

The Application Grid

Doing more with less infrastructure

by Adam Messinger and Mike Piech

Application servers, those dependable workhorses that run most enterprise Java applications, are rarely a hot topic of conversation these days. As a technology category, the application server appears to be fairly “established” and that the focus has moved elsewhere in the stack, but appearances can be deceiving.

In fact, much remains to be done at the application server layer. One area ripe for innovation is the ability for application server instances to work together to enable more rapid deployment of new applications and hardware while at the same time improving the utilization of underlying physical resources. In contrast to the traditional one-app/one-app-server/one-OS/one-machine architecture, a new approach has emerged with multiple application servers pooling and sharing lower-cost compute resources, while dynamically reallocating these resources across applications as needs evolve.



Adam Messinger is vice-president of development in the Fusion Middleware Group at Oracle. He is responsible for managing the Oracle Coherence, Oracle JRockit, Oracle WebLogic Operations Control, and other web-tier products. Prior to joining Oracle, Adam worked as a venture capitalist at Smartforest Ventures and O'Reilly AlphaTech Ventures. He is a graduate of the Stanford Graduate School of Business where he was a Sloan Fellow and of Willamette University where he was a G. Herbert Smith Scholar.

Grid computing refers to the aggregation of multiple, distributed computing resources, making them function as a single computing resource with respect to a particular computational task. Grid is a form of virtualization in the sense that it hides the details of resources and makes them appear like something different. *Application grid* applies the same concept to application servers and describes an architecture in which multiple application server instances work together to



provide a shared, dynamically allocatable pool of resources to a set of applications.

Why an Application Grid?

Before delving into what it takes to make this concept work, let's look at the motivation for seeking an alternative approach in the first place. What is the primary infrastructure challenge as it stands today? An issue widely discussed in the pages of *Java Developer's Journal* is that of stove-piped architecture whereby applications are structured as monolithic entities that are difficult to integrate and reuse. Industry adoption of SOA has gone a long way to breaking down stovepipes at the application level. SOA achieves this by decomposing applications into finer-grained services that can be connected and reused in a more

flexible way. Stove-piped resources typically remain underneath each of these SOA services — machines that are statically allocated to the entities they run. As each stovepipe (stack) is statically configured, bringing new stacks online takes a lot of effort and a big investment in hardware that will likely be underutilized.

With an application grid, the allocation of machines to applications is dynamic since it becomes easier to bring both new machines and new applications into service. With a stovepipe under an application, increasing capacity typically means adding another app server/OS/machine stack and then putting a mechanism in place to load-balance. This causes inefficiency because you don't get linear scaling — doubling the number of servers doesn't get you double the number of transac-

“ It's time for a new approach to application resourcing”

tions per second or concurrent users — because other bottlenecks come into play. By contrast new application grid-enabled application servers support clustering that scales to much higher levels.

An application grid also helps improve hardware efficiency because excess capacity can be redirected to applications that need it most. By sharing and pooling resources, an application grid allows the total compute resources required to be less than the sum of all the applications' peak demands. Since few applications hit their peak loads at the same time in most environments, shared resources can be moved from lower-demand applications to those with higher demand. Continuous, automated, dynamic adjustment of resources is one of the primary capabilities of the application grid architecture.

Finally, an application grid enables a higher quality of service. Faster response times and higher reliability, which come from the application grid's ability to parallelize computation, replicate data across distributed nodes, and reduce interruptions from network problems or Java garbage collection, allow more computation per unit of time, and improve resiliency by eliminating single points of failure and automating failover. An application grid also provides tools to manage a collection of machines in an aggregated way, enabling faster administrative response and reducing human error.

Creating an Application Grid

Sounds great, but can this be achieved with current technologies? There is certainly more work for vendors in future product releases, but much can be done today. There are four fundamental capabilities

that must be in place at the application server level: clustering, adjusting, metering, and automating.

Clustering is supported by most application servers, though with varying levels of reliability and administration. It is most often used for availability/failover: instances in a cluster divide work and replicate data, such as Web user sessions; each instance is responsible to another member of the cluster that serves as a backup. A backup server automatically takes over responsibilities in the event of the primary's failure. Clustering also allows horizontal scale-out since work is distributed (load-balanced) across the cluster.

Adjustment capability coupled with scale-out clustering is a key element of application grids. It's one thing to statically set up a set of application server instances (nodes) as a cluster and put load balancing in front of it. But when nodes can be added to or removed from the cluster while the application is running, we have the basis for dynamic scaling.

Metering, or instrumentation, complements adjustment. We need to adjust clusters for visibility into what's happening inside them. Are any computing resources near critical thresholds? Are application service levels in danger? In short, the application server, the Java Virtual Machine, and other resources must provide the right kind of information about things like memory use and latency.

Once we have dynamically adjustable clustering with good instrumentation, the linchpin of the application grid is automation. This meta-level controller plugs into the adjustment controls and metering instruments of the clusters creating an automated feedback loop of

observations and adjustments. The mechanism adds nodes to clusters in need of capacity and removes nodes from clusters with reduced need. Since each cluster is ignorant of the surrounding clusters competing for resources, the application grid controller makes allocation decisions that are optimal for the grid overall, taking into account demands, resources, and policies.

Getting Started

Many enterprises have already started down the path to an application grid by using the clustering mechanisms in contemporary application servers for horizontal scale-out and by using scripting to partially automate the addition and removal of nodes.

State-of-the-art distributed caching technologies complement these early steps by creating an even more dynamic in-memory data grid with extreme scalability. Real-time JVM technology provides the predictability and additional instrumentation for applications with microsecond latency demands. And finally, as understanding and practices around application grid mature, management technologies with increasingly sophisticated mechanisms for cross-grid optimization will continue to evolve.

The combination of accelerating business change and the agility enabled by SOA imposes increasingly volatile demands on infrastructure. At the same time, the economic climate is driving the need for greater resource efficiency. It's time for a new approach to application resourcing: application grid.

For more information on Oracle's products and services supporting application grid deployments, please visit Oracle Application Grid at oracle.com.



Mike Piech is senior director at Oracle responsible for Oracle Fusion Middleware products: Oracle WebLogic, Oracle Coherence, Oracle JRockit, and Oracle Tuxedo. Mike joined Oracle as part of the BEA acquisition, prior to which he spent seven years running product marketing at Dorado Corporation, which builds a WebLogic-based cloud solution for mortgage banking.

“ With an application grid, the allocation of machines to applications is dynamic since it becomes easier to bring both new machines and new applications into service”