A visual tour of Open Grid Services Architecture

Examine the component structure of OGSA

Level: Introductory

Jay Unger (unger@us.ibm.com), IBM Senior Technical Staff Member, IBM
Matt Haynos (mph@us.ibm.com), Program Director, Grid Strategy and Technology, IBM

August 2003

Grid computing is a promising emerging technology that is growing in mindshare and relevance in the industry. Applications that take advantage of grids are under development in both academic and commercial organizations. You can find many definitions of Grid computing (see Resources), but the essence of the grid is the federation of computing resources to accelerate application processing, plus the virtualization of these resources. At its core, the grid is all about distributed computing and resource management.

A wide array of heterogeneous resources comprise a grid, and it's important that they interact and behave in well-known and consistent ways. The need for open standards that define this interaction and encourage interoperability between components supplied from different sources was the motivation for the Open Grid Services Architecture (OGSA), specified by the Open Grid Services Infrastructure working group of the Global Grid Forum (GGF) in June 2002. In this article, we lay out the components of OGSA and explain their significance. The Globus Toolkit 3 is the first major implementation of the standard; others are under development (see Resources).

What are the objectives of OGSA?

The objectives of OGSA are to:

- Manage resources across distributed heterogeneous platforms.
- Deliver seamless quality of service (QoS). The topology of grids is often complex. Interaction of grid resources is usually dynamic. It's important that the grid provide robust, behind-the-scenes services such as authorization, access control, and delegation.
- Provide a common base for autonomic management solutions. A grid can contain many resources, with numerous combinations of configurations, interactions, and changing state and failure modes. Some form of intelligent self regulation and autonomic management of these resources is necessary.

Related content:

A developer's view of OGSI and OGSI-based Grid computing
Globus Toolkit 3.0 and the OGSI architecture
Subscribe to the developerWorks newsletter
developerWorks Toolbox subscription

Also in the Grid computing zone:

Tools and products
Articles

- Define open, published interfaces. OGSA is an open standard managed by the GGF standards body. For interoperability of diverse resources, grids must be built on standard interfaces and protocols.
- Exploit industry standard integration technologies. The authors of OGSA had foresight to leverage existing solutions where appropriate. The foundation of OGSA is Web services.

Also in the Web services zone:

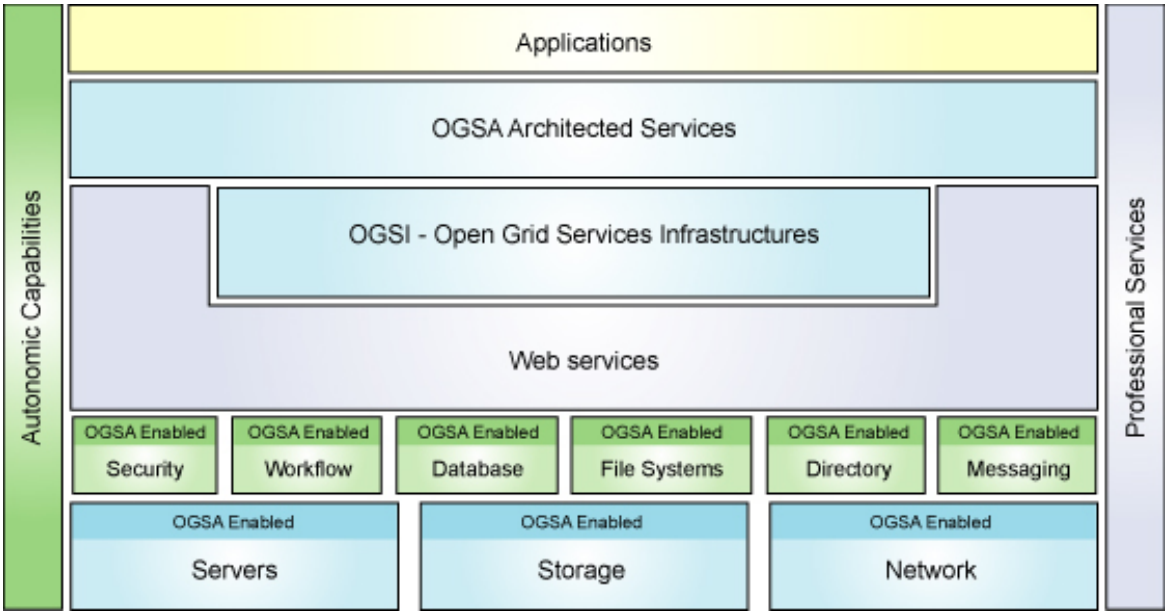
Tutorials

Tools and products

Articles

The OGSA architecture

Figure 1. OGSA main architecture



Four main layers comprise the OGSA architecture: See Figure 1. Starting from the bottom, they are:

- Resources -- physical resources and logical resources
- Web services, plus the OGSI extensions that define grid services
- OGSA architected services
- Grid applications

Let's look at these layers, one at a time.

Physical and logical resources layer
The concept of resources is central to OGSA and to grid computing in general. Resources comprise the capabilities of the grid, and are not limited to processors.

Physical resources include servers, storage, and network.

Above the physical resources are *logical resources*. They provide additional function by virtualizing and aggregating the resources in the physical layer. General purpose middleware such as file systems, database managers, directories, and workflow managers provide these abstract services on top of the physical grid.

Web services layer
The second layer in the OGSA architecture is Web services.

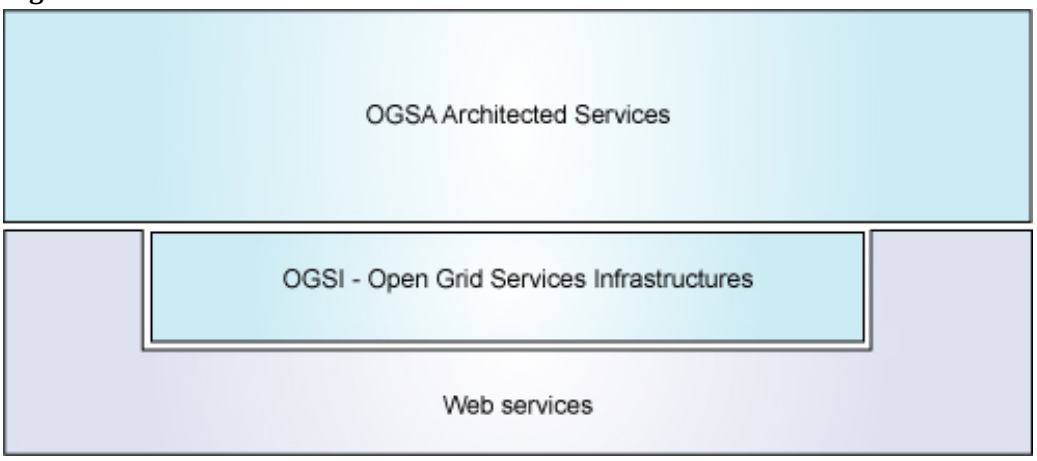
Here's an important tenet of OGSA: All grid resources -- both logical and physical -- are modeled as services. The Open Grid Services Infrastructure (OGSI) specification defines grid services and builds on top of standard Web services technology. OGSI exploits the mechanisms of Web services like XML and WSDL to specify standard interfaces, behaviors, and interaction for all grid resources. OGSI extends the definition of Web services to provide capabilities for dynamic, stateful, and manageable Web services that are required to model the resources of the grid. We'll discuss these extensions later in the article.

OGSA architected grid services layer
The Web services layer, with its OGSI extensions, provide a base infrastructure for the next layer -- architected grid services. The Global Grid Forum is currently working to define many of these architected grid services in areas like program execution, data services, and core services. Some are already defined, and some implementations have already appeared. As implementations of these newly architected services begin to appear, OGSA will become a more useful service-oriented architecture (SOA).

Grid applications layer
Over time, as a rich set of grid-architected services continues to be developed, new grid applications that use one or more grid architected services will appear. These applications comprise the fourth main layer of the OGSA architecture.

Extending Web services for grid

Figure 2. The structure of OGSA

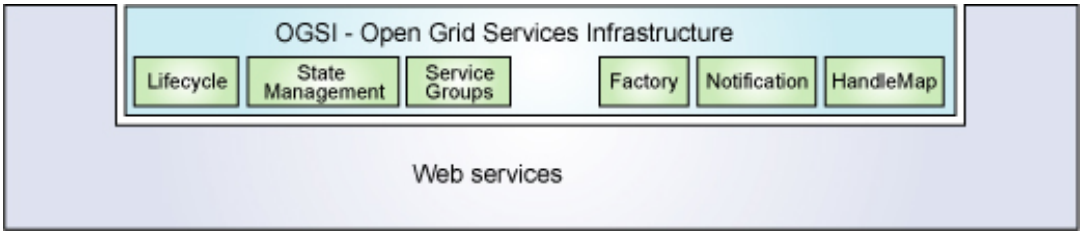


Let's look more closely at the two main logical components of OGSA -- the Web services-plus-OGSI layer, and the OGSA architected services layer. See Figure 2. Why are they separated like this? The GGF OGSA working group believed it was necessary to augment core Web services functionality to address grid services requirements. OGSI extends Web services by introducing interfaces and conventions in two main areas.

- First, there's the dynamic and potentially transient nature of services in a grid. In a grid, particular service instances may come and go as work is dispatched, as resources are configured and provisioned, and as system state changes. Therefore, grid services need interfaces to manage their creation, destruction, and life cycle management.
- Second, there's state. Grid services can have attributes and data associated with them. This is similar in concept to the traditional structure of objects in object-oriented programming. Objects have behavior and data. Likewise, Web services needed to be extended to support state data associated with grid services.

An interaction model for Web services

Figure 3. OGSI components



OGSI introduces an interaction model for grid services. OGSI provides a uniform way for software developers to model and interact with grid services by providing interfaces for discovery, life cycle, state

management, creation and destruction, event notification, and reference management. These are depicted in Figure 3. Whether a software developer is developing a grid service or an application, the OGSI programming model provides a consistent way for grid software to interact.

Let's take a closer look at the interfaces and conventions OGSI introduces.

Factory. Grid services that implement this interface provide a way to create new grid services. Factories may create temporary instances of limited function, such as a scheduler creating a service to represent the execution of a particular job, or they may create longer-lived services such as a local replica of a frequently used data set. Not all grid services are created dynamically. For example, some might be created as the result of an instance of a physical resource in the grid such as a processor, storage, or network device.

Life cycle. Because grid services may be transient, grid service instances are created with a specified lifetime. The lifetime of any particular service instance can be negotiated and extended as required by components that are dependent on or manage that service. The life cycle mechanism was architected to prevent grid services from consuming resources indefinitely without requiring a large scale distributed "garbage collection" scavenger.

State management. Grid services can have state. OGSI specifies a framework for representing this state called Service Data and a mechanism for inspecting or modifying that state named Find/SetServiceData. Further, OGSI requires a minimal amount of state in Service Data Elements that every grid service must support, and requires that all services implement the Find/SetServiceData portType.

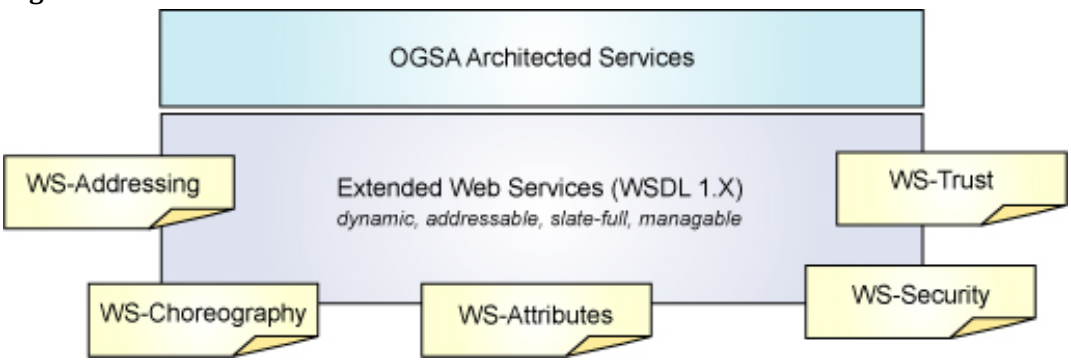
Service groups. Service groups are collections of grid services that are indexed, using Service Data, for some particular purpose. For example, they might be used to collect all the services that represent the resources in a particular cluster-node within the grid.

Notification. The state information (Service Data) that is modeled for grid services changes as the system runs. Many interactions between grid services require dynamic monitoring of changing state. Notification applies a traditional publish/subscribe paradigm to this monitoring. Grid services support an interface (NotificationSource) to permit other grid services (NotificationSink) to subscribe to changes.

HandleMap. When factories are used to create a new instance of a grid service, the factory returns the identity of the newly instantiated service. This identity is composed of two parts, a Grid Service Handle (GSH) and a Grid Service Reference (GSR). A GSH is guaranteed to reference the grid service indefinitely, while a GSR can change within the grid services lifetime. The HandleMap interface provides a way to obtain a GSR given a GSH. This might seem simple, but there are a number of associated intricacies with such a request, described in the paper "The Physiology of the Grid" (see Resources).

More about OGSI and Web services

Figure 4. OGSI and Web services

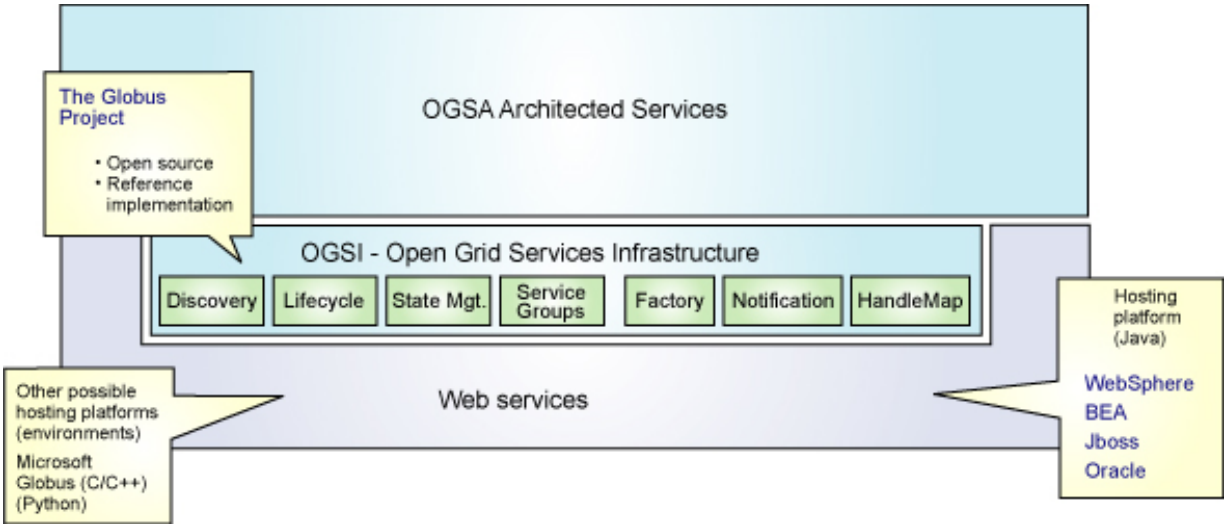


As we've seen, the OGSA architecture enhances Web services to accommodate requirements of the grid. These enhancements are specified in OGSI. As the OGSI specification was finalized and implementations began to appear, some standards organizations became interested in incorporating most of the functionality

outlined in OGSI within appropriate Web services standards. This makes good sense. A lot of what OGSI deals with, in some respects, is not unique to grid computing, but is required for building robust, service-oriented architectures. Over time, it's expected that much of the OGSI functionality will be incorporated in Web services standards. This is depicted in Figure 4, which lists several emerging Web service standards that might incorporate OGSI. In Figure 4, we refer to these enhanced Web services as *extended* Web services.

Implementations of OGSI

Figure 5. OGSI and Web services hosting



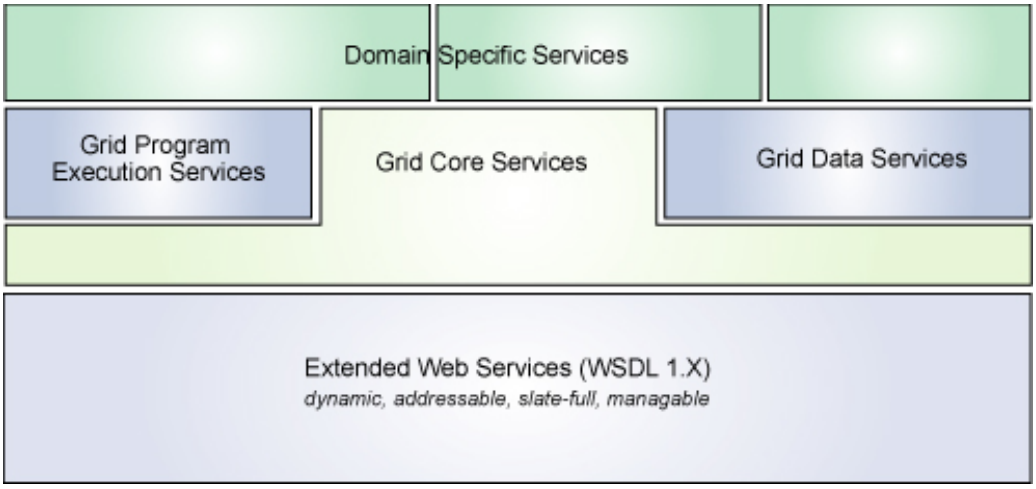
The Globus Toolkit 3 (GT3) is the first full-scale implementation of the OGSI standard. The toolkit was developed by the Globus Project, a research and development project focused on enabling the application of Grid concepts to scientific, engineering, and commercial computing. The toolkit was written in Java language using the J2EE framework. As the core of the grid service architecture, OGSI needs to be hosted on a delivery platform that supports Web services. Although OGSI was initially done in Java code and hosted within a J2EE runtime environment, nothing precludes OGSI from being implemented in other programming languages and hosted in other environments. In fact, for OGSA to grow in acceptance, OGSI will need to be enabled on multiple hosting platforms.

What platforms? In Figure 5, we see that a Java implementation of OGSI can be potentially hosted on any of several J2EE environments such as JBOSS, WebSphere, or BEA Weblogic. This is one of the distinct advantages of implementing OGSI -- and for that matter any software -- in Java technology. However, alternative platforms such as a traditional C or C++ environment or C# and Microsoft .Net are feasible hosting environments for OGSI. There are already initial implementations of OGSI running in other environments, including C#/.Net and Python.

It's expected that many of the OGSI implementations will be delivered via the open source development model and that existing reference implementations (the Globus Toolkit 3) will be used unmodified in appropriate hosting environments. Ideally, a small number of core implementations of OGSI -- one per hosting platform -- will be jointly developed by the industry and used in many products. While OGSI is novel in the respect that it extends Web services, it's not expected that software vendors will be able to differentiate themselves based on the quality of their OGSI implementations. Thus, shared development and broad use of OGSI is both intuitive and desirable.

OGSA architected services

Figure 6. The structure of OGSA architected services



OGSI is certainly an important step in developing a service-oriented architecture for grids. However, in order for useful applications to be developed, a rich set of grid services -- the OGSA architected services -- will need to be implemented and delivered by both open source efforts like the Globus project and by middleware software companies. In a sense, OGSI and the extensions it provides for Web services are necessary but insufficient for the maturation of the service-oriented architecture.

In Figure 6, we further categorize grid architected services into four categories:

- Grid core services
- Grid program execution services
- Grid data services
- Domain-specific services

The first three categories represent areas of active work by GGF research or working groups. Over time, as these services mature, domain-specific services can then be specified, which will make use of the functionality that these services supply. It's important today that the GGF working groups are concentrating on specifying a broad set of useful grid services that software vendors and developers can then begin to implement.

Let's take a closer look at each of these categories.

Grid core services

Figure 7. Grid core services

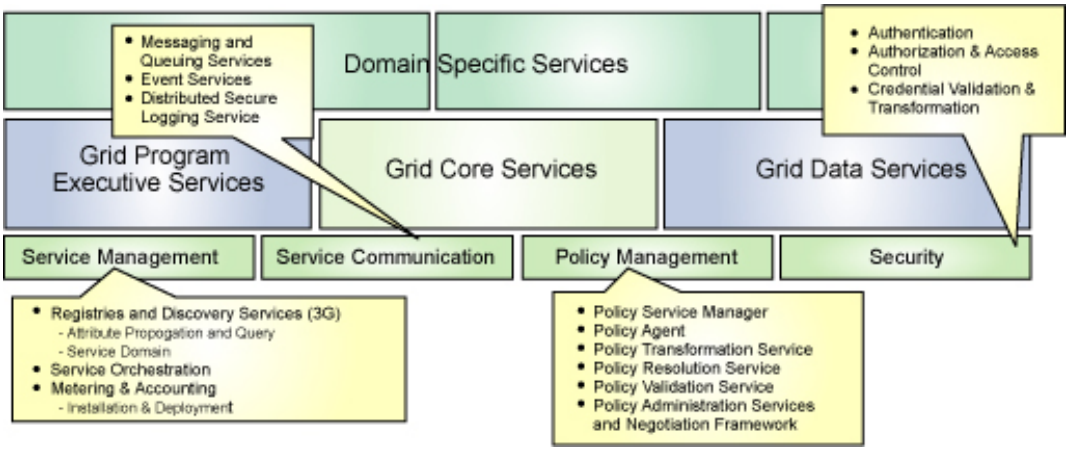


Figure 7 shows that the grid core services are composed of four main types of services:

- Service management

- Service communication
- Policy management
- Security

Unlike the OGSI functions that are largely implemented as extensions to basic Web services protocols and an interaction model, these core services are actually implemented as grid services (upon the OGSI base). These services are considered *core* primarily because it's expected they will be broadly exploited by most -- if not all -- higher-level services implemented either in support of program execution or data access, or as domain-specific services.

Service management provides functions that manage the services deployed in the distributed grid. Service management automates and assists with a variety of installation, maintenance, monitoring, and troubleshooting tasks within a grid system. It includes functions for provisioning and deploying the system components. It also includes functions for collecting and exchanging data about the operation of the grid. This data is used for both "online" and "offline" management operations, and includes information about faults, events, problem determination, auditing, metering, accounting, and billing.

Service communication includes a range of functions that support the basic methods for grid services to communicate with each other. They support several communication models that may be composed to permit effective interservice communication, including queued messages, publish-subscribe event notification, and reliable distributed logging.

Policy services create a general framework for creation, administration, and management of policies and agreements for system operation. These include policies governing security, resource allocation, and performance as well as an infrastructure for "policy aware" services to use policies to govern their operation. Policy and agreement documents provide a mechanism for the representation and negotiation of terms between service providers and their clients (either user requests or other services). These terms include specifications, requirements, and objectives for function, performance, and quality that the suppliers and consumers exchange and that they can then use to influence their interactions.

Security services support, integrate, and unify popular security models, mechanisms, protocols, and technologies in a way that enables a variety of systems to interoperate securely. These security services enable and extend core Web services security protocols and bindings and provide service-oriented mechanisms for authentication, authorization, trust policy enforcement, credential transformation, and the like.

Grid program execution and data services

Figure 8. Grid program execution services and grid data services

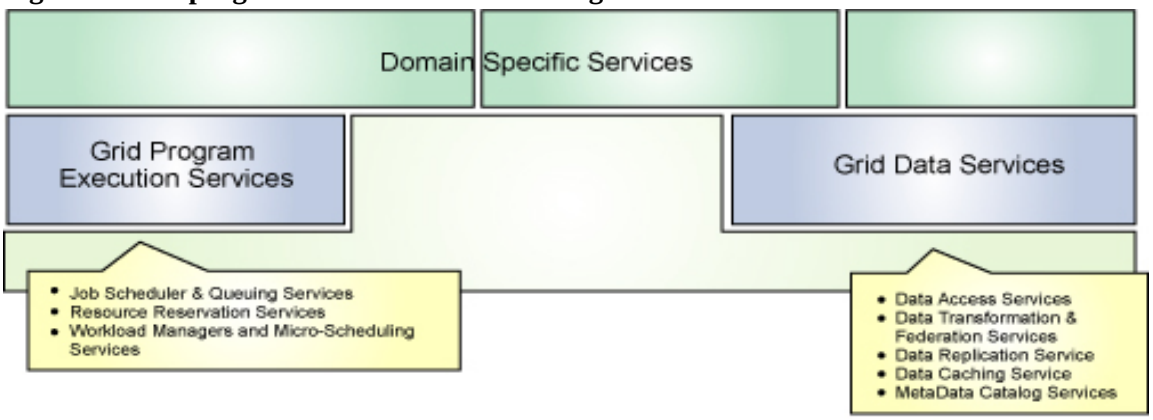


Figure 8 depicts two important classes of grid services:

- Grid program execution services
- Grid data services

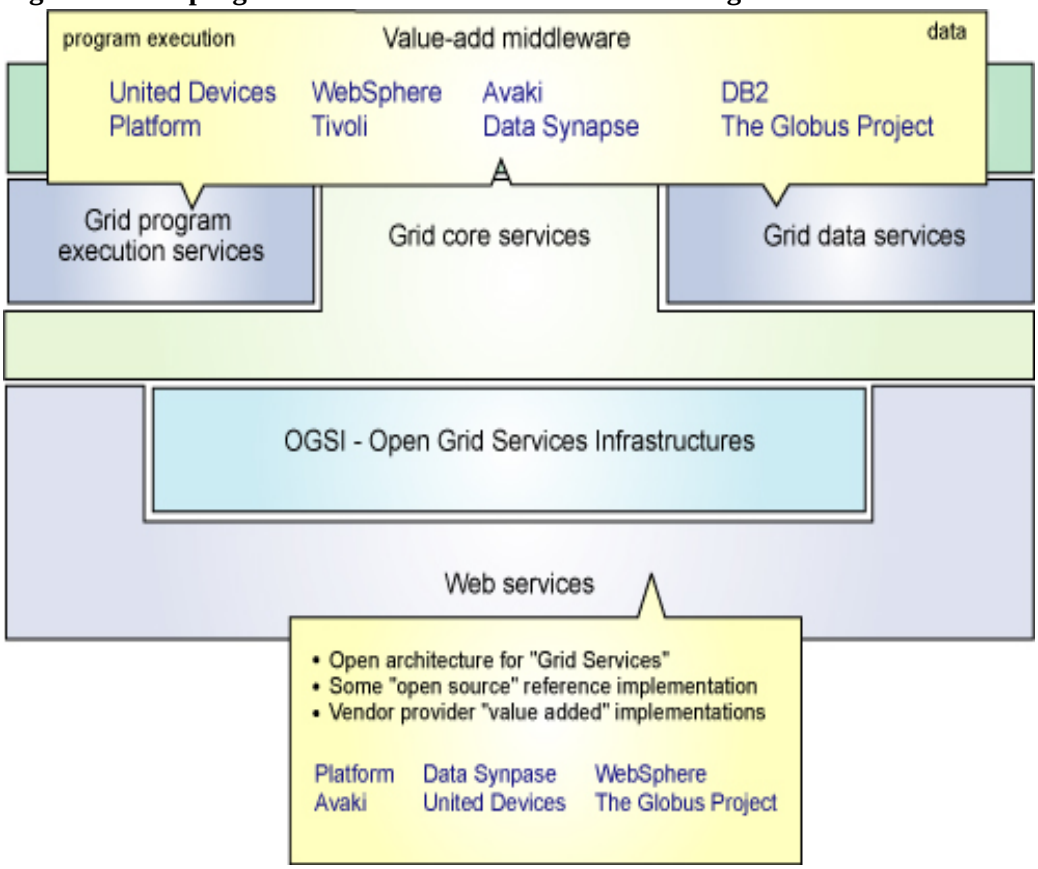
Grid program execution services. While OGSI and grid core services are generally applicable to any distributed computing system, grid program execution class is unique to the grid model of distributed task execution that supports high-performance computing, parallelism, and distributed collaboration. Principles of job scheduling and workload management implemented as part of this class of services are central to Grid computing and the ability to virtualize processing resources. We have already seen early specifications of interfaces in this category, such as the Community Scheduling Framework (CSF) announced at GGF 8 in Seattle in June 2003.

Grid data services. Complementing the compute virtualization conventions specified by program execution services are the services that make up grid data services. These interfaces support the concept of data virtualization and provide mechanisms related to distributed access to information of many types including databases, files, documents, content stores, and application-generated streams. Grid data services will exploit and virtualize data using placement methods like data replication, caching, and high-performance data movement to give applications required Quality of Service (QoS) access across the distributed grid. Methods for federating multiple disparate, distributed data sources may also provide integration of data stored under differing schemas such as files and relational databases.

In this category it's evident that OGSA is placing data resources on equivalent footing with compute resources.

Grid program execution and data services hosting

Figure 9. Grid program execution and data services hosting



Vendors probably won't compete by offering a wide range of implementations of OGSI. Instead, as part of the "fabric" of Web services implementations, vendors who offer OGSI implementations will likely directly use existing open source implementations provided by organizations like Globus, or they will integrate implementations with their hosting platform products like WebSphere, WebLogic, Apache, or .Net.

However, grid architected services provide some natural areas ripe with opportunity for vendors and

organizations to compete and differentiate themselves. This competition will create an "economy" of grid software providers whose innovation will help drive the acceptance of standards like OGSI/OGSA, and this will allow customers to build systems out of interoperable components. In addition, areas of functionality in grid program execution and data services will require innovation and novel approaches , and these will speed the market acceptance of grid solutions and provide market opportunities for vendors. In Figure 9, we see that grid core services are likely to see a mix of open source reference implementations and vendor-provided "value added" implementations. Many technologies in this area are likely to be commoditized, but areas like policy and security will provide vendors a chance to differentiate themselves.

Implementations in both grid program execution and data services are expected to consist largely of value-add implementations from companies like Platform Computing, IBM, DataSynapse, and Avaki. These areas represent significant opportunities for vendors to integrate leading middleware offerings within the OGSA framework and will allow a rich ecosystem of grid solutions to develop.

Conclusion

We've laid out and described the component structure of OGSA. With the delivery of the initial implementations of OGSI, OGSA is getting ready to accelerate its entry into mainstream commercial computing environments. Whether initiatives are referred to as organic computing, on demand computing, or adaptive computing, an open and comprehensive standard such as OGSA -- built on standard technologies -- is necessary for realizing distributed heterogeneous computing in the commercial world.

Resources

- For different definitions of Grid computing, see these resources:
  - "*What is the Grid*," by Ian Foster.
  - "*How Does One Really Characterize Grid Computing*," by Richard Freund.
  - "*An Overview of Grid Computing from Sun*."
  - "*What is Grid computing?" by IBM Corporation*.

- See a list of the technologies discussed in the OGSI Implementations Session at the Global Grid Forum GGF8 meeting in Seattle, June 2003.

- See The Physiology of the Grid.

About the authors

Jay Unger is a Senior Technical Staff Member and a lead technical architect with the IBM Grid Computing Emerging Business Opportunity organization. He has over 30 years of experience in the development of computers, operating systems, communications networks, and application systems. He joined IBM in 1974 in Kingston, New York to work on the development of IBM operating systems and was involved in the early development of MVS, JES, SNA, and coupled systems. He has worked in many technology areas, including databases and file systems, electronic publishing, imaging and document management systems, natural language processing and information retrieval, bioinformatics, imbedded systems, distributed systems and network computing, Web technologies, and high-performance computing. He is a member of the IBM Academy of Technology.

Matt Haynos is a Program Director on IBM's Grid Strategy and Technology team, based in Somers, NY. He has various responsibilities on the team covering a broad range of initiatives related to building the IBM Grid computing business. He has held a variety of technical and managerial positions within IBM in the application development, program direction, and business development areas. He holds a BA in Computer Science / Applied Mathematics and Cognitive Science from the University of Rochester, and an MS in Computer Science from the University of Vermont. He lives with his wife and two sons in Connecticut.