

SOA services in a grid and netcentric world

Skill Level: Intermediate

[Judith M. Myerson \(jmyerson@bellatlantic.net\)](mailto:jmyerson@bellatlantic.net)

Systems engineer and architect

06 Mar 2008

Get to know grid types, grid computing, and Global Information Grid (GIG). This article focuses on issues related to harnessing unused resources for computer power that's too intensive for a stand-alone machine. Explore examples of solutions, such as monitoring change in grid scale, grid coupling switch, and GIG and Service-Oriented Architecture (SOA) testing methodology.

Introduction

In the developerWorks series [Use SLAs in a Web services context](#), I discuss securing multiple Web services with a service level agreement (SLA) guarantee. Another series, [Work with Web services in enterprise-wide SOAs](#), covers consolidating your SOAs as a three-dimensional integration hug to improve speed and reliability, defense in depth for multiple SOAs, and speeding up Web services applications with the XML-binary Optimized Packaging (XOP). In the same series, I explored load-balancing Web services, cultural considerations of SOA adoption in the federal sector, and tight coupling Web services in the SOA.

Each of these shows a trend toward the optimization of resources to execute Web services in multiple SOAs. Transitioning SOA services to a grid and netcentric style is a way of harnessing and sharing unused resources from computers in the grid.

By moving Web services that connect the applications and systems to the grid, you can extend the resource capacity of multiple computers in collaboration with one another in parallel. This represents a paradigm shift from static use of a stand-alone machine's resources at one location to dynamic sharing of resources of multiple machines in parallel at any location.

In this article, you look at what grid types are, what grid computing is about, and

what the aims of GIG are. Find out what's missing from the concept and structure of grid computing, and get suggestions for solving problems.

Grid types (service)

Grid computing from a service perspective depends on what type of grid you want to use: dedicated, nondedicated, or distributed. Let's break these down and learn more about them.

Dedicated grids

A *dedicated grid* consists of dedicated hardware and computing resources used for your grid. The dedicated grid provides the most control and flexibility over your grid architecture, as you can select all aspects of the operation. It's the most flexible form of grid computing, giving you the freedom to choose the topology and networking hardware you want to use and that are most appropriate for the situation.

Nondedicated grids

Nondedicated grids use the resources and environment of an existing computing infrastructure. For example, a grid that uses the computing resources of your company when the desktop or server computers would normally be largely idle is a nondedicated grid. You have less control over the environment and network structure, because you can't change the core structure used to support the machines when they're not used in a grid. You're more likely to rely on the existing network and infrastructure and have little or no control over the networking decisions.

Distributed grids

A *distributed grid* is made up of resources of machines that can be located anywhere, internal or external, distributed over a WAN or the Internet. You have virtually no control over the network structure, but you *do* have the ability and control to ensure that the distributed components can communicate with each other effectively. The management concerns in this case are aimed more at providing access, security (including firewalls and authentication), and backup solutions for providing connectivity in the event of a failure.

Grid computing recap

IBM® was an early advocate and contributor to commercial grid computing to create a single-system image from the virtualization of distributed computing and data resources, such as processing, network bandwidth, and storage capacity. Simultaneously, the resources of many computers in a network are applied to a single problem that requires a lot of computer processing cycles or access to large

amounts of data.

Grid computing is a way to solve problems that require an enormous amount of computing power. You can think of it as distributed and large-scale cluster computing and as a form of network-distributed parallel processing. It can be confined to the network of computer workstations within a corporation across geographical boundaries, or it can be a public collaboration (for example, peer-to-peer computing).

The resources of thousands and thousands of computers can be cooperatively managed through collaboration toward a common objective. Because the load of resources can be balanced in the grid with unused resources, grid computing is like an extreme form of load balancing.

Grid computing requires the use of software that can divide and farm out pieces of a program as one large system image to several thousand computers. One concern is that if one piece of the software on a workstation fails, other pieces of the software on other workstations may fail. This is only if the single piece doesn't have a failover piece on another workstation and relies on other pieces of software to accomplish one or more grid computing task. Another concern is low latency that can result from inadequate utilization of unused resources in workstations.

Global Information Grid

Grid computing fits with the U.S. Department of Defense's (DoD) GIG vision given the heterogeneity of the DoD's systems. There are three ways of using grid computing as they pertain to GIG:

- **Computational grid:** A grid focusing on computationally intensive operations
- **Data grid:** A data computing system that deals with data—the controlled sharing and management of large amounts of distributed data
- **Equipment grid:** Where the surrounding grid is used to control the equipment remotely and to analyze the data produced

The U.S. Department of Defense defines the GIG Enterprise Services under the data grid type. This represents a paradigm shift from a system-netcentric network to a data-centric network.

Real-time decisions

GIG evolved in response to the need for an environment in which users can access, share, collect, process, store, disseminate, and manage information on demand from any location in the grid.

The GIG aims to achieve information superiority in a network-centric environment by enabling various systems and messaging-based Web services to interoperate with each other in parallel in solving problems too intensive for any stand-alone machines. GIG users can post and retrieve information and make real-time decisions rather than relying on historical information from multiple automated information systems applications.

Low latency

Intensive solutions require very high throughput with low latency, like that offered by IBM WebSphere® MQ Low Latency Messaging. These solutions address the explosion of data volumes across financial markets in high-velocity trading and analytic environments.

Designed for one-to-many multicast messaging, low-latency messaging software can deliver approximately 1 million 120-byte messages per second on Ethernet, close to 3 million 120-byte messages per second on InfiniBand, and more than 8 million smaller messages per second, all on common x86 servers. Testing has also measured very low latency of 30 microseconds for 120-byte messages delivered at 10,000 messages per second on InfiniBand or 61 microseconds on Ethernet.

What's missing from the grid

GIG lends itself to an SOA on a grid carrying information on demand. This means grid computing now relies on an open set of standards and protocols, including key SOA standards for Web services.

When these Web services are moved to the grid, these standards aren't adequate in resolving resource and performance problems at the grid level. We need something more encompassing than WS-Resource Transfer in different areas of the grid environment; in terms of what's required, we need to think about using it as a method for storing and recovering general information about grid-to-grid monitoring and management as well as security.

The problem is that Web services, normally loosely coupled, run whether the resource is scarce or not. We need to find ways to ensure that the resources on multiple workstations aren't wasted when they're in the grid. To find them, consider what's missing from the grid, then offer some solutions.

Monitoring change in grid scale

Volumes of resources can change from low to high and vice versa in the background in a nongrid environment. The resource is either scarce or isn't scarce while Web services are waiting to send or receive a message. If the change in scale isn't adequately controlled at thousands of workstations, it can have an impact on the one

system image in the grid, resulting in resource overloads.

One solution is to develop a grid monitor of how the unused resources of each workstation are harnessed and shared by other workstations. If the system finds that the unused resources on any workstation aren't properly harnessed, it should send an alert to the grid and system administrators so they can look up details in the logs for resolution.

Grid switch coupling

My article about [tight coupling Web services in an SOA](#) discusses considering a Web service with a coupling switch mechanism at the workstation level. This switch would flip to tight coupling from loose coupling when the Web service receives an alert that its corresponding resources have reached certain levels. When the Web service makes the switch, certain WS standards must be switched (for example, WS-Context for loose coupling to WS-Addressing for tight coupling).

At the grid level we can go further than that. There should be a Web service at the grid level to send an alert to specified workstations to switch from loose coupling to tight coupling of some Web services when the resources at the grid level reach certain levels. If this is reversed, there should be a Web service on a specified workstation that can send an alert to the grid when the resources of other Web services switched to tight coupling in the same machine have reached certain levels.

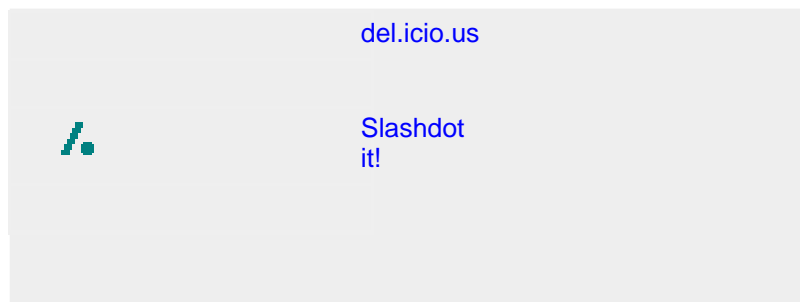
GIG and SOA testing methodology

To ensure the capabilities of GIG and SOA support the needs of the intended users, comprehensive testing is needed. The complexity of GIG and SOA enterprise services requires that methodology become more in depth and thorough. At the same time, to keep up with the rapid change and short development life cycles expected from the system builders, tests have to be ready to conduct in timescales, ranging from machine-specific functional to grid enterprise.

Conclusion

Share this...

	Digg this story
	Post to



You need a team of developers, testers, and system and grid administrators to enable SOA services to be grid and netcentric. To make the transition, you must plan ahead for the requirements and responsibilities of developing, migrating, testing, and deploying SOA services at the grid level. Resolving these issues makes your job of transitioning SOA services to the grid a lot easier. You can use IBM Rational® ClearQuest®, IBM Rational Tester for SOA Quality, IBM Rational Functional Tester, and WebSphere MQ Low Latency to increase productivity by reducing testing and defect tracking time at the grid level.

Resources

Learn

- See all of [Judith M. Myerson's](#) content, offering information on how to work with Web services in enterprise-wide SOAs.
- Check out Judith M. Myerson's developerWorks series [Use SLAs in a Web services context](#) for details about service-level agreements.
- Read "[Tight coupling Web services in the SOA](#)" (developerWorks, Jan 2008).
- Read more about the use of WS-Resource Transfer in the grid environment in the series [Building a grid system using WS-Resource Transfer](#).
- Get details about:
 - [IBM Rational ClearQuest](#)
 - [IBM Rational Functional Tester](#)
 - [IBM Rational Tester for SOA Quality](#)
 - [IBM WebSphere MQ Low Latency](#)
- Read Judith M. Myerson's [The Complete Book of Middleware](#), which focuses on the essential principles and priorities of system design and emphasizes the new requirements brought forward by the rise of e-commerce and distributed integrated systems.
- Get the business insight and the technical know-how to ensure successful systems integration by reading [Enterprise Systems Integration, Second Edition](#).
- Bring your organization into the future with [RFID in the Supply Chain](#), which explains business processes, operational and implementation problems, risks, vulnerabilities, and security and privacy.
- IBM Redbooks®: Read [Tivoli Manager for Domino V2.1 Fulfilling Service Level Agreements Using Tivoli Technology](#), for IBM Lotus® Domino® administrators, which goes into the nuts and bolts of developing a service-level agreement.
- The [SOA and Web services zone](#) on IBM developerWorks hosts hundreds of informative articles and introductory, intermediate, and advanced tutorials on how to develop Web services applications.
- Play in the [IBM SOA Sandbox!](#) Increase your SOA skills through practical, hands-on experience with the IBM SOA entry points.
- The [IBM SOA Web site](#) offers an overview of SOA and how IBM can help you get there.

- Stay current with [developerWorks technical events and webcasts](#).
- Browse for books on these and other technical topics at the [Safari bookstore](#).
- Check out a quick [Web services on demand demo](#).

Get products and technologies

- Download a [trial version of Rational ClearQuest](#).
- Download a [trial version of Rational Functional Tester](#).
- Download a [trial version of Rational Tester for SOA Quality](#).
- Innovate your next development project with [IBM trial software](#), available for download or on DVD.

Discuss

- [Participate in the discussion forum for this content](#).
- Collaborate with others who are interested in SOA in the federal sector in the [Federal SOA Institute - SOA Certification Mentoring](#) discussion forum.
- Get involved in the developerWorks community by participating in [developerWorks blogs](#), including the following SOA and Web services-related blogs:
 - [Service Oriented Architecture -- Off the Record](#) with Sandy Carter
 - [Best Practices in Service-Oriented Architecture](#) with Ali Arsanjani
 - [WebSphere SOA and J2EE in Practice](#) with Bobby Woolf
 - [Building SOA applications with patterns](#) with Dr. Eoin Lane
 - [Client Insights, Concerns and Perspectives on SOA](#) with Kerrie Holley
 - [Service-Oriented Architecture and Business-Level Tooling](#) with Simon Johnston
 - [SOA, ESB and Beyond](#) with Sanjay Bose

About the author

Judith M. Myerson

Judith M. Myerson is a systems architect and engineer. Her areas of interest include middleware technologies, enterprise-wide systems, database technologies, application development, network management, security, RFID technologies, and project management.

Trademarks

ClearQuest, Domino, IBM, the IBM logo, Lotus, Rational, Redbooks, and WebSphere are registered trademarks of IBM in the United States, other countries or both.