Prepare Your Web Site for e-business on demand™

A look at how early customers are implementing WebSphere® to create an on demand operating environment

Authors: High-Volume Web Site (HVWS) Team
Web address: ibm.com/websphere/developer/zones/hvws
Management contact: Willy Chiu wchiu@us.ibm.com
Technical contact: Noshir Wadia noshir@us.ibm.com

Date: April 29, 2003

Status: Final 1.0

Abstract: *This paper introduces how some of IBM's largest customers are starting their transition to on demand computing. It also introduces a new, early stage, limited IBM offering, IBM Server Allocation for WebSphere Application Server, developed by the HVWS group. Finally, the paper identifies what you can do now to improve resource use and ready your e-business to benefit from on demand computing.*

Contents

Executive summary

In the last few years, IBM's High-Volume Web Site (HVWS) team has engaged with many large customers to help them understand their workloads, and to design and implement highly scalable e-business infrastructures. The resulting systems operate now to meet the unpredictable workloads common to an e-business. When not meeting peak demands, however, these resulting infrastructures are often left with excess capacity. Excess capacity and the increasing pressure to improve the return on investment are combining to drive businesses to seek new opportunities to use existing resources.

Several leading-edge customers are already working with IBM to optimize their resource use and to consider how emerging on demand technologies and standards can be part of the solution. While the technologies and standards are still under development, there are actions to take now so that when the technologies and standards are available, you'll be ready to transform to on demand computing and enjoy its advantages of managing workloads dynamically, adding applications without adding computing resource, and reducing human intervention.

Among the key technologies for enabling e-business on demand are grid computing and autonomic computing. A *grid* is a collection of distributed computing resources available over a network that appears to an end user or application as one large virtual computing system. A grid can span locations, organizations, machine architectures, and software boundaries to provide virtually unlimited power, collaboration, and information access to everyone connected to the grid. One effect of grid computing can be to make network computing more like a utility. You deliver computing power to where you need it only when you need it; you pay for what you use, when you use it.

In addition to grid computing, autonomic computing is moving from concept to reality using Web services and Java™ 2 Enterprise Edition (J2EE) programming constructs. Autonomic computing is an approach to self-managed computing systems with minimal human intervention. Using a combination of grid and autonomic computing, infrastructures will be able to monitor current workload, analyze it versus historical trends and available resources, plan the reallocation of those resources, and automatically execute the movement of workload or resources to maximize computing responsiveness to meet service level objectives; in autonomic terms, this is called the autonomic control loop. This will allow server farms to act as a single *virtual* computing resource.

This paper introduces the early work of some of our customers. It also introduces a new, early stage, limited IBM offering, IBM Server Allocation for WebSphere Application Server, developed by IBM Research and the HVWS group. Finally, this paper identifies what you can do now to improve resource use and ready your e-business to benefit from on demand computing. Appendix A provides a quick introduction to grid computing and autonomic computing and identifies where you can learn more.

Future updates to this paper will expand on current customer projects and describe the lessons learned and best practices emerging from our work.

What are leading-edge customers doing?

IBM’s largest customers are finding ways to use their excess capacity as a part of their transition to e-business on demand. With help from IBM, they determine which application is the best candidate for their early work. Here are examples of three customers who have begun their transition.

- A leading financial company that offers online trading wants to be able to ‘borrow’ resources from the high-priority trading application while maintaining its service level agreements (SLA) for trading. The company plans to provide online self-service advice using the excess, available resource. The advice application will run as a lower priority application and be as responsive as it can be with available resources, which will vary based on the needs of the trading application. This application of grid and autonomic computing enables the company to offer their customers more with no additional investment, thereby improving the company's return on investment and improving its ability to compete.

- A large insurance company has the objective of making all servers available to all applications, essentially making their server farm a single virtual computing resource. By doing so, the excess capacity will serve as a ‘shock absorber’ during peaks in workload. Server grids facilitate workload management by eliminating the need to plan for and manage standby capacity.

- An automobile manufacturer has the vision of allowing a potential customer to design a car online. The new technologies enable such visions without investing in additional capacity.

The new, early stage, limited IBM offering called IBM Server Allocation for WebSphere Application Server will address these needs using open Web services standards, such as Web Services Description Language (WSDL) and Simple Object Access Protocol (SOAP). The next section introduces the technology.

IBM Server Allocation for WebSphere

The HVWS team and IBM Research are developing on demand technology using constructs of the autonomic control loop and open Web services standards. The initial offering, IBM Server Allocation for WebSphere Application Server, defines and delivers a software architecture using open standards to help customers address their business challenges. It will be available to a limited set of targeted customers for production deployment in 2Q2003. IBM intends to enhance it as the emerging Open Grid Standard Architecture (OGSA) protocols mature, and make it an integrated component of a future version of the WebSphere Application Server.

IBM Server Allocation allows the customer to run multiple mixed workloads in the same grid infrastructure. As the traffic of a higher priority workload increases to such a level that the SLA objective cannot be met, the system drains the lower priority workloads to other servers in the local or remote grid, to make additional servers available for high priority transactions. As the traffic of a higher priority workload decreases, the system schedules the lower priority workloads on to the servers where they were previously running. The primary components of the offering, as shown in Figure 1, are defined to support the goals of the autonomic control loop.

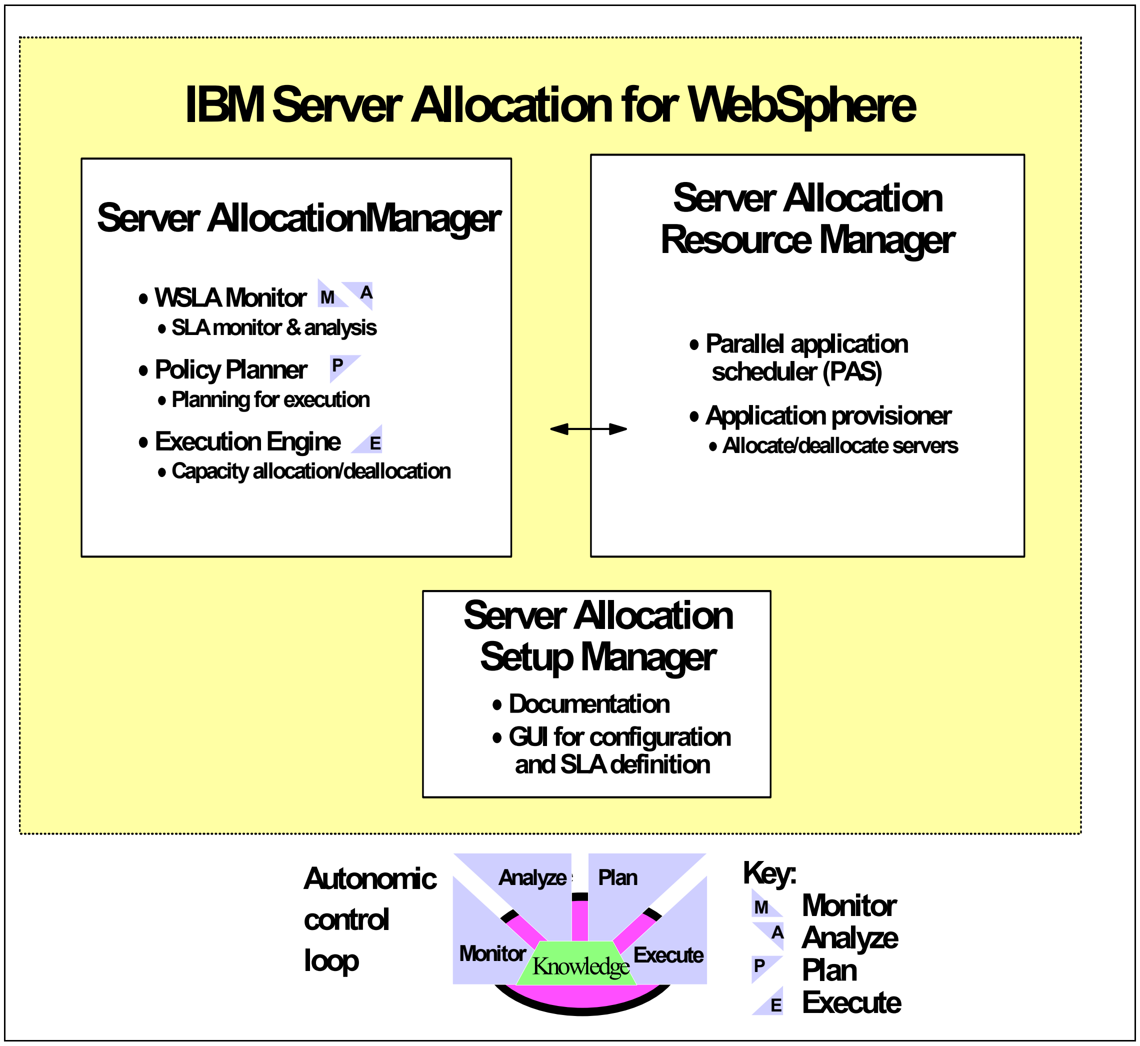

IBM Server Allocation for WebSphere
Server AllocationManager
• WSLA Monitor M A
• SLA monitor & analysis
• Policy Planner P
• Planning for execution
• Execution Engine E
• Capacity allocation/deallocation
Server Allocation Resource Manager
• Parallel application scheduler (PAS)
• Application provisioner
• Allocate/deallocate servers
Server Allocation Setup Manager
• Documentation
• GUI for configuration and SLA definition
Autonomic control loop
Analyze
Plan
Monitor
Knowledge
Execute
Key:
M Monitor
A Analyze
P Plan
E Execute

Figure 1. Components of the IBM Server Allocation for WebSphere Application Server

IBM Server Allocation has three major components:

- The *server allocation manager* component is the most sophisticated of the three. It provides the autonomic control loop functions. It contains subcomponents that monitor workloads, analyze data to determine if the workload will continue to meet its service level agreement, and if needed, plans whether resources need to be added or given back, and invokes the server allocation resource manager to execute the add/give back as needed.

- The *server allocation resource manager* adds and removes resources as needed to meet service level objectives.

- The *server allocation setup manager* facilitates setting service level agreements, and controlling the installation and operation of the Server Allocation components.

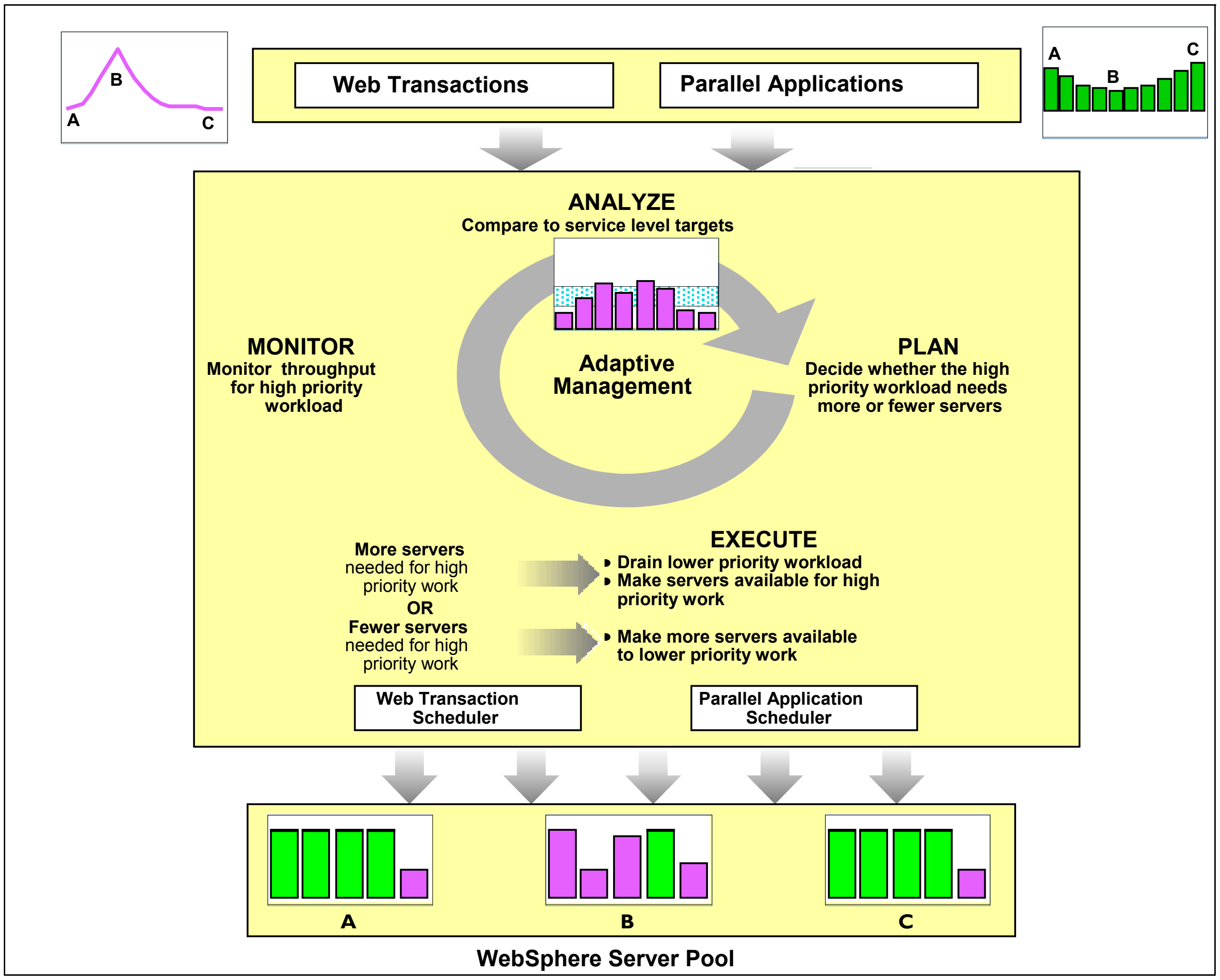
Server Allocation will include WebSphere Studio Application Developer plug-ins to help in the development of new applications or allow you to modify existing programs that are written in C, Fortran, or other programming languages. Customers can also easily develop Java applications with a new API, so that the application can take advantage of the servers in the WebSphere environment.

A demonstration of server allocation

This section summarizes the functions of server allocation by describing a demonstration available from IBM. The demonstration presents two application workloads. The transactional workload is WebSphere's standard example application Trade3. Trade3 runs on WebSphere Application Server, and simulates the common operations of an online brokerage. This class of transactional workload represents *business critical* applications. The compute intensive workload is represented by the application FAST (Financial Analytics Simulation & Test). FAST is a simulation of certain kinds of financial analysis that commercial brokerage houses offer to their customers, such as retirement portfolio analysis. A common characteristic of such applications is that the computation can be broken into units that can be processed in parallel on multiple machines. In other words, the more CPUs that are made available, the faster the application will run. This is managed by the parallel application scheduler (PAS) component of Server Allocation. PAS decomposes FAST and other applications into tasks that can be executed on the available CPUs in parallel, distributes these tasks across the CPUs, and gathers and combines the results. This class of applications falls in the *value-add* category, contrasted with those in the business critical category.

Further, the demo assumes a business rule that processing transactional workloads has higher priority than processing compute intensive workloads. This business rule implies that whenever there is a surge in traffic to the business critical application, resources are taken away from the value-add application and assigned to the business critical application.

Figure 2 presents an overview of the demonstration.



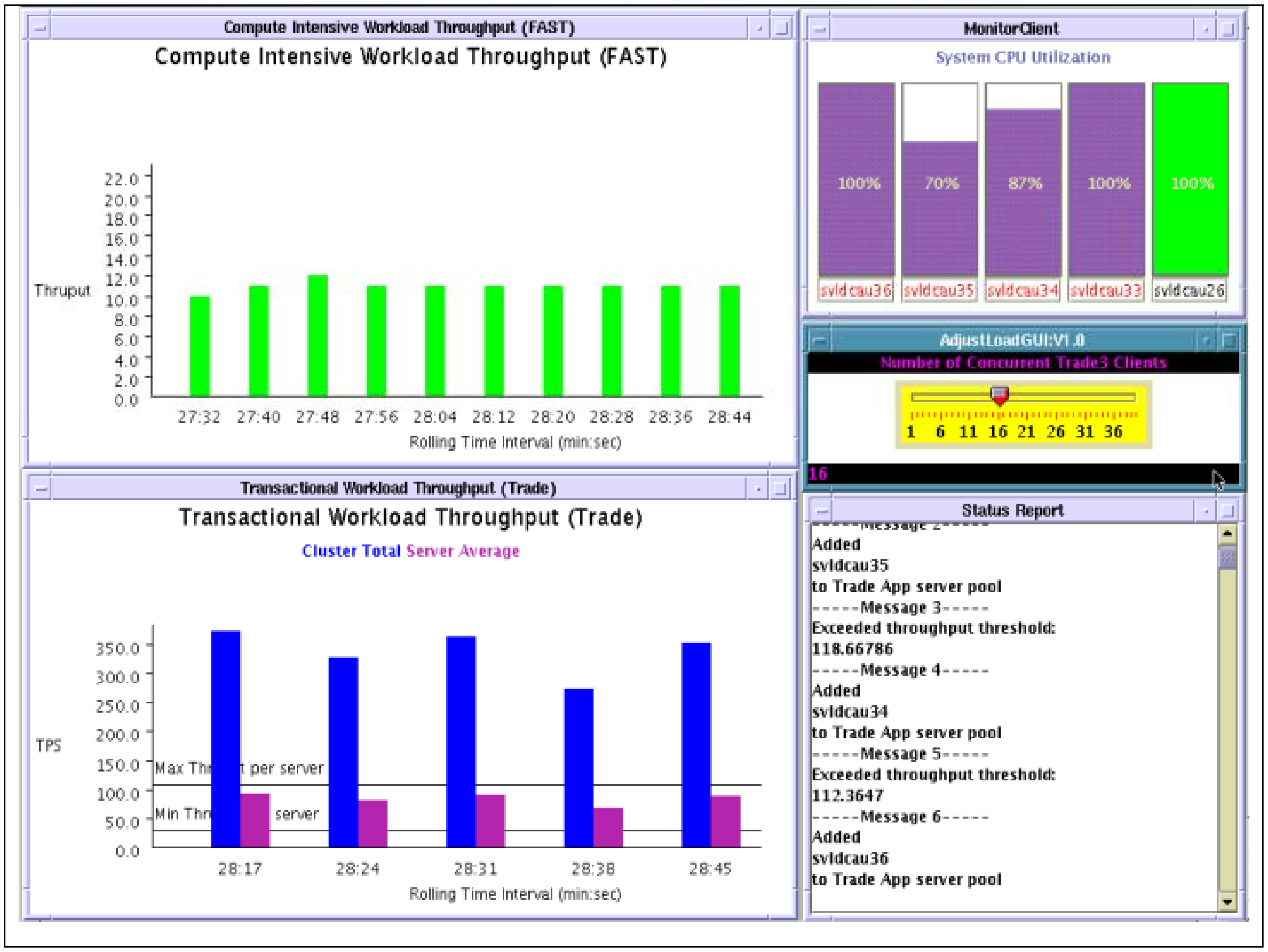
A
B
C
Web Transactions
Parallel Applications
A
B
C
ANALYZE
Compare to service level targets
MONITOR
Monitor throughput for high priority workload
Adaptive Management
PLAN
Decide whether the high priority workload needs more or fewer servers
EXECUTE
More servers needed for high priority work
OR
Fewer servers needed for high priority work
• Drain lower priority workload
• Make servers available for high priority work
• Make more servers available to lower priority work
Web Transaction Scheduler
Parallel Application Scheduler
A
B
C
WebSphere Server Pool

Figure 2. Overview of the server allocation demo

Figure 3 is a snapshot of the 'dashboard' used to demonstrate how server allocation proceeds through the process of monitoring server use, determining through analysis that the servers allocated to Trade3 can no longer meet the SLA requirement, reducing the number of servers assigned to FAST, reallocating those servers to the WebSphere Network Dispatcher for use by Trade3 until the load returns to SLA limits, and returning the reallocated servers to the FAST application.

The top left window represents the throughput of the FAST workload. The bottom left window represents the throughput of the Trade3 workload. In both these windows, the X-axis represents the progression in time. The most recent value is displayed towards the right. The bars scrolls left as time elapses. The Y-axis represents the throughput for the corresponding workload.

The top right window labeled Monitor Client represents the CPU utilization on the five servers that are used for processing the two workloads. These five servers consist of the two dedicated servers and three shared servers. The three shared servers are towards the left. The bar in purple represents the dedicated Trade3 server. The dedicated FAST server is the right most.



```
Added
svldcau35
to Trade App server pool
-----Message 3-----
Exceeded throughput threshold:
118.66786
-----Message 4-----
Added
svldcau34
to Trade App server pool
-----Message 5-----
Exceeded throughput threshold:
112.3647
-----Message 6-----
Added
svldcau36
to Trade App server pool
```

Figure 3. Demonstration dashboard for server allocation

Immediately below the Monitor Client window is the slider bar that is used to control the amount of Trade3 traffic. During the course of the demo, the traffic load increases then decreases.

Finally, at the bottom right is the Status Report window that represents the operator console. Throughout the demo this console displays messages about the server reallocations.

The color green represents FAST throughput in the top left window, and also represents the servers allocated to processing the FAST workload in the top right window. Similarly, the purple represents the Trade3 application in the throughput window, in the bottom left, and the servers allocated to processing the Trade3 workload in the top right window.

Also in the Trade3 window, the blue bar represents the total Trade3 application throughput across all available servers. The purple bar represents the average throughput per server. The top horizontal line represents the upper threshold for the SLA. The bottom horizontal line represents the lower threshold. The SLA in this demo is defined on the average throughput value, represented by the purple bar. The objective of the autonomic control loop is to have Trade3 always meet its SLA. In other words, the purple bar should stay below the upper threshold line.

During the demo, the slider bar is used to control the Trade3 load. When increased, the average value exceeds the upper threshold, indicating the SLA has been missed, and an additional server is allocated to Trade3. This reallocation is displayed in the top right window, as the green bar turns to purple. This reallocation continues until a sufficient number of servers are made available to process the Trade3 workload in order to meet the SLA.

Conversely, when the Trade3 workload is decreased, using the slider bar, the average throughput value falls below the lower threshold, indicating that additional servers are no longer required to meet the SLA. In the demo, you see the servers being returned to the FAST application, as indicated by the bars in the CPU utilization window turning from purple back to green. The premise is that no server sits idle.

Reallocating servers dynamically maintains the SLA for the business critical Trade3 application, while throughput of the FAST workload decreases in response to fewer computing resources.

Start now to be ready

It's not too early to begin your transition/transformation to an e-business on demand. You need a strategy to get you from where you are now to where you need to be to reap the benefits of an on demand e-business. Start with learning about the advanced technologies and, as you do, begin to create your strategy.

Review the quick introductions to grid and autonomic computing in this paper. Pursue the additional references to learn more. Then think about applying the elements of the autonomic control loop to your strategic planning.

Monitor: The first step is to learn as much as you can about your current workloads. Implement workload monitors so you can better understand workload characteristics and patterns. Application measurements you want to monitor include CPU utilization, response time patterns, and workload peaks and valleys. Tivoli® offers the complete IBM solution for managing the extended WebSphere environment. For precise viewing of performance metrics, users can start with the Tivoli Performance Viewer, a complimentary tool shipped with WebSphere Application Server. For ongoing production monitoring, Tivoli offers the following tools: IBM Tivoli Monitoring for Web Infrastructure (ITMf WI) provides best-practice monitoring of the key elements of WebSphere Application Server, and IBM Tivoli Monitoring for Transaction Performance (ITMfTP) provides a unique perspective of monitoring, that of the end-user.

Analyze: The second step is to analyze your workload measurements. Examine when your peaks occur. Consider what new application could be implemented to use resources available during nonpeaks. For additional information, you can refer to the two HVWS white papers related to performance: *Managing Web Site Performance* and *Design Pages for Performance*.

Plan: The third step is to plan your strategy to move toward on demand processing. Which new application(s) will you implement? On which servers? Do service level agreements need to be modified and/or created? For additional information, you can refer to the two HVWS white papers related to capacity planning: *Planning for Growth* and *High-Volume Web Site Performance Simulator for WebSphere*.

Execute: Finally, you can establish the foundation for your plans. Your IT infrastructure will need to evolve to on demand processing. You can start now making the changes necessary to facilitate your transition:

- Start by installing the monitoring software required for real-time monitoring.
- Review the levels of your software components and upgrade those needed to support your future plans. Consider software based on open standards, such as WebSphere, to facilitate integration.

Take your time creating a methodical approach and you will be ready when appropriate on demand products become available.

IBM has a number of new offerings to help you get ready and get started. To get ready, IBM offers a one and a half day e-business on demand assessment that will help you set the vision, attributes and metrics needed for your e-business on demand transformation. To get started, the IBM Server Allocation for WebSphere Application Server offering described in this paper is available to a limited number of customers beginning at the end of 2Q, 2003. Contact your local IBM sales representative for more information.

Appendix A. Introduction to grid computing and autonomic computing

Grid and autonomic computing are all the buzz. Here's a quick introduction to each. Check out the references for additional information.

Grid computing and Web services standards

A *grid* is a collection of distributed computing resources available over a network that appears to an end user or application as one large virtual computing system. A grid can span locations, organizations, machine architectures, and software boundaries to provide unlimited power, collaboration, and information access to everyone connected to the grid. The effect of grid computing is to make network computing more like a utility. You deliver computing power to where you need it only when you need it; you pay for what you use, when you use it.

Like the Internet, grids started in the scientific community but are now being deployed by business enterprises. Recently, IBM and Globus collaborated to combine open grid protocols and Web services standards, producing the Open Grid Services Architecture (OGSA). OGSA is a distributed interaction and computing architecture that is based around the grid service, enabling interoperability on heterogeneous systems so that different types of systems can communicate and share information. It leverages the emerging Web services to define the Web Services Definition Language (WSDL) interfaces. Specifically, the Grid service interface is described by WSDL, which defines how to use the service. A new tag `gsdl` has been added to the WSDL document for grid service description. The UDDI registry and WSIL document are used to locate grid service. The transport protocol SOAP is used to connect data and application for accessing grid service. All services adhere to specified grid service interfaces and behaviors.

Learn more

Developing Grid computing applications (Article)
IBM Grid Toolbox (alphaWorks download)
Installing and configuring the IBM Grid Toolbox (Free tutorial)

Autonomic computing

Autonomic computing was conceived to lessen the spiraling demands for skilled IT resources, reduce complexity and drive computing into a new era that may better exploit its potential to support higher order thinking and decision making. Immediate benefits will include reduced dependence on human intervention to maintain complex systems accompanied by a substantial decrease in costs. Long-term benefits will allow individuals, organizations, and businesses to collaborate on complex problem solving.

Fully autonomic computing systems will be:

Self-configuring: Able to adapt to dynamically changing environments
Self-healing: Able to discover, diagnose, and act to prevent disruptions
Self-optimizing: Able to tune resources and balance workloads to maximize use of IT resources
Self-protecting: Able to anticipate, detect, identify, and protect against attacks

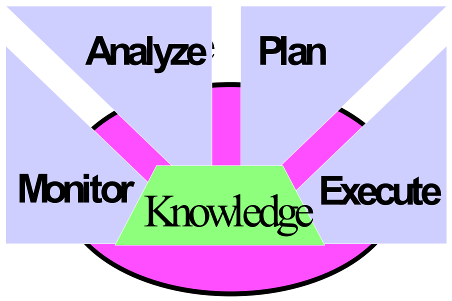
When you implement autonomic systems, you are free to focus on more strategic and higher-level issues. For the business, the core benefits of autonomic computing are improved resiliency, ability to deploy new capabilities more rapidly, and increased return from IT investments.

Autonomic computing capabilities play a critical role in the development of grid computing. Grids can become the most complex computing environments available. Autonomic computing will allow grids to be easily managed and to ensure that they deliver the levels and quality of service demanded by businesses.

Learn more

- Autonomic computing overview and resources (Collection of resources)
- Autonomic computing (Web site)
- Tivoli: Autonomic computing (Web site)
- Administration made easier: Scheduling and automation in DB2 Universal Database (Article, in PDF)

Fundamental to autonomic computing is what is known as the autonomic control loop. The goal is the ability to *monitor* current workload, *analyze* it versus historical trends and existing computing power, *plan* the reallocation of those resources, and *execute* the movement of workload or resources to maximize computing responsiveness to meet service level objectives.



The monitor function implements mechanisms that collect, aggregate, filter, and report details (metrics, topology, etc.) about the resource. The data may be sampled, as with performance metrics, or they may be unsolicited, as with event data. In addition, the monitor function can aggregate data from various external and internal sources, to enable it to produce more meaningful, and higher level, information.

The analyze function implements mechanisms that model complex situations. The analyze function may provide some learning capability, which will in turn allow it to better anticipate future situations. Examples of the analyze function include problem determination, workload forecasting, policy analysis, and event correlation.

The plan function provides a way to coordinate interrelated actions over time. A plan is constructed to achieve a goal, especially, goals related to service level agreements. The goal describes a desired state, for example, browse response time is less than 1.2 seconds. The plan specifies the actions to take and the sequence in which they should be invoked to move from the current state to the desired state. The plan function must also be aware of how long the plans it generates takes to enact and when they have to be completed.

For example, in an automatic order-processing environment, consider a situation where, 15 minutes before the end of the day, more than 2% of the orders initiated on that day have not yet been processed. In this case, the plan function could reasonably decide to provision extra elements of various types in an attempt to ensure that the outstanding orders are processed in time.

Today, this is most frequently embedded in automation scripts. For example, restart sequences are essentially a plan that takes into account the dependency of services on one another. Further, plan information is important in software distribution to properly sequence installs based on prerequisites, corequisites, and exrequisites. There is also plan functionality related to how software is sent between distribution nodes to account for constraints in network bandwidth.

One way to think of a plan is as a workflow. That is, a plan specifies a partial order of actions (for example, start an HTTP process), possibly with entry and exit conditions. This is useful in restart sequences since, for example, one should not restart a servlet until the servlet engine is running, which in turn requires a running JVM.

The execute function interprets plans and interacts with the element effectors to ensure that the appropriate actions occur. For example, if a plan is represented by a workflow, the execute function determines which actions in the workflow graph can be executed (for example, their inputs are satisfied) and then chooses the "best" one to perform. In the case of software distribution, the inputs are install dependencies and "best" may relate to balancing network loads and/or minimizing node down times.

Knowledge: Autonomic computing functions require common knowledge to work in a coordinated way. Examples of this knowledge may include the following:

- State of the managed element and management actions; for example, in the case of a printer, state may be whether it is online or offline; in the case of a Web server, state may be the number of connections currently held
- Dependencies between entities being managed; examples include: service dependencies (for example, an HTTP service requires a name service); install dependencies; and capacity dependencies
- Plans constructed to achieve management goals
- Goals, such as those expressed by service level agreements and policies, such as preferred ways of achieving a goal

References

- See all the IBM High-Volume Web Site white papers at www.ibm.com/websphere/developer/zones/hvws
- *Developing Grid computing applications, Part 1* at www-ibm.com/developerworks/library/ws-grid1/
- *e-business on demand: A developer's roadmap* at www.ibm.com/developerworks/webservices/library/i-ebodov/index.html?dwzone=webservices
- *The Physiology of the Grid* at www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf
- The WebSphere Application Server Performance Web site provides a centralized access to many helpful performance reports, tools and downloads. See www.ibm.com/software/webservers/appserv/performance.html
- IBM Redbooks have additional WebSphere performance information at ibm.com/redbooks.nsf/portals/Websphere

Notices

Trademarks

Contributors

The High-Volume Web Site team is grateful to the major contributors to this article: Ian Brackenbury, Susan Holic, Rahul Jain, Larry Lange, Linda Legregni, Noshir Wadia, and Peng Ye.