Open Grid Services Architecture: What is it?

Summary by Jonathan Ledlie on January 24, 2003

1 Big Picture

OGSA is an API that describes the interaction of Grid components. It "defines mechanisms for creating, managing, and exchanging information among entities called Grid services." Its definition is in progress and was last updated (publicly) in October. It is defined in terms of the Web Service Definition Language (WSDL).

OGSA relates to Globus as follows. Globus v2 (?) (and younger) essentially glues together various Condor and Condor-like components running at universities and research labs. It was a trial run on what might be needed to get the Grid to work. Now, what was in Globus is being generalized to the specification that is OGSA. Globus v3 will be a reference implementation of OGSA.

The Grid manifesto is essentially broken into three main papers:

1. The Anatomy of the Grid is the earliest paper and describes the motivations behind having a global computation service used by *virtual organizations*. This paper states why the Grid cannot be built using existing technologies and why something new must be created (and why they need funding to do it). "Virtual organizations" are "dynamic collections of individuals, institutions, and resources." Appears in the International Journal of Supercomputing Applications, 2001.
2. The Physiology of the Grid describes OGSA at a fairly high level. This is what I read for this meeting.
3. Grid Service Specification is an evolving specification using the Web Service Definition Language (WSDL). I have not read this.

2 Physiology of the Grid

Some points from this paper:

- Service Providers (SPs) (*e.g.,* web hosting, content distribution, application, storage) need standardization. "B2B relationships are, in effect, virtual organizations."
- Microsoft .NET and Apache Axis are mentioned together. What is Apache Axis?
- While OGSA seems to me to be an API, they never use this term.
- They imply a very flexible API. For example, that a "reference implementation" would be capable of doing certain things on any platform (*e.g.,* performing traces). However, some platforms would be more capable of doing this function than others and this capability should not be masked by the interface. I'm not sure how they plan to achieve this.
- They mention "transient services" a lot and include pings to let a service know that its consumer is still alive. They never use the word "leases" however and seem to disparage Jini. "Soft-state protocols allow state established at a remote location to be discarded eventually, unless refreshed by a stream of subsequent "keep-alive" messages." Their service creation and destruction seems weak.
- Naming. "Every Grid service instance is assigned a globally unique name, the *Grid service handle (GSH)*, that distinguishes a specific Grid service instance from all other Grid service instances that have existed, exist now, or will exist in the future." How this is achieved is left as an exercise for the reader :).
- "Unlike a GSH, which is invariant, the [Grid service reference] GSR(s) for a Grid service instance can change over that service's lifetime. A GSR has an explicit expiration time, and may become invalid at any time during a service's lifetime...."
- Much like Jini, they have Factories, Service Discovery, Registries, Notification sources and sinks.
- There seem to be many opportunities for distributed lease data structures, like a leased DHT.

2.1 The Anatomy of the Grid: Enabling Scalable Virtual Organizations

Authors: *Ian Foster, Carl Kesselman, Steven Tuecke*

The authors attempt to formulate a definition for what the Grid is trying to be and what components it needs to have in order to take that shape. In particular they argue that existing technologies do not support "virtual organizations" from being able to flexibly share one another's resources. For this reason, a new technology, the Grid, is necessary. The Grid's architecture follows an hourglass form, with broad interface definitions at the top and bottom, and narrow ones in the middle. The layered architecture, in order, is: application, collective, resource, connectivity, and fabric. Although the authors occasionally depict the architecture as a stack (and, in fact, place it next to the OSI stack for comparison), here the application can in fact make direct calls to the resource and connectivity layers. Another real deficit is that they do not differentiate between global and local perspectives (and seem to perceive of everything from a one-off global perspective). For example, it appears that collectives (collections of resources) are the same collection of resources from everyone's point of view — that my collection is your collection and vice versa. Similarly they use the centralized X.509 format instead of the more flexible and relative SPKI. This is particularly interesting because they suggest that sharing relationships between virtual organizations (VOs) "do not necessarily involve an explicitly named set of individuals" but instead may include "students." This level of indirection is what SPKI is good at; I'm not sure if it's available in X.509. They suggest that several major technology trends (*e.g.,* Internet, enterprise, distributed and p2p computing) all have much to gain from the Grid, but, strangely, they do not suggest that the Grid could benefit from them. In discussing the Grid architecture, the authors spend a great deal of their time talking about coscheduling, which is presumably of great import in the type of applications they are envisioning.

They offer a definition of the goal of the Grid: "flexible, secure, coordinate resource sharing among dynamic collections of individuals, institutions, and resources — what we refer to as virtual organizations."

The narrow components of the Grid architecture are the resource and connectivity protocols. The components in greater depth are:

Fabric Fabric components implement the local, resource-specific operations that occur on specific resources (whether physical or logical) as a result of sharing operations at higher levels. Here the authors include "enquiry" mechanisms that permit discovery of state and capabilities and "resource management" mechanisms (*e.g.,* General-purpose Architecture for Reservation and Allocation (GARA), Portable Batch System, and Condor).

Connectivity This layer enables the exchange of data between Fabric layer resources. It includes communication and authentication. Globus includes the Grid Security Infrastructure (GSI), which is mainly built on the Transport Layer Security (TLS) protocol. Local policies can be integrated via the Generic Authorization and Access (GAA) control interface.

Resource "Resource layer protocols are concerned entirely with individual resources and hence ignore issues of global state and atomic actions across distributed collections." The classes of resource protocols are (1) information and (2) management. These appear to be directly analogous to the Fabric's enquiry and resource management mechanisms. The Globus implementations of the resource level include: Grid Resource Information Protocol (GRIP, based on LDAP), Grid Resource Access and Management (GRAM, HTTP-based), and GridFTP.

Collective These are collections of resources and this layer offers services on collections. For example: directory services allow users to query for resources by name and/or attributes such as type, availability, or load; co-location and scheduling services, like Condor-G; data replication

The authors offer a nice rebuttal of many confusing points about the Grid. One in particular is that even though it is distributed, it does not need a new computing model — I bet Jim Waldo would disagree.