Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Scheduling in Grid Computing Environment

Ramanjyot Kaur* *SBBSIET(CSE Deptt)* Punjab, India
Tajinder Kaur *SBBSIET (IT Deptt)* Punjab, India
Harpreet Kaur *SBBSIET (CSE Deptt)* Punjab, India

Abstract— Grid computing is a form of distributed computing that involves coordinating and sharing computing, application, data storage or network resources across dynamic and geographically dispersed organizations. To achieve the promising potentials of tremendous distributed resources, effective and efficient scheduling algorithms are fundamentally important. The goal of grid task scheduling is to achieve high system throughput and to match the application needed with the available computing resources. This is matching of resources in a non-deterministically shared heterogeneous environment. The complexity of scheduling problem increases with the size of the grid and becomes highly difficult to solve effectively. To obtain good methods to solve this problem many algorithms have been developed. In this paper we consider all tasks scheduling algorithms for grid computing environment.

Keywords- Grid Computing, Scheduling, Makespan.

I. INTRODUCTION

Computational Grids are a new trend in distributed computing systems. They allow the sharing of geographically distributed resources in an efficient way, extending the boundaries of what we perceive as distributed computing. Various sciences can benefit from the use of grids to solve CPU-intensive problems, creating potential benefits to the entire society. With further development of grid technology, it is very likely that corporations, universities and public institutions will exploit grids to enhance their computing infrastructure. Task scheduling is an integrated part of parallel and distributed computing. Intensive research has been done in this area and many results have been widely accepted. With the emergence of the computational grid, new scheduling algorithms are in demand for addressing new concerns arising in the grid environment. In this environment the scheduling problem is to schedule a stream of applications from different users to a set of computing resources to maximize system utilization. This scheduling involves matching of applications needs with resource availability.

II. GRID COMPUTING

Grid is defined as a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements [6]. Grid technology is a growing information technology, where that the main purpose of grid is to build a kind of dynamic, distributed and heterogeneous computing environment and realize collaborative resource sharing and problem solving in dynamic and multiple virtual organizations [5]. In Figure 1 the grid computing environment is shown.

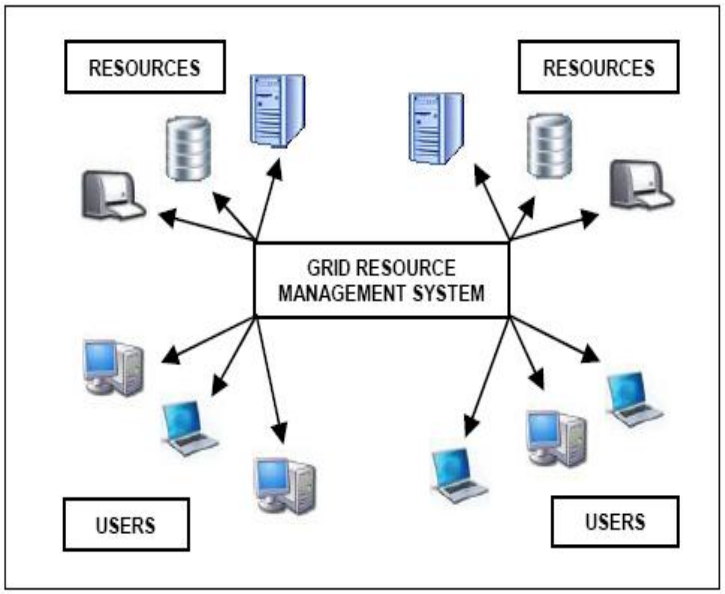

Figure 1: Grid Computing Environment

Grid computing enables large-scale resource sharing and provides a promising platform for executing tasks efficiently. In order to get optimal resource utilization and avoid overload, task scheduling with load balancing in grid computing is of interest to both users and grid systems. Load balancing mechanism aims to distribute workload evenly among computing nodes and minimize the total task execution time [3]. However, the scheduling problem with load balancing is usually computationally intractable. Grid computing is a technology that represents a kind of distributed computing. This approach is mainly used in the context of resource utilization and sharing across geographical boundaries. It helps in coordinating computing capabilities, data storage and network resources across dynamically dispersed organizational structures. In grid computing system, resources are not under the central control and can enter and leave the grid environment at any time. An effective grid resource management with good job and resource scheduling algorithm is needed to manage the grid computing system. The algorithm must consider the dynamically changes conditions in grid environment because the computational performance changes from time to time, networks connections may become unreliable, resources may join or leave the system at any time and resources may become unavailable without any notifications.

III. **GRID SCHEDULING**

Scheduling is a key concept in computer multitasking operating systems, multiprocessing operating system design and in real-time operating system design. In modern operating systems, there are typically many more processes running than there are CPUs available to run them. Scheduling refers to the way processes are assigned to run on the available CPUs [4]. This assignment is carried out by software known as a scheduler or is sometimes referred to as a dispatcher.

Scheduling includes two aspects: the task mapping and task scheduling. Task mapping is the task of machine matching, match is a logical process, not to transmit; based on task mapping results, tasks scheduling is the process that sent the task to the execution queue of designated machine [7]. The tasks of the grid can be divided into compute-intensive, correspondent intensive and compute correspondent relatively balanced [3]. Grid Scheduling is the mapping of jobs to resources as per their requirements and properties. It is proposed to solve large complex problems. Grid scheduling is an intelligent algorithm capable of finding the optimal resource for processing a job [8]. The objectives of a grid scheduler are overcoming heterogeneousness of computing resources, maximizing overall system performance, such as high resource, utilization rate and supporting various computing intensive applications, such as batch jobs and parallel applications. The grid scheduler is mainly concerned with the following: CPU utilization – to keep the CPU as busy as possible, throughput - number of process that complete their execution per time unit , turnaround time - amount of time to execute a particular process and response time - amount of time it takes from when a request was submitted until the first response is produced. Task scheduling is an integrated part of parallel and distributed computing. Intensive research has been done in this area and many results have been widely accepted. With the emergence of the computational grid, new scheduling algorithms are in demand for addressing new concerns arising in the grid environment [2]. In this environment the scheduling problem is to schedule a stream of applications from different users to a set of computing resources to maximize system utilization. There are three main phases of scheduling on a grid [1]. Phase one is resource discovery, which generates a list of potential resources. Phase two involves gathering information about those resources and choosing the best set to match the application requirements. In the phase three the job is executed, which includes file staging and cleanup. In the second phase the choice of the best pairs of jobs and resources is NP-complete problem.

IV. **COMPONENTS of GRID SCHEDULING SYSTEM**

A Grid is a system of high diversity, which is rendered by various applications, middleware components, and resources. But from the point of view of functionality, we can still find a logical architecture of the task scheduling subsystem in Grid. The scheduling process in the Grid can be generalized into three stages: resource discovering and filtering, resource selecting and scheduling according to certain objectives, and job submission [6]. As a study of scheduling algorithms is the primary concern, the focus is on the second step. Based on these observations, Figure 2 depicts a model of Grid scheduling systems in which functional components are connected by two types of data flow: resource or application information flow and task or task scheduling command flow.

A. Grid Scheduler (GS)

Grid scheduler (GS) receives applications from Grid users, selects feasible resources for these applications according to acquired information from the Grid Information Service module, and finally generates application-to-resource mappings, based on certain objective functions and predicted resource performance. Grid schedulers usually cannot control Grid resources directly, but work like brokers or agents. They are not necessarily located in the same domain with the resources which are visible to them. Figure 2 only shows one Grid scheduler, but in reality multiple schedulers might be deployed, and organized to form different structures (centralized, hierarchical and decentralized) according to different concerns, such as performance or scalability.

B. Grid Information Service (GIS)

The role of the Grid information service (GIS) is to provide such information to Grid schedulers [6]. GIS is responsible for collecting and predicting the resource state information, such as CPU capacities, memory size, network bandwidth, software availabilities and load of a site in a particular period.

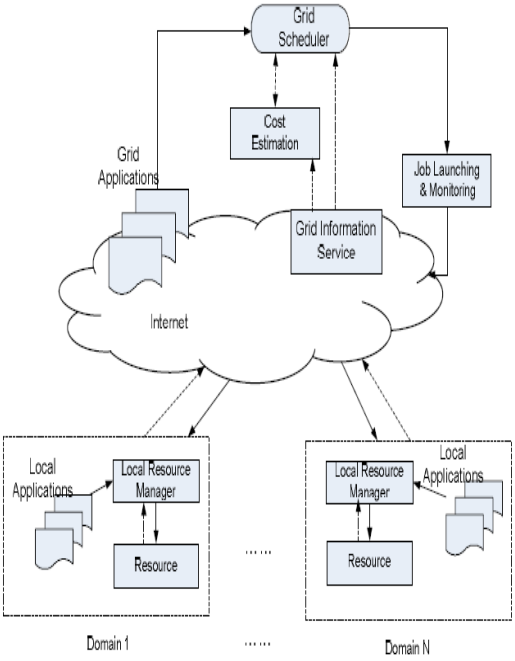

Figure 2: Components of Grid Scheduling System

C. Launching and Monitoring (LM)

The Launching and Monitoring (LM) module ,also known as the "binder" implements a finally-determined schedule by submitting applications to selected resources staging input data and executables if necessary, and monitoring the execution of the applications.

D. Local Resource Manager (LRM)

A Local Resource Manager (LRM) is mainly responsible for two jobs: local scheduling inside a resource domain, where not only jobs from exterior Grid users, but also jobs from the domain's local users are executed, and reporting resource information to GIS.

V. GRID SCHEDULING ALGORITHMS

The resource scheduling in grid is a NP complete problem. Various algorithms have been designed to schedule the jobs in computational girds. The most commonly used algorithms are OLB, MET, MCT, Min-Min, Max-Min, and MaxStd. The details of these algorithms are as follows:

A. Opportunistic Load Balancing (OLB)

The Opportunistic Load Balancing (OLB) Algorithm assigns each job in arbitrary order to the machine with the shortest schedule, irrespective of the ETC on that machine. OLB is intended to try to balance the machines, but because it does not take execution times into account it finds rather poor solutions.

B. Minimum Execution Time (MET)

Minimum Execution Time (MET) Algorithm assigns each job in arbitrary order to the machine on which it is expected to be executed fastest, regardless of the current load on that machine. MET tries to find good job-machine pairings, but because it does not consider the current load on a machine it will often cause load imbalance between the machines.

C. Minimum Completion Time (MCT)

Minimum Completion Time (MCT) Algorithm assigns each job in arbitrary order to the machine with the minimum expected completion time for the job. The completion time of a job j on a machine m is simply the ETC of j on m added to m's current schedule length. This is a much more successful heuristic as both execution times and machine loads are considered.

D. Minimum-Minimum

Minimum-Minimum Algorithm establishes the minimum completion time for every unscheduled job (in the same way as MCT), and then assigns the job with the minimum completion time (hence Min-min) to the machine which offers it this time. Min-min uses the same intuition as MCT, but because it considers the minimum completion time for all jobs at each iteration it can schedule the job that will increase the overall makespan the least, which helps to balance the machines better than MCT.

E. Maximum-Minimum

Maximum-Minimum Algorithm is very similar to Minimum-Minimum Algorithm. The minimum completion time for each job is established, but the job with the maximum minimum completion time is assigned to the corresponding machine. Max-min is based on the intuition that it is good to schedule larger jobs earlier on so they won't "stick out" at the end causing a load imbalance. However experimentation shows that Max-min cannot beat Min-min on any of the test problems used.

F. Maximum Standard Deviation (MaxStd)

In Maximum Standard Deviation Algorithm, the task having the highest standard deviation of its expected execution time is scheduled first. The task having low standard deviation of task execution has less variation in execution time on different machines and hence, its delayed assignment for scheduling will not affect overall makespan much. Moreover, the task with higher standard deviation of task execution time exhibits more variation in its execution time of different machines. A delayed assignment of such tasks might hinder their chances of occupying faster machines as some other tasks might occupy these machines earlier. It would increase the system makespan. The task having high standard deviation is assigned to the machine which has minimum completion time.

VI. **Conclusions**

The scheduling problem becomes more challenging because of some unique characteristics belonging to Grid computing. Various algorithms have been designed to schedule the jobs in computational girds. The most commonly used algorithms are OLB, MET, MCT, Min-Min, Max-Min, and MaxStd. For the complexity of tasks scheduling in grid many algorithms have been developed. These algorithms can be applied to schedule independent and dependent tasks.

References

[1] Braun, T. D., Siegel, H. J., Beck, N., Boloni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D., and Freund, R. F. (2001) 'A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems', Journal of Parallel and Distributed Computing, 61(6), pp. 810–837.

[2] Cao, J., Jarvis, S. A., et al., (2003) 'GridFlow: Workflow Management for Grid Computing', 3rd CCGrid, pp.198-205.

[3] Czajkowski, K., Fitzgerald, S. et al.,(2001) 'Grid Information Services for Distributed Resource Sharing', Tenth IEEE International Symposium on High-Performance Distributed Computing, pp.181-194.

[4] Dong, F. and Akl, S. (2007) 'A Resource Performance Fluctuation Aware Workflow Scheduling Algorithm for Grid Computing', 16th HCW in conjunction with IPDPS.

[5] El-Rewini, H., Lewis, T., and Ali, H. (1994)'Task Scheduling in Parallel and Distributed Systems'.

[6] Foster, I. and Kesselman, C. (2001) 'The Anatomy of the Grid', International Journal Supercomputer Applications, pp. 1-25.

[7] Topcuoglu, H., Hariri, S. , Wu, M. (2002) 'Performance- Effective and Low-Complexity Task Scheduling for Heterogeneous Computing', IEEE Trans. on Parallel and Distributed Systems (ITPDS), vol.13, no.3, pp.260 – 274.

[8] Yang, L., Schopf J. M., and Foster, I. (2003) 'Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments', ACM/IEEE Supercomputing, pp.31-46.

