

Service Oriented Architectures for Grid Computing Environments: Opportunities and Challenges

S. Ramaswamy, *Sr. Member, IEEE*, M. Malarvannan

Abstract— Service Oriented Architectures (SOA) support the development of software applications as loosely coupled discrete web services capable of running on multiple software and hardware platforms. Grid computing has allowed for the virtualization of both hardware and software resources, whereby it is possible to dynamically share, select, and use geographically distributed resources based on users' needs. While Grid computing offers a way of distributing the system resources dynamically, SOA provides open standards based Web services for its users to discover and use available resources across the system. Hence it is a natural fit to leverage the benefits of Grid computing architectures using SOA. This paper examines the current developments in providing SOA based open standards for Grid services and some of their limitations.

Index Terms— SOA, Grid Computing, Web Services.

I. INTRODUCTION

SOA is an implementation of business process as a set of independent but cooperating web services. The main advantage of SOA is that the interface of the services is decoupled from its implementation. This allows the consumers of the services to rely on the well-defined interface of the service that are based on open standards regardless of the software and hardware used to implement the service. Grid Computing provides a model to solve massive computational problems by sharing unused resources such as applications, operating systems, hardware, databases, storage systems, etc. that are geographically separated and belongs to different organizations and individual users. Grid Computing gives a virtualization of computation resources to its users. A key concept to realize Grid Computing model is standardization so that the computer resources can be discovered, and used. Since SOA provides the open web standards it is a natural fit to use SOA to architect Grid Computing platforms.

The paper is organized as follows. Section II presents an overview of Web Services. Section III presents an overview of

Open Grid Service Architecture(OGSA). Section IV discusses some limitations of the OGSA architecture vis-à-vis the needs of Grid Computing. Section V presents some other factors to consider while developing a grid based SOA supported service architecture. Section VI concludes the paper.

II. SOA / WEB SERVICES OVERVIEW

A service-oriented architecture is essentially a collection of services that possesses the ability to communicate with each other. A service is a well-defined, self-contained function that is not dependant upon the context or state of other such services. The major advantage of SOA using Web Services comes from its promise of interoperability. Web services have come a long way in just a few years. While initial adoption of Web services by end users was slow, organizations are now realizing that the architecture can offer faster and more cost-effective solutions to their integration and interoperability problems. Many businesses are already benefiting from Web services through reduced development and maintenance costs and faster deployment of new applications and services. The Web services approach also eliminates the complexity and expense of traditional point-to-point application integration methods.

By implementing Web Services, any application can communicate with and use a service. Web Services allow companies to provide services that can interact with one another by exposing some of their capabilities and business processes to others on the Web without constant human intervention. Hence Web Services have allowed businesses that could not previously communicate due to applications that could not interoperate to seamlessly interface with one another in providing a better quality of service to their customers. Figure 1 presents a quick overview of a web service application [4].

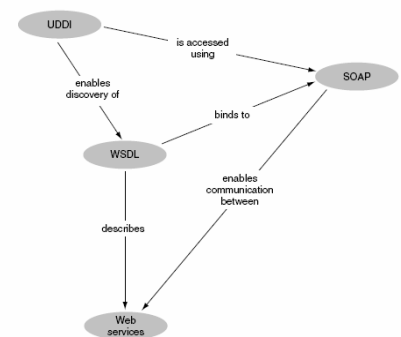


Figure 1. Web Services Overview

Manuscript received December 3rd, 2005, Revised Jan 26th 2006.

Dr. Ramaswamy is currently Professor and Chairperson of the Department of Computer Science at the University of Arkansas at Little Rock. Phone: 501-569-8134, Fax 501-569-8144, Email: srini@acm.org / srini@ieee.org.

Mani Malarvannan is CEO and cofounder of Cybelink Systems, a Minnesota based consulting company that provides technology and eBusiness solutions to its customers, URL: www.cybelink.com, Email: manim@cybelink.com.

Since all communication is in XML [5], Web Services are independent of operating system or programming language limitations. XML is for self-described and well-formed data encoding and lies at the core of Web Services; providing a common language for describing Web Services and Web service directories. The Web service protocol stack is a set of protocols used to define, discover, and implement Web Services. The Web service protocol stack is a set of 4 core protocols used to define, discover, and implement Web Services. These include: Service Transport, XML Messaging, Service Description and Service Discovery.

A. Security

Some XML security technologies are finding their way into Web Services environments. Two of these technologies are XML Encryption and XML Digital Signatures. XML Encryption builds on SSL to provide end-to-end data protection between multiple parties that supports selectively encrypting segments of a message. Different segments may be encrypted using different keys to allow only particular parties along a multi-party chain to access certain information, and certain segments may not be encrypted at all to reduce encryption/decryption processing time and energy consumption. XML Digital Signatures complements XML Encryption and provides a means to verify the authenticity and integrity of a message. XML Signature also provides a means for non-repudiation so that a sender of a message cannot disavow a sent message. WS-Security is emerging as the de-facto standard for security. In essence, it integrates and unifies multiple security models and technologies under a single umbrella, supporting interoperability.

B. Practical SOA Barriers

Consistent with observations by industry personnel, our experiences also indicate the following practical barriers.

1. **Interface immutability:** One of the major problems that arise in web services design is to make sure that the interface remains the same. Once in production, if one needs to change the method signature, either to address a different requirement or to fix a bug, it may break the existing users of that web service. So design fixes are very hard to accommodate. Though public vs. private interface design mechanisms allow immutable interfaces, it is practically challenging.
2. **Organizational Barriers:** There is still “resistance” within organizations to adopt SOA. True interoperability, like software reuse, still largely remains an illusion. Most organizational policies and inter-organizational relationships within their business-focused and IT-focused departments severe restrict the permeation of SOA to build tangible value within the organization. Other factors include personal choice issues and “blind” preferences of platform, application and vendor biases.

3. **Skill Set Barriers:** Finding the right expertise in enterprise architecting is a challenging task, requiring people with skill sets in software and information systems design, data and business process modeling.
4. **Standards Barriers:** Most SOA standards are either very limited or incomplete with respect to integration, management, and security and interoperability standards. Adoption of standard, open, general-purpose protocols and interfaces are still evolving.
5. **The HTTP Protocol:** Web services use HTTP protocol as the communication mechanism to send messages between the user and the web service. HTTP is stateless and unreliable with no success guarantees. This necessitates retry and graceful failure mechanisms. Some of the newer protocols supported by Web services (JMS, ESB, etc) allow automatic handling, but the web services built on HTTP do not. Moreover, due to the inherent nature of Internet and the HTTP protocol, the performance of the web service is not guaranteed, making them unsuitable for mission critical applications. Another issue is the parsing of complex XML. While technologies such as XML Pull Parsing (XPP) promise to better XML parsing, it is still a time-consuming process based on the complexity of your XML data transferred between the user and the web service.
6. **SOA Antipatterns:** In [7], the authors present several antipatterns that relate to design, technology, structure, and reuse issues in SOA projects.
7. **Service Modeling:** Often, developers jump into web services without understanding or modeling the business properly. Usually it is to develop a web service surrounding legacy code to expose it to the web. If the services are not designed without modeling the business processes then it may end up in a web service API wrapper around the legacy code. This approach still creates silo applications without helping leverage the benefits of SOA.
8. **Domain Objects & Components:** Developing SOA without properly modeling the fine-grained business domain objects and the coarse-grained domain components of a business processes allows the creation of applications that violate encapsulation and information hiding. While solving short-term business requirements in the end, it allows the permeation of non-reusable business services.

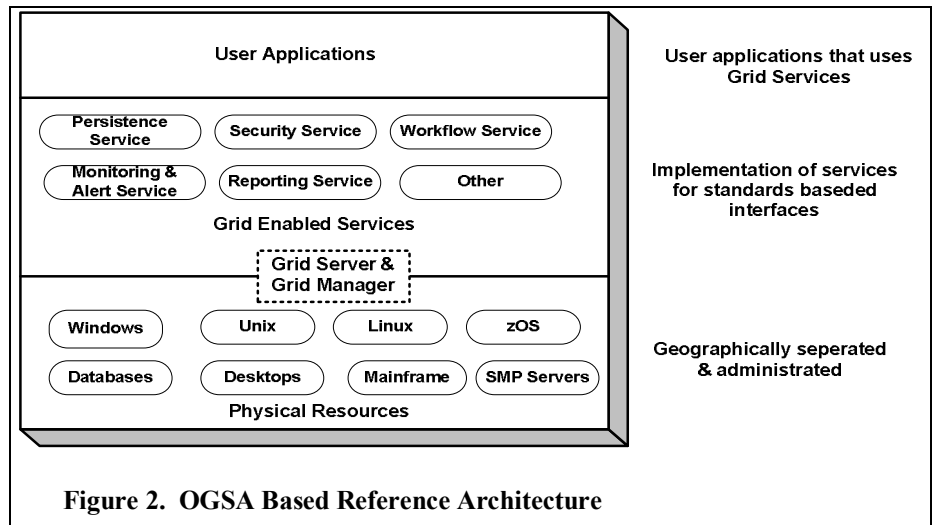


Figure 2. OGSA Based Reference Architecture

III. OGSA

OGSA [1] is a specification for distributed architecture that uses web services standards for implementing Grid Computing applications, including, distributed data systems and virtual business organizations. It is based on Grid Service interface defined by WSDL [2] that provides the necessary standards to share and use the computing resources. The standard also includes multiple bindings and implementations (Java, C, C++), which allows the services to be deployed to different software and hardware platforms. OGSA also provides security framework so that the services can be shared and used securely. Though OGSA is a comprehensive set of open standards specifications for grids it doesn't specify all the aspects of building a robust grid architecture. The OGSA architecture defines following items: (i) Grid Service interface standard: OGSA defines standards for creating grid service instances, naming, lifecycle, communication protocols, etc. It is the responsibility of the service implementers to adhere to the standards so that all the grid services can be shared and used in a standard way. (ii) Capabilities of services: Capabilities are functionalities provided by the grid services vendors. For example, a capability can be providing air ticket reservation grid service provided by a vendor that adheres to OGSA standards so that any user can find and use the service and the service vendor can charge the user of the service.

A. OGSA based Reference Architecture

Figure 2 shows the OGSA based architecture. The lowermost layers have all the physical resources that are part of the grid. As specified earlier these resources can be geographically separated and owned by either organizations or individual users. To overcome the stateless nature of SOA, Globus promotes WSRF (Web Services Resource Framework) is an attempt to define conventions for managing state so that applications discover, inspect, and interact with stateful resources in standard and interoperable ways [9,10]. The middle layer has all the services that implement the grid service interface specifications. The upper most layers are user applications. The Grid Server & Manager coordinates the physical resources with the grid services. Figure 2 shows sample of the physical resources and grid services. In an actual implementation, there will be more numbers of physical resources and grid services. The following is the typical scenario that explains how the grid architecture works; a user application uses a portal of grid service registries [3] to find appropriate grid service provider, extracts the WSDL of the service and initiates a request. The Grid Server & Manager gets the request and identifies appropriate service and passes the request for execution. Once a grid service receives a request it can either execute the request by itself or pass it to some other service for execution and gets the result. Either way a grid service with the help of Grid Server & Manager executes the request and sends WSDL response back to the user application. In this scenario the Grid Server & Manager plays a vital role in communicating the request and response between the user application and grid services. It also coordinates all the communications between the physical

resources and various grid services. For example, based on request load it can pass the request to existing grid service or create new grid services to handle the request

IV. OGSA LIMITATIONS

While OGSA provides a detailed specification to build open standards based grid service architectures, the specification doesn't provide certain key details, which are crucial, and necessary for robust service oriented grid architecture. In this section, we present some of these limitations.

A. Grid Service Transactions

For a successful implementation of grid services, a critical component is the availability of, and support for, distributed transaction capabilities as defined by the web service specifications [2]. Coordinating the successes and failures of transactions of grid services is an important missing piece. A semantically rich infrastructure to leverage common denominators between live grid applications using SOA is critical. A distributed transaction management capability would allow cross-linking data files between independent data grids. This would allow for easy adaptations enabling the design of customized solutions. This would allow scaling up grid computing applications to the volumes and diversity of the data available by leveraging pre-deployed web services.

B. Error Handling and Quality Of Service (QOS)

In [8], the author points to a three point check list for defining a grid: distributed coordinated resources, use of standard, open, general-purpose protocols and interfaces, and delivery of nontrivial qualities of service. OGSA does not specify how the error handling is accomplished among the grid services. When an user application selects a particular grid service from a grid service registry [3] it may select a particular vendor's grid service based on QOS and sends the request to that particular vendor's grid service. If for some error conditions the grid service could not satisfy the QOS as advertised in the grid service registries then the error conditions should be properly handled so that either the request is routed to some other grid service or the grid service notifies the user application. Though the OGSA addresses error handling to some extent, it doesn't specify how error conditions and degradation of QOS of grid services will be handled and notified to appropriate parties.

C. Support for existing web services

One of the vital specifications of OGSA is that all the services that conform to the OGSA must implement a Grid Service interface. This is necessary and the right step to make all the vendors conform to OGSA standards and to integrate the grid services that runs in heterogeneous environments. However, this makes it difficult to the existing web services to conform to grid service interface standards. OGSA needs to address the necessary semantic changes to be incorporated into existing web services to conform to OGSA standards.

D. Administration

OGSA specifies how the grid services are monitored and administrated but it is not clearly specified how the higher-

level grid services are monitored with the lower-level grid services (in a uniform manner). This might allow the imposition of strict requirements on the OGSA specifications. In this regard, the following questions are not adequately addressed by the OGSA specification: (i) Administration of grid services across geographical boundaries (ii) Administration of individual users' request and response, and (iii) Notification of progress of administered users' jobs.

V. OTHER FACTORS TO CONSIDER

A. Configuration and Coordination

In both Grid Computing and SOA applications, configuration (system elements, their location and interconnections, etc.) and coordination (interaction between the elements, the timing, protocols used, etc.) mechanisms are key for successful development. One of the biggest SOA challenges and cost factors has been the "delicate" nature of configuration and choreography for supporting distributed resource coordination. Within web services WS-BPEL and WS-CDL are emerging but in OGSA these specifications are not tied to grid service. For successful implementation of grid services these specifications must be integrated into OGSA. Web Service Orchestration relates to the execution of specific business processes that runs on different computers and organization boundaries. WS-BPEL is a language used to define processes that can be executed on a Web Service orchestration engine. Web Service Choreography relates to describing externally observable interactions between grid services. WS-CDL is an XML-based language that describes peer-to-peer collaborations of Grid Services participants by defining their common and complementary observable behavior; where ordered message exchanges result in accomplishing a common business goal. The WS-CDL specification is targeted for composing interoperable peer-to-peer collaborations between any type of Grid Service participants regardless of the supporting platform or programming model used.

B. Metering and Monitoring

As applications that leverage Grid computing and SOA become more prevalent, business models for commercial service providers to manage flow within and across enterprises will become critical. The leveraging of SOA in grid computing environments will allow the development of services with higher value, which in turn will drive the need for an efficient measurement model for services. This precipitates the need for better selection mechanisms and pricing structures for services by multiple service vendors.

C. Challenges brought forth by pervasive computing

SOA aspires to allow maximum reuse of services provided by associated software components, while ensuring loosely coupled interactions. Devices such as phones, PDAs, and pagers act as nodes in a pervasive computing environment. Such an environment (akin to an adhoc grid) combines intelligent computing and sensing devices with ubiquitous

network technologies to create an immersive environment made up of large numbers of mobile, locally scalable and possibly invisible, context-aware computing devices.

D. Performance Factors

Other performance factors to address in SOA for grid computing include issues such as higher quality of service, increased efficiency and productivity of users, improved resiliency / fault tolerance, reduced cost and complexity.

VI. CONCLUSIONS

In this paper we have presented SOA as a natural fit to architect grid applications. The OGSA reference architecture, and some shortfalls of OGSA, which is in its early stages of evolution are discussed. The development of antipatterns for grid computing environments is put forth. While OGSA is definitely a step in the right direction, we discussed some of the shortfalls. We also discussed several other factors that impact the widespread adoption of SOA – those that might pose similar concerns for grid computing applications.

REFERENCES

- [1] <http://www.globus.org/ogsa/>
- [2] <http://www-128.ibm.com/developerworks/library/specification/ws-tx/>
- [3] <http://www.ogsadai.org.uk/docs/R2/html/OGSA-DAI-USER-M4-UG-GDSR.html>
- [4] Web Services Architecture Requirements, <http://www.w3.org/TR/wsa-reqs/>.
- [5] Extensible Markup Language (XML), <http://www.w3.org/XML>.
- [6] W. Iverson, "Web Services in Action: Integrating with the eBay Marketplace", www.oreilly.com, June 2004
- [7] _____, "SOA Antipatterns", <http://www-128.ibm.com/developerworks/webservices/library/ws-antipatterns/>
- [8] Ian Foster, "What is a Grid? A Three Point Checklist", Grid Today, July 20, 02. <http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf>
- [9] <http://www.globus.org/wsrf/>
- [10] <http://www-128.ibm.com/developerworks/webservices/library/specification/ws-resource/>
- [11] S. Chatterjee, and J. Webber, "Developing Enterprise Web Services An Architect's Guide", Prentice-Hall.
- [12] D. F. Ferguson, T. Storey, B. Lovering, J. Shewchuk, "Secure, Reliable, Transacted Web Services: Architecture and Composition", <http://msdn.microsoft.com/library/en-us/dnwebsrv/html/wsOverView.asp>
- [13] S. J. Vaughan-Nichols, "Web Services: Beyond the Hype", IEEE Computer, February 2002.
- [14] K. P. Briman, "Like It or Not, Web Services Are Distributed Objects", Comm. of the ACM, Dec. 2004/Vol. 47, No 12.
- [15] Vogels, "Web Services are not distributed objects", IEEE Internet Comput. 7, 6 (Nov-Dec. 2003).
- [16] L. F. Cabera, C. Kurt, D. Box, "An Introduction to the Web Services Architecture and Its Specifications", Microsoft.