Gestalt of the Grid

Gregor von Laszewski[1,*], Gail W. Pieper[1], Patrick Wagstrom[1,2]
[1] Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, U.S.A.
[2] Illinois Institute of Technology, 10 West 31st Street, Chicago, IL 60616

[*] Correspondence:
gregor@mcs.anl.gov
Phone: 630 252 0472, Fax: 630 252 1997

May 12, 2002

Contents

6 Gestalt of the Grid **5**

6.1 Introduction . . . 5

6.1.1 Motivation . . . 6

6.1.2 Enabling Factors . . . 7

6.2 Definitions . . . 8

6.3 Multi-faceted Grid Architecture . . . 11

6.3.1 N-Tier Grid Architecture . . . 11

6.3.2 Role-Based Grid Architecture . . . 11

6.3.3 Service-Based Grid Architecture . . . 13

6.3.4 Grid Challenges . . . 15

6.4 Grid Management Aspects . . . 15

6.4.1 Managing Grid Security . . . 16

6.4.2 Managing Grid Information . . . 18

6.4.3 Managing Grid Data . . . 19

6.4.4 Managing Grid Execution and Resources . . . 19

6.4.5 Managing Grid Software . . . 20

6.4.6 Managing Grid Hardware . . . 20

6.5 Grid Activities . . . 21

6.5.1 Grid Activities . . . 21

6.5.1.1 Global Grid Forum . . . 22

6.5.1.2 Production Grids . . . 22

6.5.1.2.1 DOE Science Grid . . . 22

6.5.1.2.2 TeraGrid . . . 23

6.5.1.2.3 NASA Information Power Grid . . . 23

6.5.1.2.4 EuroGrid . . . 24

6.5.1.2.5 Data Grid . . . 24

6.5.1.2.6 Asia-Pacific . . . 24

6.5.2 Grid Middleware . . . 25
6.5.2.1 The Globus Project . . . 25
6.5.2.1.1 Commodity Grid Kits . . . 26
6.5.2.1.2 Open Grid Services Architecture . . . 27
6.5.2.2 Legion . . . 28
6.5.2.3 Storage Resource Broker . . . 29
6.5.2.4 Akenti . . . 29
6.5.2.5 Network Weather Service . . . 30
6.5.3 High-Throughput Computing . . . 30
6.5.3.1 Condor . . . 30
6.5.3.2 NetSolve . . . 32
6.5.3.3 Ninf . . . 33
6.5.3.4 SETI@Home . . . 33
6.5.3.5 Nimrod-G . . . 34
6.6 Grid Application . . . 35
6.6.1 Astrophysics Simulation Collaboratory . . . 35
6.6.2 Particle Physics Data Grid . . . 37
6.6.3 NEESgrid . . . 37
6.7 Portals . . . 38
6.7.1 HotPage . . . 39
6.7.2 Webflow and Gateway . . . 40
6.7.3 XCAT . . . 40
6.7.4 UNICORE . . . 41
6.7.5 JiPANG . . . 42
6.7.6 PUNCH . . . 42
6.7.7 Access Grid . . . 43
6.7.8 Commercial Grid Activities . . . 43
6.8 Future and Conclusion . . . 44

Chapter 6

Gestalt of the Grid

6.1 Introduction

The *Grid approach* is an important development in the discipline of computer science and engineering. Rapid progress is being made on several levels, including the definition of terminology, the design of an architecture and framework, the application in the scientific problem solving process, and the creation of physical instantiations of Grids on a production level. This chapter provides an overview of important influences, developments, and technologies that are shaping state-of-the-art Grid computing. In particular, we address the following questions:

- What motivates the Grid approach? (see Section 6.1.1)
- What is a Grid? (see Section 6.2)
- What is the architecture of a Grid? (see Section 6.3)
- Which Grid research activities are performed? (see Section 6.5)
- How do researchers use a Grid? (see Section 6.7.7)
- What will the future bring? (see Section 6.8)

Before, we begin our discussion, we would like to start with an observation that leads us to the title of this chapter. A strong overlap between past, current, and future research in other disciplines influences this new area and makes answers to some of the questions complex. Moreover, though we are able to define the term *Grid approach*, we need to recognize that similar to the *Gestalt approach* in psychology, we face different responses by the community to this evolving field of research. Based on the Gestalt approach, which hypothesizes that an individual's perception of stimuli has an affect on their response, we will see a variety of stimuli on the Grid approach that influence current and future research directions.

We close this introductory section with a famous picture used in early psychology experiments. If we examine the drawing in detail, it will be rather difficult to decide what the different components represent in each of the interpretations. Although hat, feather, and ear are identifiable in the figure, one's interpretation (Is it an old woman or a young girl?) is based instead on "perceptual evidence." This figure should remind us to be open to individual perceptions about Grids and to be aware of the multifaceted aspects that constitute the **Gestalt of the Grid**.

6.1.1 Motivation

To define the term Grid we first identify what motivates its development. We provide an example from weather forecasting and modeling that includes a user community with strong influence in the newest trends of computer science over the last decades.

L. F. Richardson [70, 74] expressed the first modern vision for numerical weather prediction in 1922. Within two decades, the first prototype of a prediction system had been implemented by von Neuman, Charney, and others on the first generation of computers [72]. With the increased power of computers, numerical weather prediction became a reality in the 1960s and initiated a revolution in the field that we are still experiencing. In contrast to these early weather prediction models, today the scientific community understands that complex chemical processes and their interactions with land, sea, and atmosphere have to be considered.

Several factors make this effort challenging. Massive amounts of data must be gathered worldwide; that data must be incorporated into sophisticated models; the results must be analyzed; feedback must be provided to the modelers; and predictions must be supplied to consumers (see Figure 6.1).

Analyzing this process further we observe that the data needed as input to the models based on observations and measurements of weather and climate variables are still incomplete and sophisticated sensor networks must be put in place to improve this situation.

The complexity if these systems have reached a level where it is no longer possible for a single scientsit to process and manage the entire process; the era of the lonely scientist working in seclusion is coming to an end. Today, accurate weather models are derived by the sharing the intellectual property within a community of interdisciplinary researchers.

This increase in the complexity in the numerical methods and amount of data

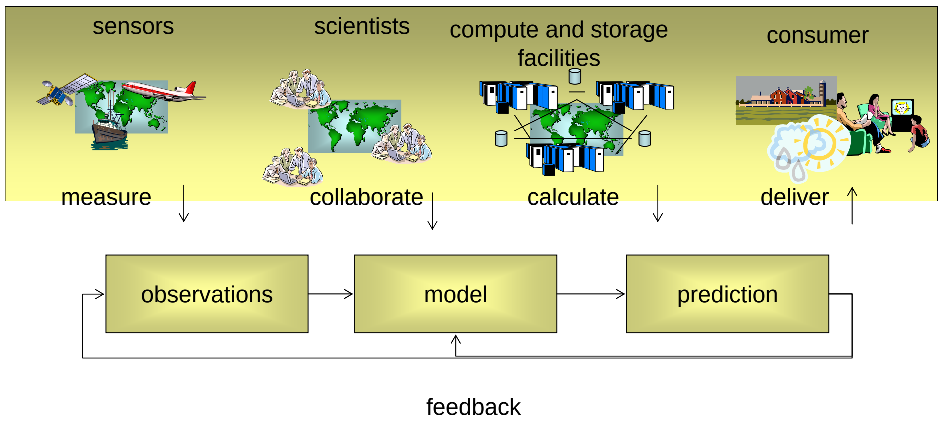

Figure 6.1: Weather forecasting is a complex process that requires a complex infrastructure.

required, along with the facet of community access, requires access to massive amounts of computational and storage resources. Although today's supercomputers offer enormous power, accurate climate and weather modeling require access to even larger resources that may be integrated from resources at dispersed locations.

Therefore, weather prediction promotes more than just the focus on making compute resources available as part of a networked environment. We have identified that need for an infrastructure that allows us to create from a dynamic, dispersed set of sensor, data, compute, collaboration, and delivery networks. Clearly, weather forecasting is a complex process that requires flexible, secure, coordinated sharing of a wide variety of resources.

6.1.2 Enabling Factors

When we look at why it is now possible to develop very sophisticated forecast models we see an increase in understanding, capacity, capability, and accuracy on all levels of our infrastructure.

Clearly, technology has advanced dramatically. Communication satellites and the Internet enable remote access to regional and international databases and sensor networks. Collaborative infrastructures (such as the Access Grid [29]) have moved exchange of information beyond the desktop. These advances have and will profoundly affect the way scientists work with each other. Compute power has also steadily increased. Indeed, for more than three decades, computer speed has doubled every eighteen months (supporting Moore's law [63]), and this trend is expected to last for at least the next decade. Furthermore, over the past five years network bandwidth has increased at a much larger rate, leading experts to believe that the network speed doubles every nine months. At the same time, the cost of production for network and computer hardware is decreasing.

We also observe a change in *modality of computer operation*. The first generation of supercomputers were high-end mainframes, vector processors, and parallel computers. Access to this expensive infrastructure was provided and controlled as part of a single institution within a single administrative domain. With the advent of network technologies, promoting connectivity between computers, and the creation of the Internet, promoting connectivity between different organizations, a new trend arose, leading away from the centralized computing center to a decentralized environment. As part of this trend, it was natural to collect geographically dispersed and possibly heterogeneous computer resources, typically as networks of workstations or supercomputers. The first connections between high-end computers to solve a problem in parallel on these machines were termed a metacomputer. (The term is believed to be originated as part of a gigabit testbed [61].) Much research in this area, some of which will be mentioned in this chapter, has been influential in shaping what we now term the Grid approach or concept. Thus, increases in capacity, capability, and modality are enabling a new way of doing *distributed science*. Additionally, the technology that was once viewed as specialized infrastructure is today becoming a commodity technology making it possible to resources for example through the use of the Internet [69] more easily.

This requirement and vision, which has become clearer over the last decades, now applies to many other disciplines that will provide commercial viability in the near future. It has a profound past, present, and future impact on several scientific disciplines, including computer science.

6.2 Definitions

In this section we provide the most elementary definition of the term Grid and its use within the community. As we have seen in the previous section the Grid approach has been guided by a complex and diverse set of requirements but at the same time provides us with a vision for an infrastructure that promotes sophisticated international scientific and business-oriented collaborations. Much research in this area, some of which will be mentioned in this chapter, has been influential in shaping what we now term the *Grid approach*:

Definition: Grid approach

> The Grid approach promotes a vision for sophisticated international scientific and business-oriented collaborations.

The term “Grid” is an analogy to the electric power grid that allows pervasive access to electric power. In a similar fashion, computational Grids provide access to pervasive collections of compute-related resources and services. As early as 1965 the designers of the Multics operating system envisioned and named requirements for a computer facility operating “like a power company or water company” [82], and others envisioned Grid-like scenarios [60]. However, we emphasize that our current understanding of the Grid approach goes far beyond simply sharing compute resources in a distributed fashion. Besides supercomputer and compute pools, Grids include access to information resources (such as large-scale databases) and access to knowledge resources (such as collaborative interactions between colleagues). Essential is that these resources may be at geographically disperse locations and may be controlled by different organizations. Thus, the following definition for a Grid is appropriate:

Definition: Grid

> An infrastructure that allows for flexible, secure, coordinated resource sharing among dynamic collections of individuals, resources, and organizations.

So far we have used the term Grid rather abstract manner. To distinguish the concept of a Grid from an actual instantiation of a Grid as a real available infrastructure we use the term production Grid. Such production Grids are typically shared among a set of users. The analogy in the electrical power grid would be a power company or agglomerate of companies that maintain their own grid while providing persistent services to the user community. Thus, the following definition is introduced:

Definition: Production Grid

> An instantiaion of a Grid that manifests itself by including a set of resources to be accessed by Grid users.

Additionally, we expect that multiple production Grids will exist and be supported by multiple organizations. Fundamental to the Grid is the idea of *sharing*.

Naturally, it should be possible to connect such Grids with each other as to share resources. Thus, it is important to define a set of elementary standards that assist to provide interoperability between production Grids.

Some production Grids are created based on the need to support a particular community. Although the resources within such a community are usually controlled in different administrative domains, they can be accessed as part of a *community production Grid*. Examples of production and community production Grids are introduced in Section 6.5.1.2.

Definition: Community Production Grid

A production Grid in which the creation and maintenance are performed by a community of users, developers, and administrators.

The management of a community production Grid is usually handled by a *virtual organization* [46], which defines the rules that guide membership and use of resources.

Definition: Virtual Organization

An organization that defines rules that guide membership and use of individuals, resources, and institutions within a community production Grid.

A typical Grid will contain a number of high end resources such as supercomputers or data storage facilities. As these resources can be consumed by users we term them in analogy to electrical power plants as follows:

Definition: Grid Plant

A high end resource that is integrated in a virtual organization and can be shared by its users.

The user on the other hand is able to access these resources through a user specific device such as a computer, handheld device, or a cell phone.

Definition: Grid Appliance

A device that can be integrated into a Grid while providing the user with a service that uses resources accessible through the Grid.

Grid appliances provide a portal that enables easy access, utilization, and control of resources available through a Grid by the user. We will define the term Grid portal in more detail in Section 6.7.

6.3 Multi-faceted Grid Architecture

A review of the literature about existing Grid research projects shows that three different architectural representations are commonly used. Each of these architectural views attempts to present a particular aspect of Grids. Thus, we believe it is important to recognize that the architecture of the Grid is multifaceted and an architectural abstraction should be chosen that fits best to describe the given aspect of the Grid research. Nevertheless, in each case one needs to consider the distributed nature and the unique security aspects.

Next we will describe in more detail these common architectural views.

6.3.1 N-Tier Grid Architecture

The N-tier application architecture provides a model for Grid developers to create flexible and reusable Grid applications. Decomposing a Grid application into tiers, allows developers to modify or add only to a specific layer, rather than to focus on the reimplementation of all parts of the application. N-tier application architectures are common within and are most often represented as part of the layer 7 of the OSI model [66]. Many Grid projects provide an N-tier architecture. The advantage of an ntier architecture is its familiarity and ints aplicability to many conceptual Grid problems that try to seperate issues between the application and the physical layer.

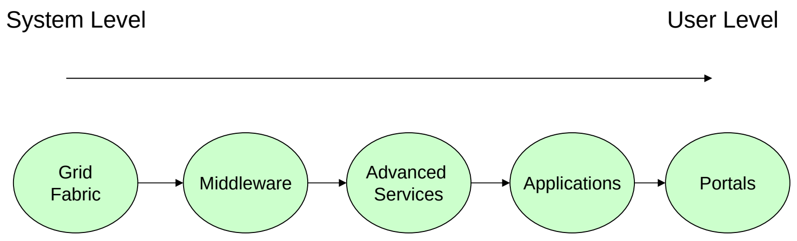

Figure 6.2: An n-tier Grid architecture based on an application users point of view.

6.3.2 Role-Based Grid Architecture

The secure access to a collectively controlled set of physical resources reused by applications motivates a role-based layered architecture [46, 47]. Within this architecture, it is easy to identify fundamental system components, specify the purpose and function of these components, and indicate how these components interact with one another. This architecture classifies protocols, services, application programming interfaces, and software development kits according to their roles in enabling resource sharing. It identifies five layers: fabric, connectivity, resource, collective, and application layer (see Figure 6.3). Interoperability is preserved by using a small standard set of protocols assisting in the secure exchange of information and data among single resources. These resources are managed by collective services in order to provide the illusion of a single resource to application designers and users.

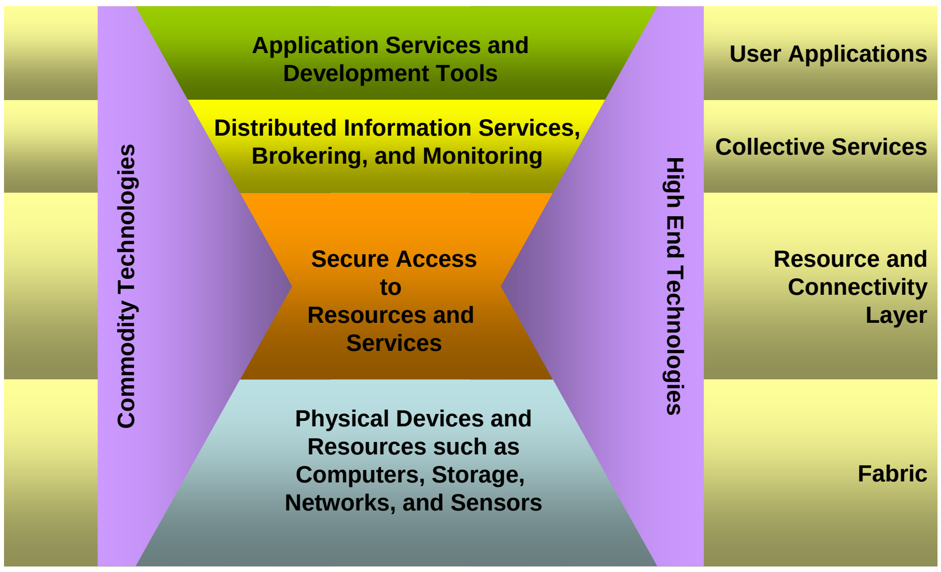
Application Services and Development Tools
Distributed Information Services, Brokering, and Monitoring
Secure Access to Resources and Services
Physical Devices and Resources such as Computers, Storage, Networks, and Sensors
Commodity Technologies
High End Technologies
User Applications
Collective Services
Resource and Connectivity Layer
Fabric

Figure 6.3: A role-based layered view of Grid architecture.

The layers within the architecture are defined as follows:

- The *fabric layer* contains protocols, application interfaces, and toolkits that allow development of services and components to access locally controlled resources, such as computers, storage resources, networks, and sensors.

- The *connectivity layer* includes the necessary Grid-specific core communication and authentication support to perform secure network transactions with the resources within the Grid fabric. This includes protocols and services allowing secure message exchange, authentication, and authorization. It is beneficial to develop a small set of standard protocols and services to provide the means of interoperability.

- The *resource layer* contains protocols that enable secure access and monitoring by collective operations.
- The *collective layer* is concerned with the coordination of multiple resources and defines collections of resources that are part of a virtual organization. Popular examples of such services are directories for resource discovery and brokers for distributed task and job scheduling.
- The *application layer* comprises the users' applications that are used within a virtual organization.

Each of these layers may contain protocols, application programming interfaces, and software development kits to support the development of Grid applications and services. A benefit of this architecture is the ability to bootstrap a complex Grid framework while successively refining it on various levels. We emphasize that this architecture can be supported with an immensely rich set of already defined application interfaces, protocols, toolkits, and services provided through commodity technologies and developments within high end computing. Reuse and extension of these standards, based on Grid specific requirements, will support the development of Grids.

6.3.3 Service-Based Grid Architecture

In near future, we will observe a shift within information technologies toward service oriented concepts. From the perspective of Grid computing, we define a service as a platform-independent software component, which is described with a description language and published within a directory or registry by a service provider (see Figure 6.4). A service requester can locate a set of services with a query to the registry, a process known as resource discovery. A suitable service can then be selected and invoked, a process known as binding [38, 41].

Definition: Service

A platform-independent software component published within a directory or registry by a service provider.

The usefulness of the service-based Grid architecture can be illustrated by scheduling a task on a computer cluster. First, we locate a set of possible resources. Next, we select a compute resource from this set where we would like to schedule our task. A criterion to select such a resource could be cost or load balance among the resources. Once a suitable resource is selected, we bind the task of execution to this resource. Figure 3 shows the parties and message exchanges that define a service-based model. An important aspect of services is the possibility to easily compose new services while using existing ones. This is enabled by the standard description, not only of the protocol, but also of the behavioral description of such a service.

Clearly, it is possible to develop complex flows between services. Since this service-based model deals with the use of asynchronous services, it will be important to deal appropriately with service guarantees in order to avoid deadlocks and other hazards.

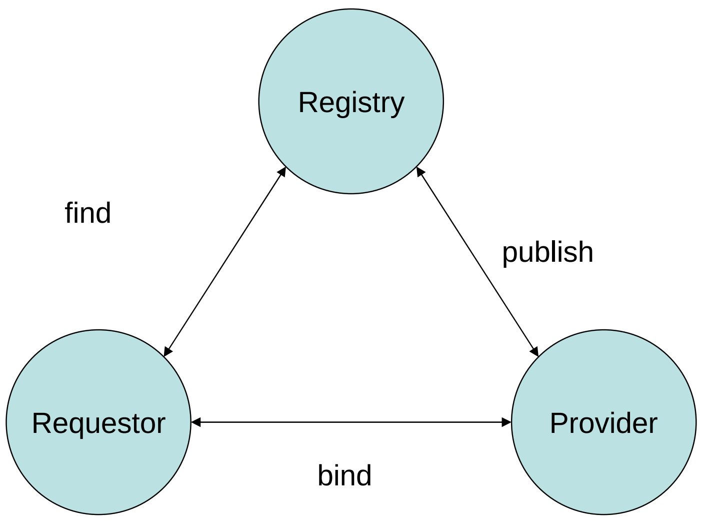

Figure 6.4: The service model allows the description of a provider service that can be published in a registry and be found and bound by a requestor.

The service-based concept has been in wide use, not only by the Grid community, but also by the business community. This fact has led to recent collaborative efforts between the Grid and the business community. An example of such an activity is the creation of the Open Grid Service Architecture, which we describe in more detail in Section 6.5.2.1.2.

6.3.4 Grid Challenges

Whatever the form of the Grid, we must consider, the dynamic, unpredictable properties of the Grid, while at the same time providing a reliable and persistent infrastructure. Additionally, we would like to enable open collaborations withouth neglecting protection of the collaboration with appropriate security restrictions. These apparent contradictions – desire for reliability vs. a potential unreliable infrastructure, or restricted vs. unrestricted access to information – provide complex challenges for Grids (see Figure 6.5). In order for Grids to become a reality, we must develop infrastructures, frameworks, and tools that address these complex management challenges and issues.

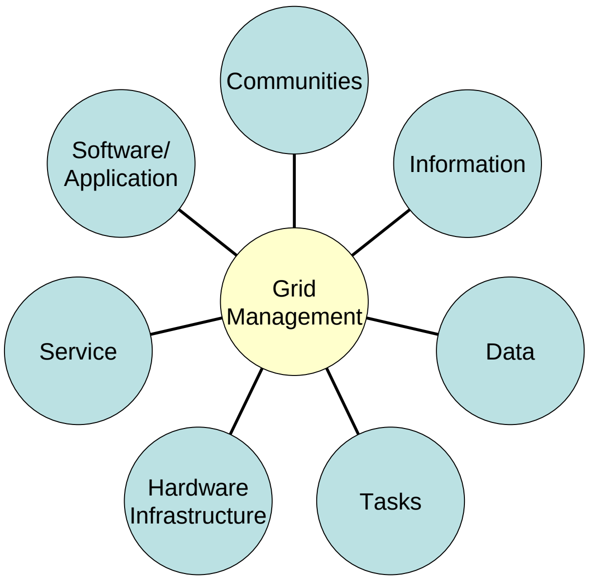

Figure 6.5: The Grid approach must deal with a complex *management challenge* in many areas.

6.4 Grid Management Aspects

A massively distributed and interconnected system entails management issues that go far beyond those of typical computers. Among these issues are the security of the system to maintain the overall integrity of the system; data and information management to ensure that the relevant data about users, systems, and experiments is available to users and programs on the Grid; execution and resource management to handle the allocation of resources and ensure that tasks are executed in a timely matter; software management to handle deployment of software packages; and hardware management to ensure that the physical base of the Grid stays running. This section addresses these issues and their relationship to the Grid.

6.4.1 Managing Grid Security

Since the Grid approach deals with heterogeneous and disperse resources and services, security aspects within Grids play an important role. Most commodity security services available today enable the interaction between two peers. The concepts used to enable this interaction are authentication, authorization, encryption, and nonrepudiation (see Figures 6.6 and 6.7)

Authentication deals with the verification of the identity of an entity within the Grid. Though this is commonly associated only with identification of a Grid user, the Grid also requires authentication of resources and services provided as part of the Grid.

Authorization deals with the verification of an action an entity can perform after authentication was successfully performed. Thus, policies must be established that determine the capabilities of allowed actions. A typical example is the use of a batch queue is allowed for a user A between 3 and 4 o'clock, but user B is allowed to use the queue only from 5 to 6 o'clock. In general, policies determine *who* can do *what*, *when*, and at *which* resource.

Encryption provides a mechanism for protecting the confidentiality of messages in transit between two peers.

Nonrepudiation deals with issues that provide data or message integrity, such as verifying that data was not changed accidentally or maliciously during message transmission.

Besides these general security issues, the Grid infrastructure poses unique requirements. For instance, it is unfeasible to authenticate via password challenges for a user on thousands of different resources.

Single sign-on is a mechanism that supports authentication to a large number of Grid resources on behalf of the user or resource by delegating the task of authentication to a service acting on behalf of the user (also called a proxy service). Such a service will typically create a temporary credential (often referred to as a secure proxy) that is used for authentication. An important factor to consider

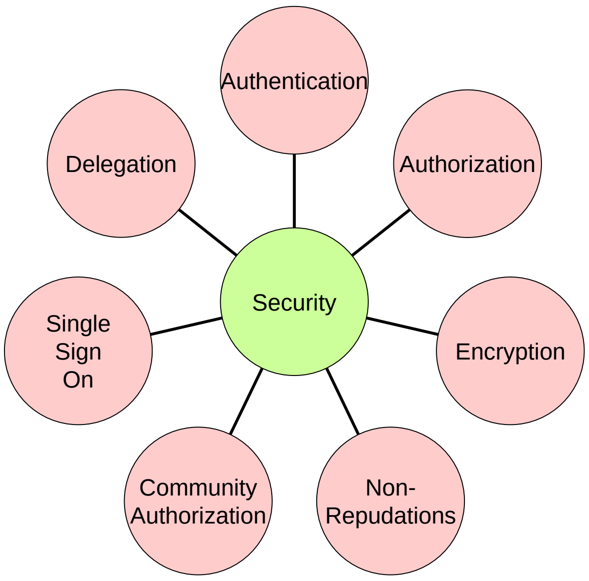

Figure 6.6: The issues to be addressed in security.

within single sign-on is that different domains may provide different local security mechanisms. Thus, any solution muse be able to deal with different identity mappings, such as Unix accounts accessible through PKI or Kerberos.

Delegation is the process of one Grid entity acting on behalf of another Grid entity. Delegation must be performed carefullybecause it is possible to create delegation chains. A simple exampleof such a chain is the initiation of a process on a resource D,initiated by a resource A, and subsequently delegated through B and C(A$\mapsto$B$\mapsto$C$\mapsto$D). In general, we observe the longer the chain, the greater the risk for misuse. Accordingly, it is desirable to create what we term limited delegation. This includes procurements for authentication restriction with more sophisticated Grid services. Thus, we can create a limited proxy that may, among other things, restrict use to a particular Grid resource.

Community authorization provides mechanisms for a virtual organization to define policies for groups of users that can be applied to enabling access control to resources by a community. This service is needed in case it is impossible or impractical to keep track of the access to a resource on a user-by-user basis. An authority that establishes trust between the peers regulates inclusion in such a community. In this sense community authorization enables single sign-on to

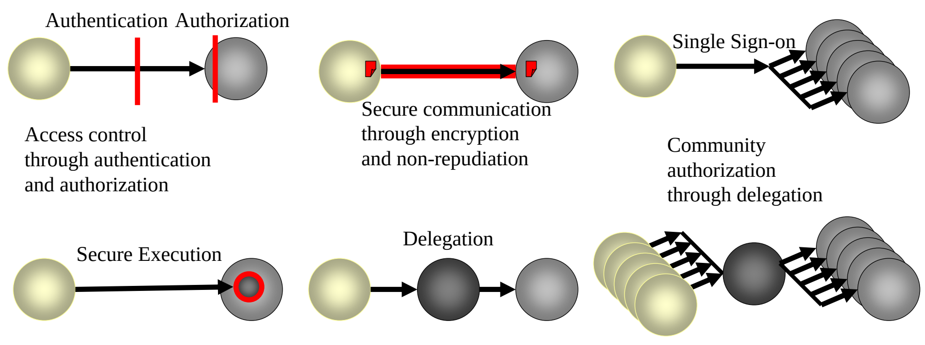

Figure 6.7: Cartoons of security concepts useful for Grids.

resources while being delegated to a trusted authority.

Secure execution is desired in environments where the user community becomes too large to handle. In these cases, it is important to provide a service that can run untrusted applications (those submitted by the users) in a trusted environment (the cluster at a compute center or a Grid); the concept of virtual machines essential for such a service.

We must consider the user community when designing a security infrastructure for applications and services running in a Grid environment. Many users are unwilling to deal with obtrusive security procedures, but at the same time expect a reasonable level of security. Hence, it is of utmost importance to present the security mechanisms to the users in an easy and mostly transparent way. A minimum level of understanding by users is necessary, so that they can specify their own security requirements and understand the security guarantees or risks of the Grid. In this respect, an *educational service* provided as part of the strategy of production Grids can offer the necessary explanations and guidance for accessing Grid resources and developing secure service.

6.4.2 Managing Grid Information

Within Grids, information about the users and the system is critical. User information helps to establish collaborative sessions, and system information helps users select the appropriate resources and applications. The availability of such information is important for the maintenance, configuration, and use of the heterogeneous and dynamically changing Grid infrastructure. Characteristics that must be imposed on such an information service to support Grids include

- uniform, flexible access to information;
- scalable, efficient access to dynamic data;and
- access to multiple information sources.

The creation of such an information service must be an integral part of each Grid toolkit and application. In the past, distributed directories have provided such a service. Often a centrally maintained relational database may serve the same purpose. In any case, the design of a scalable information service must consider the distributed nature of the Grid. Equally important is the fact that the resource owners may not wish to export the information about their system to unauthorized users. Although restricted access to information is already possible, it is not adequately addressed in the first generation of prototype production Grids.

6.4.3 Managing Grid Data

Each program executed in a Grid is dependent on data, and the data requirements for applications running on the Grid are enormous. For example, gathering the data for a meterological forcast requires the processing and storage of petabytes of data each day. To compensate for limited storage capacities at remote sites, services that perform delivery on demand may augment the data with a lifetime to limit the amount of actual data in the Grid. If the calculation cannot be performed on the server where the data is located, the user must be able to efficiently replicate that data elsewhere. Thus, a reliable file transfer service must be provided to move the data between source and destination on behalf of the issuing client. Filters can be used to reduce the amount of data during a transfer, based on metadata attached with the file. If the data can be created with less effort than the actual data transfer, it may be advantageous to augment data with pedigree information about how to regenerate the data instead of storing the data.

6.4.4 Managing Grid Execution and Resources

Calculations on resources within the Grid are controlled by execution services. The simplest execution service is part of the operating system and allows execution of jobs and tasks on a single resource. A Grid security infrastructure must be in place that provides authentication and authorization mechanisms to govern the use of this resource. Batch queuing systems provide a convenient way to extend such an execution service to a cluster, a parallel computer, or a supercomputer. To enable the use of multiple instances of such resources, a resource co-allocation mechanism is needed. Such a mechanism will identify a suitable set of resources based on the Grid information service and verify that the selected resources are available (or to fulfill the user's request if they are not), reserve the resources, and finally execute the user's task on this agglomeration of resources.

Algorithms to control the collective use of such resources may be quite complex. Since the algorithmic implications for scheduling in such an environment are an NP complete problem, heuristics may be used to solve the scheduling problem and to guarantee the execution of the tasks. Researchers are currently exploring the use of combinatorial optimization strategies, stochastic sampling, economic models, and agent-based systems. Smart services are necessary that can deal with deadlock prevention, avoidance, and QoS guarantees on the local and global scale. Often, complicated workflows must be formulated as part of the complex interdisciplinary applications run by scientists on Grids. Thus, it is necessary to provide workflow management services that allow control of the flow of data and applications as part of the problem-solving process.

6.4.5 Managing Grid Software

Deployment of applications, components, and services in a distributed heterogeneous environment is a challenging problem. Of particular concern is guaranteeing interoperability between different versions of software and libraries on already installed and operational software and services. The use of the Grid service model described earlier offers a partial solution to this problem by providing metadata to each application and service installed on the Grid that can be queried through the Grid information service. In this way, it is possible to include portability data within the infrastructure, which will be used as part of an authorization service to verify whether services or applications can interoperate.

6.4.6 Managing Grid Hardware

The resource providers are responsible for hardware management on the Grid. Notifications about downtimes and maintenance upgrades must be available through the information service in order to simplify the user's search for suitable resources with service guarantees. In general, hardware management must be augmented with an appropriate infrastructure on the hardware service provider side. Quality of service augmentations on the hardware level, such as networks, could provide a profound advantage for future Grid infrastructures.

6.5 Grid Activities

We have organized our discussion of Grid project into three classes: community activities, development toolkits, and applications (see Figure 6.8). Within each class, we describe various activities in being performed by the Grid community.

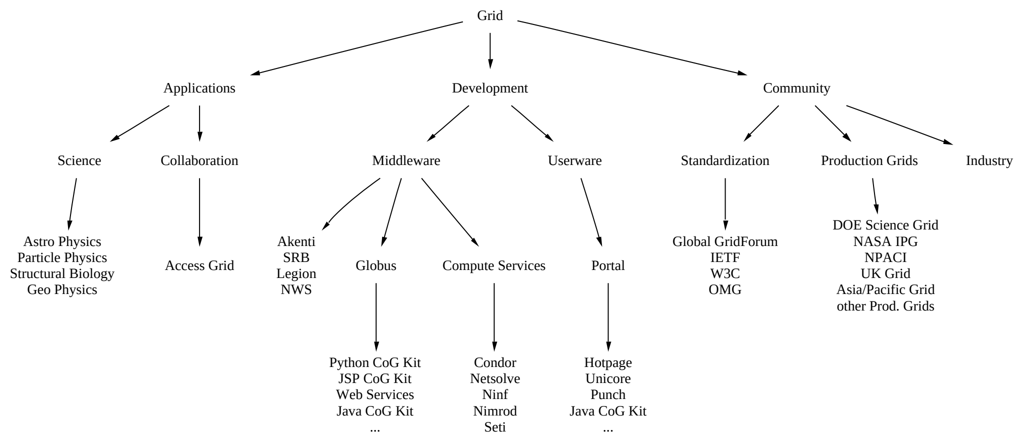

Figure 6.8: A simple classification of Grid activities: community activities, development tools and applications.

6.5.1 Grid Activities

A variety of activities are performed by the community. Each of these activities has a profound impact on the development of Grids. We identify three basic Grid user communities and the activities they perform:

- *Development:* Grid programmers who develop services in a collaborative fashion for deployment in the Grid.
- *Application:* Scientific or application users who access the services provided as part of the Grid.

- *Community Building:* Administrators who deploy services and applications in production Grids in order to make them accessible to others.

While today's Grid users include mostly large-scale scientific application users and developers, we expect that with the availability of robust Grid toolkits the community will expand to the financial sector, the health care sector, small industries, and even the common household user needing access to services resources accessible through the Grid. Thus, the Grid will be instrumental in furthering the scientific discovery process [19] while developing the next generation of *community* problem-solving environments.

6.5.1.1 Global Grid Forum

The Global Grid Forum (GGF) is an international community-initiated forum of individual researchers and practitioners working on various facets of Grids. The mission of the GGF is to promote and develop Grid technologies and applications through the development and documentation of "best practices," implementation guidelines, and standards, with an emphasis on "rough consensus and running code." The objective is to support with such standards the creation of production Grids; address infrastructure obstacles inhibiting the creation of these Grids; perform educational outreach; and facilitate the use of Grid technologies within diverse application communities. Based on the Internet Engineering Task Force model, the GGF contains several area groups and, within these areas, working groups dealing with a particular Grid-related problem. The current areas are information services, security, scheduling and management, performance, architecture, data, and applications and models. Regular meetings are held in which over two hundred organizations from more than thirty countries are represented [25].

6.5.1.2 Production Grids

A number of national and international community production Grids have been established in the past few years. Each is part of a virtual organization spanning multiple administrative domains and enabling access to high-end resources such as supercomputers, mass storage systems, and advanced instruments. A well-trained administrative staff is responsible for deploying services and components in such collectively maintained production Grids.

6.5.1.2.1 DOE Science Grid The Department of Energy (DOE) Science Grid is a pilot program to provide an advanced distributed computing infrastructure based on Grid middleware and tools to enable the degree of scalability in scientific computing necessary for DOE to accomplish its science missions. Emphasis is placed on making the construction and use of large-scale heterogeneous systems as easy as using today's desktop environments. The DOE Science Grid [40] is part of a large initiative, entitled Scientific Discovery through Advanced Computing (SciDAC) [19], that was started in FY2001. The objective of SciDAC is to develop the scientific computing software and hardware infrastructure needed for terascale computers to advance its research programs in basic energy sciences, biological and environmental research, fusion energy sciences, and high-energy and nuclear physics.

6.5.1.2.2 TeraGrid The *TeraGrid* [21] project seeks to build and deploy the world's largest, fastest, most comprehensive, distributed infrastructure for open scientific research. Upon completion, the TeraGrid will include 13.6 teraflops of Linux cluster computing power distributed at four sites: the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign; the San Diego Supercomputer Center (SDSC) at the University of California - San Diego; Argonne National Laboratory in Argonne, Illinois; and the California Institute of Technology (Caltech) in Pasadena. The TeraGrid will include other distributed facilities capable of managing and storing more than 450 terabytes of data, high-resolution visualization environments, and toolkits for Grid computing. A high-speed network, which will operate between 50 and 80 gigabits/second, will permit the tight integration of the components in a Grid. The $53 million project is funded by the National Science Foundation and includes corporate partners. The TeraGrid benefits from other Grid-related activities performed at the partner sites through the National Computational Science Alliance (Alliance) [9, 78, 10] and the National Partnership for Advanced Computational Infrastructure (NPACI) [11]. The Alliance and NPACI is supporting the TeraGrid activities through their partners and infrastructure/building activities and their current and future Grid infrastructures.

6.5.1.2.3 NASA Information Power Grid The NASA Information Power Grid project was initiated from a series of workshops in autumn of 1997. The goal is to provide seamless access to resources between NASA sites and a few selected NPACI sites for application development. These applications are likely to include aeronautics and other areas of interest to NASA, such as space sciences and earth sciences. The requirements NASA will address first are seamless access to distributed legacy applications via networks, cross-platform computational and interactive visualization of large three-dimensional data sets, intelligent and distributed data mining across unspecified heterogeneous data sources, agent technologies, privacy and security, and tools for the development of multidisciplinary systems. Additionally, NASA must deal with a number of real-time requirements for aircraft operations systems[15]. The current hardware resources included in the prototype Information Power Grid are based on Globus technology and comprise approximately 1,500 CPU nodes in six SGI Origins distributed across several NASA centers. Also included are 10-50 terabytes of securely accessible mass storage, several workstation clusters with approximately 100 CPUs, and a Condor pool with 300 workstations.

6.5.1.2.4 EuroGrid EuroGrid [14] is an application testbed for the European Grid community. It is supported as a shared cost research and technology development project between the European Commission and its eleven partner institutions. It will demonstrate the use of Grids in selected scientific and industrial communities, address the specific requirements of these communities, and highlight the benefits of using Grids. The objectives are to establish and operate a European Grid between several of Europe's High Performance Computing centers. Besides developing Grid software and applying it within state-of-the-art applications such as bio-molecular simulations, weather prediction, coupled CAE simulations, structural analysis, and real-time data processing, the alignment with commercial partners is intended to productize the software and provide supported.

6.5.1.2.5 Data Grid The DataGrid [12] project is funded by European Community. The objective is to enable next-generation scientific exploration that requires intensive computation and analysis of shared large-scale databases, from hundreds of terabytes to petabytes, across widely distributed scientific virtual communities. The initiative is led by CERN, the European Organization for Nuclear Research, together with five other main partners and fifteen associated partners. Major application areas are quantum chromodynamics, Earth observation, and human health research.

6.5.1.2.6 Asia-Pacific The ApGrid [13] is a partnership for Grid computing in the Asia Pacific region. So far, it includes about thirty institutions. One of the important objectives of ApGrid is building an international Grid testbed. The current technology plan includes the Globus Toolkit as its underlying infrastructure.

6.5.2 Grid Middleware

The collection of APIs, protocols, and software that allow creation and use of a distributed system, such as a Grid, is called *middleware*. It is at a lower level than end-user applications while being at a higher level than the underlying network transport methods. A variety of middleware packages are available, of which we shall examine a select few.

6.5.2.1 The Globus Project

Over the past few years, the Globus Project has contributed in many ways to the Grid effort. It has five thrust areas. First, the Globus Project conducts research on Grid-related issues such as resource management, security, information services, data management, and application development environments. Second, the Globus Project is developing open-source, open architecture Grid software, called the Globus Toolkit. A growing number of research institutes and companies have committed to supporting this open-source activity. Third, the Globus Project assists in the planning and building of large-scale testbeds, both for research and for production use by scientists and engineers. Fourth, the Globus Project collaborates in a large number of application-oriented efforts that develop large-scale Grid-enabled applications in collaboration with scientists and engineers. Fifth, the Globus Project is committed to community activities that include educational outreach and participation in defining Grid standards as part of the Global Grid Forum.

The Globus Toolkit is modular, enabling users to choose the components needed for the development of Grid-enabled applications.

Security is an important aspect of the Globus Toolkit. The Grid Security Infrastructure (GSI) uses public key cryptography as the basis for its functionality. It enables key security services such as mutual authentication, confidential communication, delegation, and single sign-on. GSI builds the core for implementing other Globus Toolkit services..

Communication within Globus is handled through the GlobusIO library, which provides TCP, UDP, IP multicast, and file I/O services with support for security, asynchronous communication, and quality of service. An important tool provided by the Globus Project is MPICH-G2, which supports MPI across several distributed computers. MPICHG2 was used at SC2001 in an astrophysical calculation and received the Gordon Bell Prize [55].

Information about a Grid is handled through the Metacomputing Directory

Service. The concept of a directory service for the Grid was first defined in [**?**] and later refined in [39]. The Metacomputing Directory Service manages information about entities in a Grid in a distributed fashion. The current implementation of MDS is based on the Lightweight Directory Access Protocol (LDAP). This protocol enables uniform querying of system information from a variety of system components, and for optionally constructing a uniform namespace for resource information across a system that may involve many organizations.

Resource management within Globus is handled through a layered system in which high-level global resource management services are build on top of local resource allocation services.The current Globus resource management system comprises three components: (1) an extensible resource specification language that serves as a method for exchanging information about resource requirements between all of the components in the Globus resource management architecture; (2) a standardized interface to local resource management tools including LSF, NQE, LoadLeveler, and Condor; and (3) a resource coallocation service that allows construction of sophisticated co-allocation strategies that allows use of multiple resources concurrently.

Data management is supported by integration of the GSI protocol to access remote files through, for example, the HTTP and the FTP protocols.

Data grids are supported through replica catalog services in the newest release of the Globus Toolkit. These services allow copying of the most relevant portions of a data set to local storage for faster access. Installation of the extensive toolkit is enabled through a packaging toolkit that can generate custom-designed installation distributions.

Current research activities include the creation of a community access server, restricted proxies for placing additional authorization requests within the proxy itself, data grids, quality of service, and integration within commodity technologies such as the Java framework and web services. Future versions of the Globus Toolkit will integrate the Grid architecture with Web services technologies.

6.5.2.1.1 Commodity Grid Kits The Globus Project provides a small set of useful services, including authentication, remote access to resources, and information services to discover and query such remote resource. Unfortunately, these services may not be compatible with the commodity technologies used for application development by the software engineers and scientists.

To overcome this difficulty, the Commodity Grid project is creating *Co*mmodity *G*rid Toolkits (CoG Kits) that define mappings and interfaces between Grid services and particular commodity frameworks. Technologies and frameworks of interest include Java, Python, CORBA [79], Perl, Web Services, .NET, and JXTA.

Existing Java [80] and Python CoG Kits provide the best support for a subset of the services within the Globus Toolkit. The Python CoG Kit uses SWIG in order to wrap the Globus C-API, while the Java CoG Kit is a complete reimplementation of the Globus protocols in Java. The Java CoG Kit is done in pure Java and provides the ability of using a pure Java GRAM service. Although the Java CoG Kit can be classified as middleware for integrating advanced Grid services, it can also be viewed both as a system providing advanced services currently not available in the Globus Toolkit and as a framework for designing computing portals [81]. Both the Java and Python CoG Kits are popular with Grid programmers and have been used successfully in many community projects.

6.5.2.1.2 Open Grid Services Architecture One of the major problems facing Grid deployment is the variety of different "standards", protocols, and difficult-to-reuse implementations. This situation is exacerbated by the fact that much of the Grid development has been done separately from corporate-distributed computer development. As a result, a chasm has begun to appear [52].

The Open Grid Services Architecture (OGSA) is an effort to utilize commodity technology to create a Grid architecture. OGSA utilizes the Web service descriptions as a method to bring concepts from web services into the Grid.

In OGSA, everything is a network-enabled service that is capable of doing some work through the exchange of messages. Such "services" include compute resources, storage resources, programs, networks, databases, and a variety of tools. When an OGSA service conforms to a special set of interfaces and support standards, it is deemed a "Grid service." These Grid services have the ability maintain their state; hence, it is possible to distinguish one running Grid service instance from another. Under OGSA, Grid services may be created and destroyed dynamically. To provide a reference mechanism for a particular Grid service instance and its state, each instance has a unique Grid service handler (GSH).

Because a Grid service instance may outlast the protocol on which it initially runs, the GSH contains no information about protocols or transport methods, such as an IP address or XML schema version. Instead, this information is encapsulated into a Grid service reference (GSR) which can change over time. This strategy allows the instance to upgrade or add new protocols.

To manipulate Grid services, OSGA has interfaces that handle and reference abstractions that make up OGSA. These interfaces can vary from service to service; however, the discovery interface must be supported on all services to allow the location of new Grid service instances.

Using such an object-oriented system offers several advantages. All components are virtualized, removing many dependency issues and allowing mapping of multiple logical resources into one physical resource. Moreover, because there is a consistent set of interfaces that all services must provide, construction of complex services is greatly simplified. Together these features allow for mapping of service semantics onto a wide variety of platforms and communication protocols.

When OGSA is combined with CoG kits, a new level of ease and abstraction is brought to the Grid. Together, these technologies will form the basis for the forthcoming Globus 3.0[48].

6.5.2.2 Legion

Legion is a Grid software project developed at the University of Virginia. Legion addresses Grid key issues such as scalability, programming ease, fault tolerance, security, and site autonomy. The goal of the Legion system is to support large degrees of parallelism in application code and to manage the complexities of the physical system for the user. Legion seamlessly schedules and distributes the user processes on available and appropriate resources while providing the illusion of working on a single, virtual machine.

As does other Grid middleware, Legion provides a set of advanced services. These include the automatic installation of binaries, a secure and shared virtual file system that spans all the machines in a Legion system, strong PKI-based authentication, flexible access control for user objects, and support of legacy codes execution and their use in parameter space studies.

Legion's architecture is based on an object model. Each entity in the Grid is represented as an active object that responds to member function invocations from other objects. Legion includes several core objects, such as compute resources, persistent storage, binding objects that map global to local process IDs, and implementation objects that allow the execution of machine code. The Legion system is extensible and allows users to define their own objects. Although Legion defines the message format and high-level protocol for object interaction, it does not restrict the programming language or the communications protocol.

Legion has been used for parameter studies, ocean models, macromolecular simulations, and particle-in-cell codes. Legion is also used as part of the NPACI production Grid; a portal eases the interaction with the production Grid using Legion.

6.5.2.3 Storage Resource Broker

The Storage Resource Broker (SRB) [20] developed by the San Diego Supercomputer Center is client-server middleware that provides a uniform interface for connecting to heterogeneous remote data resources and accessing replicated data sets. The SRB software includes a C client library, a metadata server based on relational database technology, and a set of Unix-like command line utilities that mimic, for example, ls, cp, and chmod SRB enables access to various storage systems including the Unix file system, archival storage systems such as UNITREE [8] and HPSS [6] and database Large Objects managed by various database management systems such as DB2, Oracle, and Illustra. SRB enables access to data sets and resources based on their attributes rather than their names or physical locations. Forming an integral part of SRB are collections that define a logical name given to a set of data sets. A Java-based client GUI allows convenient browsing of the collections. Based on these collections, a hierarchical structure can be imposed on data, thereby simplifying the organization of data in a manner similar to a Unix file system. In contrast to the normal Unix file system, however, a collection can encompass data that is stored on remote resources. To support archival mass storage systems, SRB can bind a large set of files (that are part of a collection) in a container that can be stored and accessed as single file. Additionally, SRB supports three authentication schemes: GSI, SEA (an RSA-based encryption scheme), and plain text password. Furthermore, SRB can enable access control to data to groups of users. Other features of SRB include data replication, execution of user operations on the server, data reduction prior to a fetch operation by the client, and monitoring.

6.5.2.4 Akenti

Akenti is a security model and architecture providing scalable security services in Grids. The project goals are to (1) achieve the same level of expressiveness of access control that is accomplished through a local human controller in the decision loop, and (2) accurately reflect existing policies for authority, delegation, and responsibilities. For access control, Akenti uses digitally signed certificates that include the user identity authentication, resource usage requirements (or use-conditions), user attribute authorizations (or attribute certificates), delegated authorization, and authorization decisions split among on-line and off-line entities. All of these certificates canbe stored remotely from the resources. Akenti provides a policy engine that the resource server can call to find and analyze all the remote certificates. It also includes a graphical user interface for creating use-conditions and attribute certificates.

6.5.2.5 Network Weather Service

Network Weather Service (NWS) [51] is a distributed monitoring service that periodically records and forecasts the performance of various network and computational resources over time. The service is based on a distributed set of performance sensors that gather the information in a central location. This data is used by numerical models to generate forecasts (similar to a weather forecasting). The information also can be used by dynamic schedulers to provide statistical quality-of-service readings in a Grid. Currently, the system supports sensors for end-to-end TCP/IP performance measuring bandwidth and latency, available CPU percentage, and available nonpaged memory. The forecast models include mean-based methods, which use some estimate of the sample mean as a forecast, and median-based methods, which use a median estimator, and autoregressive methods. While evaluating the accuracies of the prediction during runtime, NWS is able to configure itself and chose the forecasting method (from those that are provided with NWS) that best fits the situation. New models can be included in NWS.

6.5.3 High-Throughput Computing

High-throughput computing is an extension of the concept of supercomputing. While typical supercomputing focuses on floating-point operations per second (flops), high-throughput systems focus on floating-point operations per month or year [24]. The projects listed in this section are projects that provide increased performance for long term calculations by utilizing distributed commodity hardware in a collaborative method.

6.5.3.1 Condor

Condor is a system to utilize idle compute cycles on workstations while distributing a number of queed jobs to them. Condor focusing on high-throughput computing, rather than high performance computing [77]. Condor maintains a pool of computers while using a centralized broker to distribute jobs based on load information or preference asseret with the jobs to be executed. Condor provides a broker that identifies in the pool of resources idle computers with available resources on which to run the program (thus, the metaphor of a condor soaring over the desert looking for food).

The proper resources are found through the ClassAds mechanism of Condor. This mechanism allows each computer in the pool to advertise the resources that it has available and publish them in a central information service. Thus, if a job is specified to require 128 megabytes of RAM, it will not be placed on a computer with only 64 megabytes or RAM [24].

The ever-changing topology of workstations does, of course, pose a problem for Condor. When a user returns to his computer, he usually wants it to stop running Condor processes. To address this issue, the program uses the checkpoints described above and restarts on another host machine. Under Condor, allows the specification of elementary authorization policies, such as user A is allowed to use the machine but not user B, and the definition of a policies for running jobs in the background or when the user is not interactively using the machine. Such authorization frameworks have been successfully reused in other projects such as SETI@Home [56, 44, 43, 42].

Today, condor also includes client side brokers that handle more complex tasks such as job ordering via acyclic graphs and time management features. To prevent monopolizing the resources by a single large application Condor can use a fair scheduling algorithm. A disadvantage with the earlier condor system was that it was difficult to implement a co-allocation of resources that are not part of a workstation but of a supercomputing batch queue system. To utilize also batch queues within a pool, condor introduced a mechanism that provides the ability to integrate resources for a particular period of time into a pool. This concept is also known as glide-in, which are enebled through a Globus backend. Using this technique, a job submitted on a Condor pool may be executed elsewhere on another computing Grid. Currently Condor is working with the Globus Project to provide the necessary resource sharing [77].

Much of Condor's functionality results from the trapping of system calls by a specialized version of GLIBC that C programs are linked against. Using this library, most programs require only minor (if any) changes to the source code. The library redirects all I/O requests to the workstation that started the process. Consequently, workstations in the Condor pool do not require accounts for everyone who can submit a job. Rather, only one general account for Condor is needed. This strategy greatly simplifies administration and maintenance. Moreover, the special GLIBC library provides the ability to checkpoint the progress of a program. Nevertheless, condor provides also a mechanism that makes it possible to run jobs unchanged, but much of the advanced features such as checkpointing and restarting can not be utilized.

Additional, Grid functionality has been included with the establishment of so called Condor flocks that may represent pool in different administrative domains. Policy agreements between these flocks enable the redistribution of migratory jobs between these flocks [43, 42].

6.5.3.2 NetSolve

NetSolve, developed at the University of Tennessee's Innovative Computing Lab, is a distributed computing system that provides access to computational resources across a heterogeneous distributed environment via a client-agent-server interface [33, 16].

The entire NetSolve system is viewed as a connected non-directed graph. Each system that is attached to NetSolve can have different software installed on it. Users can access NetSolve and process computations through client libraries for C, Fortran, Matlab, and Mathematica. These libraries can access numerical solvers such as LAPACK, ScaLAPACK, and PETSc. When a computation is sent to NetSolve, the agent uses a "best-guess" methodology to determine which server to send the request to. That server then does the computation and returns the result using the XDR format [36]. Should a server process terminate unexpectedly while performing a computation, the computation is automatically restarted on a different computer in the NetSolve system. This process is transparent to the user and usually has little impact other than a delay in getting the results.

Because NetSolve can use multiple computers at the same time through non-blocking calls, the system has an inherent amount of parallelism. This, in one sense, makes it easy to write parallel C programs.

The NetSolve system is still being actively enhanced and expanded. New features included a graphical problem description file generator, Kerberos authentication, and additional mathematical libraries [26].

NetSolve's closest relative is Ninf (see Section 6.5.3.3). Work has been done on software libraries that allow routines written for Ninf to be run on NetSolve and vice versa. Currently, however, there are no known plans for the two projects to merge [33].

6.5.3.3 Ninf

Ninf (Network Information Library for High Performance Computing) is a distributed remote procedure call system with a focus on ease of use and mathematical computation. It is developed by the Electrotechnical Laboratory in Tsukuba, Ibaraki, Japan.

To execute a Ninf program, a client calls a remote mathematical library routine via a metaserver interface. This metaserver then brokers various requests to machines capable of performing the computation. Such a client-agent-server architecture allows a high degree of fail-safety for the system. When the routine is finished, the metaserver receives the data and transfers it back to the client.

The Ninf metaserver can also automatically order requests. Specifically, if multiple dependent and independent calculations need to take place, the independent ones will execute in parallel while waiting for the dependent calculations to complete.

Bindings for Ninf have been written for C, Fortran, Java, Excel, Mathematica, and Lisp. Furthermore, these bindings support the use of HTTP GET and HTTP PUT to access information on remote Web servers. This feature removes the need for the client to have all of the information and allows low-bandwidth clients to run on the network and receive the computational benefits the system offers [64].

Several efforts are under way to expand Ninf into a more generalized system. Among these efforts are Ninflet, a framework to distribute and execute Java applications, and Ninf-G a project designed a computational RPC system on top of the Globus Toolkit [71].

6.5.3.4 SETI@Home

SETI@Home, run by the Space Science Lab at the University of California Berkeley, is one of the most successful coarse grain distributed computing systems in the world. Its goal is to integrate compute resources on the Web as part of a collective of independent resources that are plentiful and can solve many independent calculations at the same time. Such a system was envisioned as a way to deal with the overwhelming amount of information recorded by the Arecibo radio telescope in Puerto Rico and the analysis of the data. The SETI@home project developed stable and user appealing screen savers for Macintosh and Windows computers and a command-line client for Unix systems [56, 62] that started to be widely used in 1999.

At its core, SETI@Home is a client-server distributed network. When a client is run, it connects to the SETI@Home work unit servers at the University of California - Berkeley and downloads a packet of data recorded from the Arecibo telescope. The client then performs a fixed mathematical analysis on the data to find signals of interest. At the end of analysis, the results are sent back to SETI@Home, and a new packet is downloaded for the cycle to repeat.

Packets of information that have been shown to have useful information are then analyzed again by the system to ensure there was no client error in the reporting of the data. In this way, the system shows resiliency toward modified clients, and the scientific integrity of the survey is maintained [57]. To date, SETI@Home has accumulated more than 900,000 CPU-years of processing time from over 3.5 million volunteers around the globe. The entire system today averages out to 45 Tflops, which makes it the world's most powerful computing system by a big margin [34]. One of the principal reasons for the project's success is its noninvasive nature; running SETI@Home causes no additional load on most PCs where it is run only during the inactive cycles. In addition, the system provides a wealth of both user and aggregate information and allows organizations to form teams for corporations and organizations, which then have their standings posted on the Web site. SETI@Home was also the first to mobilize massive amounts of participates by creating a sense of community and project the goals of the scientific project to large amounts of non scientific users.

SETI@Home was originally planned in 1996 to be a two-year program with an estimated 100,000 users. Because of its success, plans are now under way for SETI@Home II, which will expand the scope of the original project [28]. Multiple other topic such as protein folding have also been adapted [4].

6.5.3.5 Nimrod-G

Nimrod was originally a metacomputing system for parameterized simulations. Since then it has evolved to include concepts and technologies related to the Grid. Nimrod-G is an advanced broker system that is one of the first systems to account for economic models in scheduling of tasks. Nimrod-G provides a suite of tools that can be used to generate parameter sweep applications, manage resources, and schedule applications. It is based on a declarative programming language and an assortment of GUI tools.

The resource broker is responsible for determining requirements that the experiment places on the Grid and for finding resources, scheduling, dispatching jobs, and gathering results back to the home node. Internal to the resource broker are several modules:

- The *task-farming agent* is a persistent manager that controls the entire experiment. It is responsible for parameterization, creation of jobs, recording of job states, and communication. Because it caches the states of the experiments, an experiment may be restarted if the task-farming agent fails during a run.

- The *scheduler* handles resource discovery, resource trading, and job assignment. In this module are the algorithms to optimize a run for time or cost. Information about the costs of using remote systems is gathered through resource discovery protocols, such as MDS for the Globus Toolkit.

- *Dispatchers* and *actuators* deploy agents on the Grid and map the resources for execution. The scheduler feeds the dispatcher a schedule, and the dispatcher allocates jobs to the different resources periodically to meet this goal.

The agents are dynamically created and are responsible for transporting the code to the remote machine, starting the actual task and recording the resources used by a particular project.

The Nimrod-G architecture offers several benefits. In particular, it provides an economic model that can be applied to be metacomputing, and it allows interaction with multiple different system architectures, such as Globus, Legion, and Condor.

In the future Nimrod-G will be expanded to allow advance reservation of resources and use more advanced economic models such as demand-and-supply, auctions, and tenders/contract-net protocols [30].

6.6 Grid Application

At the beginning of Section 6.5.1 we divided Grid projects into three classes: community activities, toolkits (middleware), and applications. Here we focus on three applications representative for current Grid activities.

6.6.1 Astrophysics Simulation Collaboratory

The Astrophysics Simulation Collaboratory (ASC) was originally developed in support of numerical simulations in astrophysics, and has evolved into a general-purpose code for partial differential equations in three-dimensions [1, 31]. Perhaps the most computationally demanding application that has been attacked with

ASC is the numerical solution of Einsteins general relativistic wave equations, in the context, for example, of the study of neutron star mergers and black hole collisions. For this purpose, the ASC community maintains an ASC server and controls its access through login accounts on the server. Remote resources integrated into the ASC server are controlled by the administrative policies of the cite contributing the resources. In general, this means that a user must have an account on the machine on which the service is to be performed. The modular design of the framework and its exposure through a Web-based portal, permits a diverse group of researchers to develop add-on software modules that integrate additional physics or numerical solvers into the Cactus framework.

The astrophysics simulation collaboratory (ASC) pursues the following objectives [32]:

- Promote the creation of a community for sharing and developing simulation codes and scientific results;
- Enable transparent access to remote resources, including computers, data storage archives, information servers, and shared code repositories;
- Enhance domain-specific component and service development supporting problem-solving capabilities, such as the development of simulation codes for the astrophysical community or the development of advanced Grid services reusable by the community;
- Distribute and install programs onto remote resources while accessing code repositories, compilation, and deployment services;
- Enable collaboration during program execution to foster interaction during the development of parameters and the verification of the simulations;
- Enable shared control and steering of the simulations to support asynchronous collaborative techniques among collaboratory members;
- Provide access to domain-specific clients that, for example, enable access to multimedia streams and other data generated during the execution of the simulation.

To achieve these objectives, ASC uses a Grid portal based on JSP for thin-client access to Grid services. Specialized services support community code development through online code repositories. The Cactus computational toolkit is used for this work. The ASC is scheduled to be opened for users in the astrophysics community in September 2002.

6.6.2 Particle Physics Data Grid

The Particle Physics Data Grid (PPDG) [18] is a collaboratory project concerned with providing the next-generation infrastructure for current and future high-energy and nuclear physics experiments. One of the important requirements of PPDG is to deal with the enormous amount of data that is created during high energy physics experiment and must be analyzed by large groups of specialists. Data storage, replication, job scheduling, resource management, and security components supplied by the Globus, Condor, STACS, SRB, and EU Data Grid projects [12] all will be integrated for easy use by the physics collaborators. Development of PPDG is supported under the DOE SciDAC initiative (Particle Physics Data Grid Collaboratory Pilot) [18].

6.6.3 NEESgrid

The intention of NEESgrid is to build a national-scale distributed virtual laboratory for earthquake engineering. The initial goals of the project are to (1) extend the Globus Information Service to meet the specialized needs of the community and (2) develop a set of services called NEESpop, along with existing Grid services to be deployed to the NEESpop servers. Ultimately, the system will include a collaboration and visualization environment, specialized NEESpop servers to handle and manage the environment, and access to external system and storage provided by NCSA [68].

One of the objectives of NEESgrid is to enable observation and data access to experiments in real time. Both centralized and distributed data repositories will be created to share data between different locations on the Grid. These repositories will have data management software to assist in rapid and controlled publication of results A software library will be created to distribute simulation software to users. This will allow users with NEESgrid-enabled desktops to run remote simulations on the Grid [67].

NEESgrid will comprise a layered architecture, with each component being built on core Grid services that handle authentication, information, and resource management but are customized to fit the needs of earthquake engineering community.

The project will have a working prototype system by the fourth quarter of 2002. This system will be enhanced during the next years, with the goal to deliver a fully tested and operationa system 2004 to gather data during the next decade.

6.7 Portals

The term “portal” is not uniformly defined within the computer science community. Sometimes it represents integrated desktops, electronic market places, or information hubs [49, 73, 50]. We use the term portal here in the more general sense of a community access point to information and services (see Figure 6.9).

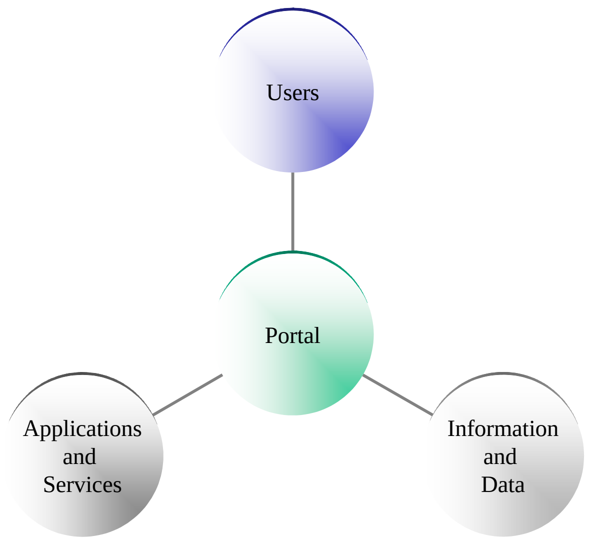

Figure 6.9: Portals provide an entry point that helps to integrate information and data, application and services.

Definition: Portal

A community service with a single point of entry to an integrated system providing access to information, data, applications, and services.

In general a portal is most useful when designed for a particular community in mind. Today, most common *Web portals* build on the current generation of Web based commodity technologies, based on the HTTP protocol for accessing the information through a browser.

Definition: Web portal

A portal providing users ubiquitous access, with the help of web-based commodity technologies, to information, data, applications, and services. The current generation of Web portals is accessed through HTTP and Web browsers.

A *Grid portal* is a specialized portal useful for users of production Grids. A Grid portal provides information about the status of the Grid resources and services. Commonly this information includes the status of batch queuing systems, load, and network performance between the resources. Furthermore, the Grid portal may provide a targeted access point to useful high-end services, such as the generation of a compute- and data-intensive parameter study for climate change. Grid portals provide communities another advantage: they hide much of the complex logic to drive Grid-related services with simple interaction through the portal interface. Furthermore, they reduce the effort needed to deploy software for accessing resources on production Grids.

Definition: Grid portal

A specialized portal providing an. entry point to the Grid to access applications, services, information, and data available within a Grid.

In contrast to Web portals, Grid portals may not be restricted to simple browser technologies but may use specialized plug-ins or executables to handle the data visualization requirements of, for example, macromolecular displays or three-dimensional high-resolution weather data displays.

A Grid portal may deal with different user communities, such as developers, application scientists, administrators, and users. In each case, the portal must support a personal view that remembers the preferred interaction with the portal at time of entry. To meet the needs of this diverse community, sophisticated Grid portals (currently under development) are providing commodity collaborative tools such as newsreaders, e-mail, chat, and video conferencing, event scheduling. Additionally, some Grid portal developers are exploiting commodity technologies such as JavaBeans and JSP, which are already popular in Web portal environments. In the following subsections we highlight several examples of well-known Grid portals and the toolkits being used to create these portals.

6.7.1 HotPage

HotPage [17] is a portal that provides a collective view of a distributed set of high-performance computing resources. The portal enables researchers easily to find information about each of the resources in the computational Grid. This information (which is stored in HTML) includes technical documentation, operational status, load and current usage, and queued jobs. Additionally, HotPage enables users to access and manipulate files and data and to submit, monitor, and delete jobs. Grid access is through the Globus Toolkit [22] or via the Network Weather Service [51]. The HotPage backend is accessed through Perl CGI scripts that create the pages requested. HotPage has been installed on a variety of production Grids, such as NPACI [11] and NASA IPG [15].

6.7.2 Webflow and Gateway

Webflow and its successor Gateway [35] are two influential projects in designing portals for Grids. They offer a programming paradigm implemented over a virtual Web-accessible Grid. An application is designed by a computational graph that is visually edited by the end-user, using Java applets. Nodes of the graph are reusable modules that written by the developers. Module users need not, however, be concerned with issues such as allocating and running the modules on various machines, creating connections among the modules, sending and receiving data across these connections, or running several modules concurrently on one machine. The Gateway system takes care of these management issues and coordinates the execution.

The Gateway system is based on a modern three-tier architecture. Tier 1 is the high-level front-end enabling visual programming, steering, run-time data analysis and visualization, and collaboration; this front-end is based on Web technologies and object-oriented commodity standards. Tier 2 is formed by distributed object-based, scalable, and reusable Web servers and object brokers and builds the middleware. Tier 3 comprises the back-end services such as execution services and data movement services.

6.7.3 XCAT

The XCAT Project [59] from Indiana University provides and implementation of the Common Component Architecture (CCA) [3] to assist in the assembly of applications using Grid resources. The CCA specification describes the construction of portable software components that can be re-used in any CCA compliant runtime frameworks. These frameworks are tuned for a variety of application environments and in some cases are designed for applications that run on massively parallel computers. Here components may be parallel objects (multiple component instances operating in synchrony and communicating with each other with MPI) or they may be highly multi-threaded and run on large shared memory, multiprocessor servers. In other cases, the frameworks are designed to enable the construction of applications from components that are distributed over a Grid. XCAT allows Grid application programmers to script complex distributed computations and package these applications with simple interfaces for others to use. Each user obtains a personal notebook for controlling the applications; the notebook is used as elementary abstraction to package applications and data scripts and parameters as part of a an Web page. The portal server has an integrated event service allowing application and Grid resource information to publish events through the Network Weather Service [51] and Autopilot [2]. XCAT has been tested on distributed simulation of chemical processes in semiconductor manufacturing and collaboratory support for X-ray crystallography. XCAT is based on Globus and uses the Java CoG Kit [53, 80] for its core security and remote task creation, and RMI over XSOAP [59] as a communication protocol.

6.7.4 UNICORE

UNICORE (UNiform Interface to COmputing REsources) [23] provides a vertical integration environment for Grids including access to resources through a portal. It is designed to assist in the workflow management of tasks to be scheduled on resources part of supercomputing centers. A UNICORE workflow comprises hierarchical assemblies of interdependent tasks, with dependencies that are mapped to actions such as execution, compilation, linking, and scripting according to resource requirements on target machines on the Grid. Besides strong authentication, UNICORE assists in compiling and running applications and in transferring input and output data. One of the main components of UNICORE is the preparation and modification of structured jobs through a graphical user interface that supports workflows. It allows the submission, monitoring, and control of the execution as part of a client that gets installed on the user's machine. Originally, UNICORE supported Web browser plug-ins, but it is now distributed as a standalone application. UNICORE is being used as the Grid infrastructure for a research project known as UNICORE Plus. This project is enhancing the original UNICORE software with new functionality to handle system administration and management, modeling of dynamic and extensible resources, creation of application-specific client and server extensions, improved data and file management functions, and runtime control of complex job chains. Metacomputing support (e.g., reservation, co-scheduling, application-level communication, and performance analysis) is also under consideration. Development to utilize Globus enabled resources within UNICORE is under development [5].

6.7.5 JiPANG

JiPANG (Jini-based Portal Augmenting Grids) [76] is both a portal system and a toolkit, providing a uniform interface layer for accessing a variety of Grid systems. JiPANG is built on top of the Jini distributed object technology. It functions as a higher-level management services to resources being managed by individual Grid systems such as Ninf [65], NetSolve [37], and the Globus Toolkit [22] via the Java CoG Kit [80]. A Java API provides the user with a uniform interface to the Grid. A specialized JiPANG browser allows the interactive access to Grid resources and services.

6.7.6 PUNCH

PUNCH (Purdue University Network-Computing Hubs) is a distributed network computer that allows users to access text and graphical applications remotely via a Web browser. PUNCH provides the ability to define several community portals, each of which serves a specific set of users [27]. When users visit a community portal, they are presented with a menu of applications that they can execute. These applications range from CPU simulators to drawing programs to complex commercial Electronic Design Automation (EDA) and mathematical analysis packages. For text-based tools, an HTML interface is provided that forwards all commands on to the actual application. This enables a quick integration of command line based applications into PUNCH. For more complex graphical applications, systems such as VNC are used to transmit the display back to the remote users [54]. Such a method has also been used by other Grid portal activities including the Access Grid (see Section 6.7.7).

At the base of PUNCH is PVFS, the PUNCH Virtual File System. By using a series of proxies over standard NFS protocols, PUNCH is able to allow near-native NFS performance over disparate networks. Also, the PVFS removes the need for individual user accounts. Instead, all files are owned by a system account with the PUNCH user of the file being identified by its position in the file system tree. This abstraction is taken further to the level of user maintenance. In a traditional distributed system, user account information would need to be propagated to all systems on the network. PUNCH solves this by maintaining a pool of UIDs on each server that are dynamically assigned to users when they begin execution of processes on a server. An accounting facility keeps track of the UIDs in use and automatically reclaims UIDs at the end of the user's session.

Based on these features, PUNCH allows different institutions to share computational resources and applications. Sharing is possible even across different administrative domains based on a limited-trust relationship that can be established between the domains. This feature allows users at multiple universities to have access to the same computer systems with small risks of exploitations [45].

6.7.7 Access Grid

The Access Grid (AG) Project develops a package of Grid Software and maintains a production Grid of resources that can be used to support human interaction. The goal of the Access Grid is to support large-scale distributed meetings, collaborative work sessions, seminars, lectures, tutorials and training. It provides the ability to include multimedia display, presentation and interaction environments, and interfaces to Grid middleware and visualization environments. This focus on group communication is in contrast to desktop based tools which focus much more in individual communication.

The environment is intended to foster both formal and informal group interactions. Large-format displays integrated with intelligent or active physical meeting rooms (also called nodes) are a central feature of Access Grid nodes. Such a physical meeting room contains the high-end audio and visual technology needed to provide a high-quality compelling user experience. There are a number of Access Grid nodes deployed world-wide that are frequently used to conduct meetings, site visits, training sessions and educational events [29].

6.7.8 Commercial Grid Activities

Many of the early Grid projects that started as research efforts are now also marketed commercially. Legion, for example, is currently marketed through Avaki (which was co-founded by the developers of Legion). Several companies have decided to include the Globus Toolkit in their Grid marketing strategies that are based on extensions or support models. Nevertheless, the Globus Toolkit will continue to be a free open-source toolkit.

Efforts such as IBM's commitment to the Web services framework, Microsoft's .Net, [7], and Sun's Web services [75] and JXTA framework [58] will be major drivers for the next generation of Grid software. The development of an Open

Grid Service Architecture together with companies such as IBM promises to integrate business and research models and processes in order to leverage from each other's technologies. Much additional work is needed to extend this early work.

6.8 Future and Conclusion

In this chapter, we have identified a vision that motivates the creation of Grids and Grid-enabled systems. We have also examined a variety of projects that address some – but not all – of the issues that must be resolved before the Grid is truly universal. In addition to the development of middleware, interfaces are needed that can be used by the application scientists to access Grids. Commodity Grid toolkits enabling access to Grid functionality on an API level such as Fortran, Java, and Python are important. Portals must also be developed to hide the complex infrastructure of Grids and allow scientists to use this infrastructure in the daily scientific exploration. The tools and technologies discussed in this chapter are but the first step in the creation of a global computing Grid.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Science Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38. DARPA, DOE, and NSF support Globus Toolkit research and development. We thank Ian Foster, Geoffrey C. Fox, Dennis Gannon, Xian-He Sun, and the members of the Computingportals working group, formerly known as Datorr and is now active as part of the Grid Computing Environment working group of the GGF for the valuable discussions leading up to this work. The Globus Toolkit and Globus Project are trademarks held by the University of Chicago.

Bibliography

[1] ASC Portal Home Page. http://www.ascportal.org.

[2] Autopilot. http://www-pablo.cs.uiuc.edu/Project/Autopilot/AutopilotOverview.htm.

[3] Common Component Architecture Forum. http://www.cca-forum.org/.

[4] Folding@home. http://folding.stanford.edu/.

[5] Grid Interoperability Project. http://www.grid-interoperability.org/.

[6] HPSS. http://www.sdsc.edu/hpss/hpss1.html.

[7] Microsoft .NET. http://www.microsoft.com/net/. part of Microsoft website.

[8] Unitree. http://www.unitree.com/.

[9] National Center for Supercomputing Applications, 1986. http://www.ncsa.uiuc.edu/.

[10] Alliance on Track to Enhance Services, 2000. http://archive.ncsa.uiuc.edu/datalink/0005/VMR.intro.html.

[11] National Partnership for Advanced Computational Infrastructure, 2000. http://www.npaci.edu/.

[12] The DataGrid Project, 2000. http://www.eu-datagrid.org/.

[13] ApGrid: Partnership for Grid Computing in the Asia Pacific Region. http://www.apgrid.org/.

[14] EUROGRID: Application Testbed for European Grid Computing, 2001. http://www.eurogrid.org/.

[15] Information Power Grid Engeneering and Research Site, 2001. http://www.ipg.nasa.gov/.

[16] Netsolve Web Page, 2001. http://www.cs.utk.edu/netsolve.

[17] NPACI HotPage, 2001. https://hotpage.npaci.edu/.

[18] Particle Physics Data Grid. http://www.ppdg.net/.

[19] Scientific Discovery through Advanced Computing (SciDAC), 2001. http://www.sc.doe.gov/ascr/mics/scidac/.

[20] Storage Resource Broker (SRB), 2001. http://www.npaci.edu/DICE/SRB/.

[21] TerraGrid, 2001. http://www.teragrid.org/.

[22] The Globus project WWW page, 2001. http://www.globus.org/.

[23] UNICORE. http://www.unicore.de/.

[24] Condor Home Page. http://www.cs.wisc.edu/condor/, Feb 2002.

[25] Global Grid Forum Web Page, 2002. http://www.gridforum.org.

[26] NetSolve Home Page. http://icl.cs.utk.edu/netsolve/, February 2002.

[27] PUNCH Home Page. http://punch.ecn.purdue.edu/, March 2002.

[28] SETI@Home Home Page. http://setiathome.ssl.berkeley.edu/, February 2002.

[29] The Access Grid Web Page. http://www-fp.mcs.anl.gov/fl/accessgrid/, 2002.

[30] ABRAMSON, D., BUYYA, R., AND GIDDY, J. A Computational Economy for Grid Computing and its Implmentation in the Nimrod-G Resource Broker. *Future Generation Computer Systems ???*, ??? (??? 2002), ??? not yet published as of this writing.

[31] ALLEN, G., BENGER, W., GOODALE, T., HEGE, H., LANFERMANN, G., MASSO, J., MERZKY, A., RADKE, T., SEIDEL, E., AND SHALF, J. Solving Einstein's Equations on Supercomputers. *IEEE Computer* (1999), 52–59. http://www.cactuscode.org.

[32] ALLEN, G., BENGER, W., GOODALE, T., HEGE, H.-C., LANFERMANN, G., MERZKY, A., RADKE, T., SEIDEL, E., AND SHALF, J. The Cactus Code: A Problem Solving Environment for the Grid. In *High-Performance Distributed Computing, 2000. Proceedings. The Ninth International Symposium on* (???, August 2000), pp. 253 –260.

[33] ARNOLD, D., AGRAWAL, S., BLACKFORD, S., DONGARRA, J., MILLER, M., SAGI, K., SHI, Z., AND VADHIYAR, S. Users' Guide to NetSolve V1.4. Computer Science Dept. Technical Report cs-01-467, University of Tennessee, Knoxville, TN, July 2001.

[34] BELL, G., AND GRAY, J. What's next in high-performance computing. *Communications of the ACM 45*, 2 (February 2002), 91–95.

[35] BHATIA, D., BURZEVSKI, V., CAMUSEVA, M., FOX, G. C., FURMANSKI, W., AND PREMCHANDRAN, G. WebFlow - a visual programming paradigm for Web/Java based coarse grain distributed computing. *Concurrency: Practice and Experience 9*, 6 (1997), 555–577.

[36] CASANOVA, H., AND DONGARRA, J. NetSolve: A Network Server for Solving Computational Science Problems. *The International Journal of Supercomputer Applications and High Performance Computing 11*, 3 (October 1997), 212–223. older versions are available.

[37] CASANOVA, H., AND DONGARRA, J. NetSolve: A Network Server for Solving Computational Science Problems. *International Journal of Supercomputer Applications and High Performance Computing 11*, 3 (1997), 212–223.

[38] CHRISTENSEN, E., CURBERA, F., MEREDITH, G., AND WEERAWARANA, S. Web Services Description Language (WSDL) 1.1, 15 March 2001. http://www.w3.org/TR/wsdl.

[39] CZAJKOWSKI, K., FITZGERALD, S., FOSTER, I., AND KESSELMAN., C. Grid Information Services for Distributed Resource Sharing. In *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing* (San Francisco, California, August 2001), IEEE Press, pp. 181–184.

[40] Doe Science Grid. http://www.doesciencegrid.org/.

[41] EHNEBUSKE, D., BOX, D., KAKIVAYA, G., LAYMAN, A., FRYSTYK, H., MENDELSOHN, N. N., THATTE, S., AND WINER, D. Simple Object Access Protocol (SOAP) 1.1, 2000. http://www.w3.org/TR/SOAP.

[42] EPEMA, D. H. J., LIVNY, M., VAN DANTZIG, R., EVERS, X., AND PRUYNE, J. A worldwide flock of Condors: load sharing among workstation clusters. Tech. Rep. DUT-TWI-95-130, Delft University of Technology, Delft, The Netherlands, 1995.

[43] EVERS, X., DE JONGH, J. F. C. M., BOONTJE, R., EPEMA, D. H. J., AND VAN DANTZIG, R. Condor flocking: load sharing between pools of workstations. Tech. Rep. DUT-TWI-93-104, Delft University of Technology, Delft, The Netherlands, 1993.

[44] FIELDS, S. Hunting for Wasted Computing Power: New Software for Computing Networks Puts Idle PC's to Work. University of Wisconsin Research Sampler, 1993.

[45] FIGUEIREDO, R. J., KAPADIA, N. H., AND FORTES, J. A. The PUNCH virtual file system: seamless access to decentralized storage services in a computational grid. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing* (San Francisco, CA, August 2001), IEEE, IEEE Press.

[46] FOSTER, I. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications 15*, 3 (August 2001), 200–222. a brief introduciton to the grid.

[47] FOSTER, I. The Grid: A New Infrastructure for 21st Century Science. *Physics Today 55*, 22 (2002), 42. http://www.aip.org/pt/vol-55/iss-2/p42.html.

[48] FOSTER, I., KESSELMAN, C., NICK, J., AND TUECKE, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers/ogsa.pdf, February 2002.

[49] FOX, G. C. Portals for Web Based Education and Computational Science, 2000.

[50] FOX, G. C., AND FURMANSKI, W. High Performance Commodity Computing. In *The Grid: Bluepirnt for a new computing infrastructure*, I. Foster and C. Kesselman, Eds. Morgam Kaufman, 1999.

[51] GAIDIOZ, B., WOLSKI, R., AND TOURANCHEAU, B. Synchronizing Network Probes to avoid Measurement Intrusiveness with the Network Weather Service. In *Proceedings of 9th IEEE High-performance Distributed Computing Conference* (August 2000), pp. 147–154. http://www.cs.ucsb.edu/ rich/publications/.

[52] GANNON, D., CHIU, K., GOVINDARAJU, M., AND SLOMINSKI, A. An Analysis of The Open Grid Services Architecture. http://www.extreme.indiana.edu/ gannon/OGSAanalysis3.pdf, March 2002.

[53] GETOV, V., VON LASZEWSKI, G., PHILIPPSEN, M., AND FOSTER, I. Multi-Paradigm Communications in Java for Grid Computing. *Communications of ACM 44*, 10 (2001), 119–125. http://www.globus.org/cog/documentataion/papers/.

[54] KAPADIA, N. H., FIGUEIREDO, R. J., AND FORTES, J. A. PUNCH: Web Portal for Running Tools. *IEEE Micro 20*, 3 (May-June 2000), 38–47.

[55] KARONIS, N. MPICH-G2 Web Page, 2001. http://www.hpclab.niu.edu/mpi/.

[56] KORPELA, E., WERTHIMER, D., ANDERSON, D., COBB, J., , AND LEBOFSKY, M. SETIat-home: Massively Distributed Computing for SETI , January/February 2001.

[57] KORPELA, E., WERTHIMER, D., ANDERSON, D., COBB, J., AND LEBOISKY, M. SETI@home-massively distributed computing for SETI. *Computing in Science & Engineering 3*, 1 (January–February 2001), 78–83.

[58] KRISHNAN, N. The Jxta solution to P2P .

[59] KRISHNAN, S., BRAMLEY, R., GANNON, D., GOVINDARAJU, M., INDURKAR, R., SLOMINSKI, A., TEMKO, B., ALKIRE, R., DREWS, T., WEBB, E., AND ALAMEDA, J. The XCAT Science Portal. In *Proceedings of SC2001* (November 10-16 2001). http://www.sc2001.org/papers/pap.pap287.pdf.

[60] LICKLIDER, J., AND TAYLOR, R. W. The Computer as a Communication Device, 1968. http://memex.org/licklider.pdf.

[61] LYSTER, P. M., BERGMAN, L., LI, P., STANFILL, D., CRIPPEN, B., BLOM, R., PARDO, C., AND OKAYA, D. Casa Gigibit Supercomputing Network: CALCRUST Three-Dimensional Real-Time Multi-Dataset Rendering. In *Presented at Supercomputing '92* (Minneapolis, MN, 17-20 November 1992).

[62] MOLNAR, D. The SETIat-home Problem. *ACM*, 1 (jan 2001).

[63] MOORE, G. E. Cramming More Components Onto Integrated Circuits. *Electronics 38*, 8 (April 19 1965), 114–117.

[64] NAKADA, H., SATO, M., AND SEKIGUCHI, S. Design and Implementations of Ninf: towards a Global Computing Infrastructure. *Future Generation Computing Systems 15*, 5–6 (1999), 649–658.

[65] NAKADA, H., SATO, M., AND SEKIGUCHI, S. Design and Implementations of Ninf: towards a Global Computing Infrastructure. *Future Generation Computing Systems 15*, 5-6 (1999), 649–658.

[66] The Seven Layer of the OSI Model. http://www.iso.org, http://www.webopedia.com/quick_ref/OSI_Layers.html.

[67] PRUDHOMME, T., KESSELMAN, C., FINHOLT, T., FOSTER, I., PARSONS, D., ABRAMS, D., BARDET, J.-P., PENNINGTON, R., TOWNS, J., BUTLER, R., FUTRELLE, J., ZALUZEC, N., AND HARDIN, J. NEESgrid: A Distributed Virtual Laboratory for Advanced Earthquake Experimentation and Simulation: Scoping Study. Tech. Rep. 2001-02, NEES, February 2001.

[68] PRUDHOMME, T., AND MISH, K. D. NEESgrid: A Distributed Virtual Laboratory for Advanced Earthquake Experimentation and Simulation: Project Execution Plan. Tech. Rep. 2001-02, NEES, June 2001.

[69] RESOLUTION, F. N. C. Definition of ”Internet”. http://www.itrd.gov/fnc/Internet_res.html, 24 October 1995.

[70] RICHARDSON, L. F. *The Collected Papers of Lewis Fry Richardson*. Cambridge University Press, Cambridge, 1993. 2 volumes.

[71] SEKIGUCHI, S. Ninf Project Home Page. http://ninf.apgrid.org/, February 2002.

[72] SHUMAN, F. G. History of Numerical Weather Prediction at the NMC. *Weather and Forecasting, 4* (1989).

[73] SMARR, L. Infrastructures for Science Portals, 2001. http://www.computer.org/internet/v4n1/smarr.htm.

[74] SOMERVILLE, R. C. *The Forgiving Air: Understanding Environmental Change*. University of California Press, Berkeley, 1996.

[75] Web services made easier. http://java.sun.com/xml/webservices.pdf. Article Published by Sun Microsystem.

[76] SUZUMURA, T., MATSUOKA, S., AND H.NAKADA. A Jini-based Computing Portal System. http://matsu-www.is.titech.ac.jp/ suzumura/jipang/.

[77] TEAM, C. *Condor Version 6.2.2 Manual*, 6.2.2 ed. University of Wisconsin-Madison, Professor Miron Livny, 7367 Computer Science, 1210 West Dayton St, Madison, WI 53706-1685, 2001.

[78] TOWNS, J. The Alliance Virtual Machine Room, 2001. http://archive.ncsa.uiuc.edu/SCD/Alliance/VMR/.

[79] VERMA, S., GAWOR, J., VON LASZEWSKI, G., AND PARASHAR, M. A CORBA Commodity Grid Kit. In *2nd International Workshop on Grid Computing in conjunction with Supercomputing 2001 (SC2001)* (Denver, Colorado, November 12 2001). http://www.globus.org/cog.

[80] VON LASZEWSKI, G., FOSTER, I., GAWOR, J., AND LANE, P. A Java Commodity Grid Kit. *Concurrency and Computation: Practice and Experience 13*, 8-9 (2001), 643–662. http://www.globus.org/cog/documentation/papers/cog-cpe-final.pdf.

[81] VON LASZEWSKI, G., FOSTER, I., GAWOR, J., LANE, P., REHN, N., AND RUSSELL, M. Designing Grid-based Problem Solving Environments and Portals. In *34th Hawaiian International Conference on System Science*. Maui, Hawaii, 2001. http://www.mcs.anl.gov/ laszewsk/papers/cog-pse-final.pdf.

[82] VYSSOTSKY, V. A., CORBAT, F. J., AND GRAHAM, R. M. Structure of the Multics Supervisor. In *Joint Computer Conference, AFIPS Conf. Proc 27* (1965), p. 203. http://www.multicians.org/fjcc3.html.