Grid Middleware

Gregor von Laszewski[1,2] and Kaizar Amin[1,3]

[1]Mathematics and Computer Science Division,
Argonne National Laboratory
[2]Computation Institute, University of Chicago
[3]Department of Computer Science, University of North Texas

in Middleware for Commnications edited by Qusay H. Mahmoud

Corresponding author:
Gregor von Laszewski
Argonne National Laboratory
9700 S. Cass Ave
Argonne, IL 60439
gregor@mcs.anl.gov
Tel: 630 252 0472
Fax: 630 252 1997

MIDDLEWARE FOR COMMUNICATIONS

Contributors

Gregor von Laszewski,
Argonne National Laboratory, Argonne, IL 60439, U.S.A.,
Computation Institute, University of Chicago, Chicago IL 60637, U.S.A.
Kaizar Amin,
Argonne National Laboratory, Argonne, IL 60439, U.S.A.,
Department of Computer Science, University of North Texas, Denton, TX 76203, U.S.A.

Contents

Part I

Grid Middleware

1 Grid Middleware

Gregor von Laszewski[1,2] and Kaizar Amin[1,3]

[1]Mathematics and Computer Science Division,
Argonne National Laboratory
[2]Computation Institute, University of Chicago
[3]Department of Computer Science, University of North Texas

> The reason why we are on a higher imaginative level is not because we have finer imagination, but because we have better instruments.
>
> —Alfred North Whitehead [1]

The Grid creation of middleware is an essential aspect of the software engineering effort to create an evolving layer of infrastructure residing between the network and applications [2]. In order to be effective, this Grid middleware manages security, access, and information exchange to

- develop collaborative approaches for a large number of users, including developers, users, and administrators, that can be extended to the larger set of Internet and network users;
- transparently share a wide variety of distributed resources, such as computers, data, networks, and instruments, among the scientific, academic, and business communities; and
- develop advanced tools to enable, manage, and expedite collaboration and communication.

Scenarios that influence the state-of-the art middleware development include challenging worldwide efforts to deliver scientific experiment support for thousands of physicists at hundreds of laboratories and universities [3]. Such an advanced environment is possible only if the middleware supports high-end and high-performance resource requirements to allow managing and analyzing petabytes of data while using end-to-end services under quality and security provisions.

In this chapter we present information about research activities to develop such computing environments that are supported through Grid middleware. We address the following questions:

- What does the term Grid mean?

- How does the Grid middleware help to develop a collaborative science and business environment?
- What trends exist in Grid middleware development?
- What applications exist that use Grid middleware?
- Where can we find more information about Grids?
- What will the future bring?

1.1 THE GRID

Because of the recent popularity of the term *Grid* we revisit its definition. Throughout this chapter we follow the definition as introduced in [2].

In general, we distinguish between (a) the *Grid approach*, or paradigm, that represents a general concept and idea to promote a vision for sophisticated international scientific and business-oriented collaborations and (b) the physical instantiation of a *production Grid* based on available resources and services to enable the vision for sophisticated international scientific and business-oriented collaborations. Both ideas are often referred to in the literature simply as *Grid*. In most cases the usage of "Grid" is apparent from its context.

The term "Grid" is chosen in analogy to the electric power grid that allows pervasive access to electric power. Similarly, computational Grids provide access to pervasive collections of compute-related resources and services. The concept of the Grid is not a new one; it was suggested in the mid-sixties to imagine a computer facility operating "like a power company or water company" [4, 5]. However, we stress that our current understanding of the Grid approach is far beyond simply sharing compute resources [6] in a distributed fashion; indeed, we are dealing with far more than the distribution of a single commodity to its customers. 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). We believe it is crucial to recognize that Grid resources also include actual scientists or business consultants that may share their knowledge with their colleagues [2]. Middleware must be developed to support the integration of the human in the loop to guide distributed collaborative problem solving processes. In order to enable the Grid approach, an infrastructure must be provided that allows for flexible, secure, coordinated resource sharing among dynamic collections of individuals, resources, and organizations.

When a *production Grid* is created in order to support a particular user community, we refer to it as *community production Grid*.

Although the resources within such a community may be controlled in different administrative domains, they can be accessed by geographically and organizantionaly dispersed community members. The management of a com-

munity production Grid is handled as part of a *virtual organization* [7, 8]. While considering different organizations, we have to make sure that the policies between the organizations are properly defined. Not only may organizations belong to different virtual organizations, but also part of an organization (organizational unit) may be part – but not all – of a virtual organization (Figure 1.1). This is just one aspect addressed within the creation of a Grid; we will address many more challenging issues throughout the chapter.

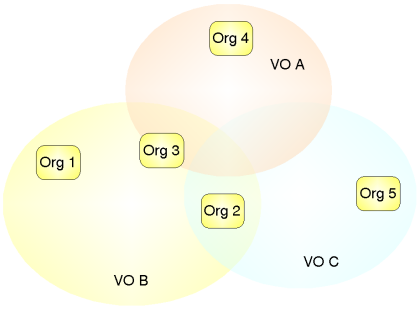

Fig. 1.1 Virtual organizations typically contain multiple organizations with complex policy rules. The Grid provides mechanisms to transparently include resources from a number of organizations into a virtual organization.

Although typical Grids contain high-performance resources such as supercomputers, we avoid the term highperformance and replace it in favor of high-end in order to stress the fact that performance is just one of many qualities required of a Grid. Fault tolerance and robustness are other qualities that are viewed as important.

In parallel to the electrical power Grid, we term producers and contributers of resources a *Grid plant*. Intermediaries that trade resource commodities are termed *Grid brokers*. Users are able to access these resources through *Grid appliances*, devices and tools that can be integrated into a Grid while providing the user with a service that enables access to Grid resources. Such appliances can be as simple as computers, handheld devices, or cell phones, but may be as complex as collaborative spaces enabling group-to-group communication [9] or sensor networks as part of sophisticated scientific infrastructure [10]. Many Grid appliances are accessed and controlled through sophisticated portals that enable easy access, utilization, and control of resources available through a Grid by the user.

One important concept that was originally not sufficiently addressed within the Grid community was the acknowledgment of *sporadic and ad hoc Grids* that promote the creation of time-limited services. This concept was first formulated as part of an initial Grid application to conduct structural biology and computed microtomography experiments at Argonne National Labora-

tory's Advanced Photon Source (APS) [11, 12, 13]. In these applications, it was not possible to install, on long term basis, Grid-related middleware on the resources, because of policy and security considerations. Hence, besides the provision for a pervasive infrastructure, we require Grid middleware to enable sporadic and ad hoc Grids that provide services with limited lifetime. Furthermore, the administrative overhead of installing such services must be small, to allow the installation and maintenance, to be conducted by the nonexpert with few system privileges. This requirement is common in many peer-to-peer networks [14] in which connections between peers are performed on an ad hoc or sporadic basis.

1.2 GRID ARCHITECTURE

In order to develop software supporting the Grid approach, it is important to design an architecture that corresponds with our vision.

A review of the literature [2] shows that a Grid architecture combines traditional architectural views. Such architectural views include layers, roles, and technologies. However, we believe it is important to recognize that the architecture of the Grid is multifaceted and an architectural abstraction should be chosen that best suites the area of the particular Grid research addressed by the architectural view.

The multifaceted aspects of the Grid motivate an architecture that is (a) *layered*, allowing us to bootstrap the infrastructure from a low to a high level, (b) *role-based*, allowing us to abstract the Grid functionality to interface with single resources to a collective multiple resources, (c) *service oriented* allowing a convenient abstraction model that others can easily deploy, create, and contribute to.

Let us consider the architectural diagram depicted in Figure 1.2. As mentioned previously, a virtual organization contains resources that are governed by policies and accessed by people that are part of the virtual organization. The *Grid fabric* 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 *application layer* comprises the users' applications that are used within a virtual organization. Additional layers within the *Grid middleware* are defined from top down as follows [2, 7]:

- The *connectivity layer* includes the necessary Grid-specific core communication and authentication support to perform secure network transactions with multiple resources within the Grid fabric. This includes protocols and services allowing secure message exchange, authentication, and authorization.

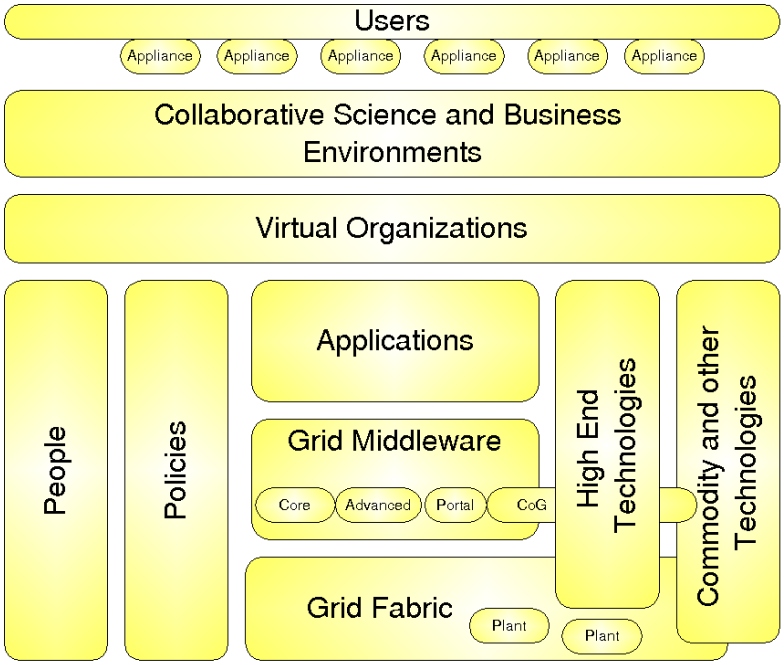

Fig. 1.2 Grid middleware is an evolving layer of software infrastructure residing between the Grid Fabric and applications.

- 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.

A benefit of this architecture is the ability to bootstrap a sophisticated Grid framework while successively improving it on various levels. Grid middleware is supported with an immensely rich set of application interfaces, protocols, toolkits, and services provided through commodity technologies and developments within high end computing. This interplay between different trends in computer science is an essential asset in our development for Grids, as pointed out in [15].

Recently it is become apparent that a future Grid architecture will be based on the Web services model, but will also need additional convenient frameworks of abstraction [16]. From the perspective of Grid computing, we define a service as a platform-independent software component that is self describing and published within a directory or registry by a service provider

(see Figure 1.3). A service requester can locate a set of services with a query to the registry, a process known as resource discovery. Binding allows a suitable service to be selected and invoked [17, 18, 19].

The usefulness of the service-based Grid architecture can be illustrated by scheduling a task in a virtual organization containing multiple supercomputers. First, we locate sets of compute resources that fulfill the requirements of our task under quality of service constraints. Next, we select one of these sets and establish a service binding to reserve the resources. Finally, we execute the task. Clearly, it is desirable to develop complex flows between services. Since this service-based model involves the use of asynchronous services, it will be important to deal appropriately with service guarantees in order to avoid deadlocks and other computational hazards.

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 is described in more detail in Section 1.7.7.

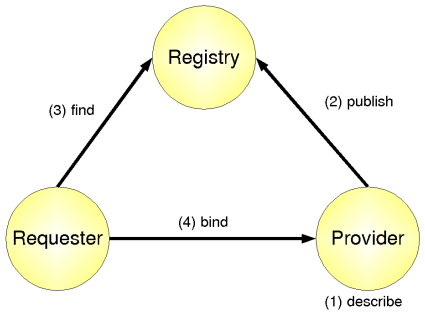

Fig. 1.3 The service model allows the description of the provider service by the provider that can be published in a registry and be found and bound by a requester.

1.3 GRID MIDDLEWARE SOFTWARE

Developing middleware is not an easy undertaking. It requires the development of software that can be shared, reused, and extended by others in a multitude of higher-level frameworks. Choosing the right paradigm for a software engineering framework in which the middleware is developed provides the means of its usefulness within the community. Hence, developing just an API as part of the middleware development is possible but does not provide the abstraction level needed in the complex Grid approach. Figure 1.4 depicts a useful subset of technologies that should be provided by state-of-the-art Grid

middleware. It includes protocols, data structures, and objects that can be accessed through convenient APIs and classes. Appropriate schemas need to be available to describe services and to deliver components for easy reuse in rapid prototyping of the next generation of Grid software. In many cases it is not sufficient to concentrate just on one of these aspects. It is important that any middleware that is developed strives to fulfill the needs of the user community reusing this middleware. In case of the Grid the user community is so diverse that it is essential to also address the diverse needs of the community while choosing the appropriate frameworks and other tools that must be part of the middleware layer [20].

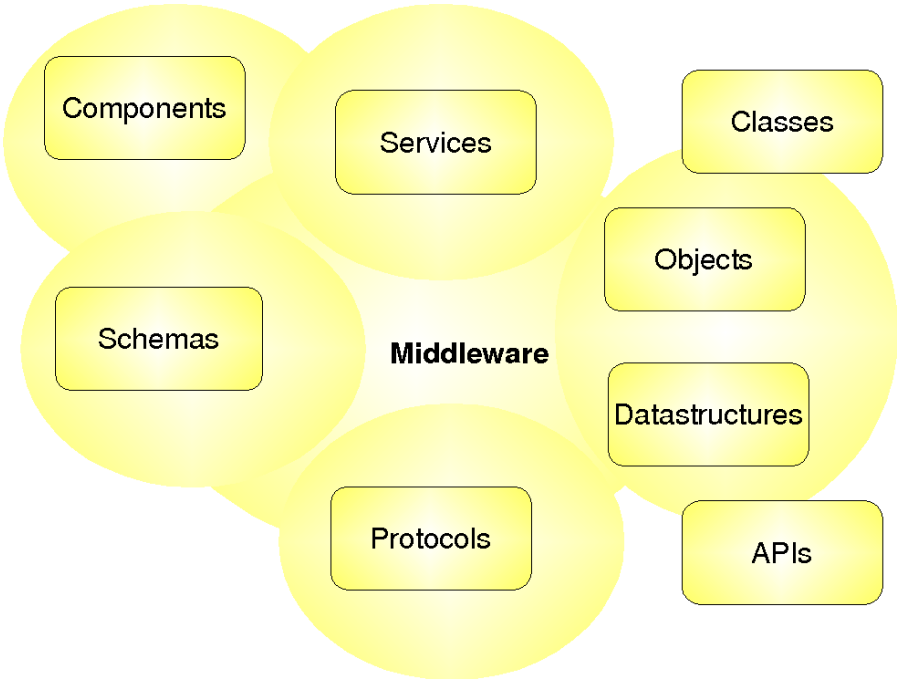

Fig. 1.4 Grid Middleware needs to support a variety of technologies in order to accommodate a wide variety of uses.

1.4 GRID MIDDLEWARE CHALLENGES

The current challenge in developing middleware software for the Grid is to address the complexity of developing and managing Grids. As we address a wide user community ranging from application developers to system administrators, and also a wide range of infrastructure as part of diversified virtual organizations, we must identify a path that leads to an integral software en-

vironment developed by the community. Besides emphasizing the interoperability issues, one needs to be concerned about the maintenance, management and policy issues that go beyond the actual delivery of a technical solution. In many cases it may be difficult to establish such management collaboration because of the complex and contradictory operating guidelines institutions may have. Hence, building a Grid has not only a technical challenge but also a social challenge.

1.5 GRID MIDDLEWARE STANDARDIZATION

In order for any middleware to achieve acceptance with a large community, it must be developed and maintained with the help of a community standards body. In the early days of the Grid development, there was no standard for Grid development; thus, several non-interoperable Grid frameworks resulted [7, 21, 22, 23]. The Global Grid Forum (GGF) [24] has presented itself as the much required body that coordinates the standardization of the Grid development.

The GGF is a community-initiated forum of more than five thousand individual researchers and practitioners working on distributed computing or Grid technologies. The primary objective of GGF is to promote and support the development, deployment, and implementation of Grid technologies and applications through the creation and documentation of "best practices" – technical specifications, user experiences, and implementation guidelines. Additionally, the efforts of the GGF are aimed at the development of a broadly based integrated Grid architecture that can serve to guide the research, development, and deployment activities of emerging Grid communities. Defining such an architecture will advance the Grid approach through the broad deployment and adoption of fundamental basic services and by sharing code among different applications with common requirements.

1.6 GRID MIDDLEWARE SERVICES

Now that we have an elementary understanding on what the Grid approach constitutes, we ask the question which elementary Grid services must we provide as part of Grid middleware and which services must be enhanced to make the middleware more useful.

1.6.1 Elementary Grid Middleware Services

The most elementary Grid services include *job execution services*, *security services*, *information services*, and *file transfer services*.

Job execution services provide mechanisms to execute jobs on a remote resource while appropriate security mechanisms are applied to allow only authenticated and authorized users to use the desired compute resources. Job execution services also allow users to interface with batch queuing systems [25]. Hence, a job execution service must be able to submit a job asynchronously and be able to check its status at a later time. As multiple remote resources may be part of the Grid, jobs must have a unique identifier so that they can be properly distinguished.

Security services that are part of the Grid infrastructure should provide the ability to perform authentication and authorization to protect resources and data on the Grid. Encryption and other elementary security mechanism are usually included in Grid security services through the use of commodity technologies such as PKI or Kerberos. Because of the large number of resources in the Grid, it is important to establish policies and technologies that allow the users to perform a *single sign-on* to access the Grid. The Globus Grid security infrastructure provides the framework for such a mechanism [26]. A user can sign on through a single interaction with the Grid. A security credential is created on his behalf that is used through a delegation process to sign onto other resources. Naturally, the user must be authenticated to use the resource. In general, all Grid services must integrate Grid security mechanism that can be established by reusing middleware protocols, APIs, and data structures so they can be reused to develop more sophisticated security services.

Information services provide the ability to query for information about resources and participating services on the Grid. Typically a virtual organization maintains a Grid Information Service relevant to its user community. Community production Grids may maintain different information services exposed through different protocols. This is in part motivated by the different scale and functionality of the diverse user communities as part of a virtual organization. An application user may be interested in knowing only on which computer he may be able to compile a sophisticated FORTRAN 90 application. Other users may want to know on where to execute this application under certain resource constraints. Hence an information services framework must be able to deal with the diverse nature of queries directed to it. Many different information services can be integrated and the capabilities be announced and made accessible to the users of the community Grid. Services such as querying the amount of free memory for a compute resource may pose immediately problems if the information service is not properly implemented. Assume a user queries the amount of memory every millisecond, but the memory change is updated only every second. Many resources are wasted for a useless query. Hence, as pointed out in [27], information services must provide protocols and mechanisms that can dynamically change their behavior toward unreasonable requests by the users. We expect that such smart Grid services will become increasingly available and developed by the large Grid community. Agglomeration and filter services as part of the information

service framework can be used to join or reduce the amount of information that is available to the user. The capabilities of such services must be self-described, not only through the availability of schemas and objects that can be retrieved, but also through the semantics attached with the information through metadata annotations.

File transfer services must be available to move files from one location to the other location. Such file transfer services must be sophisticated enough to allow access to high-performance and high-end mass storage systems. Often such systems are coupled with each other, and it should be possible to utilize their special capabilities in order to move files between each other under performance considerations. Striping and parallel streams are some obvious strategies to be included in file transfer services between file servers. Ideally the middleware for file transfers must be exposed to the user community as simply as possible through, for example, a command like "copy file A from server Y to file B on server Z as fast as possible." New protocols and services must be developed that support this notion. An API that provides this functionality is not sufficient.

1.6.2 Advanced Grid Management Services

With the help of the basic Grid middleware services, a variety of more advanced services can be developed. Such services include *file Management*, *task Management*, and *information management*. Such an advanced service may manage the agglomeration and filtering of information for a community. There exist many more examples of such advanced services. As pointed out in the preceding section, users wish to manage file transfers in an easy and transparent fashion through the invocation of an elementary set of protocols and APIs. These should hide the underlying complexity of the Grid as much as possible from the user but should provide enough flexibility that individual control about a file transfer can be issued. Users may want to use advanced file transfer services that dynamically adapt to changing network conditions and fault situations [28], without user intervention.

Task management of multiple tasks prepared by a user is a common problem. Envision a computational biologist conducting a comparative study in genome analysis. As part of the problem-solving process, multiple, similar experiments must be run. These could involve many thousands of subtasks. Organizing and scheduling these tasks should be performed by a sophisticated service that utilizes the capabilities of the Grid by reusing the current generation of Grid services.

Taking a closer look at our two examples, we observe that in Grid literature a distinction is made between file transfers, task execution, and information queries. However, we believe this distinction is useful only for developers of basic Grid middleware and not for the larger community developing even more advanced Grid services. We have demonstrated in [27] that an abstraction of

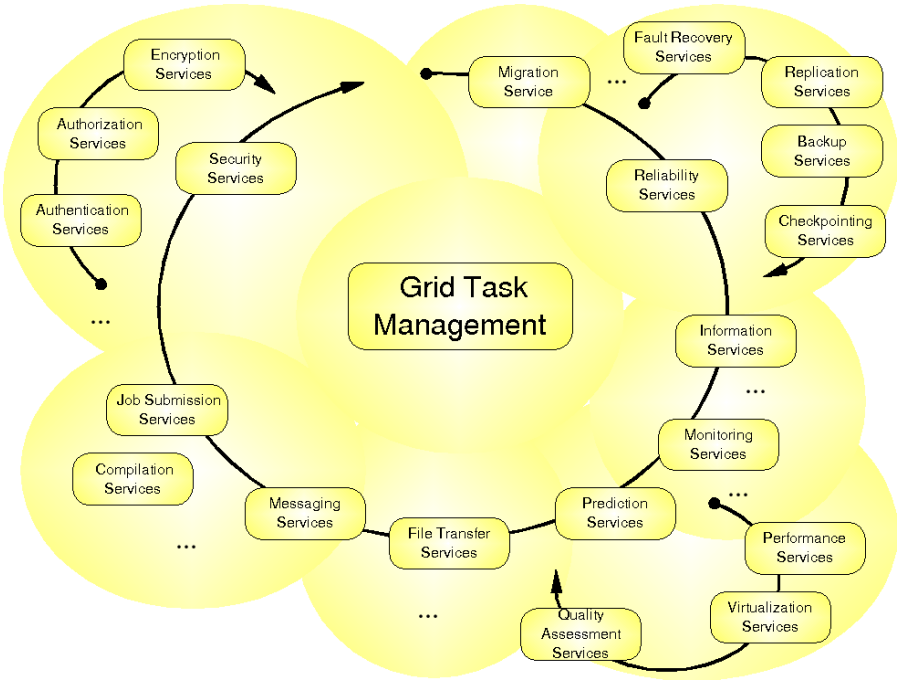

Fig. 1.5 Grid task management is a complex issue as part of any Grid middleware software.

all of these can be, and should be, performed. Using such an abstraction will allow us to introduce more easily uniform service specifications that integrate higher level functionality such as fault tolerance and self-configuration. Without this abstraction the development of future Grid middleware is rather limited and will not address the more complex management issues.

Although middleware typically does not contain user-level applications, one must recognize the fact that middleware must be provided to simplify the development of user applications that reuse Grid middleware [29, 30, 31]. As part of this effort tools are created within the NSF Middleware Initiative (NMI) to provide portal middleware for Grids [29]. Additional efforts are already under way to provide similar functionality as part of Java Beans by, for example, the Java CoG Kit.

As we also address not only the application developer or the Grid middleware developer, but also those maintaining and monitoring Grids a

Significant effort also is currently being spent in developing tools that make the task of the Grid system administrator easier. Administrative and monitoring services are under development facilitating the maintenance of accounts and the monitoring of the services running at remote locations.

1.7 GRID MIDDLEWARE TOOLKITS

Currently the community is building Grid middleware toolkits that address some of the complex issues related to the Grid approach. These efforts are still in their infancies. However, they have made great progress in the past years. In this section we briefly outline a subset of these developments in order to introduce several different toolkits.

1.7.1 Globus Toolkit

The Globus Project has contributed significantly to the Grid effort through (a) conducting research on Grid-related issues such as resource management, security, information services, data management, and application development environments; (b) developing the Globus Toolkit as part of an open-source development project; (c) assisting in the planning and building of large-scale testbeds, both for research and for production use by scientists and engineers; (d) contributing as part of collaborations in a large number of application-oriented efforts that develop large-scale Grid-enabled applications in collaboration with scientists and engineers; and (e) participating in community activities that include educational outreach and Grid standards as part of the Global Grid Forum.

The Globus Toolkit has evolved from a project developing APIs (Nexus and DUROC) to a project that also develops protocols and services (MDS, GRAM, and their uses through the Java CoG Kit). During the past few years the Globus Toolkit has evolved from an API-oriented solution to a more protocol and service-oriented architecture.

The Globus Toolkit includes the following features:

Security is an important aspect of the Globus Toolkit [2]. 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 (in the C-based Globus Toolkit) is handled through the GlobusIO API that 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.

Information about a Grid is handled in the Globus Toolkit version 2 through the Metacomputing Directory Service. The concept of a distributed directory service for the Grid was first defined in [32] and later refined in [33]. The Metacomputing Directory Service manages information about entities in a Grid in a distributed fashion. The implementation of MDS in GT2 is based on the Lightweight Directory Access Protocol (LDAP). In [27] we have shown

that the Globus architecture can be significantly simplified while combining the information services with a job submission service. We expect this concept to be introduced into future versions of the Globus Toolkit. In version 3 of the toolkit this protocol is being replaced by XML while providing similar functionalities as the original toolkit.

Resource management within the Globus Toolkit is handled through a layered system in which high-level global resource management services are built on top of local resource allocation services [2]. The current Globus Toolkit resource management system comprises three components: (1) an extensible resource specification language for exchanging information about resource requirements among the components in the Globus Toolkit resource management architecture; (2) a standardized interface to local resource management tools including, for example, PBS, LSF, and Condor; and (3) a resource coallocation API that may allow the construction of sophisticated co-allocation strategies that enable use of multiple resources concurrently (DUROC) [25, 34, 35].

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 software that can generate custom-designed installation distributions.

1.7.2 Legion

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

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.

1.7.3 Condor

Condor is a high-throughput system for utilizing idle compute cycles on workstations while distributing a number of queued jobs to them. While typical supercomputing focuses on floating-point operations per second, high-throughput systems focus on floating-point operations per month or year [23, 36, 23]. Condor maintains a pool of computers while using a centralized broker to distribute jobs based on load information or preference asserted with the jobs to be executed. 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 controls and publish them in a central information service. In order not to disable interactive use of a workstation, the machine must be made available quickly to the console users.

To address this issue, the program uses checkpoints and restarts on, another host machine. 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. Similar authorization frameworks have been successfully reused in other projects such as SETI@Home [37, 38, 39, 40].

Today, Condor also includes client side brokers that handle more complex tasks such as job ordering via acyclic graphs and time management features. Such features have been demonstrated earlier in Webflow and the Java CoG Kit. To prevent monopolizing the resources by a single large application, Condor uses a fair scheduling algorithm.

To integrate resources maintained as part of batch queues, Condor introduced a mechanism that provides the ability to temporarily integrate resources for a particular period of time into a Condor pool. This concept, also known as *glide-in*, is enabled through a Globus Toolkit resource management backend. Using this technique, a job submitted on a Condor pool may be executed elsewhere on another computing resource [36].

Much of Condor's capability 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 changes to the source code. 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 cannot be used [2].

Additional Grid capabilities have been provided, called Condor flocks, that represent pools in possibly different administrative domains. Policy agreements between these flocks enable the redistribution of jobs between these flocks [39, 40].

1.7.4 SETI@Home

SETI@Home, a middleware toolkit and application created by the Space Science Laboratory at the University of California-Berkeley, is one of the most successful coarse-grained distributed computing systems. Its goal is to integrate compute resources on the Web as part of a collective resource that can solve many independent calculations at the same time, similar in spirit to Condor. In contrast to Condor, however, it deals with a specialized application-oriented functionality. The system was designed to handle 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 [41, 42] that started to be widely used in 1999. SETI@Home is a client-server distributed network. When a client connects to the SETI@Home work unit servers, a packet of data recorded from the Arecibo telescope is downloaded to the requesting client. 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 the server and a new packet is downloaded while repeating the cycle.

Packets of information that have been shown to have valuable information are then analyzed again to ensure there was no client error in the reporting of the calculation performed on the data. Hence, the system shows resiliency toward modified clients while maintaining scientific integrity [41]. SETI@Home has accumulated more than 900,000 CPU-years of processing time from over 3.5 million volunteers around the world. The entire system today averages out to 45 Tflops, which makes it the world's most powerful computing system. However, it is suitable only for coarse-grained problems [43]. 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 numbers of participants by creating a sense of community and project the goals of the scientific project to large numbers of nonscientific users. Seti@Home has also been adapted to other scientific domains [37, 44]; hence, it has become not only an application but also a middleware toolkit.

1.7.5 Network Weather Service

Network Weather Service (NWS) [45] is a specialized and limited middleware toolkit and service that periodically records and forecasts the performance of network and computational resources. The service is based on a distributed set of computational performance sensors that gather performance related 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 choose the forecasting method (from those that are provided with NWS) that best fits the prediction. NWS can be extended by including new models.

NWS addresses just one aspect of the Grid approach; however, it can enhance the usefulness of, for example, Legion or the Globus Toolkit.

1.7.6 Commodity Grid Kits

The Globus Project provides an elementary set of Grid middleware. Unfortunately, these services may not be compatible with the commodity technologies used for application development by other Grid middleware developers.

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 [46], Perl, Web services, .NET, and JXTA.

Existing Python and Java CoG Kits [47, 48, 49] 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. 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 [49]. Both the Java and Python CoG Kits are popular with Grid programmers and have been used successfully in many community projects. The usefulness of the Java CoG Kit has been proven during the design and implementation of the Globus Toolkit version 3 (GT3). Today the Java CoG Kit has been made integral part of the GT3 release. Some features that are not included in the GT3 release but can be downloaded from the Java

CoG Kit Web pages [50] are the availability of prototype user interfaces that are designed as Java beans and can be easily integrated in interface definition environments for rapid prototyping. As these components work on GT2 and GT3. The CoG Kit provides a new level of abstraction for Grid middleware developers (see Figures 1.6 and 1.7). In contrast to the Globus Toolkit, the Java CoG Kit has extended the notion of middleware while not only providing middleware for elementary Grid services, but also middleware for Grid portal development. A variety of other tools are build on top of the Java CoG Kit [51, 52, 53].

1.7.7 Open Grid Services Architecture

.

The Open Grid Services Architecture (OGSA) [54] initiative of the GGF defines the artifacts for a standard service-oriented Grid framework based on the emerging W3C defined Web services technology [55]. A service-oriented Grid framework provides a loosely coupled technology- and platform-independent integration environment that allows different vendors to provide Grid-enables services in a variety of technologies, yet conforming to the GGF-defined OGSA standards.

Web services, as defined by W3C [56], comprise "a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols". In other words, a Web service is a network-enabled software component that can be described, published, discovered, and invoked.

Web services are described through the Web services description language (WSDL) [17], providing an XML format for defining Web services as set of endpoints operating on messages. Web service descriptions can be published and discovered on centrally available service registries using protocols such as universal description, discovery, and integration (UDDI) [57]. Alternatively, the service descriptions can be published locally within the service hosting environment and discovered by using distributed protocols such as the Web service inspection language (WSIL) [58]. Web services can be invoked by defined operations as a part of service descriptions by passing XML message inputs wrapped in standard Internet protocols such as the simple object access protocol (SOAP) [18].

OGSA has introduced the concept of a Grid service as a building block of the service-oriented framework. A Grid service is an enhanced Web service that extends the conventional Web service functionality into the Grid domain. A Grid service handles issues such as state management, global service naming, reference resolution, and Grid-aware security, which are the key requirements for a seamless, universal Grid architecture. To facilitate the de-

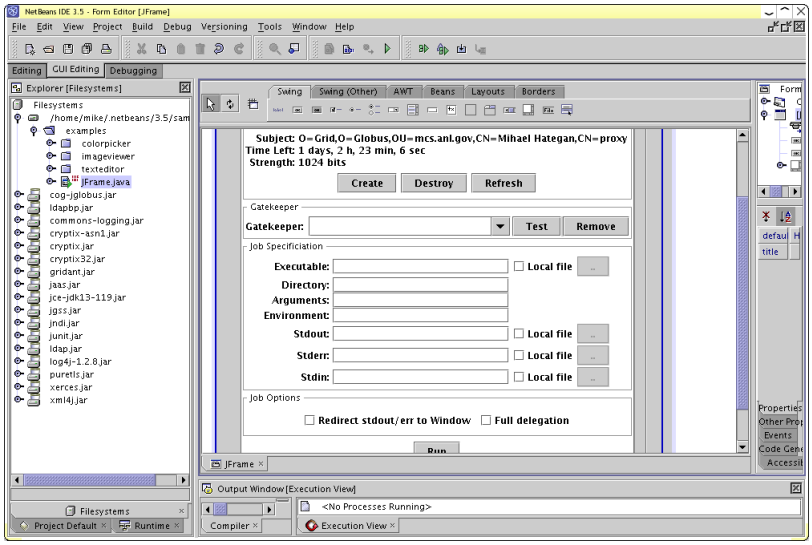

Fig. 1.6 The Java CoG Kit contains a number of JavaBeans that provides the next level of Grid middleware to develop simple, yet powerful components for Grid users.

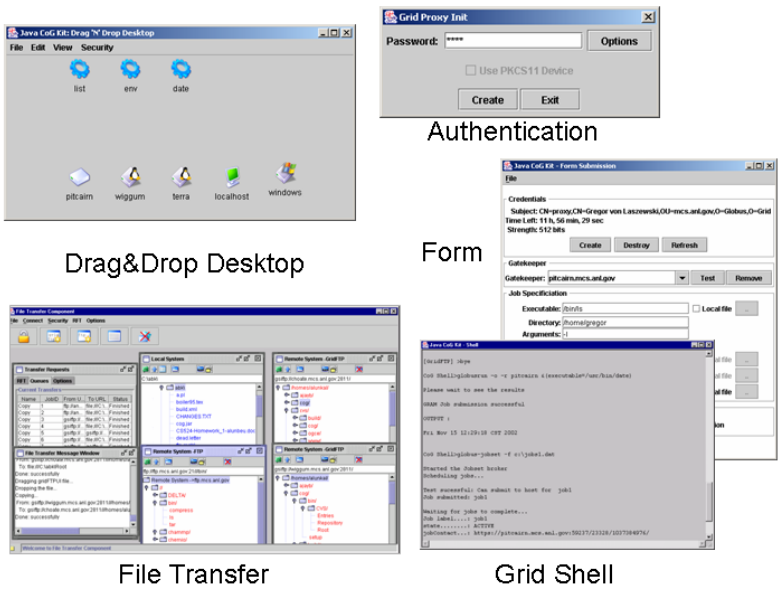

Fig. 1.7 Selected Java CoG Kit GUI components include convenient prototype interfaces to perform file transfer, job submissions, and service access while interfacing with Grid middleware.

scription of Grid services, GGF has extended the conventional WSDL schema to incorporate Grid-enabled attributes. The Grid-aware WSDL is called as *Grid Service Description Language* (GSDL). Hence, an OGSA-compliant Grid architecture will be composed of Grid services described using GSDL, published and discovered on specialized Grid registries, and invoked, by using an Internet protocol such as SOAP.

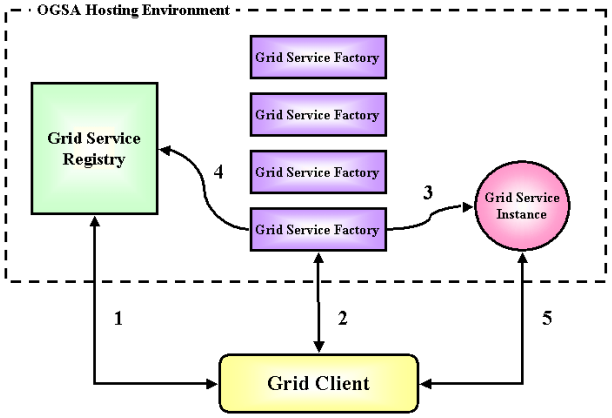

Fig. 1.8 Typical usage pattern for OGSA services. (1) The client accesses the central Grid registry and discovers the handle for a Grid service factory capable of providing the required functionality. (2) It then invokes the instance-creation operation of the Grid service factory. (3) The factory creates a Grid service instance and provides the client a unique network handle (GSH) for this instance. (4) The factory then registers the instance handle with the Grid registry so that other clients can discover it. This step is optional. (5) The client uses the GSH to communicate with the Grid service instance based on the operations promised in its description (GSDL).

Additional features provided by OGSA include the following:

Service Factories and Instances. The Web services paradigm addresses the discovery and invocation of persistent services. The Grid paradigm supports the creation, management, and termination of transient service instances handled at runtime. The OGSA framework defines entities and patterns between these entities to support of such dynamic management of transient instances. OGSA implements the factory pattern, whereby persistent Grid service factories enable the creation of transient Grid service instances. The factory and the service instance collectively provide constructs for lifetime negotiation and state management of the instance. Every service instance has a universally unique identity, also called the Grid service handle (GSH), that provides a network pointer to that service instance. As shown in Figure 1.8 a typical

usage pattern in OGSA is as follows. A user identifies a central Grid service registry (using some out-of-band methodology), selects a persistent Grid service factory appropriate for the functionality desired, negotiates the instance lifetime with the factory, and invokes the instance-creation operation on the factory. The factory creates a service instance and provides, the client with a unique GSH. From that point on the client uses the GSH to communicate with the corresponding service instance directly.

Service Data. To support discovery, introspection, and monitoring of Grid services, OGSA introduced the concept of service data. Every Grid service instance has a set of service data elements associated with it. These elements refer to metadata used internally by the OGSA framework, as well as the runtime data retrieved by clients and other external services. In other words, the Service Description Elements (SDEs) of a Grid service instance represent its internal and external state. In an object-oriented terminology the SDEs in a Grid service are equivalent to instance variables in an object.

1.7.7.1 Notification OGSA also provides the classic *publish/subscribe* style communication mechanism with Grid services. Hence, clients can subscribe to receive notification messages based for any activity on a predefined notification topic or change in the data value of SDEs. OGSA provides mechanisms for a *notification source* and *notification sink* to allow asynchronous, one-way delivery of messages from the source to the sink. Hence, OGSA shifts from a client-server Web service model to a peer-to-peer Grid service model.

1.8 PORTAL MIDDLEWARE FOR GRIDS

The term "portal" is not uniformly defined within the computer science community. Sometimes it is used interchangeably for integrated desktops, electronic market places, or information hubs [59, 60, 61]. We use the term portal in the more general sense of a community access point to information and services. A portal typically is most useful when designed for a particular community in mind.

Web portals build on the current generation of Web-based commodity technologies, based on the HTTP protocol for accessing the information through a browser.

A *Grid portal* is a specialized portal for users of production Grids or Grid-enabled applications. 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 commu-

nities 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.

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. These advanced portals may be the Web portals of tomorrow. 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 portal middleware must provide commodity collaborative tools such as news-readers, e-mail, chat, and video conferencing, and event scheduling.

It is important to recognize the fact that Grid middleware must also include software that allows the creation of portals for Grids. Such middleware has in the past been relying on the Java CoG Kit and can, through abstractions, continue to build on the convenient functions provided. Several efforts are using Jetspeed as part of this middleware to enable Grid portals [62, 63, 64].

1.8.1 Grid Middleware to Deploy Virtual Organization Portals

Compute center portals provide a collective view of and access to a distributed set of high-performance computing resources as part of a high performance computing center. Typical candidates are HotPage [65], and UNICORE [21].

HotPage 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. Hence it combines information provided by Grid middleware services and other informational databases maintained as part of the virtual organization. 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 [66] or the Network Weather Service [45]. 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 [67] and NASA IPG [31]. Future versions of Hotpage are expected to be implemented in Java by using the Java CoG Kit, OGSA, and Jetspeed.

UNICORE (UNiform Interface to COmputing REsources) provides a vertical integration environment for Grids, including access to resources through a Java Swing framework. It is designed to assist in the workflow management of tasks to be scheduled on resources part of supercomputing centers similar to [20]. 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 tar-

get 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. New functionality is developed 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. The ability to utilize Globus Toolkit enabled resources within UNICORE [63] while reusing the Java CoG Kit is under development. Future developments will also include GT3.

1.9 APPLICATIONS USING AND ENHANCING GRID MIDDLEWARE

In this section we focus on three applications representative of current Grid activities reusing and enhancing Grid middleware.

1.9.1 Astrophysics

The Astrophysics Simulation Collaboratory is an example of a Grid application requiring large numbers of tightly coupled resources that are used as part of large parameter studies. The Astrophysics Simulation Collaboratory (ASC) was originally developed in support of numerical simulations in astrophysics. It has let to the development of a general-purpose code for partial differential equations in three dimensions [68, 69, 70].

The astrophysics simulation collaboratory (ASC) pursues the following objectives [71]: (a) promote the creation of a community for sharing and developing simulation codes and scientific results; (b) enable transparent access to remote resources, including computers, data storage archives, information servers, and shared code repositories; (c) 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; (d) distribute and install programs onto remote resources while accessing code repositories, compilation, and deployment services; (e) enable collaboration during program execution to foster interaction during the development of parameters and the verification of the simulations; (f) enable shared control and steering of the simulations to support asynchronous collaborative techniques among collaboratory members; and (g) 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 communicate these objectives as part of a collaboratory, 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 [2].

1.9.2 Earthquake Engineering

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 application specific services, reusing existing Grid services. Ultimately, the system will include a collaboration and visualization environment, specialized servers to handle and manage the environment, and access to external system and storage provided by NCSA [72].

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 [73].

NEESgrid comprises 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.

1.9.3 High Energy Physics Grids

A number of large projects related to high energy research have recognized the potential and necessity of Grids is part of their sophisticated problem solving infrastructure. Such projects include Particle Physics Data Grid (PPDG) [74], international Virtual Data Grid Laboratory (iVDGL) [75] and the European DataGrid [3].

PPDG 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 Toolkit, Condor, STACS, SRB, and EU DataGrid projects 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) [30, 74].

The goal of iVDGL is to establish and utilize a virtual laboratory comprising heterogeneous computing and storage resources in the United States, Europe, and other regions linked by high-speed networks and operated as a single system for the purposes of interdisciplinary experimentation in Grid-enabled, data-intensive scientific computing. iVDGL is aiming at the physics community while providing future production services, which may also designed in the Grid Physics Network (GriPhyN) [76]. Integration of the two projects will provide a community production Grid.

In general, the communities addressed by such Grids require sharing and modifying large amounts of data.

Many of these projects reuse the concept of virtualization, which is well understood by the processor design community in addressing the design of memory hierarchies. In Grid projects, however, the virtualization goes beyond the concept of tertiary storage space and includes not only file systems but also the annotation of data with metadata. This metadata allows the scientists to formulate complex context sensitive queries returning the appropriate information. As result, some of these queries have been generated by state-of-the art algorithms that are too resource intensive to be run redundantly by the many users. Thus, it is beneficial to store the results in a cache-like system that can be queried rather than generated. On the other hand, some of the calculations performed on the data may actually be cheaper and less resource intense than storing the data and their intermediate results. In both cases it is important to annotate the data and the way it has or can be created with metadata. Efforts such as [77, 78, 79] are reporting on progress in this area. In many such efforts [80, 81] lessons learned from parallel computing while mapping an abstract direct acyclic graph specification onto a physical environment are used. Through late mapping that postpones the mapping in successive steps, the dynamical of the actual Grid resources are optimized. At present, however, these efforts do not address the more formal aspects such as deadlocks or resource over- and underprovision.

1.10 CONCLUDING REMARKS

In this chapter, we have concentrated on several aspects of middleware for Grids. We have seen that the Grid approach and the creation of Grid middleware are a natural continuations of established parallel and distributed computing research. We highlighted a number 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 application scientists to access Grids. These interfaces are provided by portal middleware toolkits that allow the easy integration of Grid middleware in ongoing application development efforts. Based on the efforts in the community we observe the trend that many advanced tools will be part

of the middleware of tomorrow. Hence the level of abstraction included in middleware to enable collaborative research is expected to rise significantly over the next years. Also critical are commodity Grid toolkits, enabling access to Grid functionality in the area of Web services, programming paradigms and frameworks, as well as the integration with other programming languages The tools and technologies discussed in this chapter are the first step in the creation of a global computing Grid.

Acknowledgments

This work was funded in part by the U.S. Department of Energy, Office of Advanced Scientific Computing, SciDAC program "CoG Kits: Enabling Middleware for Designing Science Applications, Web Portals, and Problem-Solving Environments," under Contract W-31-109-ENG-38. Globus Project is a trademark held by the University of Chicago. Globus Toolkit is a registered trademark held by the University of Chicago. We thank Gail Pieper and Jens Vöckler for comments on this chapter.

References

1. *Science and the Modern World.* New York: Free Press, 1967, page 58.

2. G. von Laszewski, G. Pieper, and P. Wagstrom, "Gestalt of the Grid," in *Performance Evaluation and Characterization of Parallel and Distributed Computing Tools*, ser. Series on Parallel and Distributed Computing. Wiley, 2003, (to be published). [Online]. Available: http://www.mcs.anl.gov/~laszewsk/bib/papers/vonLaszewski--gestalt.pdf

3. "The DataGrid Project," 2000. [Online]. Available: http://www.eu-datagrid.org/

4. V. A. Vyssotsky, F. J. Corbat, and R. M. Graham, "Structure of the Multics Supervisor," in *Joint Computer Conference, AFIPS Conf. Proc 27*, 1965, p. 203. [Online]. Available: http://www.multicians.org/fjcc3.html

5. J. Licklider and R. W. Taylor, "The Computer as a Communication Device," 1968. [Online]. Available: http://memex.org/licklider.pdf

6. L. Smarr and C. Catlett, "Metacomputing," *Communications of the ACM*, vol. 35, no. 6, pp. 44–52, 1992.

7. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputing Applications*, vol. 15, no. 3, 2002. [Online]. Available: http://www.globus.org/research/papers/anatomy.pdf

8. I. Foster, "The Grid: A New Infrastructure for 21st Century Science," *Physics Today*, vol. 55, no. 22, p. 42, 2002. [Online]. Available: http://www.aip.org/pt/vol-55/iss-2/p42.html

9. "The Access Grid Web Page," Web Page. [Online]. Available: http://www-fp.mcs.anl.gov/fl/accessgrid/

10. "NEESgrid Homepage," Web Page. [Online]. Available: http://www.neesgrid.org/

11. G. von Laszewski, M.-H. Su, J. A. Insley, I. Foster, J. Bresnahan, C. Kesselman, M. Thiebaux, M. L. Rivers, S. Wang, B. Tieman, and I. McNulty, "Real-Time Analysis, Visualization, and Steering of Microtomography Experiments at Photon Sources," in *Ninth SIAM Conference on Parallel Processing for Scientific Computing*, San Antonio, TX, 22-24 Mar. 1999. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/papers/vonLaszewski--siamCmt99.pdf

12. I. Foster, J. Insley, G. von Laszewski, C. Kesselman, and M. Thiebaux, "Data Visualization: Data Exploration on the Grid," *IEEE Computer*, vol. 14, pp. 36–41, Dec. 1999.

13. G. von Laszewski, M. Westbrook, I. Foster, E. Westbrook, and C. Barnes, "Using Computational Grid Capabilities to Enhance the Ability of an X-Ray Source for Structural Biology," *Cluster Computing*, vol. 3, no. 3, pp. 187–199, 2000. [Online]. Available: ftp://info.mcs.anl.gov/pub/tech_reports/P785.ps.Z

14. A. Oram, Ed., *Peer-To-Peer*. O'Reiley, 2001.

15. G. von Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke, "CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids," in *ACM Java Grande 2000 Conference*, San Francisco, CA, 3-5 June 2000, pp. 97–106. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/papers/cog-final.pdf

16. G. von Laszewski and K. Amin, "Java CoG Kit GridSDK," Web PAge. [Online]. Available: http://www.globus.org/cog/projects/gridsdk/

17. R. Chinnici, M. Gudgin, J.-J. Moreau, and S. Weerawarana, "Web Services Description Language Version 1.2," July 2002, w3C Working Draft 9. [Online]. Available: http://www.w3.org/TR/2002/WD-wsdl12-20020709/

18. J. Scheinblum, "An introduction to SOAP," Web Page, 2001. [Online]. Available: http://builder.cnet.com/webbuilding/0-7704-8-4874769-1.html

19. D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, "Simple Object Access Protocol (SOAP) 1.1," May 2000. [Online]. Available: http://www.w3.org/TR/SOAP

20. G. von Laszewski, "A Loosely Coupled Metacomputer: Cooperating Job Submissions Across Multiple Supercomputing Sites," *Concurrency, Experience, and Practice*, vol. 11, no. 5, pp. 933–948, Dec. 1999. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--CooperatingJobs.ps

21. "Unicore," Web Page. [Online]. Available: http://www.unicore.de/

22. A. S. Grimshaw and W. A. Wulf, "The Legion Vision of a Worldwide Virtual Computer," *Communications of the ACM*, vol. 40, no. 1, pp. 39–45, January 1997. [Online]. Available: http://legion.virginia.edu/copy-cacm.html

23. "Condor: High Throughput Computing," Web Page. [Online]. Available: http://www.cs.wisc.edu/condor/

24. "The Global Grid Forum Web Page," Web Page. [Online]. Available: http://www.gridforum.org

25. "Portable Batch System," Web Page, Veridian Systems. [Online]. Available: http://www.openpbs.org/

26. R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch, "A National-Scale Authentication Infrastructure," *IEEE Computer*, vol. 33, no. 12, pp. 60–66, 2000.

27. G. von Laszewski, J. Gawor, C. J. Peña, and I. Foster, "InfoGram: A Peer-to-Peer Information and Job Submission Service," in *Proceedings of the 11th Symposium on High Performance Distributed Computing*, Edinbrough, U.K., 24-26 July 2002, pp. 333–342. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/papers/infogram.ps

28. B. Allcock and R. Madduri, "Reliable File Transfer Service," Web Page. [Online]. Available: http://www-unix.globus.org/ogsa/docs/alpha3/services/reliable_transfer.html

29. "NSF Middleware Initiative," Web Page. [Online]. Available: http://www.nsf-middleware.org/Middleware/

30. "Scientific Discovery through Advanced Computing (SciDAC)," Web Page, 2001. [Online]. Available: http://scidac.org/

31. "Information Power Grid Engeneering and Research Site," Web Page, 2001. [Online]. Available: http://www.ipg.nasa.gov/

32. G. von Laszewski, S. Fitzgerald, I. Foster, C. Kesselman, W. Smith, and S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Computations," in *Proceedings of the 6th IEEE Symposium on High-Performance Distributed Computing*, 5-8 Aug. 1997, pp. 365–375. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/papers/fitzgerald--hpdc97.pdf

33. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing." in *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed*

Computing. San Francisco, CA: IEEE Press, 7-9 Aug. 2001, pp. 181–184, http://www.globus.org/research/papers/MDS-HPDC.pdf.

34. "Load Sharing Facility," Web Page, Platform Computing, Inc. [Online]. Available: http://www.platform.com/

35. K. Czajkowski, I. Foster, and C. Kesselman, "Co-allocation Services for Computational Grids," in *Proceedings of the 8th IEEE Symposium on High Performance Distributed Computing*, 1999.

36. C. Team, *Condor Version 6.2.2 Manual*, 6th ed., University of Wisconsin-Madison, Professor Miron Livny, 7367 Computer Science, 1210 West Dayton St, Madison, WI 53706-1685, 2001. [Online]. Available: http://www.cs.wisc.edu/condor/manual/v6.2/condor-V6_2-Manual.pdf

37. "SETI@Home Home Page," Web Page, Feb. 2002. [Online]. Available: http://setiathome.ssl.berekeley.edu/

38. S. Fields, "Hunting for Wasted Computing Power: New Software for Computing Networks Puts Idle PC's to Work," University of Wisconsin Research Sampler, 1993. [Online]. Available: http://www.cs.wisc.edu/condor/doc/WiscIdea.html

39. X. Evers, J. F. C. M. de Jongh, R. Boontje, D. H. J. Epema, and R. van Dantzig, "Condor flocking: Load sharing between pools of workstations," Delft University of Technology, Delft, The Netherlands, Tech. Rep. DUT-TWI-93-104, 1993. [Online]. Available: http://citeseer.nj.nec.com/evers93condor.html

40. D. H. J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne, "A worldwide flock of condors: load sharing among workstation clusters," Delft University of Technology, Delft, The Netherlands, Tech. Rep. DUT-TWI-95-130, 1995. [Online]. Available: http://citeseer.nj.nec.com/epema96worldwide.html

41. E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky, "SETI@home-massively distributed computing for SETI," *Computing in Science & Engineering*, vol. 3, no. 1, pp. 78–83, January–February 2001.

42. D. Molnar, "The SETIat-home Problem," *ACM*, no. 1, jan 2001. [Online]. Available: http://www.acm.org/crossroads/columns/onpatrol/september2000.html

43. G. Bell and J. Gray, "What's next in high-performance computing," *Communications of the ACM*, vol. 45, no. 2, pp. 91–95, Feb. 2002. [Online]. Available: http://doi.acm.org/10.1145/503124.503129

44. "Folding@home." [Online]. Available: http://folding.stanford.edu/

45. B. Gaidioz, R. Wolski, and B. Tourancheau, "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. [Online]. Available: http://www.cs.ucsb.edu/~rich/publications/

46. S. Verma, J. Gawor, G. von Laszewski, and M. Parashar, "A CORBA Commodity Grid Kit," in *2nd International Workshop on Grid Computing in conjunction with Supercomputing 2001 (SC2001)*, Denver, Colorado, November 12 2001. [Online]. Available: http://www.globus.org/cog

47. G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8-9, pp. 643–662, 2001. [Online]. Available: http://www.globus.org/cog/documentation/papers/cog-cpe-final.pdf

48. V. Getov, G. von Laszewski, M. Philippsen, and I. Foster, "Multi-Paradigm Communications in Java for Grid Computing," *Communications of ACM*, vol. 44, no. 10, pp. 119–125, Oct. 2001. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/bib/papers/vonLaszewski--cacm.pdf

49. G. von Laszewski, I. Foster, J. Gawor, P. Lane, N. Rehn, and M. Russell, "Designing Grid-based Problem Solving Environments and Portals," in *34th Hawaiian International Conference on System Science*, Maui, Hawaii, 3-6 Jan. 2001. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/bib/papers/vonLaszewski--cog-pse-final.pdf

50. "The Commodity Grid Project," Web Page. [Online]. Available: http://www.globus.org/cog

51. A. Jhoney, M. Kuchhal, and Venkatakrishnan, "Grid Application Framework for Java (GAF4J)," IBM Software Labs, India, Tech. Rep., 2003. [Online]. Available: https://secure.alphaworks.ibm.com/tech/GAF4J

52. M. Govindaraju, S. Krishnan, K. Chiu, A. Slominski, D. Gannon, and R. Bramley, "XCAT 2.0 : A Component Based Programming Model for Grid Web Services," in *Submitted to Grid 2002, 3rd International Workshop on Grid Computing*, 2002. [Online]. Available: http://www.extreme.indiana.edu/xcat

53. H. Nakada, M. Sato, and S. Sekiguchi, "Design and Implementations of Ninf: towards a Global Computing Infrastructure," *Future Generation Computing Systems*, vol. 15, no. 5-6, pp. 649–658, 1999.

54. I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed

Systems Integration," Web page, Jan. 2002. [Online]. Available: http://www.globus.org/research/papers/ogsa.pdf

55. "World Wide Web Consortium," Web Page. [Online]. Available: http://www.w3.org/

56. "Web Services Gloassary," Web Page, May 2003, w3C Working Draft. [Online]. Available: http://www.w3c.org/TR/ws-gloss/

57. "Universal Description, Discovery and Integration of Business for the Web," Web Page. [Online]. Available: http://www.uddi.org/

58. K. Ballinger, P. Brittenham, A. Malhotra, W. Nagy, and S. Pharies, "Web Services Inspection Language (WS-Inspection) 1.0," Web Page, November 2001. [Online]. Available: http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html

59. G. C. Fox, "Portals for Web Based Education and Computational Science," 2000.

60. L. Smarr, "Infrastructures for Science Portals," 2001. [Online]. Available: http://www.computer.org/internet/v4n1/smarr.htm

61. G. C. Fox and W. Furmanski, "High Performance Commodity Computing," in *The Grid: Bluepirnt for a new computing infrastructure*, I. Foster and C. Kesselman, Eds. Morgam Kaufman, 1999.

62. "The Jetspeed Webpage," Web page. [Online]. Available: http://jakarta.apache.org/jetspeed/

63. "Grid Interoperability Project," Web Page. [Online]. Available: http://www.grid-interoperability.org/

64. D. Snelling, S. van den Berghe, G. von Laszewski, P. Wieder, J. MacLaren, J. Brooke, D. Nicole, and H.-C. Hoppe., "A unicore globus interoperability layer," Web page, Nov. 2001. [Online]. Available: http://www.grid-interoperability.org/D4.1b_draft.pdf

65. "NPACI HotPage," Web Page, 2001. [Online]. Available: https://hotpage.npaci.edu/

66. "The Globus Project," Web Page. [Online]. Available: http://www.globus.org

67. "National Partnership for Advanced Computational Infrastructure," Web Page. [Online]. Available: http://www.npaci.edu/

68. G. von Laszewski, M. Russell, I. Foster, J. Shalf, G. Allen, G. Daues, J. Novotny, and E. Seidel, "Community Software Development with the Astrophysics Simulation Collaboratory," *Concurrency and*

Computation: Practice and Experience, vol. 14, pp. 1289–1301, 2002. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/bib/papers/vonLaszewski--cactus5.pdf

69. "ASC Portal Home Page," http://www.ascportal.org.

70. G. Allen, W. Benger, T. Goodale, H. Hege, G. Lanfermann, J. Masso, A. Merzky, T. Radke, E. Seidel, and J. Shalf, "Solving Einstein's Equations on Supercomputers," *IEEE Computer*, pp. 52–59, 1999. [Online]. Available: http://www.cactuscode.org

71. G. Allen, W. Benger, T. Goodale, H.-C. Hege, G. Lanfermann, A. Merzky, T. Radke, E. Seidel, and J. Shalf, "The Cactus Code: A Problem Solving Environment for the Grid," in *High-Performance Distributed Computing, 2000. Proceedings. The Ninth International Symposium on*, Pitsburg, PA, August 2000, pp. 253 –260. [Online]. Available: http://xplore2.ieee.org/iel5/6975/18801/00868657.pdf?isNumber=18801

72. T. Prudhomme and K. D. Mish, "NEESgrid: A Distributed Virtual Laboratory for Advanced Earthquake Experimentation and Simulation: Project Execution Plan," NEES, Tech. Rep. 2001-02, June 2001. [Online]. Available: http://www.neesgrid.org/html/TR_2001/NEESgrid_TR.2001-02.pdf

73. T. Prudhomme, C. Kesselman, T. Finholt, I. Foster, D. Parsons, D. Abrams, J.-P. Bardet, R. Pennington, J. Towns, R. Butler, J. Futrelle, N. Zaluzec, and J. Hardin, "NEESgrid: A Distributed Virtual Laboratory for Advanced Earthquake Experimentation and Simulation: Scoping Study," NEES, Tech. Rep. 2001-02, February 2001. [Online]. Available: http://www.neesgrid.org/html/TR_2001/NEESgrid_TR.2001-04.pdf

74. "Particle Physics Data Grid," Web Page, 2001. [Online]. Available: http://www.ppdg.net/

75. "The International Virtual Data Grid Laboratory," Web Page. [Online]. Available: http://www.ivdgl.org/

76. "GriPhyN - Grid Physics Network," Web page. [Online]. Available: http://www.griphyn.org/index.php

77. G. von Laszewski, B. Ruscic, P. Wagstrom, S. Krishnan, K. Amin, S. Nijsure, R. Pinzon, M. L. Morton, S. Bittner, M. Minkoff, A. Wagner, and J. C. Hewson, "A Grid Service Based Active Thermochemical Table Framework," in *Third International Workshop on Grid Computing*, ser. Lecture Notes in Computer Science, vol. 2536. Baltimore, MD: Springer, 18 Nov. 2002, pp. 25–38. [Online]. Available: http://www.mcs.anl.gov/~laszewsk/bib/papers/vonLaszewski--cmcs.pdf

78. C. Pancerella, J. D. Myers, T. C. Allison, K. Amin, S. Bittner, B. Didier, M. Frenklach, J. William H. Green, Y.-L. Ho, J. Hewson, W. Koegler, C. Lansing, D. Leahy, M. Lee, R. McCoy, M. Minkoff, S. Nijsure, G. von Laszewski, D. Montoya, R. Pinzon, W. Pitz, L. Rahn, B. Ruscic, K. Schuchardt, E. Stephan, A. Wagner, B. Wang, T. Windus, L. Xu, and C. Yang, "Metadata in the collaboratory for multi-scale chemical science," in *2003 Dublin Core Conference: Supporting Communities of Discourse and Practice-Metadata Research and Applications*, Seatle, WA, 28 Sept.2 Oct. 2003.

79. I. Foster, J.-S. Vöckler, M. Wilde, and Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying and Automating Data Derivation," in *14th International Conference on Scientific Database Management*, Edinburgh, 2002.

80. E. Deelman, J. Blythe, Y. Gil, and C. Kesselman, "Pegasus: Planning for Execution in Grids," ISI, Tech. Rep. TR-2002-20, November 2002.

81. "DAGMan (Directed Acyclic Graph Manager)," Web Page. [Online]. Available: http://www.cs.wisc.edu/condor/dagman/

Biographies

Gregor von Laszewski is a scientist at Argonne National Laboratory and a fellow at the University of Chicago Computation Institute. He specializes in Grid computing. He is the principal investigator of the Java CoG Kit, which provided the first defacto standard for accessing Globus through Java. He obtained his B.S. and his M.S in computer science at University of Bonn, Germany. In 1996, he received a Ph.D. form Syracuse University, where he developed a loosely coupled metacomputing environment allowing seamless access to several supercomputing centers through workflows. He has published more than forty papers and technical reports in the area of distributed and Grid computing.

Kaizar Amin is a Ph.D. student in the computer science department at the University of North Texas. He received his B.E in computer engineering from Mumbai University, India and M.S in computer science from the University of North Texas. His research interest includes adaptive middleware for Grid computing, peer-to-peer Grids, Grid workflows, and Grid mobility using mobile agents.