with their size and deadline

Figure 2a shows the Jobs arrived at a given interval of time. Size of the jobs is given in Millions of instructions per second. The deadline for each job is given in seconds. The last column shows the number of millions of instructions executed per second in order to complete the job within the deadline.
We are having four resources available at the time when the above jobs arrived. Then, for the above set of jobs four Directed acyclic graphs are generated. The following are the Directed acyclic graphs generated for the given set of jobs.

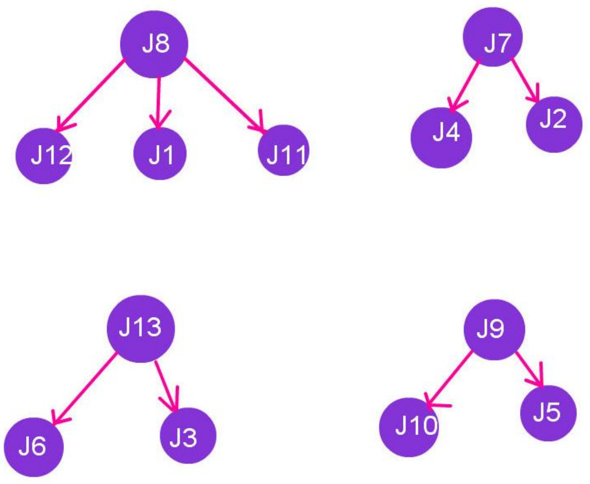
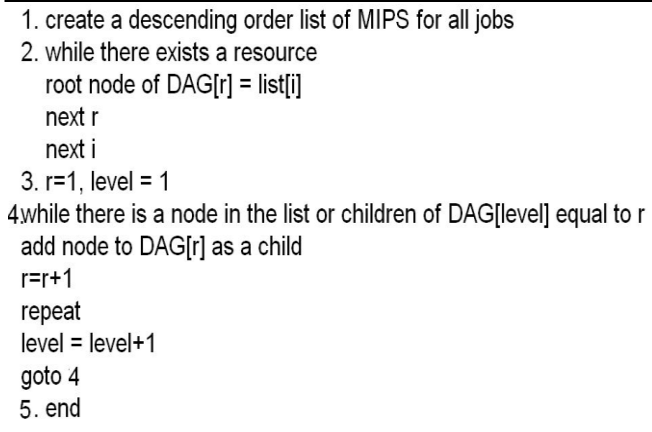


Figure 2b: Directed acyclic graphs generated for the set of jobs shown in figure 2a

While creating the directed acyclic graphs the algorithm takes root nodes in descending order of Millions of instructions per second. As we have four resources there will be four roots and they are J8, J7, J13 and J9. J 12 is also having the same millions of instructions per second, but we start retrieving from the beginning. Hence, J9 is the root node. Once the root nodes are added the next four jobs are extracted from the list and added one to each of the directed acyclic graphs at the next level. We continue adding to the same level till number of children at each level is equal to number of resources available at the time of creating directed acyclic graph. Once children at each level is equal to number of resources we start adding the jobs to the next level. This is the procedure involved in creating directed acyclic graphs for the independent set of jobs. This way we are creating directed acyclic graphs and any workflow scheduling algorithm can allocate resources for the jobs shown in figure 2b.

5. CONSTRUCTING DIRECTED ACYCLIC GRAPH FOR THE GIVEN SET OF INDEPENDENT JOBS

The following is an algorithm for constructing directed acyclic graph for the given set of independent jobs. The sorted list contains fourth column of figure 2a and r is the number of resources available at that particular instant. Level indicates to which level we are adding the nodes of the list. Figure 3 shows the algorithm for the procedure described above.



```
1. create a descending order list of MIPS for all jobs
2. while there exists a resource
     root node of DAG[r] = list[i]
     next r
     next i
3. r=1, level = 1
4.while there is a node in the list or children of DAG[level] equal to r
  add node to DAG[r] as a child
  r=r+1
  repeat
  level = level+1
  goto 4
5. end
```

Figure 3: Algorithm for Constructing directed acyclic graph of given set of jobs

The algorithm shown in figure 3 generates directed acyclic graphs equal to the number of resources available. The directed acyclic graphs constructed are used in scheduling of independent jobs. The main idea behind creating such a set of directed acyclic graphs is to make use of the same scheduling algorithm for workflow of jobs and independent jobs.

6. SIMULATION ENVIRONMENT AND RESULTS

The Algorithms for comparison DCP-G (Dynamic Critical path for grids) and GA (genetic algorithm) are implemented on a machine based on Pentium 1.6GHz with 120GB HDD and RAM of 1GB on Microsoft Windows XP. This experiment is carried out on a simulated grid environment provided by GridSim.
The Simulation environment consists of resources. There are four resources having different processing capabilities in terms of Millions of Instructions per Second (MIPS).
PRAG is compared with DCP-G and GA the results are recorded on graphs.
We evaluate the scheduling decisions on the basis of number of gridlets completed in the given amount of time. The resources that can complete the execution within the deadline are identified and then they are sorted in descending order if the prior constraint is time, when the prior constraint is cost the resources are sorted in descending order.
The following are the graphs drawn for comparison.
Figure 4a shows the graph between number of jobs and total number of rejections.
From the figure we can see that workflow scheduling algorithm is more efficient than scheduling independent jobs by using greedy approach.

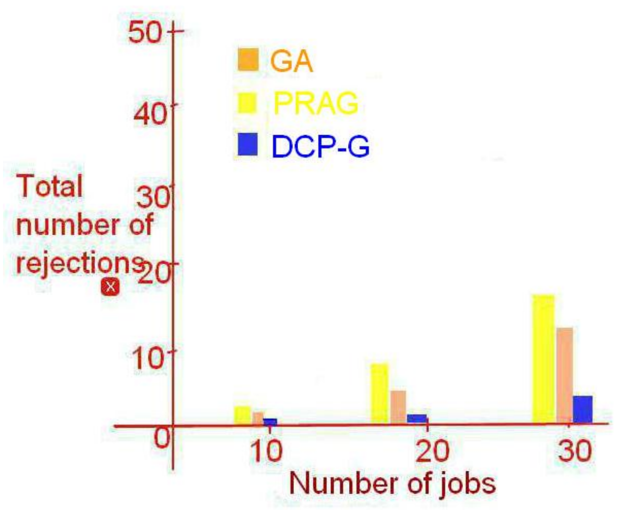


Figure 4a Graph between number of jobs and number of rejections

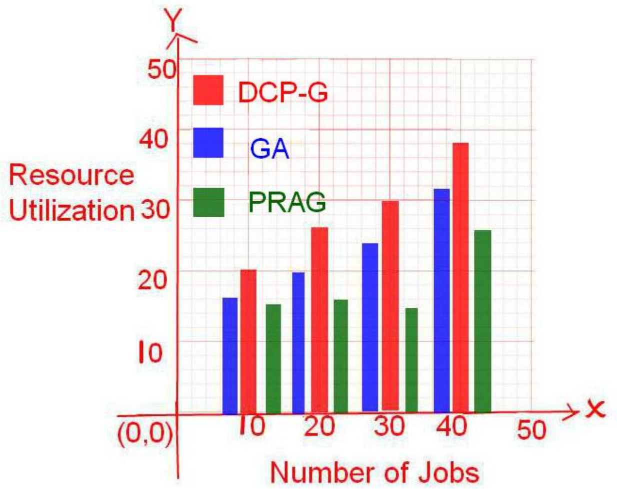


Figure 4b: Graph between number of jobs versus resource utilization

From figure 4b it is evident that resource utilization is high in workflow scheduling algorithms like DCP-G and GA than independent job scheduling algorithm like PRAG.
Based on the results shown above and literature survey we found that there is a need to develop a workflow scheduling algorithm that creates workflow for the independent set of jobs and allocates resources for the entire workflow.

7. CONCLUSIONS AND FUTUREWORK

In this paper we have presented a new technique for constructing a directed acyclic graph for the set of independent jobs given at a particular interval. We also tested the given set of jobs using PRAG, DCP-G and GA. We have observed that workflow scheduling algorithms like genetic algorithm and DCP-G are more efficient in terms of resource utilization. The number of jobs rejected are very less when compared to the independent job scheduling algorithm PRAG.

We also observed that existing workflow scheduling algorithms can generate DAG only for the workflow. We would like to develop a workflow scheduling algorithm which works for independent set of jobs and workflow of jobs. When a set of independent jobs come it creates a directed acyclic graph as discussed in constructing directed acyclic graph for the independent set of tasks using deadlines for scheduling. Thus the algorithm we are going to develop in future works for independent set of jobs and workflow of jobs. The main idea behind this paper is to use the same algorithm for scheduling of workflow and independent jobs to grid resources.

REFERENCES

[B+05] **Jim Blythe, Sonal Jain, Ewa Deelman, Ken Kennedy, Anirban Mandal** - *Task scheduling strategies for workflow based applications in grids*, IEEE International Symposium on Cluster Computing and Grid (CCGrid), 2005, IEEE Press.

[KS13] **Srikala Keshetti, Ramachandram Sirandas** - *Pre-emptive resource allocation in grid computing*, Proceedings of Information & Communication Technologies (ICT), 2013 IEEE, pp 240 – 243.

[M+05] **A. Mandal et al.** - *Scheduling Strategies for Mapping Application Workflows onto the Grid*, High Performance Distributed Computing, 2005. HPDC-14. Proceedings, 14th IEEE International Symposium, 24-27 July 2005, pp 125 - 134.

[RVB07] **Mustafizur Rahman, Srikumar Venugopal, Rajkumar Buyya** - *A dynamic critical path algorithm for scheduling scientific workflow applications on global grids*, E-SCIENCE '07 Proceedings of the Third IEEE International Conference on e-Science and Grid Computing, pp 35-42.

[YB06a] **Jia Yu, Rajkumar Buyya** - *Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms*, Scientific Programming - Scientific Workflows, Volume 14 Issue 3,4, December 2006, pp 217-230.

[YB06b] **J. Yu, R. Buyya** - *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.

[YBR08] **Jia Yu, Rajkumar Buyya, Kotagiri Rammanoharrao** - *Workflow scheduling algorithms for grid computing*, Metaheuristics for Scheduling in Distributed Computing Environments, Studies in Computational Intelligence Volume 146, 2008, pp 173-214.