International Journal of Advanced Science and Research
ISSN: 2455-4227
Impact Factor: RJIF 5.12
www.allsciencejournal.com
Volume 3; Issue 2; March 2018; Page No. 100-104

A priority based workflow management in grid computing system

Kamal Vohra[1], Dr. Vikram Bali[2], Dr. KK Paliwal[3]
[1] Student, Computer Science and Engineering, Panipat Institute of Engineering and Technology, Panipat, Haryana, India
[2, 3] Computer Science and Engineering, Panipat Institute of Engineering and Technology, Panipat, Haryana, India

Abstract
Grid workflow is executed on diverse resource whose interactions are highly complicated and hardly predicted. Often the user and the workflow middleware services want to be informed about the performance behaviour of the workflows, as early as possible, so that they can steer the execution of the workflow to compensate the performance analysis service that supports tracing execution, analyzing performance overhead in workflows. We present how the user and the Grid workflow middleware can utilize the distributed performance analysis service in order to optimize the execution of workflows.

Keywords: cloud computing, genetic algorithm, scheduling, task scheduling

1. Introduction
The Grid is emerging as a wide-scale, distributed computing infrastructure that promises to support all the resource sharing and coordinated problem solving in dynamic, multi-institutional Virtual Organization [6]. Such an approach to network computing is known by several names: Meta Computing, Scalable Computing, Global Computing, Internet Computing, and more recently Peer-to-Peer Computing. With the growth in the amount of data to be computed, computational problems are getting more and more complex and costly in terms of time needed. Grid computing can integrate and utilize heterogeneous [7, 8] computing resources which are connected through networks without the limitation of geography. Thus grid computing is widely used to solve large-scale computational problems. Unlike traditional cluster computing, computational capabilities of resources in grid computing environments are usually different. A load balancing algorithm [2] attempts to improve the response time of user's submitted applications by ensuring maximal utilization of available resources. The main goal is to prevent, if possible, the condition where some processors are overloaded with a set of tasks while others are lightly loaded or even idle. Although load balancing problem in conventional distributed systems has been intensively studied, new challenges in Grid computing still make it an interesting topic and many research projects are under way. This is due to the characteristics of Grid computing and the complex nature of the problem itself. Load balancing algorithms in classical distributed systems, which usually run on homogeneous and dedicated resources, cannot work well in the Grid architectures. Grids has a lot of specific characteristics, like heterogeneity, autonomy, scalability, adaptability and resources computation-data separation, which make the load balancing problem more difficult. Grid's resources may be located at distant places. The main advantage of Grid computing in terms of its characteristics is that we can connect different resources which are a far distinct placed from each other. Resources in a grid belong to many different organizations that allow other organizations (i.e. users) to access them. Non local resources can thus be used by applications, promoting efficiency and reducing costs. By sharing the resources we save the idle time of the resources by providing jobs automatically to the resource which are kept free from a longer time. Load balancing let the resources to be shared. Sharing of resources provide a better way to provide the good synchronization between the different resources so that the idle time of Processor should be saved. A grid must be built with standard services, protocols and inter-faces thus hiding the heterogeneity of the resources while allowing its scalability. Without such standards, application development and pervasive use would not be possible. Since the number of users within a system is increased, new security mechanisms are needed to ensure that malicious code cannot legitimate services running on the grid. As the number of users increased, there is also need to improve the security system. A grid federates a large number of resources contributed by individual machines into a greater total virtual resource. For applications that are grid-enabled, the grid can offer a resource balancing effect by scheduling grid jobs on machines with low utilization. This feature can prove invaluable for handling occasional peak loads of activity in parts of a larger organization. Grid is a large, dynamic, heterogeneous and collaborative environment. Grids integrate networking, communication, computation and information to provide a virtual platform for computing and data management. Machines in a Grid are typically grouped into autonomous administrative domains that communicate via high-speed communication links. Scheduling, in such an environment, requires a different approach than presently used in distributed systems [3]. This approach must take into account that resources in a Grid typically do not belong to the same administrative domain. Therefore, the individual demands of the participants need to be observed. This requires a new technological approach. Access to resources is typically

subject to individual access, accounting, priority, and security policies of the resource owners. [1] Moreover a Grid resource is a very general concept, it is not restricted only to CPU cycles, but also network bandwidth, disk space, memory, files, etc. In a broad sense a Grid resource is anything that may be used through a standard Grid interface, i.e. an Application Programming Interface and a protocol. In a Grid environment there is the need of a set of basic functions and services that take care of all these constraints that means accepting requests for resources and assign specific machine resources to a request from the overall pool of Grid resources for which a user has access permission. This set of services is called Grid Resource Management System (GRMS). The RMS manages the pool of resources that are available to the Grid, i.e. the scheduling of processors, network bandwidth, and disk storage. In a Grid, the pool can include resources from different providers thus requiring the RMS to hold the trust of all resource providers. Basically, the RMS is constituted by the middleware, tools and services that allow disseminating resource information discover suitable resources and scheduling resources for job execution. The designs of a RMS have to consider aspects such as: site autonomy, heterogeneity, extensibility, allocation/co-allocation, scheduling, and online control. The solution of complex problems can require various kinds of resources located on different sites that can use different operating systems as well as different local resource management systems which lead to significant differences in functionality. A resource management solution must support the frequent development of new domain-specific management structures, without requiring changes to cod installed at participating sites. Applications have computational requirements that can be satisfied by using one or more resources that could be allocated simultaneously at several sites. Site autonomy and possibility of failure during allocation introduces a need for specialized mechanisms for allocating resources, initiating computation on those resources, and monitoring and managing those computations.

2. Problem Formulation

With the advent of Grid and application technologies, scientists and engineers are building more and more complex applications to manage and process large data sets, and execute scientific experiments on distributed resources. Such application scenarios require means for composing and executing complex workflows. The grid contains tasks which execute on heterogeneous and distributed resources. So handle this type of Problems workflow management is required. The main significance of Priority Based Technique is to handle the order of execution of workflow is to make the execution takes place faster. Many efforts have been made towards the development of workflow management systems for Grid computing.

3. Objectives

Objectives of Dissertation are to divide the Workflow into tasks and tasks are assigned priorities. We have used the static values for workflow and produced a static model for workflow management. 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. The proposed algorithm will represent workflow in DAG then assign the priority to each level DAG.

There are three type of algorithm for this:
Static algorithm
Dynamic algorithm
Proposed Priority Calculation Method

The proposed load balancing schemes/methodologies for distributed system focus on centralized based method only, ignoring decentralization. Load balancing algorithms for decentralized based method need to be investigated.
Techniques developed should be for the decentralized approaches also by the use of different types of dynamic load balancing algorithms [4].
New schemes with higher scalability and efficiency in terms of communication overhead, heterogeneity need to be developed for the distributed systems. Selecting the appropriate load balancing policy for distributed systems is fundamental to providing load balancing in distributed system. However, the decision depends on the computation, storage and communication capability of the devices.
Proposed Priority Calculation Method**:** The method has two phases namely, compile-time phase and run-time phase. The Compile-time phase of the algorithm has three stages, such as level sorting, task prioritization and processor selection.
The main objective is to propose a priority based workflow management algorithm that can handle heterogeneous workflow.

4. Present Work

A. Proposed Algorithm

The proposed algorithm for adaptive load balancing in computational grid is as follows:
Assumptions:

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. Rate of data transfer between the processors: R (p×p)

B. Algorithm

1. Begin
2. Read the DAG, associated attributes values, and the number of processor p;
3. Level sort the given DAG;
4. For all task vkin the DAG do
5. Begin
6. Compute ULC, DLC and ACC values for the task vk;
7. Compute DLC (vj) = Max {Data (vi, vj)}, where vi Î pred (vj).
8. Compute ULC (vj) = Max {Data (vj, vk)}, where vkÎ succ (vj).
9. Compute LC (vk) = max{LC (vj)} + ULC (vk) + DLC (vk), where vjÎbpred (vk);
10. Insert the task into the priority queue based on the LC value such that the tasks in Lower level are placed in the

priority queue first than the tasks in the higher level and tie if any, is broken using the ACC value.

C. Proposed system model

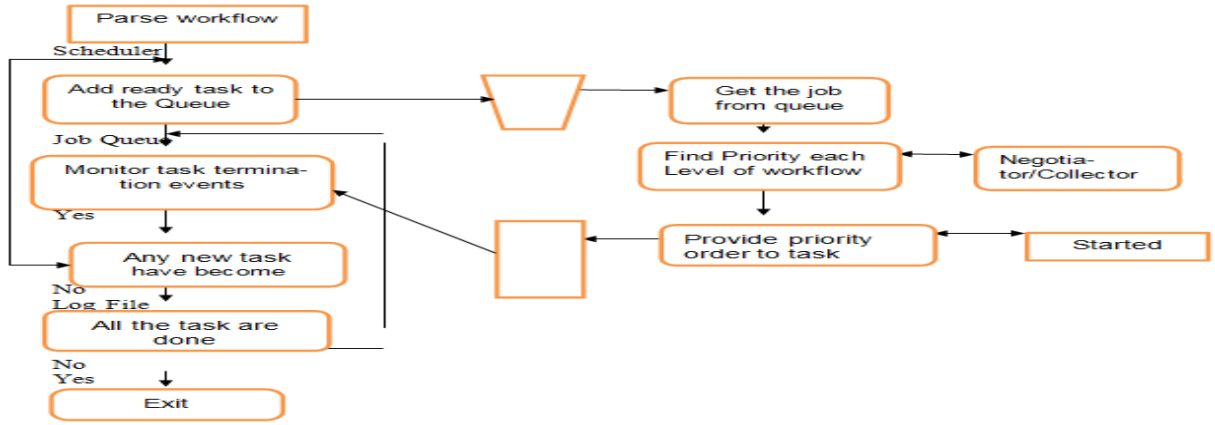

Fig 1: Sequence Diagram

D. Methodology

Basic Requirements:
Software or Resource Version Required
NetBeans IDE version 6.9 or higher
Java Development Kit (JDK) version 6 or version 7
Here the basic parameters of the proposed work are presented respective to the simulation environment. The system is implemented in java with the help of NetBeans. NetBeans is an IDE for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML. It is also an application platform framework for Java desktop applications and others.
The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM.
The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeansPlatform (including the NetBeans IDE itself) can be extended by third party developers.
Setting up the project:

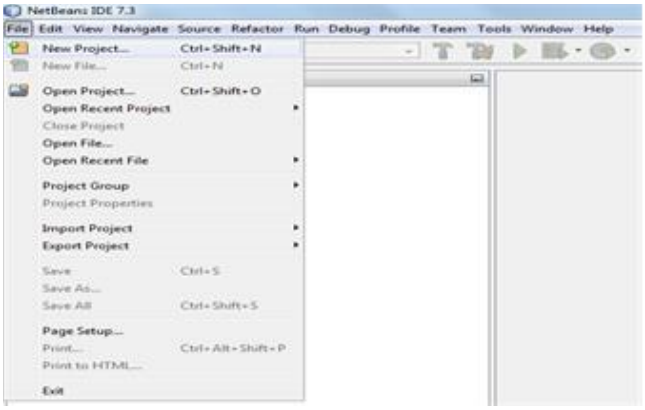

Fig 2: To Create a New Project

To create an IDE project

1. Start NetBeans IDE
2. In the IDE, choose File > New Project (Ctrl-Shift-N)
3. In the New Project wizard, expand the Java category and select Java Application. Then click Next.
4. In the Name and Location page of the wizard, do the following:
   In the Project Name field, type Network.
   Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
   In the Create Main Class field, type network.Network.
5. Click Finish.

The project is created and opened in the IDE. You should see the following components:
The Projects window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
The Source Editor window with a file called Network open.
The Navigator window, which you can use to quickly navigate between elements within the selected class.

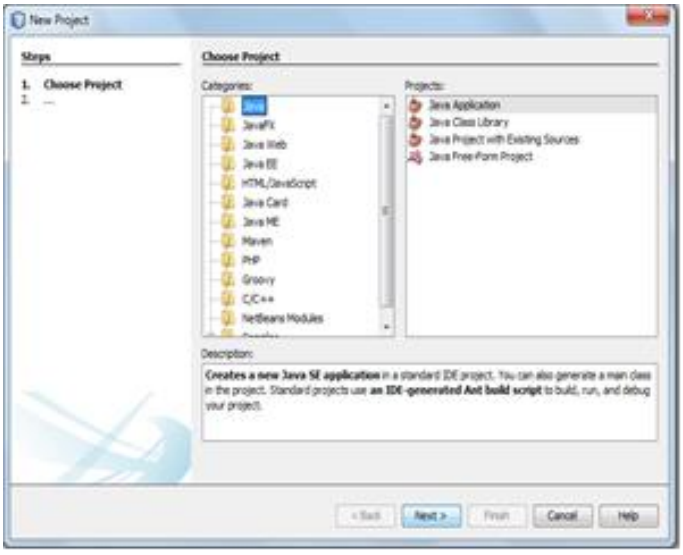

Fig 3: New Project wizard

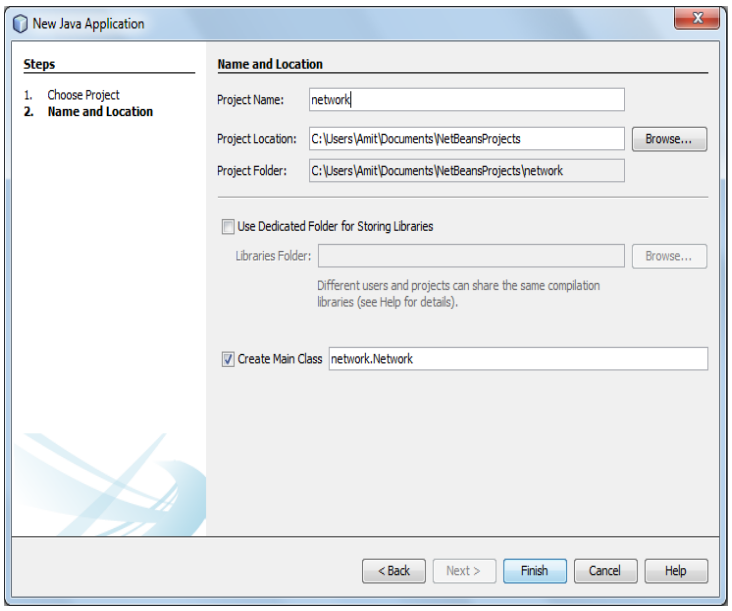

Fig 4: Name and Location page of wizard

5. Results

A. Parameters

Following parameters are used during simulation of priority calculation algorithm:

Table 1: Execution environment

Execution Environment	Java Net Beans 6.9

Table 2: Simulation parameter taken

No. of Workflow	**1**
Workflow Division in task	10

The table 1 has shown the execution environment java net beans 6.9. The table 2 showed the simulation parameter. The task at top level of workflow is assigned level 1. All the tasks that becomes ready for execution when the tasks at level 1 complete are assigned level 2 and so on. All the tasks at level l having a predecessor at level l-1.Tasks at the same level are not dependent of each other. In this context a parallel application is generally modeled by a precedence-constrained task graph, which is a directed acyclic graph (DAG) with node and edge weights (the nodes represent the tasks and the directed edges represent the execution dependencies as well as the amount of communication).

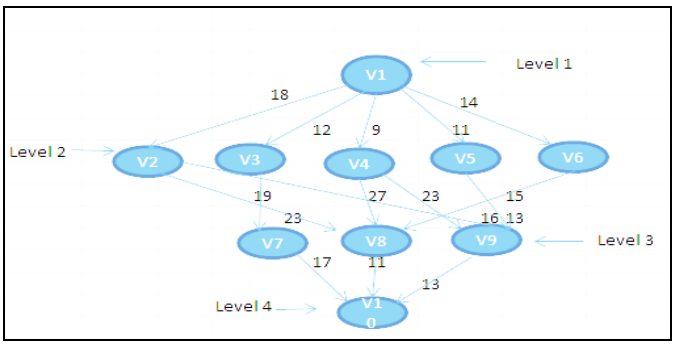

Fig 5: Workflow representations with help of DAG

The figure 5 represented workflow with the help of DAG. A Priority based Static Model is proposed to deal with the above problem. As shown in fig 5, 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. The table 4 shows Task List Sorted by level. We find out the priority at each level of DAG based on parameters such as DLC, ULC& LC. As shown in Table 4 V1 contains only one Task so priority at this level will be one similarly at next level it contains tasks such as V2, V3, V4, V5 & V6 so after applying the algorithm it sort out the order of execution at this level and same for next level. The order of execution will depends on priority order of Task.

Table 3: 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

6. Conclusion and Future Work

In this paper, priority based task scheduling [5, 6] 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. We have used the static values for workflow and produced a static

model for workflow management. 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. The future work can be extended to priority based dynamic workflow management. In dynamic workflow management the values of workflow would be computed dynamically, i.e. during the execution of jobs and thus would produce a total automated solution for workflow management. Also the distributed environment can be another prominent area where our algorithm holds the possible impact and use. This research work can be extended for priority based cloud computing. Where we have to manage large clouds and priority assignment is complicated task. So we have a vast and deep future scope of our present work.

Table 4: 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

7. References

1. Hotovy S, Schneider D, O'Donnell T. Analysis of the early workload on the cornell theory, ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems. 1996; 24(1):272-273.
2. Abderezak T, Hussien A, Challal, Y. Performance Evaluation of Load Balancing in Hierarchical Architecture for Grid Computing Service Middleware**,** IJCSI International Journal of Computer Science Issues March Foster. 2011; 8(2):1.
3. Casauant TL, Kuhl JG. A Tamnomy of Scheduling in General-Purpose Distributed Computing Systems, Institute of Electrical and Electronics Engineer Transaction on Software Engineering. 1988; 14(1l):1578-88.
4. Han Zhao, Xinxin Liu, Xiaolin Li. DLBEM: Dynamic Load Balancing Using Expectation-Maximization, IEEE, 2008.
5. Hu J, Marculescu R. Energy-Aware Communication and Task Scheduling for Network on-Chip Architectures under Real-Time Constraints, in Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, 2004.
6. Ilavarasan E, Thambidurai P. Low complexity performance effective task scheduling algorithm for heterogeneous computing environments". Journal of Computer Science, Science Publications. 2007; 3(2):94-103.
7. Iverson M, Ozguner F, Follen G. Parallelizing Existing Applications in a Distributed Heterogeneous Environments, Proc. Heterogeneous Computing Workshop, 1995, pp. 93-100.
8. Huang J, Lee SY. A Theoretical Approach to Load Balancing of a Target Task in a Temporally and Spatially Heterogeneous Grid Computing Environment Springer Berlin Heidelberg, 2002.