A Review: Scheduling in Grid Environment
Dilpreet Kaur, Balwinder Singh
M.Tech (Final year student), Asst. Professor
Yadwindra College of Engg. and Technology, Talwandi Sabo, Punjab

Abstract: Grid computing came into being and is currently an active research area. One motivation of Grid computing is to aggregate the power of widely distributed resources, and provide non-trivial services to users. To achieve this goal, an efficient Grid scheduling system is an essential part of the Grid. Rather than covering the whole Grid scheduling area, this survey provides a review of the subject mainly from the perspective of scheduling algorithms. In this review, the challenges for Grid scheduling are identified. Then various Grid scheduling algorithms are discussed from different points of view, such as static vs. dynamic policies, objective functions, applications models, adaptation, QoS constraints, and strategies dealing with dynamic behavior of resources, and so on.

Keywords: Scheduling, Grid, Local, Global, Static, Dynamic, Heuristic.

I. INTRODUCTION
The popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we use computers today. These technical opportunities have led to the possibility of using geographically distributed and multi-owner resources to solve large-scale problems in science, engineering, and commerce. Recent research on these topics has led to the emergence of a new paradigm known as Grid computing. To achieve the promising potentials of tremendous distributed resources, effective and efficient scheduling algorithms are fundamentally important. Unfortunately, scheduling algorithms in traditional parallel and distributed systems, which usually run on homogeneous and dedicated resources, e.g. computer clusters, cannot work well in the new circumstances.

A. Overview of the Grid Scheduling Problem
A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. It is a shared environment implemented via the deployment of a persistent, standards-based service infrastructure that supports the creation of, and resource sharing within, distributed communities. Resources can be computers, storage space, instruments, software applications, and data, all connected through the Internet and a middleware software layer that provides basic services for security, monitoring, resource management, and so forth. Resources owned by various administrative organizations are shared under locally defined policies that specify what is shared, who is allowed to access what, and under what conditions. The real and specific problem that underlies the Grid concept is coordinate resource sharing and problem solving in dynamic, multi-institutional virtual organizations. From the point of view of scheduling systems, a higher level abstraction for the Grid can be applied by ignoring some infrastructure components such as authentication, authorization, resource discovery and access control. Thus, the following definition for the term Grid adopted: “A type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous and heterogeneous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.”

To facilitate the discussion, the following frequently used terms are defined:

- A task is an atomic unit to be scheduled by the scheduler and assigned to a resource.
- The properties of a task are parameters like CPU/memory requirement, deadline, priority, etc.
- A job (or metatask, or application) is a set of atomic tasks that will be carried out on a set of resources. Jobs can have a recursive structure, meaning that jobs are composed of sub-jobs and/or tasks, and sub-jobs can themselves be decomposed further into atomic tasks. In this paper, the term job, application and metatask are interchangeable.
- A resource is something that is required to carry out an operation, for example: a processor for data processing, a data storage device, or a network link for data transporting.
- A site (or node) is an autonomous entity composed of one or multiple resources.
- A task scheduling is the mapping of tasks to a selected group of resources which may be distributed in multiple administrative domains.

II. TAXONOMY OF SCHEDULING IN GRID
Casavant et al’s hierarchical taxonomy [1] for scheduling algorithms in general purpose parallel and distributed computing systems will be suitable for grid computing as grid is also a form of parallel and distributed system. The taxonomy is given in the figure 1.

Local vs. Global Scheduling
In general, at the highest level, scheduling algorithms can be classified into local and global scheduling algorithms. Local scheduling algorithms deal with allocation and execution of the processes that are resident on a single CPU. On the other hand, information about the system or resources is used to allocate the processes to multiple processors in global scheduling algorithms. As grid computing deals with coordination of multiple processors
that are geographically distributed, grid scheduling falls under global scheduling, Fig. 1.

![Hierarchical Taxonomy of Scheduling Algorithms](image)

**Fig. 1** Hierarchical Taxonomy of Scheduling Algorithms [1]

### III. STATIC VS. DYNAMIC SCHEDULING

In the case of static scheduling, all the information regarding the tasks and resources such as execution time of the tasks, speed of the processor are available by the time the application is scheduled. In this type of scheduling, it is easy to program from the scheduler’s point of view. But in the case of dynamic scheduling, the execution time of the tasks may not be known due to the direction of branches, number of iterations in the loop etc. So, the task has to be allocated on the fly as the application executes. Both static and dynamic scheduling are widely adopted in the grid. Here, system need not be aware of the run time behavior of the application before execution. Static scheduling will be useful for predictive analyzes, impact studies, postmortem analyzes etc. Here it is assumed that each machine executes a single task at a time in the order in which the tasks are assigned and the size of the meta task and the number of machines in the heterogeneous computing environment are static and are known priori. Static scheduling algorithm uses a cluster of processors for critical path tasks which is useful where communication costs are not arbitrarily heterogeneous. Fault tolerant static scheduler that uses task duplication for grid applications. Application of swarm intelligence such as Ant Colony Optimization to grid scheduling, Dynamic scheduling algorithms should be able to solve job failure, unexpected arrival of high priority jobs, resource failure, activation of new resources, varying workload on resources, changing resource properties, changing priority or deadline of the job etc. Dynamic scheduling algorithm for grid has been developed by incorporating the prediction strategy into a static scheduling algorithm. Wasp colony’s interaction with environment to dynamic job shop scheduling has been applied and a waps based scheduling decision mechanism has been constructed. Scheduling policies for desktop grid systems involving different levels of heterogeneity that utilize the solution to a linear programming problem. Parallel applications consisting of independent tasks have been considered here. Aline P. Nascimento et al has focused on scheduling policies of an application management system which is embedded into MPI applications. There are applications with large number of lightweight jobs. The overall processing of these types of applications involves high overhead time and cost in terms of job transmission to and from grid resources and job processing at the grid resources. To overcome this difficulty, a dynamic job grouping mechanism based on the processing requirements of each application, grid resources availability and their processing capability.

### IV. SCHEDULERS FOR GRID ENVIRONMENT

A number of tools have been developed for scheduling in grid computing systems. Some of the tools include Nimrod-G, Condor-G, GRaDS, Legion, NetSolve, Sun Grid Engine etc.

#### A. Nimrod-G

Nimrod was the first tool to use heterogeneous resources in a grid which was created as a research project funded by the Distributed Systems Technology Centre. Nimrod-G is a grid aware version of Nimrod which takes advantage of the features such as automatic discovery of resources in globus toolkit. It is used for automated modeling and execution of parameter sweep applications in grid. It uses resource management and scheduling algorithms based on economic principles and also supports user defined deadline and budget constraints. The components of Nimrod-G include client or user station, parametric engine, scheduler, dispatcher and job-wraper [2]. Client or User Station acts as an user interface for controlling an experiment under consideration. It is also possible to run multiple instances of the same client at different locations. Parametric engine is a central component that controls the entire experiment. It also maintains the state of the whole experiment and it is recorded in persistent storage so that it allows the experiment to be restarted if the node running nimrod goes down. Next comes the scheduler which is responsible for resource discovery, resource selection and job assignment. According to scheduler’s instruction, the dispatcher initiates the execution of a task on the selected resource. Job wrapper starts execution of the task on the assigned resource and sends the result back to the parametric engine via dispatcher. David Abramson et al in [3] has used Nimrod-G to manage all operations associated with remote execution including resource discovery, trading, scheduling etc.

#### B. Condor-G

Condor is a specialized workload management system for compute intensive jobs that provides a job queuing mechanism, scheduling policy, priority scheme, and resource monitoring and resource management. It is used to manage a cluster of dedicated compute nodes and also exploits wasted CPU power from idle desktop workstations. It can be used to seamlessly combine all of an organization’s computational power into one resource. It is the product of Condor Research Project at the University of Wisconsin-Madison and was first installed...
as a production system in the UW-Madison Department of Computer Sciences. Condor incorporates many of the emerging grid-based computing methodologies and protocols. Condor-G is fully interoperable with resources managed by Globus [4]. When a user submits a job to the Condor, it is executed on a remote machine within the pool of machines available to Condor where security of the remote machines is preserved by Condor through remote system calls. Condor can be useful on a range of small to large network sizes. On a single machine, it pauses the job when the user uses the machine for other purposes, and it restarts the job if the machine reboots. On a small dedicated cluster, it functions as a cluster submission tool. James Frey et al in [5] has used Condor G for handling job management, resource selection, security and fault tolerance.

C. GRaDS

Grid Application Development Software (GRaDS) Project with support from NSF Next Generation Software Program has been developing tools for construction of applications on the grid easier. This led to the development of a prototype software infrastructure called GrADSoft that runs on top of Globus and facilitates scheduling, launching and performance monitoring of tightly coupled Grid applications. In GrADS, the end user just submits their parallel application to the framework for execution. The framework schedules the application to appropriate set of resources, launching and monitoring the execution and also rescheduling the applications on different set of resources if necessary. Anirban Mandal et al has launched and executed EMAN, a Bio imaging workflow application onto the grid [6]. F.Berman et al in [7] has presented an extension to GrADS software framework for scheduling workflow computations that has been applied to a 3-D image reconstruction application etc.

D. Legion

Legion is an object based, meta systems software project at the University of Virginia that began in late 1993. It has been created to address key issues such as scalability, programming ease, fault tolerance, site autonomy, security etc. It was also designed to support large degrees of parallelism in application code and manage the complexities of the physical system for the user. It also allows applications developers to select and define system-level responsibility. Anand Natrajan et al in [8] has presented a grid resource management of legion for scheduling all compute objects as well as data objects on machines whose capabilities match the requirements, while preserving site autonomy as well as recognizing usage policies.

E. NetSolve

The NetSolve system from the University of Tennessee’s Innovative Computing Laboratory was to address the ease of use, portability and availability of optimized software libraries for high performance computing. It enables users to solve complex scientific problems remotely, by managing networked computational resources and using scheduling heuristics to allocate resources to satisfy the requests [9]. The Primary goal of NetSolve was to make an easy access to grid resources. NetSolve, which is a client-agent server system provides remote access to hardware as well as software resources. The agent maintains a list of all available servers and performs resource selection for client requests and also ensures load balancing of the servers. Even though locating appropriate resources to the request is a challenge in grid computing, the NetSolve agent uses knowledge of the requested service, information about the parameters of the service request from the client, and the current state of the resources to get possible servers and return the servers in sorted order [10].

F. Sun Grid Engine

Sun Grid Engine (SGE) is the foundation of Sun Grid Utility Computing system, made available over Internet in the United States in 2006, later available in many other countries. It is used on high performance computing cluster is used for accepting, scheduling, dispatching and managing remote and distributed execution of large numbers of standalone or parallel user jobs. It also schedules the allocation of distributed resources such as processors, memory, disk space etc. Some of the features of SGE include advance reservation of resources, multi clustering, job submission verifier on both client and server sides, topology aware scheduling, job and scheduler fault tolerance etc. Goncalo Borges et al in [11] has presented a work developed to integrate SGE with the EGEE (Enabling Grids for E-Science) middleware. EGEE is the world’s largest operating grid infrastructure serving thousands of multi science users with robust, reliable and secure grid services worldwide.

V. GRID SIMULATION TOOLS

As grid computing is more loosely coupled, heterogeneous and geographically dispersed, getting all these resources working together for a single scientific or business problem may be difficult for research purpose which requires repeated evaluation of some strategies. Here comes the simulators which provide users with practical feedback when developing real world systems. This allows the developer to determine correctness and efficiency of the proposed system before it is actually constructed and also overall cost of developing the real system also diminishes. There are many simulation tools for grid computing such as simgrid, gridsim, optorsim, bricks etc available for evaluating applications and network services for grid systems.

A. SimGrid

The SimGrid project was started in 1999 which provides core functionalities for simulation of distributed applications in heterogeneous distributed environments. The main aim of the project was to facilitate research in the area of distributed and parallel application scheduling on distributed computing platforms that were ranging
from simple network of workstations to computational grids. SimGrid v1 prototyped scheduling heuristics and SimGrid v2 extended the capabilities of its predecessor by transitioning from a wormhole model to analytical one. Application Programming Interface was also added to study non-centralized scheduling and other kinds of concurrent sequential processes. GRAS (Grid Reality and Simulation) API was added in v3.0 where distributed applications can be developed within the simulator. Henri Casanova in [12] has used Simgrid for the study of scheduling algorithms for distributed application.

B. GridSim

Since in grid environment, resources and users are distributed across multiple organizations with their own policies, it is impossible to perform scheduler performance evaluation in a repeatable and controllable manner, Rajkumar Buyya et al. has developed Java based discrete-event grid simulation toolkit [13]. The toolkit allows modeling and simulation of entities in parallel and distributed computing systems users, applications, resources and resource broker for design as well as evaluation of scheduling algorithms. Some of the functionalities of gridSim include incorporating failures of grid resources during runtime, supporting advance reservation of a grid system, incorporating auction model, incorporating extension of data grid into GridSim, incorporating network extension into GridSim etc.

C. OptorSim

A grid simulator designed to test dynamic replication strategies and appropriate scheduling of jobs was developed as a part of European Data Grid project. OptorSim which has the structure of EDG, includes the following elements to achieve a realistic simulated environment. These include storage resources where data can be kept, computing resources to which jobs can be sent, scheduler to decide to which resource the job has to be sent, the network which connects the sites and finally replica management. It also incorporates peer to peer messaging system which is used by some of the optimization algorithms for conducting auctions [14].

D. Bricks

It is a java based performance evaluation system for scheduling algorithms and frameworks of high performance global computing systems. It consists of a scheduling unit that allows simulation of various behaviors of resource scheduling algorithms, programming modules for scheduling, processing schemes for networks and servers etc. Users can also construct and alter the script using building bricks within the script for testing and evaluating simulations [15].

VI. APPLICATIONS

There are many scientific problems that require grid environment to get solved. It provides an environment to solve problems in physics, chemistry, nuclear fusion, earth science, space, human health, agriculture, medicine, education, research etc. In medical and biomedical fields, grid computing is useful in digital x-ray image analysis, radiation therapy simulation and protein folding. In chemistry, problems related to quantum chemistry, organic chemistry and polymer modeling makes use of grid computing. In physics, high energy physics, theoretical physics, lattice calculations, combustion and neutrino physics use grid environment.

VII. CONCLUSION

This paper has given a detailed study on scheduling with its evolution, application and importance. Also, in this paper, a study on taxonomy of scheduling has been given in different perspectives and the performance of scheduling algorithms has been discussed. The software that support scheduling in real grid environment as well as in simulated environment are also given. Thus this paper gives a detailed survey on scheduling with its applications. It is observed that heuristic based algorithms and in particular, population based heuristics are most suitable for scheduling the tasks in the grid environment. But there are population based heuristics which are complex in nature and takes a long execution time. For instance, ant colony optimization, when run in a normal PC, it takes hours to execute an algorithm to schedule more than 1000 processes. This algorithm even though gives better results compared to other population based heuristics such as genetic algorithm, due to its longer execution time of the algorithm, some other algorithm which executes faster has to be considered. Particle Swarm Optimization, Frog Leap Algorithm are some of the population based heuristics which can be used for scheduling in grid.

REFERENCES


