Summary: Integration of Legacy Grid Systems with Emerging Grid Standards [1]

A. Grimshaw, W. Kang, D. Merrill, M. Morgan
Department of Computer Science, University of Virginia

1. Introduction

The Open Grid Services Architecture (OGSA™) addresses the need for standardization of diverse grid services by defining a set of core capabilities and behaviors needed by loosely coupled, service-oriented Grid architectures. These OGSA standards and interfaces are based on ubiquitous, platform-neutral, technologies like SOAP, XML, and Web Services.

We expect that over time there will be many "native" implementations of OGSA services. At the same time, a number of grid systems already exist such as Globus [3], Legion [4], EGEE , NAREGI [5], and Unicore [2] that expose similar functionalities. Rather than rush to build native OGSA implementations, we believe many projects will choose to build OGSA-compliant proxies to allow for OGSA interoperability. This would facilitate the composition of islands of internally-proprietary grids that are connected by OGSA interfaces and mechanisms. Proxies will mediate and translate between standard OGSA service abstractions and legacy services and capabilities.

In the following sections, this summary briefly describes our OGSA proxy implementation that interconnects legacy Legion grids with grids that support the emerging OGSA specifications. Specifically we have constructed proxy services that support OGSA-ByteIO [6], WS-Directory[7], and WS-Naming[8].

2. Background

2.1 Relevant OGSA specifications. The Web Services Resource Framework (WSRF) is a group of cross-cutting specifications that grid service implementations can employ to facilitate some of the simpler, more mundane pieces of functionality (i.e., lifetime management, state and meta-data management and inspection, grouping, and notification).

The WS-Addressing specification gives a standard way of indicating or addressing stateful web service resources. At the heart of the WS-Addressing specification is the EndpointReferenceType (EPR) data-structure. WS-Naming supplements the WS-Addressing specification by providing for the ability to compare and rebind EPRs.

The OGSA-ByteIO specification defines a service interface for reading and writing sequences of bytes. It provides the means for treating data resources as POSIX-like files (e.g., reading, writing, truncating, seeking, etc.).

WS-Directory is a specification for lightweight directory resources that map string names to addressable entities that are identified by WS-Addressing EPRs. These directory resources can be composed hierarchically to describe and manage a namespace of grid/web resources (e.g., a grid file system namespace).

2.2 Legion. Legion is a distributed system in which entities (files, processors, storage devices, networks, users, etc.) are modeled as communicating objects. Every Legion object instance is defined and managed by its *class* object. Class objects are managers and policy makers and have system-level responsibility for creating new instances, instance lifetimes, and supplying bindings (network address data) for instances to client objects.

Legion objects are identified by a three-level naming scheme. Each object is assigned an *object address*: a mutable list of network addresses. Because they are not static, objects are also assigned (at creation) static, globally unique, location-independent names called Legion Object IDentifiers (LOIDs).

To facilitate "human-friendly" organization of Legion objects, Legion supports a hierarchical directory service, *context space*, which lets users assign arbitrary Unix-like string paths to objects. Context space is composed of *context objects*, each of which provides a mapping of strings to LOIDs (which may identify other context objects).

[1] Parts of this summary have been excerpted from Grimshaw, A.S., et al, *Integration of Legacy Grid Systems with Emerging Grid Standards*, UVA CS TR CS-2006-07, 2006

1-4244-0307-3/06/$20.00 ©2006 IEEE.

Legion provides objects called *Basicfiles* that model Unix files. Available operations include those to read/write blocks of data, append blocks of data, truncate files, or to obtain information about the file.

3. Implementation

To achieve interoperability, the proxy implementations explicitly solve two different problems: name mapping (translating from Legion LOIDs to WS-Addressing EndpointReferenceTypes and vice-versa) and interface-translation (translating the methods, parameters, and data structures involved in order to achieve functional congruence).

3.1. Name-translation. The name-translation solution is two-fold. The first issue is how to empower OGSA clients to refer to Legion objects. This is done by minting an EPR whose *address* field identifies the URI for the Legion inbound proxy and whose *abstract name* field is assigned the Legion LOID of the target Legion object.

The second issue is how to enable Legion clients to refer to OGSA resources. This is accomplished by architecting the outbound proxy to be the "class" of all external WSRF resources. Because Legion classes are responsible for object creation, management, and binding, the outbound (relative to Legion) proxy can maintain the EPRs of all external WSRF resources for which it has minted LOIDs. When a Legion client requests an object address for one of these LOIDs, the outbound proxy simply returns its own binding (since it will act as the communication proxy for all outgoing requests). Hence the outbound proxy implements the interface of a Legion class as well as the set of target object types for which it can proxy.

3.2. Interface-translation. The interface-translation problem must be addressed individually for each type of service interface that requires interoperability.

In the case of I/O functionality, the translation between Byte-IO and Legion BasicFiles was very straightforward. The interface abstractions were nearly identical in functionality, thus requiring only syntactic translation (converting method names, parameter ordering, and marshalling functionally-equivalent data structures).

Namespace interoperability, however, was complicated by object and resource references that might be nested within returned data. For example, inbound proxies must mint EPRs for any Legion LOID object references returned during Context lookup(). Similarly, outbound proxies must create LOIDs and maintain LOID–EPR mappings for any EPR resource references returned during WS-Directory lookup().

Another challenge was achieving interoperability for the "cross-cutting" interfaces: specifically the Legion *object-mandatory* methods (concerning attributes, interfaces, access-control, etc.) and WSRF (concerning resource-properties, lifetimes, etc.). Interface mismatch prevents a complete mapping of functionality, requiring the proxies to "fill in holes" by providing reasonable responses itself rather than proxying the request to the target.

4. Results and conclusions

We created a testbed consisting of a Legion grid and a lightweight WS-based filesystem grid (composed of of WS-Directory and OGSA-ByteIO resources using the WSRF.Net runtime). We were able to "mount" the respective namespaces into each other and to demonstrate both transparent namespace interoperability (i.e., clients performing "ls", "cd", "pwd" operations) and I/O interoperability (i.e., clients performing "cat" and overwrite operations). It was determined that proxy overhead was minimal in comparison to the latency imposed by the native OGSA and Legion implementations. In future work, we plan to better address scalability as well as investigate security-related interoperability. We conclude that, at least in the short run, OGSA service definitions can provide new interoperability value to legacy grid implementations via reasonable proxy construction.

5. References

[1] Foster, I., et al. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*. 2002.

[2] Snelling, D., *Unicore and the Open Grid Services Architecture*, in *Grid Computing: Making The Global Infrastructure a Reality*, F. Berman, A.J.G. Hey, and G. Fox, Editors. 2003, John Wiley. p. 701-712.

[3] Foster, I. and C. Kesselman, *Globus: A Metacomputing Infrastructure Toolkit.* International Journal of Supercomputing Applications, 1997. **11**(2): p. 115-128.

[4] Grimshaw, A.S., *The Legion Vision of a Worldwide Virtual Computer.* Communications of the ACM, 1997. **40**(1): p. 39-45.

[5] NAREGI, *NAREGI English Home Page.* 2006.

Box, D., et al., *Web Services Addressing (WS-Addressing).* 2004, W3C.

[6] Morgan, M. *ByteIO Specification 1.0*. 2005 [cited; Available from: https://forge.gridforum.org/projects/byteio-wg/document/draft-byteio-rec-doc-v1-1/en/4.

[7] Morgan, M. *WS-Directory Specification - Draft*. 2005 [cited; Available from: https://forge.gridforum.org/projects/ogsa-naming-wg/document/draft-wsdir-rec-doc-v1/en/1.

[8] GGF, *WS-Naming Specification*. 10 August 2005, Global Grid Forum, GFD-WS-Naming WG, http://forge.gridforum.org/projects/ws-naming-wg.