Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 4, April 2014, pg.634 – 639

REVIEW ARTICLE

A Review: Grid Computing

Ankit Punia
M.Tech Student, Department of Computer Science and Applications, M. D. University, Rohtak-124001, Haryana, India
Email Id: ankitpunia@gmail.com

Ms. Pooja Mittal
Assistant Professor, Department of Computer Science and Applications, M. D. University, Rohtak-124001, Haryana, India

Abstract* - A grid is a hardware and software infrastructure that allows service oriented, flexible, and seamless sharing of diverse network of resources. A grid is to compute data intensive tasks and provides faster throughput and scalability at lower costs. The aim of grid computing is to provide an affordable approach to large-scale computing problems. The geographically isolated computational resources combined within a grid viewed as a virtual supercomputer. This paper tells about the various types of grid, Grid Architecture, OGSA, Fault Tolerance, Load Balancing, and various challenges in grid computing.

***Keywords* — Grid Computing, Virtual Organizations, OGSA, Fault Tolerance, Load Balancing**

I. INTRODUCTION

A parallel architecture in which shared resources across a network acts as functions like one large supercomputer. A grid allows unused CPU capacity in sharing machines to be allocated to one application that is computation intensive and programmed for parallel processing. Grid computing is peer to peer computing and distributed computing. The grid computing gives us yet another way of sharing the computer resource and yields us the maximum benefit at the time and speed efficiency. Grid computing enables multiple applications to share computing infrastructure, resulting in much greater scalability, flexibility, cost, power efficiency, performance, and availability at the same time. [1].

In 1998, Ian Foster and Carl Kesselman [2] provided a first definition “A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.” In a later article [9], the authors refined definition to address social and policy issues, stating that Grid computing concerned with “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.” The key idea is the ability to negotiate resource-sharing arrangements among a set of joining parties (providers and consumers) and then to use the resulting resource pool for some purpose. Grid computing is an emerging technology where computational and information resources shared and managed by various organizations in widespread locations (virtual organizations). Grid computing offers valuable services to work groups such as scientific researchers, airplane designers, drug-research firms. Resources usually owned by different people, communities or organizations with varied administrative policies in grid environments. To manage resources in such a distributed environment is a complex task because of geographical distributed and dynamic behavior of resources. The complexity of grid environment growing because of increase in number of projects and applications that followed in this domain. The demands exerted on grid are highly dynamic in nature and varies from application to application. For example, one application may need large number of CPUs and another application is memory intensive. This makes it difficult to identify idle resources, and the resource to application mapping. In such scenarios it is important that grids managed by a well-defined scalable and resource management system.

Grid Types

Computing grid: This grid designed to provide as much computing power as possible. This environment usually provides services for filing, surveying and managing jobs. In a computational grid most machines are high-performance servers.
Data grid: A data grid stores and provides reliable access to data across multiple organizations. It manages the physical data storage, data access policies and security issues of the stored data.
Service Grid: A service grid provides services that not covered by a single machine. It connects users and applications into workgroups and enables real-time interaction between users and applications with a virtual workspace. Service grids include on-demand, collaborative and multimedia grid systems.

II. CHARACTERISTICS OF GRID

Ian Foster defines the main characteristics of a grid as follows:

Decentralized control: Decentralized control on the resources and enables different administration policies and local management systems within the grid.

Open technology: A grid should use of open protocols and standards.

High quality service: A grid provides high quality of service in performance, availability and security.

III. BENEFITS OF GRID COMPUTING

In recent years, the IT infrastructure is facing a huge amount of stress because of the significant increase in transaction volumes. So, there are needs in collaboration and virtualization of resources and policies. All these business needs drive the need and use of the grid in enterprises. There are mainly four distinct benefits of using grids are resource utilization, performance and scalability, management and reliability, and virtualization.

IV. GRID ARCHITECTURE MODEL

In this section, we will discuss the “grid problem”, the ideas of a “virtual organization” and the architecture defined to solve the “grid problem”.

A. The GRID PROBLEM

Grid Computing is trying to solve the problems associated with resource sharing among a set of individuals or groups. These grid computing resources include data storage, power, hardware, on-demand software, and applications. The real problems involved with resource sharing are resource discovery, event correlation, authentication, authorization, and access mechanisms. These problems become more complicated when the Grid Computing introduced a solution for utility computing, where industrial applications and resources become available as shareable.

B. Basic Idea of Virtual Organizations (VO)

VO defined as a set of individuals or institutions around a set of resource–sharing rules and conditions [2]. All these virtual organizations share some common concerns and needs, but may vary in size, scope, duration, sociology, and structure. The members of any virtual organization negotiate on resource sharing based on the rules and conditions. These rules defined to share the resources from the automatically constructed resource pool. Assigning users, resources, and organizations from different domains to a virtual organization is one of the key technical challenges in Grid computing. This includes a resource discovery mechanism, such as resource-sharing methods, specification of rules and conditions for member assignment and access control among the participants.

C. Grid Architecture

Grid architecture developed for the establishment, management and cross-organizational resource sharing within a virtual organization. It identifies basic parts of grid, defines the roles of components and shows how each component interacts with one another.
Fig.1 illustrates layered grid architecture and its relationship to the internet protocol architecture.

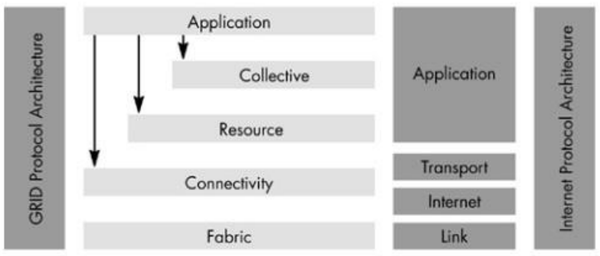

Fig. 1

Fabric Layer: It defines the resources which are shareable. It includes data storage, networks, catalogs and other computational resources.

Connectivity Layer: It defines the core communication and authentication protocols needed for grid-specific networking services.

Resource Layer: This layer uses the communication and security protocols defined by the networking communications layer, to control the secure negotiation, initiation, monitoring, metering, accounting and payment involving sharing across individual resources.

Collective Layer: It is responsible for all global resource management and interaction with a collection of resources. This protocol layer imposes a wide variety of sharing behaviors using a few Resource layer and Connectivity layer protocols.

Application Layer: These are user applications, which formed by using the services defined at each lower layer. Such an application can directly access the resources, or can access the resource through the Collective Service interface APIs (Application Provider Interface).

V. OPEN GRID SERVICE ARCHITECTURE (OGSA)

OGSA which recently proposed and actively developed by the Globus Project, intended to be a basic technology for building Grid systems. Interoperability and implementation independence may achieve through web services. The basic service semantics included in the Open Grid Services Infrastructure (OGSI) used as building blocks of Globus Toolkit 3.0 (GT3). The Global Grid Forum OGSA Working Group is discussing the higher level set of services that used in Grids and that integrated into the OGSA framework [3].

The objective of OGSA activity is to make the standard for all Grid services. It will give the standard way for discovery, inspection and interoperability with service. It is not required that all external communication with a service has to be done by means of XML/SOAP. The service may use any other communication protocols when high performance is necessary. It may however use OGSA mechanisms only for advertising itself to enable discovery by other services or use Service Data model to expose and publish some information for external consumers.

It is a layered architecture with separation of functionalities at each layer. As we can see from the fig., the core architecture layers are OGSI, which provides the base infrastructure. OGSA core platform services are a set of standard services including policy, logging, service-level management and so on. The high-level application services and services use these lower layer core platform components and OGSI that become part of a resource sharing grid.

Kunszt [4] points out some issues resolved in the development of OGSA for aspects such as availability, robustness, scalability, measurability, interoperability, compatibility, service discovery, manageability and changeability. Dialani et al. [5] proposed a fault tolerance mechanism for Web services that used to support Grid services fault tolerance. A service may become overloaded on a node at certain times if large numbers of users concurrently request a service instance. Zhang et al. [6] propose the Grid mobility service that we feel may be a solution to the service overloading problem. A service could be mobile code that can move from node to node in Grid environment. When necessary, a service can move to another more lightly loaded node to spawn service instances.

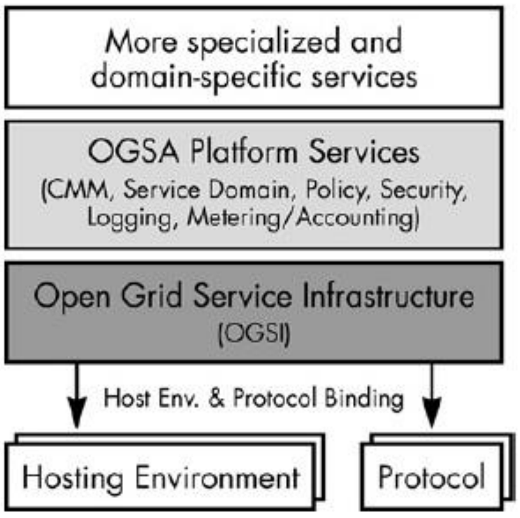
More specialized and domain-specific services
OGSA Platform Services
(CMM, Service Domain, Policy, Security, Logging, Metering/Accounting)
Open Grid Service Infrastructure (OGSI)
Host Env. & Protocol Binding
Hosting Environment
Protocol

Fig. 2

Major Goals of OGSA:

- Identify the use cases that can drive the OGSA platform components
- Identify and define the core OGSA platform components
- Define hosting and platform-specific bindings
- Define models and resource profiles with interoperable solutions.

VI. EVOLUTION OF GRID TECHNOLOGY

Grid technologies have emerged from some 10 years of research and development in both academia and industry, which further continues today. As showed in Fig 3, we can distinguish four distinct phases in this evolution.

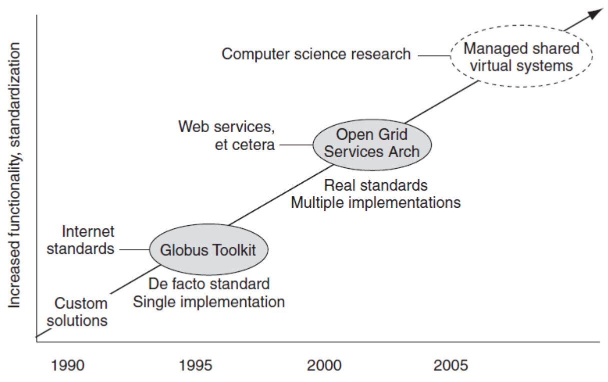

Fig. 3

Globus Toolkit. From 1997 onward, the open source Globus Toolkit version 2 (GT2) [7] emerged as the de facto standard for Grid computing. Focusing on usability and interoperability, GT2 defined and implemented the protocols, APIs, and services used in thousands of Grid deployments worldwide. By providing solutions to common problems such as authentication, resource discovery, and resource access, GT2 speeded up construction of real Grid applications. The GT2 protocol suite leveraged existing Internet standards for transport, resource discovery, and security. Some of the GT2 protocol suite arranged in formal technical specifications, reviewed within standards bodies, and instantiated in multiple implementations: notably, the GridFTP [8] data transfer protocol and elements of the Grid Security Infrastructure. However, in general, GT2 "standards" were neither formal nor subject to public review. Similar comments apply to other important Grid technologies that emerged during this period, such as the high-throughput computing system.

VII. FAULT TOLERANCE AND LOAD BALANCING

Fault tolerance is ability to perform its work correctly even in the presence of faults and it makes the system more dependable. The chance of a failure in grid is much greater than in traditional parallel computing as the resources spread geographically. There are two methods for detecting the fault in grid resource: the push or the pull model. In former one, grid components periodically send messages to failure detector, announcing that they are alive. The fault detector recognizes that failure has occurred at grid component without any such message from any grid component. In pull model, the failure detector sends live requests ("Are you alive?" messages) periodically to grid components [9].

Checkpoint-recovery and job replication are two techniques used in fault tolerance. Check pointing depends on the system's MTTR (Mean Time to Repair) while Replication depends on sites to run replicas.

Check pointing: The check pointing is the technique for providing fault-tolerance on unreliable systems. Check pointing is a snapshot of entire system state to restart the application after happening of some failure. The checkpoint can store periodically on temporary as well as stable storage [10]. Frequent check pointing may increase the overhead, while lazy check pointing may lead to loss of significant computation. Decision on size of the check pointing interval and the check pointing technique is a complicated task. The various types of check pointing optimization are:

Full check pointing or Incremental check pointing Unconditional periodic check pointing or Optimal (Dynamic) check pointing

Synchronous (Coordinated) or asynchronous (Uncoordinated) check pointing Kernel, Application or User level check pointing.

Replication: This technique based on an assumption that single resource much susceptible to failure compared to simultaneous failure of multiple resources. Unlike check pointing, the replication avoids task recomputation by executing several copies of the same task on more than one compute stations. The job replication and finding out best number of replicas involves many technical concerns. The task replication in grids has studied in [12].

Static vs. Dynamic replication: In static replication [10], when any replica fails, it's not replaced by any other replica. The number of replicas of original task decided before starting the execution. In dynamic replication, new replicas can generate during run time.

Active vs. passive replication: In active replication, the state of replicas kept closely synchronized and replicas service the same requests in parallel and undergo the same state transitions [9]. In Passive replication, primary replica services requests for clients and other replicas kept as standby and can take over a primary failure [11].

LOAD BALANCING

Load balancing feature always integrated into system to avoid processing delays and over commitment of resources. These applications built with schedulers and resource managers. The workload push towards resources, based on resources available, and then pulls jobs from the schedulers depending on their availability. This load balancing involves partitioning of jobs, identifying the resources, and queuing of the jobs. There are cases when resource reservations needed, as well as running multiple jobs in parallel. It supports for failure detection and management. The grid model contains the features of heterogeneous and homogeneous systems, and different coupling choices like tightly coupled and loosely coupled systems. The main aim of load balancing techniques is to adjust the load effectively to its surrounding. Zaki et al. [15] judge many CPU execution speeds and balance the jobs effectively. Hendrickson and Devine [14] evaluate some of the large groups of dynamic load balancing (DLB) techniques. For the heterogeneous systems, they give more focus on the connections of arrangement with different execution time.

Load balancing algorithms defined by their implementation of the following policies [13]:

Information policy: specifies load information, when it collected and from where.

Triggering policy: decides the proper moment to start a load balancing.

Location policy: uses results of the resource type policy to find a suitable partner for a server or receiver.

Selection policy: defines tasks that migrated from overloaded resources to idlest ones.

VIII. CHALLENGES OF GRID COMPUTING

The most important challenge in grid is to handle variation present in the infrastructure. Complexity arises because of the decentralized control, underlying hardware and software resources. It also arises to deal with faults in grid, grid middleware such as resource broker, security and privacy mechanism, local policies and usage patterns of the resources and so on. Programming in Grid environment introduces new challenges that not faced in parallel computers, such as multiple administrative domains, new failures, and large variations in performance.

REFERENCES

[1] H. Kargupta and C. Kamath and P. Chan, Distributed and Parallel Data Mining: Emergence, Growth, and Future Directions, In: Advances in Distributed and Parallel Knowledge Discovery.

[2] Ian Foster and Carl Kesselman (eds),The Grid: Blueprint for a New Computing Infrastructure, 1st edition, Morgan Kaufmann Publishers, San Francisco, USA (1 November 1998), ISBN: 1558604758.

[3] GlobalGrid Forum: http://www.ggf.org

[4] Kunszt, Peter Z. (April 2002).The Open Grid Services Architecture – A Summary and Evaluation, http://edms.cern.ch/file/350096/1/OGSAreview.pdf.

[5] Dialani, V., Miles, S., Moreau, L., Roure, D.D. and Luck, M. (August 2002). Transparent Fault Tolerance for Web Services Based Architectures. Proceedings of 8th International Europar Conference (EURO-PAR '02), Paderborn,Germany. Lecture Notes in Computer Science, Springer-Verlag.

[6] Zhang, W., Zhang, J., Ma, D., Wang, B. and Chen, Y. (2004). Key Technique Research on Grid Mobile Service. Proceedings of the 2nd International Conference on Information Technology for Application (ICITA 2004), Harbin,China.

[7] Foster, I., and Kesselman, C., Globus: A metacomputing infrastructure toolkit,International Journal of Supercomputer Applications 11(2), 115–129, 1998.

[8]Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Liming, L., and Tuecke, S., GridFTP: Protocol Extension to FTP for the Grid. Global Grid Forum, 2001.

[9] Y. Li, Z. Lan, “Exploit failure prediction for adaptive fault-tolerance in cluster”. In: Proceedings of the sixth IEEE International symposium on cluster computing and the grid, Vol 1, May 2006, pp.531-538.

[10] Oliner, A.J., Sahoo, R.K., Moreira, J.E., Gupta, M.: “Performance Implications of Periodic Checkpointing on Large-Scale Cluster Systems”, In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Washington, 2005.

[11] Y. Wang, Y. H. jin, W. Guo, W. Q. Sun, W. S. hu and M. Y. Wu, (2007) “Joint Scheduling for Optical Grid Applications”, Journal of Optical Networking, Vol. 6, pp. 304-318.

[12] S. Agarwal, R. Garg, M. Gupta, and J. Moreira, “Adaptive Incremental Checkpointing formassively Parallel Systems,” In Proceedings of the 18th Ann. International Conf. Supercomputing, Nov. 2004.

[13]. Leinberger, W., G. Karypis, V. Kumar and R. Biswas, 2000. Load balancing across nearhomogeneous multi-resource servers. In 9th Heterogeneous Computing Workshop, pp: 60-71.

[14]. Berman, F., G. Fox and Y. Hey, 2003. Grid Computing: Making the Global Infrastructure a Reality. Wiley Series in Comm. Networking & Distributed System.

[15]. Foster, I. and C. Kesselman, 1997. Globus: a metacomputing infrastructure toolkit. Intl. J. Super- Computer and High Performance Computing Applications, 11: 115-128.