Dominique A. Heger, Fortuitous Technology, (dom@fortuitous.com), Austin, TX, 2006

An Introduction to Grid Technology – Vision, Architecture, & Terminology

Introduction

In today's economic environment, any organization aims at reducing the time-to-market and any associated cost factors. At the same time, constraints on processing power, and the limitations on existing computing infrastructures hamper the implementation of efficient and effective IT solutions. It is increasingly important today to explore new avenues to utilize the existing IT resources. In many industries, such as financial services, manufacturing, or life sciences, significant improvements to the bottom line have been realized via adopting some form of Grid computing. In a nutshell, the basic idea behind Grid computing is to interconnect and utilize available processor, storage, or memory sub-components of distributed computing systems to solve larger problems more efficiently. The benefits of Grid computing are (1) cost savings, (2) improved business agility by decreasing the time-to-market (delivering actual results), and (3) enhanced collaboration and sharing of resources among departments or institutions.

The basic concept of Grid networking is not new, what *is* relatively new is the current transition of Grid networking from R&D into the commercial market. This article introduces the vision of Grid networking, and the rather pragmatic evolution that was taken to realize that vision. The report elaborates on the Grid architecture and technologies that surround the Grid paradigm. Further, the report addresses some of the issues surrounding Grid applications. To reiterate, the vision of the Grid movement is to virtualize the computing landscape, focusing on the main goal of creating an actual *utility computing model* that is distributed over a set of resources. In general, a single compute node includes some basic elements such as a processor, some storage (I/O) subsystem, an operating system, and some form of network/interconnect interface. The basic concept of Grid computing is to establish a similar environment, over a distributed area, incorporating heterogeneous elements such as server nodes, storage devices, and network components in a scaleable, wide-area spanning compute infrastructure. The software that coordinates the participating components and elements is analogous to the operating system in a single compute or server environment.

Virtualization of Computing Resources

Virtualizing the compute resources yields a scaleable (and flexible) pool of processing and data storage components that can be utilized to improve efficiency. Moreover, it aims at generating a competitive advantage by streamlining product development, allowing an organization to focus on their core business. Over time, Grid environments will enable the creation of virtual organizations and advanced Web services, as partnerships and collaborations become more critical in strengthening each link in the value chain. It has to be pointed out though that nowadays there are many definitions for Grid computing, but the core concept revolves around the "*aggregation of geographically dispersed computing, storage, and network resources, coordinated to deliver improved performance, higher quality of service, better utilization, and easier access to data. It enables virtual, collaborative organizations, sharing applications and data in an open, heterogeneous environment*".

The common denominator for all Grid deployments is the network layer. The shared network fabric is the component that connects all of the resources in a Grid, and hence has a profound impact on the success of a Grid implementation. Therefore, high-speed data networking technologies have been the cornerstone for the evolution of the Grid technology. Distributed Grid systems demand high-speed connectivity and provide very low latency behaviors. High-performance Ethernet switching elements are paramount in meeting these requirements. Hence, the statement can be made that components such as (bundled) Gigabit Ethernet interfaces or 10-Gigabit Ethernet connections are considered a necessity to achieve the performance goals. Over time, as the Grid infrastructure evolves from cluster systems to (1) virtualized enterprise data centers to (2) large distributed, wide-area spanning deployments, the underlying network infrastructure has to grow in a cost-effective manner, be scaleable, and meet the performance requirements.

Grid computing evolved out of the academic research community and the national defense industry, where researchers have to process vast amounts of data as efficiently as possible (mostly in simulation related projects). Utilizing the original concept of Grid computing, arrays of computational power and storage are constructed from a network of many small (and sometimes widespread) compute nodes. The resulting infrastructure is used to perform large calculation and operation based studies, where the workload generator (application) can be decomposed into independent units of work. This approach allows massive computational projects to achieve results that otherwise could not be completed, even on today's largest super-computers. To re-emphasize, as the concept has evolved, Grid computing has gained some acceptance in the commercial marketplace. Institutions with both, large and small networks have adopted Grid techniques to (1) reduce the aggregate execution time and (2) to enable resource sharing.

Types of Grid Environments

There are three basic types of Grid environments discussed in the market today. (1) The compute Grid, (2) the data Grid, and the (3) utility Grid. On the other hand, from an application perspective, there are two types of Grid environments, the compute and the data Grids, respectively. From a topology perspective, the argument being made is that there are additional Grid types, such as the cluster systems, intra-Grids, extra-Grids, and inter-Grids. In reality, clusters, intra-Grids, extra-Grids, and inter-Grids can be better defined as representing stages of the Grid evolution. In each of these stages, it is feasible to support compute Grids, data Grids, or a combination of both types. The majority of the early Grid deployments have focused on enhancing computation, but as data Grids provide easier access to large, shared data sets, data Grids are becoming increasingly important.

A *compute Grid* is essentially a collection of distributed computing resources (within one or multiple locations). The compute resources are aggregated to act as a unified processing resource (virtual super-computer). The process of interconnecting these resources into a unified pool involves the coordination of the usage policies, job scheduling, queuing issues, Grid security, as well as user authentication. The main benefit of a compute Grid is to construct an infrastructure that allows efficient processing of compute-intensive jobs, by utilizing existing resources.

A *data Grid* provides distributed, secure access to current data. Data Grids enable managing (and efficiently utilize) database information from distributed locations. Much like a compute Grid, data Grids also have to rely on software components to insure secure access and enforce usage policies. Data Grids (such as compute Grids) can be deployed within one or across multiple domains. Data Grids eliminate the necessity to move, replicate, or centralize data. Actual data Grids are being used today, primarily serving collaborative research communities.

The evolution from compute Grids to data Grids is an important factor in repositioning Grid applications from education and R&D to large enterprises. This transition is an indicator that the market (and the technology itself) is maturing. From a networking perspective, the impact of data Grids includes a tighter integration of storage protocols and high-performance networking. As the middle-ware components for Grid infrastructures mature, coupled with the recent advances in data networking equipment, the statement can be made that the necessary momentum for evolving Grids from local clusters to distributed, wide-area systems is well defined. The first stage of Grid computing revolves around clusters. Based on the true definition of a Grid that includes terms like *distributed and heterogeneous*, it is debatable whether cluster systems should actually be considered as a Grid solution. Semantics aside, nevertheless, clusters are paramount in the evolution of Grid computing. Clusters are often defined by using terms such as distributed or parallel file systems or by a collection of homogeneous server nodes that are aggregated for increased performance. Cluster systems are largely being used in application domains such as simulation-based testing and evaluation. The majority of the newer cluster systems utilize high-speed interconnects such as bundled Gigabit Ethernet, Myrinet, InfiniBand, or any other (sometimes proprietary) high-throughput network interface. Low-latency switching is critical in maintaining application performance across the cluster fabric.

To reiterate, cluster systems are critical in the evolution of Grid computing. As in order to move to the next phase (*intra-Grids*), these cluster systems have to be interconnected. By interconnecting individual

cluster systems, it is feasible to enable the establishment of enterprise and inter-departmental Grid systems. The creation of intra-Grids puts additional strain on the controlling software layer (middle-ware) and the underlying network infrastructure. The middle-ware has to have a better understanding of the resource allocation, based on the additional complexity of the processor-sharing relationships. Further, latency issues at the network layer become more challenging as well, shifting the focus to even higher throughput capable interconnects. In a next phase, extra-Grids will reflect cluster or intra-Grids that are connected among geographically distributed sites within either a single or among several organizations. The two important distinctions made for an extra-Grid are (1) geographic distribution and (2) inter-enterprise relationships. Because of these relationships, extra-Grids are sometimes referred to as partner Grids. In such a partner Grid, as the data can be shared among different organizations, authentication, policy management, and security issues become even more critical, a fact that has to be addressed by the Grid middle-ware layer. Further, load balancing, topology discovery, network (LAN/WAN) throughput, and application awareness are other factors that have to be considered to provide adequate performance.

The final stage in the evolution of Grid computing revolves around the *inter-Grid* paradigm. This stage reflects the most powerful phase of Grid evolution, as it embodies two of the primary visions for Grid computing, (1) the utility computing infrastructure and (2) the Grid services providers. Although this is the final stage (which has not yet been reached from a commercial perspective), it has to be pointed out that inter-Grids do exist today in the R&D domain (the TeraGrid as an example). Inter-Grid setups surpass all the other phases in regards to the relative complexity requirements. To illustrate this point, compared to a setup where a couple of institutions participate in an extra-Grid environment, an inter-Grid may service thousands of users. As a result, all of the complex requirements at the middle-ware and the network layer are increased significantly. Grid setups in this stage are also referred to as utility Grids. At this time, two significant statements about the status quo of Grid implementations can be made. (1) Cluster systems are evolving toward intra-Grid setups in the commercial world. (2) Complex inter-Grids are being built, tested, and deployed in the R&D domain. These are signs that the Grid momentum is still building, and the market is moving forward. The key to successfully penetrate the market will be to establish early deployments, so that the technology can evolve in parallel with the market requirements.

Grid Layers

Figure 1: GRID Structures

Descriptions of the Layered Grid Structure
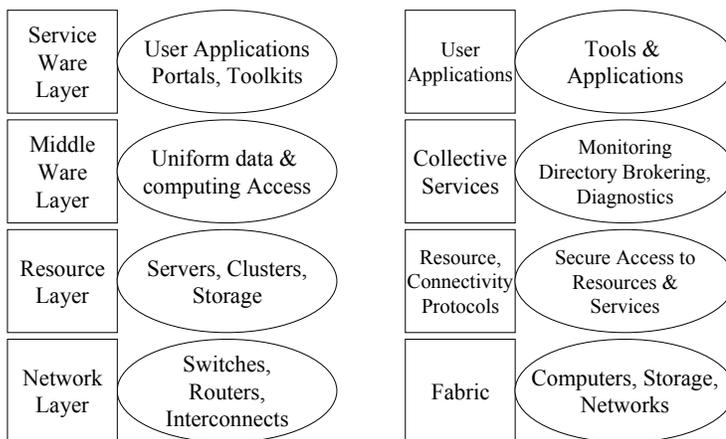
At the base of the Grid architecture, the bottom layer so to speak, is the network, which assures the connectivity for the resources in the Grid (see Figure 1). On top of the network layer is the resource layer, which incorporates the actual resources that are part of the Grid (mainly computers and storage systems).

The middle-ware layer (positioned above the resource layer) provides the tools that enable the various elements (servers, storage, and networks) to participate in a unified Grid environment. The middle-ware layer can be described as the intelligence that brings the various elements in the Grid together. The top layer of the Grid structure represents the application layer, which includes all different user applications (science, engineering, business, financial), as well as portals and development toolkits that are supporting the Grid applications. It is the top layer that the users of the Grid are actually working with.

In most Grid architectures, the application layer also provides the *service-ware* functionality. The service-ware represents a set of general Grid management functions that are utilized to measure the amount of time a user spends on the Grid. Further, the service-ware acts as the billing functionality for the Grid services (assuming a commercial model), and keeps track of who is providing the resources and who can use them. It has to be pointed out that the service-ware is part of the top layer, as it is an entity the user interacts with, whereas the middle-ware components reflect a hidden layer that the user dose not necessarily has to worry about. There are other ways to describe this layered Grid structure. To illustrate, the Grid community uses the term fabric for all the physical infrastructure of the Grid, including the computers and the communication network. Within the middle-ware layer, distinctions can be made between a layer of resource and connectivity protocols, and a higher layer of collective services. Resource and connectivity protocols handle all the Grid specific network transactions (among different computers and any other resources on the Grid). This is accomplished via communication protocols, which let the resources communicate with each other, enabling the exchange of data as well as authentication protocols, which provide secure mechanisms for verifying the identity of both users and resources. The collective services are also based on protocols, actual information protocols, which obtain information about the structure and state of the resources on the Grid, and management protocols which negotiate access to resources in a uniform way. These services include:

- Updating the directories that represent the available resources
- Providing brokering services (buy and sell resources)
- Monitoring the Grid and diagnosing any potential issues
- Replicating key data so that multiple copies are available at different locations for ease of use
- Providing membership/policy services, keeping track of who is allowed to do what and when

To reiterate, in all the Grid schemes, the top layer represents the applications layer. Applications rely on all the other layers to run on the Grid. To illustrate, assuming a Grid application that analyzes data that is distributed among several independent files. Hence, the application has to:

1. Obtain the necessary authentication credentials to open the files (resource and connectivity protocols)
2. Query an information system and replica catalogue to determine where the copies of the files can be located on the Grid, as well as where the computational resources necessary to conduct the analysis are available, and most conveniently located (collective services)
3. Submit actual requests to the fabric, the appropriate computers, storage systems, and network components to extract the data, initiate computations, and provide the results (resource and connectivity protocols)
4. Monitor the progress of the various computation tasks and data transfer components. Notifying the user when the analysis is completed. Detecting and responding to any potential failure conditions (collective services)

In order to accomplish all the above discussed scenarios, an application written for a single CPU system will have to be *substantially adapted* in order to invoke the proper services, use multiple distributed components simultaneously, and utilize the required protocols. In other words, a Grid requires companies and institutions to heavily invest time and money into getting their applications ready for a Grid environment. Plus, it has to be pointed out that not every application is suitable for running on a Grid.

Grid Applications

It is paramount to clearly define the type of applications that have the potential to run in a Grid environment, so that the right Grid project can be chosen, realistic expectations can be set, and performance

goals can be established and met. Typically, applications that are good candidates for a Grid implementation take many hours (possibly days or weeks) to execute (large problem sizes). In some circumstances, the task is so big that it can not be completed at all even given today's processor capacity. The reason behind the long run time may be due to the application requiring many replicated runs of the same fundamental tasks, such as identical processing on many subgroups of a large data file, or certain types of optimizations or statistical simulations, respectively. Another example of a long-running task may be the one where many independent tasks have to execute against the same large data source (scoring or risk analysis projects). In general, an application should possess one or more of the following characteristics to be considered a good candidate for a Grid implementation.

1. Problem to be solved results in a long execution time
2. Problem to be solved Involves many replicated runs of the same fundamental task
3. Problem to be solved requires processing vast amounts of data
4. The problem to be solved allows the decomposition into multiple execution units and/or data subsets

Many (especially scientific) applications involve repeating the same fundamental task many times against unique subsets of the data pool. While the execution of a single task against a single subset of the data may execute rather quickly, repeating the same task against thousands or even millions of subsets of the data pool impacts the aggregate execution time. These types of problems are massively parallel and the applications that solve them are very well suited to a Grid implementation, as the replicated tasks can be distributed across the Grid and executed in parallel, which greatly reduces the aggregate execution time. Each of the fundamental tasks distributed across the Grid has to have access to all the required (input) data. Sometimes the input data can be small (MB's), and other times the input data may be rather large (GB's). In order to achieve the highest possible efficiency level, the compute nodes have to spend the majority of the time processing compute tasks rather than processing any communication related activities. Compute tasks that require substantial data movement generally do not perform very well in a Grid environment. The data has to be either distributed to the nodes prior to running the application, or the data has to be made available via a shared network storage solution. It has to note that there have been many recent advances in data storage hardware (switches, controllers, disks) and software (advanced parallel file systems for clusters) that provide faster access to data, and hence actively contribute to the success of a Grid implementation.

Globus Grid Toolkit

Practically all major (R&D) Grid projects today are built based on protocols and services provided by the Globus Toolkit, a software solution which is being developed by the Globus Alliance, which involves primarily Ian Foster's team at Argonne National Laboratory and Carl Kesselman's team at the University of Southern California in Los Angeles. The toolkit provides a set of software tools to implement the basic services and capabilities required to construct a computational Grid, such as security, resource location, resource management, and communications. Below is a list of the primary elements included in the Globus Toolkit.

- *Globus Resource Allocation Manager (GRAM)*. The GRAM processes the resource requests, and allocates the resources necessary for application execution. The component further manages active jobs that are running on a Grid, and returns updated capability information to the Monitoring and Discovery Service (MDS).

- *Monitoring and Discovery Service (MDS)*. A Light-way Directory Access Protocol (LDAP) based service that allows querying system information from a variety of components (processing capacity, bandwidth capacity, types of storage). MDS enables the collection of element specific information for use in a Grid environment. It allows the optional construction of a uniform namespace (for resource information purposes) across a system that may involve many organizations.

- *Grid Security Infrastructure (GSI)*. Provides a secure authentication and communication facility over a Grid network. In addition, it supports security across organizational boundaries, and single sign-on capabilities for the users of a Grid, including delegation of credentials for computations that involve

multiple resources. GSI is based on public key encryption (X.509 certificates), and secure sockets layer (SSL) entities, with extensions for Grid-specific applications.

- *Grid Resource Information Service (GRIS).* Provides a uniform approach to querying resources on a Grid for their current configuration, capabilities, and status information, respectively. Such resources may include server systems, storage and network components, or database systems.

- *Grid Index Information Service (GIIS)*. Provides a method to coordinate arbitrary GRIS services to provide a consistent system image, which can be explored by Grid enabled applications. In addition to providing a consistent system image, subsets of GRIS services (such as all the storage entities within a specific subset of a Grid) can be defined.

- *GridFTP*. A high-performance, secure, and robust data transfer mechanism for Grid environments. GridFTP is based on FTP (the File Transfer Protocol), but is provided with extensions for Grid-specific requirements. Additional features include third-party control over data transfer, parallel data transfer, striped data transfer, and partial file transfer.

- *Replica Catalog*. Provides a mechanism for maintaining a catalog of data-set replicas. This service is accomplished by providing actual mapping information among logical file names and one or more copies of the files location on physical storage.

- *Replica Management*. Provides a mechanism that couples the Replica Catalog and the GridFTP technologies, allowing applications to generate and manage replicas of large data-set entities.

To reiterate, many of the functions performed by the Grid middle-ware and/or the Grid server appliances today (including security, quality of service, fault detection and recovery, policy management, scheduling and queuing, authentication, topology discovery, resource allocation, and load balancing) are natural tasks for today's networking products and protocols. Hence, many of the protocols and functions defined by the Globus Toolkit are analogous to protocols that already exist in the networking and storage environment, but the toolkit provides the services optimized for Grid-specific deployments. The synergy between Grid protocols defined at the middle-ware layer and protocols at other layers (networking and storage), are expected to pragmatically converge over time. Applications enabled for Grid computing represent an actual convergence point for storage, networking, and computing resources, respectively. There are two main reasons for the strength and popularity of the Globus toolkit. (1) The Grid will have to support a wide variety of applications that have been developed according to different programming paradigms. Rather than providing a uniform programming model for Grid applications, the Globus Toolkit reflects an object-oriented approach, providing a set of services from which developers can choose based on a company's particular needs. The tools can be introduced one at a time (into existing programs) to allow the application to gradually become increasingly Grid enabled. To illustrate, an application may exploit some of the Globus features such as GRAM for resource management or GRIS for information services, without necessarily utilizing the Globus security or replica management systems, respectively. (2) Similar to the Linux operating system, the providers of the Globus Toolkit are distributing the software under the open-source licensing agreement. This allows other institutions and single contributors to utilize the software at no cost, as well as to contribute via improvements to the toolkit.

Conclusion - Benefits of Grid computing

There are several economic and business factors that are contributing to the heightened interest in the development and deployment of Grid computing. Based on the Internet and E-commerce, today's society is inundated with data. As the available data repository grows bigger and wider, the window of opportunity for capturing and translating the available data into information shrinks rapidly. Computing applications in many industries involve processing vast amounts of data and/or performing large numbers of repetitive computation operations that exceed the capabilities of existing platforms. To utilize data analysis techniques to achieve a high level of business intelligence, and improve the decision making process, data has to be analyzed in a much more timely manner. Today's business requirements demand a much larger sample size for analysis to provide the best possible accuracy level achievable. The challenges

that IT faces today include budget cuts, server consolidation, hardware provisioning and overall administration, all factors that may further drive the interest in implementing Grid computing. The convergence of recent hardware and software advances has made resource virtualization feasible, hence made it easier to construct and deploy a Grid infrastructure. On the hardware side, advances include networked storage devices and low-cost, modular hardware components (such as blade systems). On the software side, the advances include improvements in networking, Web services, databases, application servers and management frameworks. While Grid computing may not be the solution for every application or institution, Grid computing has to be considered as an innovative solution that provides:

1. Scalability of applications, as long-running applications can be decomposed along either execution unit boundaries and/or data subsets, and hence be executed in parallel, economizing on the aggregate execution time.

2. Scalability in the number of users, as multiple users can access a virtualized pool of resources to obtain the best possible response time by maximizing the utilization of the computing resources.

3. By implementing Grid computing technology, organizations can optimize the return on investment and lower the cost of ownership

Hence, to summarize, Grid solutions reflect three main categories beneficial to organizations and institutions. (1) Cost saving, as it is feasible to leverage and exploit unutilized or underutilized compute power and/or storage capacity within a networked environment. (2) Improved business agility by decreasing the time to process the data and deliver the results. By delivering results faster, a company is provided with the insight and agility to adjust to a changing market place. (3) Enhanced collaboration as IT resources can be shared and utilized collectively. This collaboration insures to efficiently as well as effectively resolve compute-intensive problems.

References

1. Avery, P., Foster, I., Gardner, R., Newman, H. and Szalay, "A. An International Virtual-Data Grid Laboratory for Data Intensive Science". GriPhyN, 2001

2. Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C., "Grid Information Services for Distributed Resource Sharing". In IEEE International Symposium on High Performance Distributed Computing, (2001), IEEE Press

3. Foster, I., "Grid Technologies & Applications: Architecture & Achievements", Argonne National Laboratory, 2002

4. Foster, I. and Kesselman, C. "A Data Grid Reference Architecture". GriPhyN, 2001

5. Foster, I. and Kesselman, C. "Globus: A Toolkit-Based Grid Architecture. In The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 1999, 259-278.

6. Grid @ CERN: http://gridcafe.web.cern.ch/gridcafe/GridatCERN/gridatcern.html

7. Oracle, "Oracle Grid Computing", Oracle Business While Paper, 2003

8. Smith, W., Gunter, D., "Simple LDAP Schemas for Grid Monitoring", NASA Ames Research Center, 2001

9. The Globus Alliance: http://www.globus.org/