Grid Technology:
Going Beyond Web Service Functionality and Capability

April 8, 2004

ABSTRACT
In this paper, the benefits of grid technology are examined by comparing certain functionalities and capabilities found in Web Services with those found in Grid Services. Grid Services build on the conventions and practices used in Web Service development together with the harnessing of power and uniformity of structure based in grid technologies. The argument here is not if all that is possible using grid technologies and Grid Services is also possible using Web Services, rather it is a question of what these grid solutions can offer beyond the current scope of Web Services.

Differences between the two are clearly apparent in the abstraction inherent in the Grid Service naming scheme and the level of functionality guaranteed with Grid Services, but not Web Services. The distinction in usability is also quite notable in many real-world cases in which grid technologies are preferred to Web Services. The examples of the Condor Project of the University of Wisconsin demonstrates the advantage in scalability and customization available using grid technologies while the example of Axyz Animation, Inc. showcases the capability of grid technologies for the small-to-mid size business. It becomes quite clear that the advantages offered by grid technology in both uniformity and structure, as well as real-world applications, give a clear cut edge which is not likely to be taken away by Web Services any time soon.

1. INTRODUCTION
The Open Grid Services Infrastructure (OGSI) Version 1.0 defines a Grid Service as "a Web Service that conforms to a set of conventions (interfaces and behaviors) that define a client interacts with a Grid Service."[7] This definition gives the impression of Grid Services as merely a subset of or extension to traditional Web Service functionality. So the question is raised, can Grid Services, or grid technologies in general, offer anything above and beyond that which is capable with Web Services alone? If not, then why do they exist? It is the contention of this paper that a clear distinction between the two services does in fact exist. The difference, however, is not seen in extreme variances of capability, but rather in their respective real-world applications. Notable differences also exist in the conventions and standards found in Grid Services which provide a level of abstraction and uniformity for service creation and management practices not seen in traditional Web Services.

These differences are evident when discussing issues such as service load balancing, data encapsulation, scalability, and simplicity of design and implementation. The first distinction between traditional Web Services and Grid Services is evident in the markets in which each of these services is prevalent and the reasons for their success. Differences in the level of structure behind each service are also clear when comparing their respective naming schemes and service mappings. However, the true test is the application of each of these technologies in real-world scenarios. Grid Services do enjoy an advantage and are preferred for applications striving for high-throughput for a range of end-users, such as the Condor Project, and also in small-to-mid size businesses with intensive computing needs such as large amounts of frame rendering, like Axyz Animation, Inc. These differences leave no doubt as to whether or not grid technologies have a place in the current technological market. With elegant structure and abstraction in addition to guidelines for uniformity and high-level development, grid technologies do go far above and beyond that which is capable with Web Services alone.

2. MEETING MARKET DEMANDS
2.1 Grid Technology
Grid technology strives to provide location and platform independent computing power and resources without compromising security, privacy, resource availability, uniform user interface, performance, and accounting issues.[7]

The current market for grid technologies is primarily comprised of the following groups [2]:

- Users who have compute-intensive tasks and do not have the in-house resources
- Users who require access to applications on an ad-hoc basis which does not justify the investment in providing self-owned equipment
- Virtual organizations which have transitory needs to share information/data or resources
- Resource providers who have spare computing capacity (compute or storage or applications) which they wish to better utilize and possibly to delay further capital investment
- Resource providers whose business is to provide GRID resources
- Resource users who have sporadic compute/data intensive tasks who wish to utilize or avoid expenditure on rarely used expensive high performance equipment.

Simply stated, large numbers of grid technology users are single end-users, small businesses, or entities with limited resources or funds.

2.2 Web Services
Web Services, on the other hand, are being pushed in a large part by the likes of larger corporations and market leaders including IBM and Microsoft.[2] These industry leaders are looking to

increase security, reliability, and interoperability of Web Services. Web Services are a gold mine in the eyes of these technology giants because of their potential use in a wide range of applications, from mobile devices to large-scale Internet retailing. In order to achieve the interoperability between companies and infrastructures, strict conventions and standards must be implemented and followed.

In 2003, IBM, Microsoft, and BEA Systems developed "The Business Process Execution Language for Web Services" (BPEL4WS) which is intended to provide more automated Web Services.[2] These companies understand the need for such standards if Web Service technology is to ever spread. The ultimate interoperability they look to reach will enable IBM and the likes to provide customers with somewhat simple and understandable yet powerful and effective solutions to e-commerce and other application needs.

The challenge faced by IBM and other developers then is how to add more security, reliability, and transaction capabilities to Web Services without adding unnecessary complexity.

2.3 Is There a Difference Here?

While it may seem like the two technologies just described have few relevant differences to speak of, there are in fact key distinctions to be made. The reason IBM and other large corporations are pushing for Web Service standards to be established is because that would make higher level development and use of Web Services easier. Grid Services, as will be seen in the next section, are built on a solid foundation of naming and interface structure that allows for this easier higher level development. The reason for the push by end clients for grid technologies becomes evident when considering projects such as Condor-G, discussed later in this paper, which provides the end user with a level of customization and policing that is not available with Web Services alone. These subtle tweaks and nuances ultimately make grid technologies functional and capable beyond the scope of Web Services.

3. DIFFERENCES

3.1 Named Service Instances

Both Web Services and Grid Services seek to provide uniformity and consistency for application and system functions. Grid Services go further by following the specifications for named service instances and a two-level naming scheme. The two-levels of the naming scheme are formed by a Grid Service Handle (GSH) and a Grid Service Reference (GSR).[7] Each service instance is given an abstract Grid Service Handle. This abstraction makes specific information about the service instance such as its location, implementation, operational status, etc. inaccessible knowing only the Grid Service Handle. The Grid Service Reference contains the method and address of delivery specific to the grid service instance.

This level of abstraction makes it possible to have both stateless and stateful grid service instances. A stateless grid service instance makes each transformation or mapping of an input to output independent of any previous calls on that service instance. Also possible are stateful grid service instances in which output may be somehow tied to or dependent on the history of calls on the instance.[3]

Such functionality is also possible using traditional Web Service structure. The xml file of the service could be modified to make the session stateful with an optional timeout period.[3] The difference is in the flexibility grid services enjoy as a result of the abstraction and two-level naming scheme. Each Grid Service Handle is represented by a URI (Universal Resources Identifier) which allows for many different naming schemes to be used. This greater flexibility in naming allows further flexibility in how Grid Service Handles are linked to their respective Grid Service References. This allows for other approaches to be taken by Grid Services when dealing with method and service mapping.

3.2 Method Mapping and Load Balancing

Other than abstraction and elegance, what advantages does this two-level naming scheme give Grid Services over Web Services? Since Grid Service Handles are abstract mappings to grid service instances containing no specific information about the instance characteristics, it is possible for the mapping to change dynamically allowing Grid Service Handles to resolve to different Grid Service References over time.[7] This dynamic mapping capability makes it possible for grid service instances to be accessed from different locations, since it is just the Grid Service Handle reference which contains the mapping.

Another notable advantage of this two-level naming scheme is the ability to map on a one-to-many relationship where a single Grid Service Handle can map to many Grid Service References. This creates an opportunity to handle issues such as load balancing and anticipated congestion or failure. By distributing the load of instances among available services, per service performance also increases.[2]

Traditional Web Service technology would require more explicitly stated mappings to achieve even similar load balancing capability. A typical approach taken when using a cluster of machines is to have a separate load balancing system in place below the Web Service layer to handle fluctuations and distribution.[1] Other Web Services handle load balancing with a load balancing system which contains XML tag switching capability. This approach, however, will typically distribute load in a round-robin fashion, not taking into account other sources of congestion or anticipated failure as is the case with the Grid Service scheme.

This and similar solutions may at first seem to promise equal capability for Web Services. However, it is clear that the uniformity and consistency found in the two-level naming scheme of Grid Services greatly surpasses any ad-hoc system put together for Web Services in several areas. The use of Grid Services removes the need for any additional load balancing system while also ensuring data encapsulation and security for grid instances through the use of both Handles and References. By creating such uniformity among naming and binding practices, Grid Services also become better suited for use in a wider possible set of services.

3.3 Guaranteed Interfaces

Another critical difference between Web Services and Grid Services using grid technology is a sort of guarantee of functionality. Both Web Services and Grid Services must be deployed and consequently discovered to be of any use. The

method in which the two go about this process is another major area of distinction.

Web Services traditionally use Web Service Definition Language (WSDL) and Universal Description, Discovery, and Integration (UDDI) for discovering currently available interfaces. Grid Services, on the other hand, must expose certain mandatory services and data elements including functions to query and manipulate a specific Service Data Element (SDE). Each SDE contains metadata for the service and necessary state information.[2]

The OGSI, which provides industry standards for Grid Services, defines a set of mandatory services for creating, managing, and exchanging information among Grid Service entities.[7] Also contained in the specification is a set of guidelines for the infrastructure used when building Grid Services. Because of this detailed layout, assumptions can be made when designing higher level applications about what functionality is guaranteed to be in place when using any given service. This assumed uniformity and level of functionality removes a lot of the availability checking and service testing that is needed when working with Web Services. It also provides a solid foundation for Grid Services to build on knowing that a basic level of functionality will remain in any case.

The capability and importance of these Grid Service SDEs goes beyond service description and discovery. In addition to the basic mechanism for discovering and monitoring the Grid Service, the metadata and state information contained gives a wealth of information. The metadata may contain relevant policy information used in scheduling and the state information can carry usage statistics essential in achieving the load balancing discussed earlier.[2]

These dynamic SDEs tie in even further with the load balancing and dynamic mappings made available by the two-level naming scheme. Contrary to the Web Service process of static method deployment, dynamic SDEs can use the XML tag switching capability to respond to increased requests and load for particular methods or services. The potential of this dynamic discovery is immeasurable especially when creating systems in which the full range of necessary services is impossible to know prior to deployment. This room for future expansion from a grounded point of uniformity is a great advantage for Grid Services over Web Services which may ultimately lead to the more rapid spread and use of the grid technology based services.

4. REAL-WORLD EXAMPLES

4.1 The Condor Project

The Condor Project at the University of Wisconsin converts collections of distributively owned workstations and dedicated clusters into a distributed high-throughput facility.[9] The Condor-G package takes advantage of grid technologies to achieve this high-throughput and harness extra CPU power wherever it is available for any client looking for it. By installing the Condor-G software on a user's workstation, the user is able to tap into the resources of the Grid while also contributing to it.

Why not use Web Services to provide this functionality with a standard interface and seemingly simpler client-server interactions? The 'Personal Condor' running on each user's machine gives that user power and control beyond the scope of the traditional Web Service architecture. The user is able to implement specific policies about the jobs they submit for processing or about jobs able to use their processor power. For example, the user can specify that jobs may only run on their machine when they are away from it or that jobs by another certain user may never run on their machine.

The Condor-G, the Condor system using Grid computing resources, is able to obtain this functionality at the cost of slightly more complicated deployment procedures.[9] The condor_submit process allows the user to specify any policies or preferences they would like enforced. Such restriction and policing could be accomplished with the addition of large amounts of server code or password protection schemes using Web Service ideas. However, the massive scale of the Condor project makes either possibility completely unfeasible. So by slightly increasing the complexity per user, the Condor is able to offer a service of vast computing power to users that is not possible with typical Web Service structure.

4.2 Axyz Animation, Inc.

Axyz Animation, Inc. of Toronto, Canada does post-production work specializing in digital special effects for commercials, television shows, and films. The process of frame-by-frame image rendering is a compute-intensive task which many digital effects companies handle with specialized "render farm solutions." These "render farm solutions" typically involve a large cluster of dedicated workstations to be brought in and a customized user interface for users to harness the cluster power.[9] For a smaller company like Axyz Animation, however, such a custom solution is not a viable economic option. Vice-President and senior animator John Coldrick was looking for "a flexible solution that didn't require re-inventing the wheel as far as distributed processes went, yet was flexible enough for us to implement things the way we wanted."[9]

This is the position of many small-to-mid size companies in the animation industry. Coldrick found the flexibility he was looking for by switching to the Sun ONE Grid Engine Software. The customizable package allowed all currently available CPU power to be harnessed while it also gave animators control over when their workstation was "available" for such work. Animators are then able to achieve shorter rendering times while not sacrificing other imaging or editing tasks on their personal machine.

The last main advantage worth pointing out in this case was the time and cost that went into development. This is another crucial area where Grid Service packages offer huge advantages over the often complicated and costly Web Service implementations. For arguments sake, consider if Axyz Animation, Inc. could have achieved equivalent performance and benefits from Web Services as they did with their grid engine software. The cost of designing and implementing such a Web Service goes well beyond the bounds of a small graphics company merely looking to harness extra CPU power. However, by choosing Grid Services, "I [Coldrick] was able to set up a single group Cluster Grid in approximately two weeks of spare time after regular working hours, not very long at all."[9] Using simple script

wrappers, achieving the friendly and easy user interface of Web Services, Coldrick was also able to adapt to their specific requirements while keeping the user interface familiar for all the staff who worked with it without requiring the resources of the Web Service approach.

5. COUNTER ARGUMENTS

5.1 Web Service Clusters

Many argue that projects such as Condor would be easily implemented and maintained using Web Services if they had a large enough cluster available for processor power. However, the customizable access and policing possible at the end user level does not have to do with the cluster environment. Rather, the flexible user capabilities stem from the ability of Grid Services to map abstract instance Handles to References. This level of indirection along with multiple service mapping enables efficient use of the grid power behind the Condor project. Efficient handling of such a large scale of grid power is not as easy to implement using standard Web Services without this uniform naming and mapping. Using grid technology, Condor is able to manage a computing system with multiple owners, multiple users, and no centralized administrative structure. [2]

5.2 Syntax and Semantics

Many Web Service enthusiasts claim the naming scheme and guaranteed service interfaces of Grid Services are nothing more than "syntactic sugar."[3] The extensions provided by merging grid technology with core Web Service concepts go far beyond added elegance and abstraction. The naming conventions and guaranteed level of functionality cross service boundaries to provide a uniform and solid foundation upon which it is easily built. Higher level development using Grid Services is thus less of a complicated and intricate task. This ultimately means shorter time needed for development and deployment and more cost effective solutions.

As in the example of Axyz Animation, Inc., the abstraction provided by Grid Services allowed the Sun grid solution to be customized without significant costs. It also allowed higher level wrappers to be written to mask underlying structure to users of the new system, just as in typical Web Services, but with the added computing power to solve their rendering needs. This was so easily accomplished because all offered services maintained a core level of functionality. The ability to effectively customize a solution on a limited budget is a task faced by many small-to-mid sized companies with some range of compute-intensive tasks. Grid services with grid technology, such as the Sun ONE Grid package, are where they consistently turn for such needs.

6. CONCLUSION

As was stated at the onset of this paper, it is not an argument of whether or not all that is possible using grid technology and Grid Services is also possible using traditional Web Services. It is a question of whether or not grid technology does in fact have anything to offer beyond what can be considered reasonably feasibly using current Web Service technology. The argument made thus far examines Grid Services, taking advantage of Web Service features and underlying grid technology and power, in comparison with traditional Web Service structure and capability. The two-level naming scheme and guaranteed level of service functionality behind Grid Services provides a necessary level of structure and uniformity which will undoubtedly make the difference in upcoming years as the technology extends and advances. While ad-hoc solutions and software extensions may temporarily allow Web Services to boast similar performance and functionality, they will ultimately not be able to match the growth and flexible tasks possible using Grid Services.

Projects such as Condor-G have already shown the capabilities of grid technologies in a large-scale setting when looking for user customization and policing. Small-to-mid size companies such as Axyz Animation, Inc. have also realized the benefit of grid technologies in their vast amount of power yet relative simplicity. With cost being such a large issue for such smaller organizations, grid technologies offer a way to find the CPU power and flexibility they are looking for without the still high cost of Web Service development and deployment. With such a strongly laid foundation built on details, structure, and uniformity, there is no question that grid technology does go beyond the capability and functionality of Web Services alone, and will not be fading away or falling behind anytime in the near future.

7. REFERENCES

[1] Chaudhary, A., Saleem, M., and Bukhari, H. Web Services in Distributed Applications – Advantages and Problems. Ghulam Ishaq Khan Institute of Engineering Sciences and Technology.

[2] Grid Technology – A Guide. Retrieved April 2, 2004 from http://www.gridoutreach.org.uk/docs/pdfs/gridtech.pdf

[3] Grimshaw, Andrew, and Tuecke Steve. Grid services extend Web services: a solid foundation for service consumer reliability. Web Services Journal (August 2003).

[4] Ferguson, D., and Lovering, B. Secure, Reliable, Transacted Web Services: Architecture and Composition. Retrieved April 2, 2004 from http://www-306.ibm.com/software/solutions/webservices/pdf/SecureReliableTransactedWSAction.pdf

[5] Foster, I.., Kesselman, C., and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3), 2001.

[6] Foster, I.; Kesselman, C.; Nick, J.; and Tuecke, S. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration." Open Grid Services Infrastructure WG, Global Grid Forum, June 22, 2002.

[7] Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman C., Maquire, T., Sandholm, T., Snelling, D., Vanderbilt, P., Eds. "Open Grid Services Infrastructure v1.0", Global Grid Forum.

[8] Software Solutions – Grid Computing. Retrieved April 2, 2004 from http://wwws.sun.com/software/grid/

[9] The Condor Project Homepage. Retrieved April 2, 2004 from http://www.cs.wisc.edu/condor/