Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# Priority Based Workflow Management in Grid Computing System

**Er. Gaurav Gulati , Er. Tarun Kumar Dhiman**
*Dept. of CSE*
*KUK, Kurukshetra , India*

***Abstract— Grid is offered to solve huge and complex problems, and type of parallel computing technology in which the nodes are almost completely independent of each other in terms of resources. A priority-based workflow management algorithm in grid is proposed in this paper. The aim of this research is to divide a workflow into tasks and tasks are assigned priorities. We have defining the static values for workflow and generate a static model for workflow management. Results show that when we provide workflow our algorithm provides the priorities assigned to each task. These priorities are assigned on the basis of Uplink Cost, Downlink Cost and Average computation Cost. These three parameters are taken as input from the user and the output comes as task order.***

***Keywords*: Workflow Management, Directed Acyclic Graph, Load Balancing, Task Priority, Scheduling**

## I. INTRODUCTION

Grid computing is significant way to solve large-scale and complicate problems since Ian Foster proposed the concept of it. The term Grid generally referring to some form of system framework into which hardware or software components can be plugged, and which allow easy configuration and formation of new functionality from existing components. In grid multiple numbers of same types of computers clustered together. Grid can provide users with integrated information and application services to realize resource sharing and collaboration in this virtual environment. A grid [1] is defined as a group of nodes that are connected to each other through a number of connecting paths for communication**.** Thus grid computing is widely used to solve large-scale computational problems.

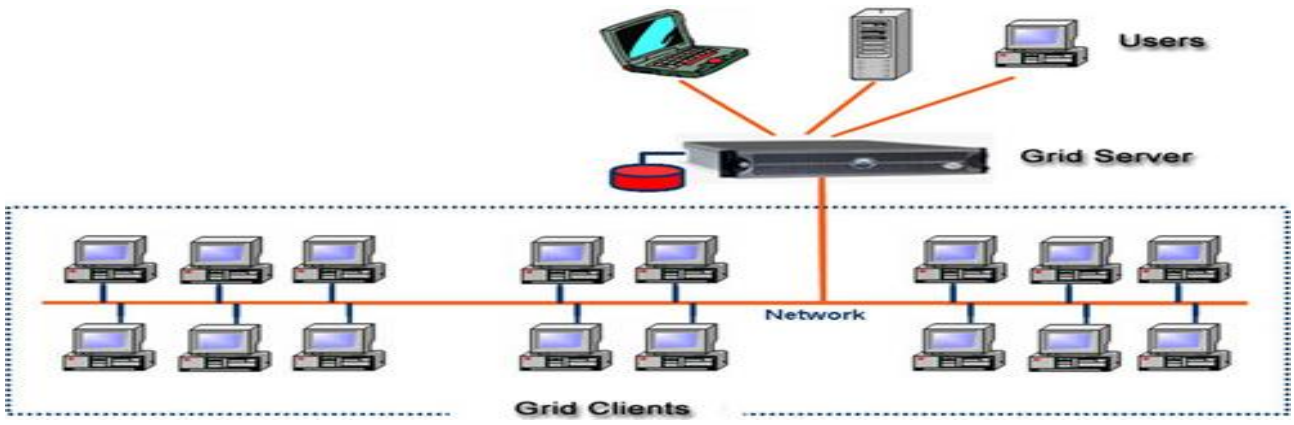


**Fig 1.1: Grid Architecture**

Grid applications are usually divided into many interdependent subtasks in real applications. Every single subtask is processed dependently 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. Load Balancing [5] is crucial to computational grids. It is a mapping strategy that efficiently equilibrates the task load into multiple computational resources in the network based on the system status to improve performance.

### A. Directed Cyclic Graph (DAG)

DAG is usually used to illustrate the data dependency among subtasks, and it can be used to solve task scheduling problems. Actually, the most common Grid workflow can be modelled as simple Task Directed Acyclic Graphs (DAG), where the order of execution of tasks is determined by dependencies Each DAG node represents the execution of a component, characterized by a set of attributes such as an estimate of its cost and possible requirements on the target execution platform, while DAG directed edges represent data dependencies between specific application components.

### B. Grid Workflow Management

Grid workflows are an emerging research field in the Grid community. The workflow [2] is expressed as a set of tasks with dependencies between them. A task becomes executable when its dependencies are met and may be scheduled by submitting it to a resource for execution. The workflow design includes key factors involved in the workflow at build-

 

time. A workflow is composed by connecting multiple tasks according to their dependencies. Workflow structure, also referred as workflow pattern, indicates the temporal relationship between tasks. In general, the workflow can be represented as DAG and non-DAG. In DAG-based workflow, 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.

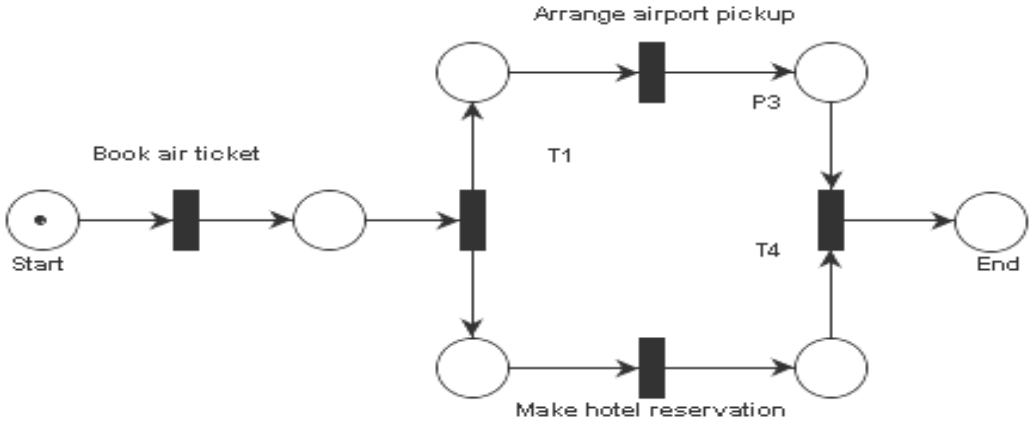


**Figure 1.2 Practical model of workflow**

In addition to all patterns contained in a DAG-based workflow, non-DAG workflow also includes iteration structure, in which sections of workflow tasks in an iteration block are allowed to be repeated. Iteration is also known as loop or cycle. Iteration structure is quite frequent in scientific applications, where one or more tasks needed to be executed repeatedly. These four types of workflow structure, namely sequence, parallelism, choice and iteration, can be used to construct many complex workflows. Moreover, sub-workflows can also use these types of workflow structure as building blocks to form a large-scale workflow. A full or partial concrete model can be generated just before or during workflow execution according to the current status of resources. Additionally, in some systems, every task in the workflow is concretized only at the time of task execution. However, concrete models may be needed by some end users who want to control the execution sequence

C. **Workflow Scheduling**

Task scheduling [4] in distributed computing systems into 'local' task scheduling and 'global' task scheduling. Local scheduling involves handling the assignment of tasks to time-slices of a single resource whereas global scheduling involves deciding where to execute a task

## II. TASK PRIORITY PROBLEM

A Problem is divided into tasks according to their dependencies and can be represented with the help of DAG.A DAG is represented by tuples such as G (V, E, P, T, C) where

*a)* Number of tasks: *V* and the *c*omputation cost matrix of the DAG: *T(V×V)*
*b)* Amount of data to be transferred between the tasks: *D (v×v)*
c) Number of processors in the systems: *P*
d) Communication Edges from one task to another task *i.e. E($v_x$,$v_y$)*
e) Communication cost associated with edges :C

The proposed model has two stages, such as level sorting, task prioritization phase .In the level sorting phase, the given DAG is traversed in a top-down fashion to sort tasks 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 = (V, E), level 0 contain entry tasks. Level I consist of all tasks vk such that for all edges ($v_y$, $v_z$), task $v_y$ is in a level less than i and there exists at least one edge ($v_y$,$v_z$) such that vj is in level i-1. The last level comprises some of the exit tasks. For implementation, it is assumed that there is one entry task and one exit task for a DAG. If there are multiple entry or exit tasks, the multiple tasks can be connected to a dummy task with zero computation cost and zero communication cost edges. In the task prioritization phase, priority is computed and assigned to each task of the task graph. The attributes used to assign the priority to a task are the Down Link Cost (DLC), the Up Link Cost (ULC), the Link Cost (LC) and the Average Computation Cost (ACC) of the task. The DLC of a task is the maximum data (input) received by a task from all its immediate predecessor tasks. The DLC for all tasks at level 0 is 0 and for all other tasks at level l, the DLC of a task $v_y$ is computed using the Eq 1.

$$DLC(vj) = Max\{Data(v_x,v_y)\}, \text{ where } v_x \hat{I} pred(v_y) . \quad (1)$$

The ULC of a task is the max data to be transferred from a task to all its immediate successors. The ULC for exit task is 0 and for all other tasks at level l, it is computed using the Eq 2.

$$ULC (v_y) = Max \{Data(v_y, v_z)\}, \text{ where } v_z \hat{I} succ(v_y). \quad (2)$$

 

The LC of a task is the sum of DLC, ULC and maximum LC of its immediate predecessor tasks. The LC of a task is calculated using the Eq 3.

$$LC(v_y) = 0, \text{ for entry task, otherwise} = \max\{LC(v_z)\}+ULC(v_y)+DLC(v_y), \text{ for all } v_z \hat{I} \text{ pred}(v_y). \quad (3)$$

The Average Computation Cost (ACC) of a task vi is the average of computation cost on all the m available processors and it is computed using the Eq (4).

$$ACC(vi) = \sum T(v_x, pj) \quad (4)$$

Priority is assigned to all the tasks at each level i, based on its LC value. At each level, the task with the highest LC value receives the highest priority followed by the task with next highest LC value and so on in the same level. While assigning priority, if two tasks are having the same LC value, then the tie is broken based on the ACC value. The task with maximum ACC value receives higher priority than the task with the lower ACC value.

**Proposed Algorithm**

1. Start
2. Read the DAG, associated attributes values, and the number of processor P;
3. Level sort the given DAG;
4. For all task $v_z$ in the DAG do
5. Begin
6. Compute ULC, DLC and ACC values for the task vk ;
7. Compute DLC(vj) = Max{Data($v_x$,$v_y$)}, where $v_x$ Î pred($v_y$) .
8. Compute ULC (vj) = Max {Data($v_y$, $v_z$ )where $v_z$ Î succ($v_y$).
9. Compute LC(vk) = max{LC($v_y$)} + ULC($v_z$) + DLC($v_z$), where $v_y$Î bpred($v_z$);
10. Put the task into the priority queue based on the LC value such that the tasks in Lower level are positioned in the priority queue first than the tasks in the higher level and tie if any, is broken using the ACC value.
11. End

The figure 2.1 represented workflow with the help of DAG. A Priority based Static Model is proposed to deal with the above problem. As shown in fig 2.1, it contains total ten tasks as V1 & V10 are entry and exit task. After applying the algorithm it sort out the priority to task at each level.

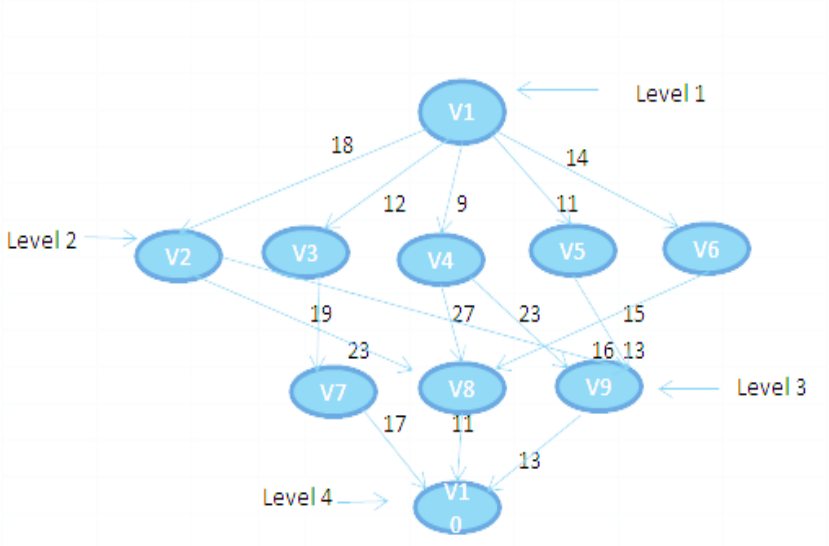


**Fig 2.1 Workflow representations with help of DAG**

**Table 2.1 DAG REPRESENTATION**

| Vertex | Successor | Predecessor |
|---|---|---|
| V1 | V2,V3,V4,V5,V6 | None |
| V2 | V8,V9 | V1 |
| V3 | V7 | V1 |
| V4 | V8 | V1 |
| V5 | V9 | V1 |
| V6 | V8 | V1 |
| V7 | V10 | V3 |
| V8 | V10 | V4 |
| V9 | V10 | V5 |
| V10 | None | V7,V8,V9 |

 

**Table 2.2 TASK LIST SORTED BY LEVEL**

| Level | Task | DLC | ULC | LC | ACC | Priority |
|---|---|---|---|---|---|---|
| 1 | V1 | 0 | 18 | 18 | 13 | 1 |
| 2 | V2 | 18 | 19 | 55 | 16 | 1 |
| 2 | V3 | 12 | 23 | 53 | 14 | 3 |
| 2 | V4 | 9 | 27 | 54 | 12 | 2 |
| 2 | V5 | 11 | 13 | 42 | 11 | 5 |
| 2 | V6 | 14 | 15 | 47 | 16 | 4 |
| 3 | V7 | 23 | 17 | 93 | 11 | 1 |
| 3 | V8 | 27 | 11 | 93 | 10 | 2 |
| 3 | V9 | 23 | 13 | 91 | 16 | 3 |
| 4 | V10 | 17 | 0 | 110 | 14 | 1 |

**Table 2.3 :TASK LIST SORTED BY LEVEL AND BY PRIORITY**

| Level | Task | DLC | ULC | LC | ACC | Priority |
|---|---|---|---|---|---|---|
| 1 | V1 | 0 | 18 | 18 | 13 | 1 |
| 2 | V2 | 18 | 19 | 55 | 16 | 1 |
| 2 | V4 | 9 | 27 | 54 | 12 | 2 |
| 2 | V3 | 12 | 23 | 53 | 14 | 3 |
| 2 | V6 | 14 | 15 | 47 | 16 | 4 |
| 2 | V5 | 11 | 13 | 42 | 11 | 5 |
| 3 | V7 | 23 | 17 | 93 | 11 | 1 |
| 3 | V8 | 27 | 11 | 93 | 10 | 2 |
| 3 | V9 | 23 | 13 | 91 | 16 | 3 |
| 4 | V10 | 17 | 0 | 110 | 14 | 1 |

Table 2.2 & 2.3 shows task List Sorted by level then by priority .we can find out the priority at each level of DAG based on parameters such as DLC, ULC & LC. As shown in table 2.3, task list sorted by level then assign priority at each level so now order of execution of workflow will be priority based.

## III. Conclusions

In this paper, priority based task scheduling workflow management has been implemented over a grid. Basically, workflow is difficult to be divided in tasks in case of grid computing as the size of jobs are not known a priori and the selection criteria are also a bottleneck. Also the assignment of priorities to tasks is an issue as the level of each task is not known and also to differentiate between entry and exit task is difficult. So what we have done is, Workflow has been divided into tasks and tasks are assigned priorities. Results show that when we provide workflow, algorithm provides the priorities assigned to each task. These priorities are assigned on the basis of Uplink Cost, Downlink Cost and Average computation Cost. These three parameters are taken as input from the user and the output comes as task order.

**Refrences**

[1] Yadav,K., Jindal, D. and Singh, R.(2013) "*Job Scheduling in Grid Computing*" *International Journal of Computer Applications"* 69(22):13-16, May.

[2] Marton, I, Karoczkal , K. , Kozlovzsky. , M.(2012) "*Enabling Generic Distributed Computing Infrastructure Compatibility for Workflow Management"* International Journal of Computer Science.

[3] Suri, P.K. , and Singh, M . (2010)" *An Efficient Decentralized Load Balancing Algorithm for Grid*," IEEE 2nd International Advance Computing Conference, March.

[4] Gizem, Aksahya & Ayese, Ozcan (2009) *Communications & Networks*, Network Books, ABC Publishers.

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

[6] Werstein , paul., Situ, Hailing. and Huang, Zhiyi.(2006) "*Load Balancing in a Cluster Computer*," in Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06), pp. 569-577

[7] Talukder, A. K. and Yavagal , R.R. (2006) "*Mobile Computing: Technology, applications and service creation*", Tata McGraw Hill.

[8] Bharadwaj, V., Ghose , D. , G. Robertazzi, T.(2003) *"Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems"* Springer Berlin Heidelberg. January.

 

[9] Ian Foster. (2002) "what is the grid"? A three point checklist, Grid Today, vol.1, pp.6-12, 2002

[10] Casauant, T.L., Kuhl, J. G.,(1988) "*A Tamnomy of Scheduling in General-Purpose Distributed Computing Systems*," Institute of Electrical and Electronics Engineer Transaction on Software Engineering, vol. 14(1l), pp. 1578-88, November .

 