Web Services and the Grid

LEE LIMING

Web services are not new to the Grid community. Research teams and some product developers have been developing and using Web services in Grid applications for several years now. The emergence of Grid-motivated standards—WSRF and WS-I, for example—have recently sped up the adoption of Web services technologies. The availability of several WSRF implementations in the Globus Toolkit 4.0 and related software distributions has facilitated this adoption. As clusters become increasingly essential elements of the Grid's physical fabric, Web services are becoming essential elements of the Grid's application development toolset. Given the importance of clusters and Web services to Grids, cluster owners and operators need to understand the implications of Web services on the applications that run on their clusters.

Why Web Services?

Web services are aimed at making it easier to develop and deploy distributed, interoperable application services. Like DCE, CORBA, DCOM, Jini, and others before them, Web service technologies provide a framework for developing distributed applications that run on the Internet. Unlike its predecessors, the Web services model appears to be nearing the universal acceptance required for it to become a part of the essential fabric out of which the Internet and the Grid are built. The Web services movement is usually viewed as a natural extension of the Web movement that matured in the early 1990s. The World Wide Web, based on HTTP and HTML, provides a very simple but powerful interface for sharing information on the Internet. It is very easy to set up and maintain a Web server, and it is very easy to use a Web browser to access servers anywhere on the Internet. Now that the Web is essentially ubiquitous to the Internet, it has became a platform on which all sorts of information services—both free and commercial—can be built.

While fundamental to the Web, HTTP and HTML are very simple. They don't address many of the challenges of developing sophisticated online services, which means that different services use different (and incompatible) mechanisms to do the same things. Web services technologies—like SOAP and WSDL—provide more of the solutions to these problems, so applications based on Web services do more things the same way and are more interoperable. The W3C and OASIS standards organizations are largely focused on increasing the interoperability between services and applications on the Web in much the same way that the IETF has done for the Internet.

This sounds very familiar to members of the Grid community, who have been working for almost as long on promoting interoperability between services and applications on the Grid. The Global Grid Forum established a team a few years ago to formulate an Open Grid Services Architecture (OGSA), which uses interoperable Web services mechanisms to provide essential Grid interoperability features such as monitoring, lifetime management, and universal naming. The resulting collaboration between GGF, IETF, OASIS, and W3C has produced the WS-Resource Framework (WSRF) and WS-Notification (WSN) family of specifications, which form the basis for a new generation of Web services that have essential Grid interoperability features.

All of this adds up to good news for the developers of distributed applications in service of the e-Science and e-Business movements. WSRF and WSN, in combination with existing Web services standards like SOAP, WSDL, UDDI, and WS-Security, make it possible to develop applications that have a high degree of interoperability without requiring an enormous amount of planning and negotiation with partners that has to be revisited for every new application.

How to Host Web Services

The capabilities offered by Web services are very exciting for application developers. For cluster operators, the most pressing issue posed by Web services is probably, "What do I need to do to support Web service-based applications on my cluster?"

Most Web services require a hosting environment that consists of the computer on which it runs, a web server program, and a Web services "container." The container consists of extensions to the web server program. Elements of the container include support for SOAP messaging and servlets. An open-source hosting environment might consist of a Linux system, an Apache web server program, Apache Axis (for SOAP processing) and Apache Tomcat (a servlet container). A commercial hosting environment might consist of a Solaris system running Sun's Java Application Server Platform. The Globus Toolkit 4.0 includes a standalone Web services container that will host WSRF and WSN services. There are many other hosting environment combinations.

Containers are language-specific. Web services can be developed in a variety of languages, including Java, C++, Python, and Perl, and each language has a set of containers that can be used. Containers may also be embedded in standalone applications; in this scenario, running the application starts the container so that the Web service interfaces are available while the application runs.

Configuring a cluster to support Web services includes establishing a node on which the web server and associated Web services will run (typically a head node), and installing the software for the web server and the container. Choosing the software is currently a tricky business, as the standards for Web services are still being established and different applications may require containers with different properties. In most cases, Web service-based applications will either include their own containers (standalone or embedded containers) or come with a list of containers in which they can be run.

Note that unless the Web service has been designed to use the back end nodes on the cluster, it will only use the cluster's head node. Web services that require significant processing power should be written to submit tasks to back end nodes via the cluster's scheduler or other tools. The application developer must explicitly implement this capability. The GRAM service in the Globus Toolkit 4.0 is an example of a Web service that can distribute tasks to back end nodes via a variety of scheduler interfaces.

The steps for deploying a Web service differ for different containers. In most cases, one can add a new Web service to an existing container by installing the files that implement the service and then adjusting the configuration of the container software to tell it where the files are. In some cases, the container must be shut down and restarted to add or remove a service. In others, services can be added or removed dynamically. It is important to understand, however, that most containers require the system administrator's involvement in service deployment. (Any user can start up a standalone or embedded container, of course, but in this case the container will have to use a free port number.)

Most commercial Web service hosting environments include tools to monitor the status of the container and any Web services that have been deployed within it. Web services that support the WS-ResourceProperties specification (one of the WSRF specifications) can be monitored using their resource properties interfaces with tools like WebMDS, which provides a web browser interface to WSRF monitoring data.

Implications for Cluster Owners

The increasing popularity of Web services for distributed application development has two fundamental implications for cluster administrators. First, distributed applications are becoming more popular. Distributed applications have different use patterns and requirements from traditional high-performance computing (HPC) applications, so cluster administrators will see different demands on their clusters. Second, Web services use different mechanisms to deploy software applications and initiate tasks on clusters, so administrators will see new kinds of configuration requests from users and will need to account for new uses of their clusters.

With traditional HPC applications, an application run uses a single cluster in isolation to perform tasks on behalf of a single user. The application's installation can be tailored to fit the system and users are then told how to run the application on the cluster. Application runs can be performed on a scheduled basis at the convenience of the system without much consideration given to the user's schedule. Distributed applications break these conventions in several ways. A single application run may use several clusters in tandem, requiring each cluster to work with the others in a prescribed manner. The application must be installed on each cluster in a consistent way, minimizing system-specific idiosyncrasies. Application runs may involve multiple systems requiring each system to service application requests on demand rather than delaying them indefinitely until other processes are completed. Finally, distributed applications that follow a service-oriented design are often used by many users simultaneously, leading to service requests from different users arriving at the system in apparently random order, wreaking havoc with traditional approaches to per-user priorities.

Traditional HPC systems usually require users to log in to a system and then launch an application by submitting a request to the scheduler or queuing system. Remote job submission interfaces follow this model closely, though they compress the login and job launch steps into a single action. Some accounting and auditing mechanisms depend on this model, keeping track of logins, jobs executed, and data stored by each user. Distributed applications following the Web services model break this model as well. Users consume cluster resources by making service requests rather than by logging in and submitting jobs. These service requests can be authenticated and mapped to local users, but the processes that service the requests are launched as daemons or service handlers rather than as jobs. The data that results from multiple service requests may or may not be attributable to individual users. Users may make substantial use of a cluster's resources without ever actually logging into the cluster or seeing a shell prompt. (Users may not even be explicitly aware that they are using the cluster if their applications aren't designed to inform them of details at that level.) Accounting and auditing systems may need to be modified to track the resources used by authenticated service requests.

As noted above, deploying a Web service is quite different from deploying a traditional HPC application. If no existing container is suitable for hosting the application, a new container must be set up. The container must be configured to recognize the new service once it has been installed. New services will likely be accompanied by new users, and most of those new users will be using the Web services interfaces rather than explicitly logging in to the system. These users' Grid identities must be mapped to local system IDs for authorization and accounting purposes.

In some cases, users will deploy their own applications that have Web service interfaces. These applications may have embedded containers or use standalone containers that users can install and launch themselves on non-standard port numbers. Cluster administrators need to be aware that these services may be used by people other than the user who launched the job. It may become necessary to adjust or update usage policies to limit this kind of use of the system or to establish new standards for how authentication, authorization, and accounting are performed in these cases.

Conclusions

This month's column has raised a number of issues about how Web services will change the traditional use models for cluster resources. Some readers will undoubtedly infer from this that the increasing use of Web services is threat that must be eliminated. That would be a mistake with potentially very large consequences. In truth, based on the anecdotal experiences of early Grid application deployment projects like Grid3 (*www.ivdgl.org/grid3*) it seems likely that service-oriented applications—such as those based on Web services—will lead to significantly greater use of clusters (i.e., more business) than traditional, manually launched applications. Early efforts to gear clusters to become high-power hosting environments for these types of applications will position administrators well in a service-oriented era.

The challenges are neither trivial nor monumental. Commercial system vendors (Sun, HP, IBM, Oracle, Microsoft) are already providing the first generation of Web service hosting environments and management tools, and some of their customers and application developers are already exploring the possibilities of these new systems. Early adopters will find these tools a bit rough around the edges and lacking in some of the features familiar to HPC system administrators. Over time, however, the current tools may well become the basis for the next *status quo*, and those who participate now will help determine what that status quo ultimately looks like.

Globus Toolkit is a registered trademark held by the University of Chicago. This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38 and under Contract DE-AC03-76SF0098 with the University of California; by the National Science Foundation; by the NASA Information Power Grid program; and by IBM.

Lee Liming is manager of the Distributed Systems Laboratory (DSL) of Argonne National Laboratory, part of the Globus Alliance.