Grid Computing 4

Grid Computing is defined as controlled and coordinated resource sharing and problem-solving in dynamic, multi-institutional virtual organizations (VOs). These sharing resources, ranging from simple file transfers to complex and collaborative problem-solving, are accomplished within controlled and well-defined conditions and policies. The dynamic groupings of individuals, multiple groups, or organizations that define the conditions and rules for sharing are called *virtual organizations,* or *VOs*.

What Is Grid Computing?

Grid Computing enables the virtualization of distributed computing and data resources such as processing, network bandwidth, and storage capacity to create a single system image, granting users and applications seamless access to vast IT capabilities. Grid Computing utilizes untapped resources of a computing device, without interruption to whatever the end user may be doing at that moment in time on the computer. Just as an Internet user views a unified instance of content via the Web, a Grid Computing user essentially sees a single, large, virtual computer. At its core, Grid Computing is based on an open set of standards and protocols—for instance, OGSA—that enables communications across heterogeneous, geographically dispersed environments. With Grid Computing, organizations can optimize computing and data resources, pool them for large-capacity workloads, share them across networks, and enable collaboration.

Grid Computing is important to on demand business; however, Grid Computing is not always required to create on demand business solutions. That being said, Grid Computing is one way of delivering very powerful on demand business solutions. Grid Computing, in fact, shares many of the same technologies with Autonomic Computing; for example, dynamic provisioning of resources. It is important to note, however, as we explore the interesting and slightly diverse subjects in this chapter, that one can deliver on demand business solutions without developing and/or integrating Grid Computing applications.

In the previous chapter, we discussed Autonomic Computing strategic perspectives. In this chapter, we will introduce and define the "Grid Computing problem," discuss the core concepts of a *virtual organization* (VO), and define the Grid Protocol Architecture that will solve the Grid Computing problem. This chapter will present some very innovative work [Foster01] that defines the Grid Computing anatomy. In addition, this chapter will examine Grid Computing in relation to other distributed technologies such as the Web, framework(s) providing services, server clusters, and P2P (peer-to-peer) computing.

THE VO CONCEPT IN GRID COMPUTING IS KEY

One of the significant operational concepts in Grid Computing is the implementation of the *VO.* This involves the dynamic computational-oriented task of defining groupings of individuals, such as multiple groups or organizations. Although this is perhaps simple to understand in theory, it remains complex across several dimensions. The complexities involved in this dynamic assembly revolve around identifying and bringing together the humans who initially defined the conditions to instantiate the grid. This includes automated functionalities of the rules, the policies, and the specific conditions affecting operations in the Grid that are at the core surrounding processing and sharing of information with those individuals in a VO.

The simplest way to think about this advanced Grid Computing concept is captured in the term "VO." This type of grouping serves as the basis for identifying and managing the Grid Computing groups that are associated with any particular community of Grid Computing end-users.

The Grid Computing Problem

Grid Computing has evolved as an important field in the computer industry by differentiating itself from distributed computing with an increased focus on resource sharing, coordination, and high-performance orientation. Grid Computing tries to solve the problems associated with resource sharing among a set of individuals or groups.

Grid Computing resources include computing power, data storage, hardware instruments, on demand business software, and applications. Problems associated with resource sharing among a set of individuals or groups are resource discovery, event correlation, authentication, authorization, and access mechanisms. These problems become proportionately more complicated when the Grid Computing solution is introduced as a solution for Utility Computing. Utility Computer in a grid is where end-users in the grid VO share industrial applications and resources.

This commercial Utility Computing concept spanning across Grid Computing services has introduced a number of challenging problems to the already complicated Grid Computing problem domains. These challenging problems include service level management features, complex accounting, utilization metering, flexible pricing, federated security, scalability, open-ended integration, and a multitude of very difficult networking services to sustain. It is key to understand that networking services can no longer be taken for granted, as these very important services now become the central nervous system for the enablement of all worldwide Grid Computing environments.

The Concept of Virtual Organizations (VOs)

Understanding the concept of a VO is the key to understanding Grid Computing. A VO is a dynamic set of individuals and/or institutions defined around a set of resource sharing rules and conditions [Foster02]. All VOs share some commonality, including common concerns and requirements, but they may vary in size, scope, duration, sociology, and structure.

The members of any VO negotiate resource sharing based on defined rules and conditions. This allows the members of the VO to share the resources from the automatically constructed grid resource pool. Establishing the VO with users, resources, and organizations from different domains across multiple, worldwide geographics is one of the fundamental technical challenges in Grid Computing. This complexity includes the definitions of the resource discovery mechanism, resource sharing methods, rules and conditions by which this sharing can be achieved, security federation and/or delegation, and access controls among the participants of the VO. As you can see, this challenge is complicated across several dimensions.

Let us explore two examples of VOs to better understand their common characteristics:

1. Thousands of physicists from different laboratories join together to create, design, and analyze the products of a major detector at CERN, the European high-energy physics laboratory. This group forms a "data

grid," with resource sharing among intensive computing, storage, and network services to analyze petabytes of data created by the detector at CERN.

2. A company wants to do financial modeling for a customer based on the data collected from various data sources, both internal and external to the company. This specific VO customer may need a financial forecasting capability and advisory capability on its investment portfolio, which is based on actual historic and current real-time financial market data. This financial institution customer can then be responsive by forming a dynamic VO within the enterprise for achieving more benefit from advanced and massive forms of computational power (i.e., application service provider [ASP]) and for data (i.e., data access and integration provider). This dynamic, financially-oriented VO can now reduce undesirable customer wait time while increasing reliability on forecasting by using real-time data and financial modeling techniques.

By closely examining the VOs, we can infer the following: the number and type of participants, the resources being shared, the duration, the scale, and the interaction pattern between the participants. All these attributes vary between any one single VO and another. At the same time, we can also infer that there are common characteristics among competing and sometimes distrustful participants that contribute to their VO formation. They may include [Foster02] some of the following items for consideration:

1. *Common concerns and requirements on resource sharing*—A VO is a well-defined collection of individuals and/or institutions that shares a common set of concerns and requirements. For example, a VO created to provide financial forecast modeling shares the same concerns on security, data usage, computing requirements, resource usage, and interaction patterns.
2. *Conditional, time-bound, and rules-driven resource sharing*—Resource sharing is conditional, and each resource owner has full control on making the resource available to the sharable resource pool. These conditions are defined based on mutually understandable policies and access control requirements (authentication and authorization). The number of resources involved in the sharing may dynamically vary over time based on the policies defined.
3. *Dynamic collection of individuals and/or institutions*—Over its period of time, a VO should allow individuals and/or groups into and out of the collection, provided they all share the same concerns and requirements on resource sharing.

4. *Sharing relationship among participants is P2P in nature*—The sharing relationship among the participants in a VO is P2P, which emphasizes that a resource provider can become a consumer to another resource. This introduces a number of security challenges, including mutual authentication, federation, and delegation of credentials among participants.
5. *Resource sharing is based on an open and well-defined set of interaction and access rules*—Open definition and access information must exist for each sharable resource for better interoperability among the participants.

The above characteristics and non-functional requirements of a VO lead to the definition of an architecture for the establishment, management, and sharing of resources among participants. As we will see in the next section, the focus of the Grid Protocol Architecture is to define an interoperable and extensible solution for resource sharing within a VO.

The Grid Protocol Architecture

A new architecture model and technology were developed for the establishment, management, and cross-organizational resource sharing within a VO. This new architecture, called the *Grid Protocol Architecture,* identifies the basic components of a Grid Computing system, defines the purpose and functions of such components, and indicates how each of these components interacts with the others [Foster02]. The main focus of the architecture is on the interoperability among the resource providers and users to establish the sharing relationships. This interoperability means common protocols at each layer of the architecture model, which leads to the definition of a Grid Protocol Architecture, as shown in Figure 4.1. This Grid Protocol Architecture defines common mechanisms, interfaces, schema, and protocols at each layer, by which users and resources can negotiate, establish, manage, and share resources.

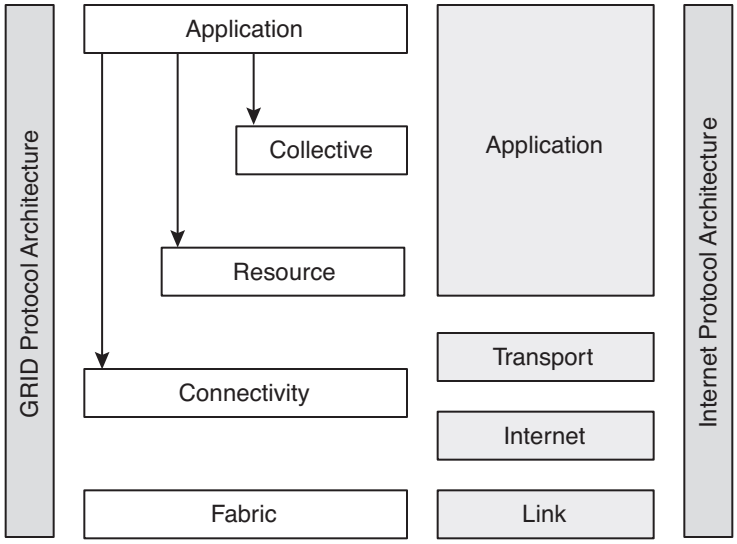

FIGURE 4.1 The layered Grid Protocol Architecture and its relationship to the IP architecture [Foster02].

Figure 4.1 illustrates the component layers of the architecture with specific capabilities at each layer. Each layer shares the behavior of the component layers described next. As we can see in this illustration, each of these component layers is compared with its corresponding IP layer(s) to further clarify its capabilities.

Fabric Layer: Interfaces to Local Resources This layer defines the resources that can be shared. These could include computational resources, data storage, networks, catalogs, and other system resources. These resources can be physical resources or logical resources by nature.

Typical examples of logical resources found in a Grid Computing environment are distributed file systems, computer clusters, distributed computer pools, software applications, and advanced forms of networking services. Logical resources are implemented by their own internal protocol (e.g., Network File System [NFS] for distributed file systems and Logical File System [LFS] for clusters). These resources then comprise their own network of physical resources.

Although there are no specific requirements for a particular resource that relate to integrating itself as part of any Grid Computing system, it is recommended to have two basic capabilities associated with the integration of resources; these basic capabilities should be considered "best practices" of Grid Computing disciplines:

1. Provide an "inquiry" mechanism that allows for the discovery of the resource against its own capabilities, structure, and state of operations. These are value-added features for resource discovery and monitoring.
2. Provide appropriate "resource management" capabilities to control the QoS (Quality of Service) that the Grid Computing solution promises, or has been contracted to deliver. This enables the service provider to control a resource for optimal manageability, such as (but not limited to) the following: starting and stopping activation, resolving problems, configuration management, load balancing, workflow, complex event correlation, and scheduling.

Connectivity Layer: Manages Communications The connectivity layer defines the core communication and authentication protocols required for Grid Computing-specific networking services transactions. Communication protocols, which include aspects of networking transport, routing, and naming, assist in the exchange of data between fabric layers of respective resources. The authentication protocol builds on top of the networking communication services to provide secure authentication and data exchange between users and respective resources.

The communication protocol can work with any of the networking layer protocols that provide the transport, routing, and naming capabilities in networking services solutions. The most commonly used network layer protocol is the Transmission Control Protocol/Internet Protocol (TCP/IP) stack; however, this discussion is not limited to that protocol. The authentication solution for VO environments requires significantly more complex characteristics. The following describes these characteristics:

1. *Single Sign-On (SSO)*—This allows any of multiple entities in the Grid Protocol Architecture fabric to be authenticated once, so the user can then access any available resources in the fabric layer without further user authentication intervention.
2. *Delegation*—This provides the ability to access a resource under the current user's permissions set, and the resource should be able to relay the same user credentials (or a subset of the credentials) to other resources respective to the chain of access.
3. *Integration with local resource-specific security solutions*—Each resource and hosting environment has specific security requirements and solutions that match the local environment. This may include (for example) Kerberos security methods, Windows security methods, Linux security methods, or UNIX security methods. Therefore, to provide proper secu-

rity in the Grid Protocol Architecture fabric model, all Grid Computing solutions must provide integration with the local environment and respective resources specifically engaged by the security solution mechanisms.

4. *User-based trust relationships*—In Grid Computing, establishing an absolute trust relationship among users and multiple service providers is very critical.
5. *Data security*—Data security provides both data integrity and confidentiality. The data passing through the Grid Computing solution, no matter what complications may exist, should be made secure using various cryptographic and data encryption mechanisms. These mechanisms are well-known in the world of technology, across all global industries.

Resource Layer: Shares a Single Resource

This layer utilizes the communication and security protocols defined by the networking communications layer to control the secure negotiation, initiation, monitoring, metering, accounting, and payment involved in the sharing of operations across *individual resources*. The way this works is the resource layer calls the fabric layer's functions to access and control the local resources. This layer only handles the individual resources, and hence ignores the global state and atomic actions across the other resources, which in the operational context becomes the responsibility of the collective layer.

There are two primary classes of resource layer protocols; these protocols are key to the operation and integrity of any single resource:

- Information protocols—These protocols are used to get information about the structure and operational state of a single resource, including configuration, usage policies, SLAs, and the state of the resource. In most situations, this information is used to monitor the resource capabilities and availability constraints.
- Management protocols—The important functionalities provided by the management protocols are:
  - Negotiating access to a shared resource is paramount. These negotiations can include the requirements for QoS, advanced reservation, scheduling, and other key operational factors.
  - Performing operation(s) on the resource, such as process creation, is critical. Data access is also a very important operational factor.
  - Acting as the service/resource policy enforcement point for policy validation between a user and resource is critical to the integrity for the operations.

- Providing accounting and payment management functions on resource sharing is mandatory.
- Monitoring the status of an operation and controlling the operation, including terminating the operation and providing asynchronous notifications on operational status, is extremely critical to the operational state of integrity.

It is recommended that these resource level protocols be minimal from a functional overhead point of view, and they should focus on the functionality each provides from a utility aspect.

Collective Layer: Coordinates Multiple Resources

The collective layer is responsible for all global resource management and interaction with any resource collection. This protocol layer implements a wide variety of sharing behaviors utilizing a small number of resource layer and connectivity layer protocols.

Some key examples of the common, more visible collective services in a Grid Computing system are:

- *Discovery services*—These services enable VO participants to discover the existence and/or properties of the specific available VO's resources.
- *Co-allocation, scheduling, and brokering services*—These services allow VO participants to request the allocation of one or more resources for a specific task, for a specific period of time, and to schedule those tasks on the appropriate resources.
- *Monitoring and diagnostic services*—These services provide the VO's resource failure recovery capabilities, monitoring of the networking and device services, and diagnostic services that include common event logging and intrusion detection. Another important aspect of this topic relates to the partial failure of any portion of a Grid Computing environment. It is critical to understand any and all *business impacts* related to any partial failure immediately, as the failure begins to occur—all the way through its corrective healing stages.
- *Data replication services*—These services support the management aspects of the VO's storage resources; they maximize data access performance with respect to response time, reliability, and cost.
- *Grid-enabled programming systems*—These systems allow familiar programming models to be utilized in Grid Computing environments while sustaining various Grid Computing networking services. These networking services are integral to the environment; they address

resource discovery, resource allocation, problem resolution, event correlation, network provisioning, and other very critical operational concerns related to Grid Computing networks.

- *Workload management systems and collaborative frameworks*—These provide multi-step, asynchronous, multi-component workflow management. This is a complex topic across several dimensions, yet it is a fundamental area of concern for enabling optimal performance and functional integrity.
- *Software discovery services*—These services provide the mechanisms to discover and select the best software implementation(s) available in the Grid Computing environment, and those available to the platform based on the problem being solved.
- *Community authorization servers*—These servers control resource access by enforcing community utilization policies and providing respective access capabilities by acting as policy enforcement agents.
- *Community accounting and payment services*—These services provide resource utilization metrics; they also generate payment requirements for members of any community.

As we can observe based on the previous discussion, the capabilities and efficiencies of these collective layer services are based on the underlying layers of the protocol stack. These collective networking services can be defined as general-purpose Grid Computing solutions that have been narrowed down to domain and application-specific solutions. As an example, one such service is accounting and payment, which is most often very specific to a domain or application. Other notable and very specialized collective layer services include schedulers, resource brokers, and workload managers.

Applications Layer: User-Defined Grid Computing Applications

These are user applications that are constructed by utilizing the services defined at each lower layer. Such an application can either access a resource directly or through the Collective Service API.

Each layer in the Grid Protocol Architecture provides a set of APIs and SDKs for the higher layers of integration. It is up to the application developers whether to use the collective services for general-purpose discovery and other high-level services across a set of resources or to start working directly with exposed resources. These user-defined Grid Computing applications are (in most cases) domain-specific and provide specific solutions.

The Grid Protocol Architecture and Its Relationship to Other Distributed Technologies

It is a known fact in technology that there are numerous well-defined and well-established technologies and standards developed for distributed computing. This foundation has been a huge success (to some extent) until we entered the domain of heterogeneous resource sharing and the formation of VOs.

Based on our previous discussions, the Grid Protocol Architecture is defined as coordinated and highly automated, dynamically sharing resources for a VO. It is appropriate that we turn our attention at this stage toward a discussion regarding how this architectural approach differs from distributed technologies, how the two approaches complement each other, and how we can leverage the best practices of both approaches.

Our discussion will now begin to explore notions of widely implemented distributed systems, including the following: World Wide Web environments, application and storage service providers, distributed computing systems, P2P computing systems, and clustering systems.

The World Wide Web

Numerous open and ubiquitous technologies are defined for the World Wide Web (TCP, Hypertext Transport Protocol [HTTP], SOAP, and XML) that in turn makes the Web a suitable candidate for the construction of VOs. However, as of now, the Web is defined as a browser/server messaging exchange model, and lacks the more complex interaction models required for a realistic VO.

Some areas of interest include: Delegation of authority, complex authentication mechanisms, and event correlation mechanisms. Once the browser-to-server interaction matures, the Web will be suitable for the construction of Grid Computing portals to support multiple VOs. This will be possible because the basic platforms and layers of technology will remain the same.

Distributed Computing Systems

The major distributed technologies, including CORBA, J2EE, and the Distributed Communication Object Model (DCOM) are well-suited for distributed computing applications; however, these technologies do not provide a suitable platform for sharing resources among the members of a VO. Some of the notable drawbacks include resource discovery across virtual participants, collaborative and declarative security, dynamic construction of the VO, and the scale factor involved in potential resource sharing environments.

Another major drawback in distributed computing systems involves the lack of interoperability among the technology protocols. However, even with these perceived drawbacks, some distributed technologies have attracted considerable Grid Computing research attention toward the construction of Grid Computing systems, the most notable of which is Java JINI.[1] The JINI system is focused on a platform-independent infrastructure that delivers services and mobile code to enable easier interaction with clients through service discovery, negotiation, and leasing.

Application and Storage Service Providers

Application and storage service providers normally outsource their business and scientific applications and services, as well as very high-speed storage solutions to customers outside their organizations. Customers negotiate with these highly effective service providers on QoS requirements (i.e., hardware, software, and network combinations) and pricing (i.e., utility-based, fixed, or other pricing options).

Normally speaking, these types of advanced service arrangements are executed over some type of a VPN (Virtual Private Network) or dedicated line by narrowing the domain of security and event interactions. This is oftentimes somewhat limited in scope because the VPN or private line is very static in nature. This, in turn, reduces the visibility of the service provider to a lower and fixed scale, with the lack of complex resource sharing among heterogeneous systems and inter-domain networking service interactions.

That being said, the introduction of the Grid Computing principles related to resource sharing across VOs, along with the construction of VOs yielding inter-domain participation, will alter this situation. Specifically, this will enhance the utility model of Application Service Providers/Storage Service Providers (ASPs/SSPs) to a more flexible and mature value proposition.

Peer-to-Peer (P2P) Computing Systems

Similar to Grid Computing, P2P computing is a relatively new computing discipline in the realm of distributed computing. Both P2P and distributed computing are focused on resource sharing, and are now widely utilized throughout the world by the home, commercial, and scientific markets. Some of the major P2P systems are SETI@home[2] and file-sharing system environments (e.g., Napster, Kazaa, Morpheus, and Gnutella).

1. For more details on JINI and related projects, visit *www.jini.org*
2. Details on the SET1@home project can be found at *http://setiathome.ssl.berkeley.edu/*

The major differences between Grid Computing and P2P computing are centered on the following notable points:

1. They differ in their target communities. Grid Computing communities can be small with regard to the number of users, yet will yield more focused applications with higher levels of security requirements and application integrity. On the other hand, P2P systems define collaboration among a larger number of individuals and/or organizations, with a limited set of security requirements and a less complex resource sharing topology.
2. Grid Computing systems deal with more complex, more powerful, more diverse, and more highly interconnected sets of resources than P2P environments.

The convergence of these areas toward Grid Computing is highly probable since each discipline deals with the same problem of resource sharing among the participants in a VO. There has been some work done to date in the Global Grid Forum (GGF), which is focused on the merger of these complementary technologies for the interests of integrating a larger audience.

Cluster Computing Clusters are local to a domain and are constructed to solve inadequate computing power. Clustering is related to the pooling of computational resources to provide more computing power by parallel execution of the workload. Clusters are limited in scope with dedicated functionality and local to the domain; they are not suitable for resource sharing among participants from different domains. The nodes in a cluster are centrally controlled, and the state of the node is aware of the cluster manager. This forms only a subset of the Grid Computing principles of more widely available intra-/inter-domain communication and resource sharing.

Summary

This chapter introduced the *Grid Computing problem* in the context of a *VO*, combined with the proposed *Grid Protocol Architecture*. The overall Grid Protocol Architecture described in this chapter is effectively designed for controlled resource sharing with better interoperability among participants. This is, in practice, one commonly accepted and globally suggested solution to resolve the Grid Computing problem. This introduction to Grid Computing helps us to better understand aspects of the Grid Protocol Architecture's design, while understanding the Grid Computing protocols necessary for resource sharing with maximum interoperability.

In addition, this chapter discussed the relationships between Grid Computing systems and distributed systems by emphasizing exactly how these varying disciplines complement each other, while at the same time differ with respect to each other.

The next chapter will explore the future of Grid Computing, outlining more of the tactical perspectives of the technology, as well as the strategic perspectives.