International Journal of Contemporary Research in Computer Science and Technology (IJCRCST) e-ISSN: 2395-5325
Volume2, Issue 8 (August '2016)

STUDY ON GRID COMPUTING ENVIRONMENTS AND RESOURCE RESERVATION

S.Sivakumar,
Research Scholar,
Periyar University,
Salem, Tamilnadu.India.

Dr.D.Maruthanayagam,
Assistant Professor,
Department of Computer Science,
Sri Vijay Vidyalaya College of Arts & Science,
Dharmapuri,Tamilnadu,India.

Abstract: Grid is a distributed high performance computing paradigm that offers various types of resources (like computing, storage, communication) to resource-intensive user tasks. These tasks are scheduled to allocate available Grid resources efficiently to achieve high system throughput and to satisfy user requirements. Providing guaranteed QoS for Grid services using resource reservation and allocation is an important feature for today's service Grid. In this paper, we discussed about grid computing, resource reservation in grid and explore some existing resource reservation mechanisms for resource reservation problems employed in Grid systems.

Keywords: Grid Computing, Resource Management System (RMS), Co-Scheduling, Co-Reservation and Co-Allocation.

I.INTRODUCTION

The term Grid comes from an analogy to a power Grid. When you plug an appliance into a receptacle, you expect that you will be supplied with electricity of correct voltage, whereas you needn't care where the power comes from and how it is generated. In mid-1990s, inspired by the pervasiveness, reliability, and ease of use of electricity, Foster et al [1] began exploring the design and development of an analogous computational power Grid for wide-area parallel and distributed computing.

Since Grid computing is still emerging, the concept of Grid computing itself is evolving. The definition given by Foster [2],[1],[3] is widely accepted and frequently referred. According to Foster, Grid computing strives to aggregate diverse, heterogeneous, geographically distributed and multiple-domain-spanning resources to provide a platform for transparent, secure, coordinated, and high-performance resource-sharing and problem solving. The resources that Grid computing is attempting to integrate are various. They include supercomputers, workstations, databases, storages, networks, software, special instruments, advanced display devices, and even people.

Based on Web Service, the Open Grid Services Architecture (OGSA) [3] [6] and its associated implementation, the Globus toolkit [4] [5], are becoming the de facto standard of Grid service and Grid environment for application development, respectively.

Grid computing is a promising paradigm with the following potential advantages.

- **Exploiting underutilized resources:** Studies have shown that most low-end machines (PCs and workstations) are often idle: utilization is as low as 20% [6]. And even for servers only 50% of their capacity is utilized [7]. Grid computing provides a platform to exploit these underutilized resources and thus has the possibility of increasing the efficiency of resource usage. A simple case is that we can run a local job on a remote machine elsewhere in the Grid if the local machine is busy.
- **Distributed supercomputing capability:** The parallel execution of parallel applications is one of the most attractive features of computational Grids. A wide spectrum of applications is parallel in nature and these applications are intended to be computation-intensive. In Grid systems, there are a large number of computational resources available for one parallel application, such that different jobs within the application can be executed simultaneously on a suite of Grid resources.
- **Virtual organizations for collaboration:** Another important contribution of Grid computing is to enable the collaboration among wider-area members. Grid computing provides the infrastructure to integrate heterogeneous systems to form a virtual organization. Under the virtual organization, sharing is not limited to computational resources, but also includes various resources, such as storages, software, databases, special equipments, and so on. Furthermore, the sharing is more direct through using the uniform interfaces. Although sharing in a virtual organization is quite direct, security and local policy are guaranteed. Local resources are protected securely against those who are not authorized to access.
- **Resource balancing:** After joining a Grid, users will have a dramatically larger pool of resources available for their applications. When the local system is busy with a heavy load, part of the workloads can be scheduled to other resources in the Grid. Thus the function of resource balancing is achieved. This feature proves to be invaluable for handling occasional peak loads on a single system.
- **Reliability:** High-end conventional computing systems use expensive hardware to increase reliability. In the future, Grid computing provides a complementary approach to achieving high-reliability nevertheless with little additional investment. The resources in a Grid can be relatively inexpensive, autonomous and geographically dispersed. Thus, even if some of the resources within a Grid

encounter a severe disaster, the other parts of the Grid are unlikely to be affected and remain working well.

A) Grid System Taxonomy

Grid computing is still a very general concept. Different institutes or organizations devise different Grid systems to meet their specific needs. Grid computing can be used in a variety of ways to address various kinds of application requirements. According to the distinct targeted application realms, Grid systems can be classified into three categories. But there are actually no hard boundaries between these Grid categories. Real Grids may be a combination of two or more of these types. The three categories of Grid systems are described below.

- **Computational Grid:** A computational Grid is a system that aims at achieving higher aggregate computational power than any single constituent machine. According to how the computing power is utilized, computational Grids can be further subdivided into *distributed supercomputing* and *high throughput* categories. A distributed supercomputing Grid exploits the parallel execution of applications over multiple machines simultaneously to reduce the execution time. A high throughput Grid aims to increase the completion rate of a stream of jobs through utilizing available idle computing cycles as many as possible.
- **Data Grid:** A data Grid is responsible for housing and providing access to data across multiple organizations. Users are not concerned with where this data is located as long as they have access to the data. For example, you may have two universities doing life science research, each with unique data. A data Grid would allow them to share their data, manage the data, and manage security issues. European DataGrid [8] Project is one example of Data Grids.
- **Storage Grid :** A storage Grid attempts to aggregate the spare storage resources in Grid environments and provides users transparent and secure storage services. On the other hand, Grids can be built in all sizes, ranging from just a few machines in a department to groups of machines organized as hierarchy spanning the world. Grids can be classified into three categories according to the topology of Grid. The relationship between the three Grid topologies is illustrated in Figure 1-1.
- **IntraGrid :**A typical intraGrid topology exists within a single organization. The single organization could be made up of a number of computers that share a common security domain, which are connected by a private high-speed local network. The primary characteristics of an intraGrid are a single administrative domain and bandwidth guarantee on the private network. Within an intraGrid, it is easier to design the scheduling system, since an intraGrid provides a relatively static set of computing resources and communication capability between machines.
- **ExtraGrid:** An extraGrid couples two or more IntraGrids. The extraGrid typically involves more than one administrative domains, and the level of management complexity increases. The primary characteristics of an ExtraGrid are dispersed security, multiple domains, and remote/WAN connectivity. Within an ExtraGrid, the resources become more dynamic. A business would benefit from an ExtraGrid if there was a business initiative of integrating with external trusted business partners.
- **InterGrid:** An InterGrid has an analogy with the Internet. It is the most complicated form of Grid topology. The primary characteristics of an interGrid include dispersed security, multiple domains and WAN connectivity. A business may deem an InterGrid necessary if there is a need for a collaborative computing community, or simplified end to end processes with the organizations that will use the interGrid.

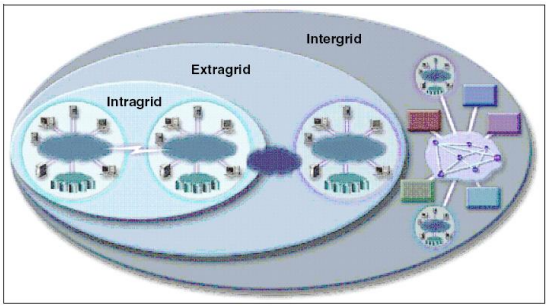

Figure 1. IntraGrid, ExtraGrid, and InterGrid

Both the targeted application realms and Grid topology will fundamentally impact the design and development of Grid systems. In this survey, we speak of scheduling in the context of computational Grids since we will concentrate on the problem of scheduling user applications for execution on a suite of computational resources.

II RESOURCE RESERVATION

Resource reservation is an important task to be performed by the Grid Resource Management System. Planning and Reservation is the process of analyzing the job and determining the resources required for successful execution of the job. Based on these results resources are reserved seamlessly to the user. Resource management is a complex task involving security and fault tolerance along with scheduling. It is the manner in which resources are allocated, assigned, authenticated, authorized, assured, accounted, and audited. Resources include traditional resources like compute cycles, network bandwidth, space or a storage system and also services like data transfer, simulation etc.

Following are the requirements that a Grid RMS (Resource Management System) must satisfy in order to perform resource reservation:

- A Grid RMS needs to schedule and control the resources on any element in the network computing system environment.
- Grid RMS should predict the impact that an application's request will have on the overall resource pool and quality of service guarantees already given to other applications.
- Grid RMS should preserve *site autonomy*. Traditional resource management systems work under the assumption that they have complete control on the resource and thus can implement the mechanisms and policies needed for effective use of that resource. But the Grid resources are distributed across separate administrative domains. This results in resource heterogeneity, differences in usage, scheduling policies and security mechanisms.
- Grid RMS must ensure Co-allocation of the resources. Co-allocation is the problem of allocating resources in different sites to an application simultaneously.

Different administrative domains employ different local resource managements systems like NQE (Network Queuing Environment), LSF (Load Sharing Facility) etc. A grid RMS should be able to interface and interoperate with these local resource management systems.

- In a Grid system resources are added and removed dynamically. Different types of applications with different resource requirements are executed on the Grid. Resource owners set their own resource usage policies and costs. This necessitates a need for negotiation between resource users and resource providers so a grid RMS should enable such negotiation.
- The resource management framework should allow new policies to be incorporated into it without requiring substantial changes to the existing code.
- The Grid RMS is also responsible for ensuring the integrity of the underlying resource and thus enforces the security of resources. The resource management system must operate in conjunction with a security manager.

II. CO-SCHEDULING, CO-RESERVATION AND CO-ALLOCATION

Meta-schedulers are characterized by receiving requests from users and schedule applications to local cluster scheduler. They also have the function of orchestrate multi-cluster configurations [10]. The use of advance reservation in multi-schedulers requires efficient algorithms to co-scheduling, co-reservation and co-allocation.

a) Co-Scheduling

Scheduling determines where and when jobs are executed and how many resources will be allocated. Co-scheduling can be understood as the ability to schedule jobs to execute at the same time at different processing nodes [10]. There are two types of co-scheduling [11], Gang and Loosely.

- Gang: schedule process of a job at the same time at different allocated nodes or processors to this job.
- Loosely: originated from NOW (Network of Workstations) environments, seeks to solve the problems to scheduling serial and parallel workload together. This approach is based on the local scheduling and on communication organization of the involved nodes.

b) Co-Reservation

The co-reservation function is characterized by the coordination use of cluster resources in sequence as a grid configuration [10]. Usually are adopted timeslot tables to show the percentage of available resources allocated during a determinate period of time. These tables help to control current allocations and future reserves [12].

c) Co-Allocation

Co-allocation can be defined as the simultaneous use of grid resources through multi-clusters [10]. According to Netto and Buyya [11], co-allocation is the resource allocation process from multiple administrative domains in a coordinated.

III. RESOURCE RESERVATION ARCHITECTURE

The Advanced reservation of resources (ARR) module enables users to require a reservation in advance resources from a cluster to be used in the future. The Figure 2 shows the insertion of the ARR module in the Cluster Manager in order to maintain the basic structure of InteGrade. The ARR is composed of service modules which are responsible for its operation. These modules are: Reservation Service, Execution Service and Notification Service. The Reservation Service deals with the management of reservation request. In other words, it verifies if there is a free time interval and reallocates the reservation if an interval requested is not available. To reach this goal, the service receives from User Node information necessary for executing an application, originally provided by InteGrade. Examples are: application type (sequential, parametric or parallel), quantity for resources to application execution, identification of application in the Application Repository and the information relating to advanced reservation and stored them in the database. The information required to advanced reservation is:

- start-time: execution start time;
- end-time: execution end time;
- user: user name to handle stored reservations;
- Email: user e-mail to notifications.

The Execution Service is responsible for requesting the beginning of application execution. Then, the service searches the execution information in the database and sends it to Cluster Manager which starts the procedures to application execution. The Notification Service warns users, by e-mail, when the process failure or in the initialization or completion of application execution. The following section shows the application execution process in InteGrade using the Advanced Reservation of Resource module.

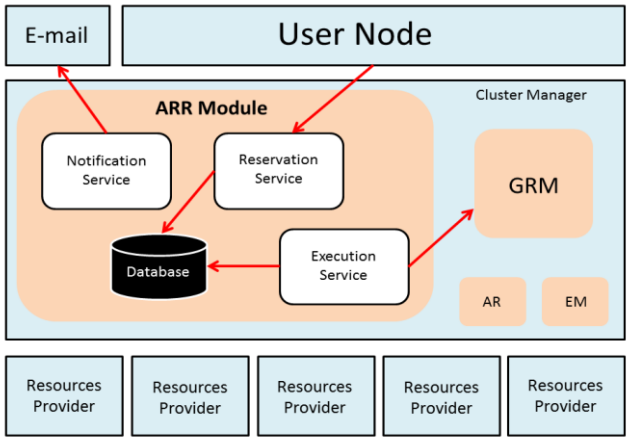

Figure 2. ARR Architecture

In standard **resource reservation schemes**, when the scheduler assigns a task to a resource, it records a reservation for that resource in its database. The resource is considered allocated to the task for an unspecified amount of time, starting at the time the task-to-resource assignment is made. As far as the scheduler is concerned, the resource remains booked for the task until the scheduler is informed about the task completion. This situation is shown in Fig. 3, where the scheduler waits for a *release message* to arrive from the resource before it can allocate the resource to a new task. To allocate the resource to a new task, the scheduler must send an *allocation message* to the

application telling the user-side where the data and code required for the execution of the task should be transferred. The data and code are then transferred to the resource, which takes some additional communication time. During the transmission of the allocation message and during the transmission of the data, the resource remains (unnecessarily) inactive. When all the data have arrived at the resource, the task begins execution. When the task is completed, the resource remains again (unnecessarily) inactive until the release message arrives at the scheduler, which can then allocate it to another task.

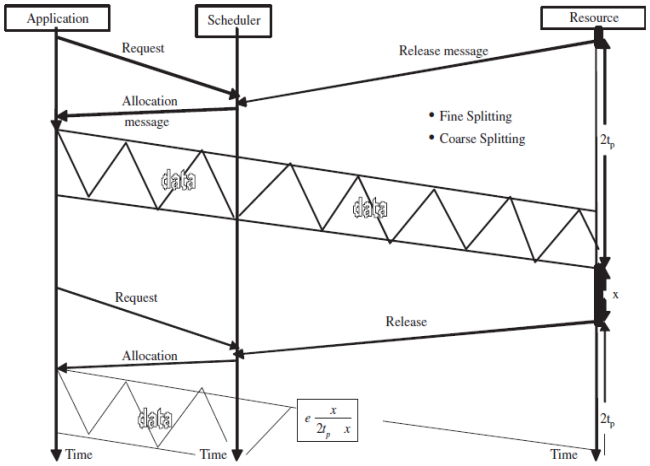

Figure 3: Possible scenario when a standard resource reservation scheme is used.

Average time denoted by 2*tp* that elapses between the time a resource sends the release message to the scheduler to inform it about the completion of a task and the time all the data required to execute the next task arrive at that resource. Also denoted the *x* is mean execution time of a task on that resource. It is then clear that when a standard resource reservation scheme is used, the efficiency with which a resource is used is at most

$$e = x / 2tp+x \qquad \text{--- (1)}$$

Note that the efficiency factor *e* may be considerably smaller than 1, and it decreases as *x* decreases or as 2*tp* increases (2*tp* is at least as large as the roundtrip propagation delay). In order for the Grid to be useful for a number of different applications, we would like to be able to use fine grain computation (where *x* is small) and also be able to use remote resources (where 2*tp* is large), both of which correspond to small values for the efficiency factor *e*. Thus, standard resource reservation algorithms fundamentally restrict the efficiency with which Grid resources are used.

IV.CONCLUSION

The Grid provides ability to access, use and manage various virtual organizations heterogeneous resources across multiple domains and institutions where requests are served from external users with local users. Due to the extremely heterogeneous and complex computing environments, availability of resources may possibly fluctuate in Grid environments. Advance reservation for grid resources allows users to gain *concurrent access* for their applications to be executed in parallel. It also guarantees the availability of resources at the specified times in the future. The goal of this paper is to understand the basic concepts of grid and resource reservation in grid then investigate various methods for designing effective and efficient resource reservation systems for Grid environments.

REFERENCES

[1]. Foster and C. Kesselman, editors. The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers, 1998.
[2]. Foster, C. Kesselman, and S. Tueche. The anatomy of the Grid. Intl. J. Super-computer Applications, 2001.
[3]. Foster, C. Kesselman, J. Nick and S. Tuecke. The physiology of the Grid: an open Grid services architecture for distributed systems integration. http://www.globus.org/research/ papers/ ogsa.pdf.
[4]. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. International Journal of Supercomputer Applications, 11(2):115-128, 1997.
[5]. Globus Project website, http://www.globus.org
[6]. Global Grid Forum, http://www.ggf.org
[7]. Fracine Berman, High-Performance Schedulers, in The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers, 1998.
[8]. W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, Data Management in an International Data Grid Project, Proceedings of the first IEEE/ACM International Workshop on Grid Computing, (Springer Verlag Press, Germany), India, 2000.
[9]. M.A.S. Netto and R. Buyya. Resource co-allocation in grid computing environments. Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications. IGI Global, 2009.
[10]. J. Qin, M.A.R. Dantas, and M. Bauer. Application programmers interactions enhancing a meta-scheduler in multi-clusters environment. Unpublished.
[11]. A.C. Sodan. Loosely coordinated coscheduling in the context of other approaches for dynamic job scheduling: a survey. Concurrency and Computation: Practice and Experience, 17(15):1725{1781, 2005.
[12]. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In Quality of Service, 1999. IWQoS'99. 1999 Seventh International Workshop on, pages 27{36. IEEE, 1999.
[13]. Selection of Best Resources and Advance Resource Reservation for Complex Jobs in Grid Computing Environment Lata Ghadavi, Viren Patel ,Mehsana, Gujarat, Volume - 5 | Issue - 1 | Jan Special Issue - 2015 | ISSN - 2249-555X

AUTHORS PROFILE

S. Sivakumar received his M.Phil Degree from Alagappa University, Karaikudi in the year 2005. He has received his M.C.A Degree from Periyar University, Salem in the year 2001. He is working as Assistant

Professor, Department of Computer Science, PGP College of Arts & Science, Namakkal, Tamilnadu, India. He is pursuing his Ph.D Degree at Periyar University. Salem, Tamilnadu, India. His areas of interest include Grid Computing and Data Mining.

Dr.D.Maruthanayagam received his Ph.D Degree from Manonmanium Sundaranar University, Tirunelveli in the year 2014. He has received his M.Phil, Degree from Bharathidasan University, Trichy in the year 2005. He has received his M.C.A Degree from Madras University, Chennai in the year 2000. He is working as Assistant Professor, Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India. He has 14 years of experience in academic field. He has published 1 book, 16 International Journal papers and 23 papers in National and International Conferences. His areas of interest include Grid Computing, Cloud Computing and Mobile Computing.