On Demand Business solutions
White paper

WebSphere® software

Enabling operational excellence for your application infrastructure — delivered at the speed of business.

December 2006

Contents

Introduction

In today's business environment, your organization must optimize the quality of service that it provides, while minimizing IT expenditures. You have to respond quickly to customer and market demand, while using all of your available assets. And your IT infrastructure must be able to keep pace with your changing business needs – with a minimum of human intervention.

As a business executive, you prefer an environment where:

- *Your business can reduce costs – and still be prepared for future growth.*
- *You can reuse existing resources, including people skills, hardware, software and infrastructure.*
- *Your infrastructure assets could run themselves intelligently, leaving people more time to do other productive work.*
- *Your infrastructure is streamlined and always open for business.*

In much the same way, if you're an administrator in an application server environment, you want:

- *Your application server to be able to determine whether an application running on it has a memory leak and can restart itself when necessary.*
- *Your applications to know how much processor power is needed at any given time based on measurements of incoming traffic to the Web site, preconfigured service policies and application characteristics.*
- *To know the status of your whole cell at a glance.*
- *Your application server to support your actual heterogeneous application environment, rather than a prefabricated homogeneous setup.*
- *Your application environment to support the deployment of multiple versions of applications without interruption in service.*
- *To write a robust Java™ 2 Platform, Enterprise Edition (J2EE) application that could cache data effectively, no matter what the size of the underlying data sets.*
- *To more effectively and securely share cached data with everything that needs it.*

IBM has a long history of providing leading-edge middleware that addresses high availability, superior functionality and reliability – without sacrificing the consistent, predictable performance you need to maintain a competitive advantage. IBM WebSphere® Extended Deployment, Version 6.0.2 delivers on this promise by creating an extension package to the already robust, enterprise-class capabilities of the IBM WebSphere platform, such as the industry-leading application server IBM WebSphere Application Server Network Deployment and IBM WebSphere Application Server for z/OS, as well as IBM WebSphere Process Server.

This white paper provides an overview of WebSphere Extended Deployment software. It describes the core function of the product that can help you derive the most value from your complex computing environment. It also explains how businesses of all sizes, with wide-ranging needs, can use WebSphere Extended Deployment software to increase business flexibility and reduce IT complexity.

An optimized IT application infrastructure

By extending the capabilities of the WebSphere platform, WebSphere Extended Deployment can help you increase server-utilization rates, increase the performance of your transactional infrastructure and enable more-robust management of your deployments, while enhancing the quality of service available to your entire application software stack (see Figure 1).

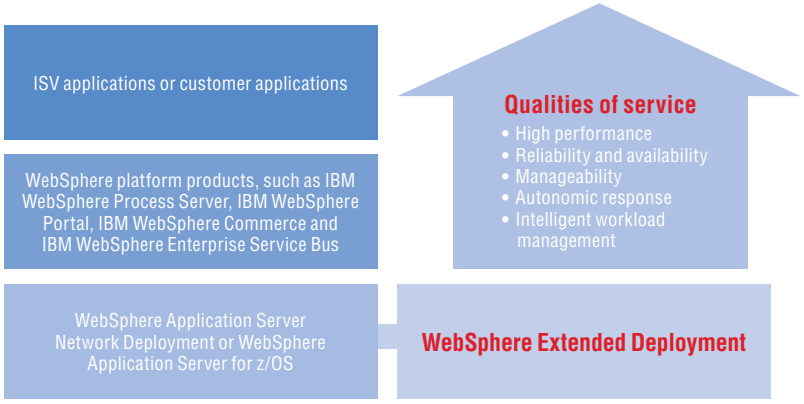

Figure 1. WebSphere Extended Deployment extends the capabilities of the WebSphere platform.

WebSphere Extended Deployment capabilities can help you handle the IT scalability and performance challenges of today's business environment. Taking advantage of the principles and concepts of proven IBM systems, and years of IBM research and client experience, WebSphere Extended Deployment enables:

- *Dynamic operations*
- *Extended manageability*
- *High-performance computing*
- *Business flexibility*

Dynamic operations

WebSphere Extended Deployment enables your application environment to scale as needed with the virtualization of WebSphere resources and the use of a goals-directed infrastructure, helping you increase the speed at which your company can adapt to business demands.

Extended manageability

WebSphere Extended Deployment offers simpler and improved management of complex system operations with advanced, meaningful, real-time visualization tools and gradual, controlled implementation of autonomic computing capabilities, helping you reduce the cost of managing IT resources.

High-performance computing

WebSphere Extended Deployment enhances the performance and throughput of critical applications, while enabling near-linear scalability for high-end transaction processing, helping you improve customer-service levels. All of these capabilities are delivered in a highly available framework to help ensure that you are open for business 24x7.

Business flexibility

WebSphere Extended Deployment enables heterogeneous business environments to efficiently support mixed workloads, such as batch, compute-intensive, and online transaction processing (OLTP). It also supports mixed application-server types, including both WebSphere Application Server and software from other vendors, while providing the ability to run on and integrate a broad set of platforms.

WebSphere Extended Deployment, Version 6.0.2 enables a business grid – a dynamic, goals-directed, high-performance application environment for running mixed application types and workload patterns on WebSphere Application Server.

Dynamic-operations capabilities help increase responsiveness and flexibility

WebSphere Extended Deployment is designed to deliver dynamic operations through several key capabilities – the virtualization of WebSphere environments, autonomic managers and a goals-directed infrastructure. A virtualized WebSphere environment helps increase the flexibility of your solution, enabling rapid response to changing business needs through the dynamic allocation of WebSphere resources, while protecting your critical application workloads.

WebSphere Extended Deployment implements a virtualized environment by creating pools of resources that can be shared among applications, helping to optimize usage and simplify overall deployment. As resources are needed for expected (or unexpected) spikes in workload demand, application resources can be allocated where they're needed most. This capability enables better use of the computing resources that you already own and can potentially allow you to run more applications on the machines that you already have. To protect and categorize workloads, WebSphere Extended Deployment complements dynamic operations by providing the ability to define IT goals and policies for handling application workloads. This capability enables traffic that is deemed more important to receive better quality of service, providing its users with a better overall end-to-end user experience.

Share resources among applications to optimize utilization and simplify deployment

WebSphere Extended Deployment redefines the relationship between traditional J2EE constructs. Instead of running all of your available servers at all times so that you can handle peak traffic conditions, you can allow WebSphere Extended Deployment to determine the right number of servers to run based on your configured goals to provide optimal IT-infrastructure utilization. The virtualized infrastructure is predicated on two WebSphere Extended Deployment constructs: *node groups* and *dynamic clusters*.

A *node group* is a defined set of servers with some common characteristic, such as access to a network or database. Within a node group, one or more dynamic clusters are created, which are essentially nothing more than application-deployment targets. The computing power represented by a node group is divided among its member dynamic clusters. This distribution of resources is modified autonomically, according to business goals, to compensate for changing workload patterns. This loosely defined relationship between node groups and dynamic clusters extends your available resources and provides the foundation for the autonomic abilities of WebSphere Extended Deployment.

Differentiate application service levels according to your business requirements

The goals-directed infrastructure capabilities of WebSphere Extended Deployment mean that user requests are classified, prioritized, queued and routed to servers based on application-service policies. Application performance is optimized according to these policies, which directly reflect business goals and relative importance to your organization. You can state what applications are important to you, and these applications get the highest-priority access to your WebSphere resources at the right time. This capability can help you ensure, for example, that your critical transactions get the best quality of service.

A *service policy* is an explicit, configurable contract through which operating goals and guidelines are fed into a WebSphere environment. Through service policies, WebSphere Extended Deployment introduces the capability to designate the business importance of different request types. Among other things, this capability can help balance your resources optimally in a controlled manner during periods of heavy workloads.

Increase business value by combining z/OS Workload Manager and WebSphere Extended Deployment

Many implementers of high-end IBM System z™ environments question the relationship between the IBM z/OS® Workload Manager and the service goals delivered in WebSphere Extended Deployment — both deliver policy-applied business goals to workloads. By using z/OS Workload Manager and WebSphere Extended Deployment together, however, you can increase their value to your business. z/OS Workload Manager addresses WebSphere Application Server for z/OS work as part of the greater System z enterprise workloads, whereas WebSphere Extended Deployment focuses only on applying goals to your WebSphere workload. As a result, when using WebSphere Extended Deployment on z/OS, you gain the value of z/OS Workload Manager enterprise-wide policies, as well as the more granular definition of WebSphere application workloads.

WebSphere Extended Deployment, Version 6.0.2 includes policy support for long-running workloads to help ensure that the autonomic infrastructure recognizes goals affiliated with differentiated workloads. The product also extends policy support for traditional workloads to include manifestations as lists of Uniform Resource Identifier (URI) filters, authenticated identities, IP addresses or query parameters. This capability gives you finer-grained control of how the traffic arriving at a Web site is characterized, and as a result, supports a more granular declaration of service policies for Web site traffic. Also, support has been provided for Internet Inter-ORB Protocol (IIOP) and Java Message Service (JMS) traffic, which helps widen the scope of control that you have on traffic flowing into the enterprise.

When demands exceed the available computing power, the WebSphere Extended Deployment infrastructure uses service goals to determine which workloads are critical and which can be sacrificed. This capability helps ensure that critical requests maintain a high quality of service, helping to achieve graceful and controlled degradation. (See Figure 2.)

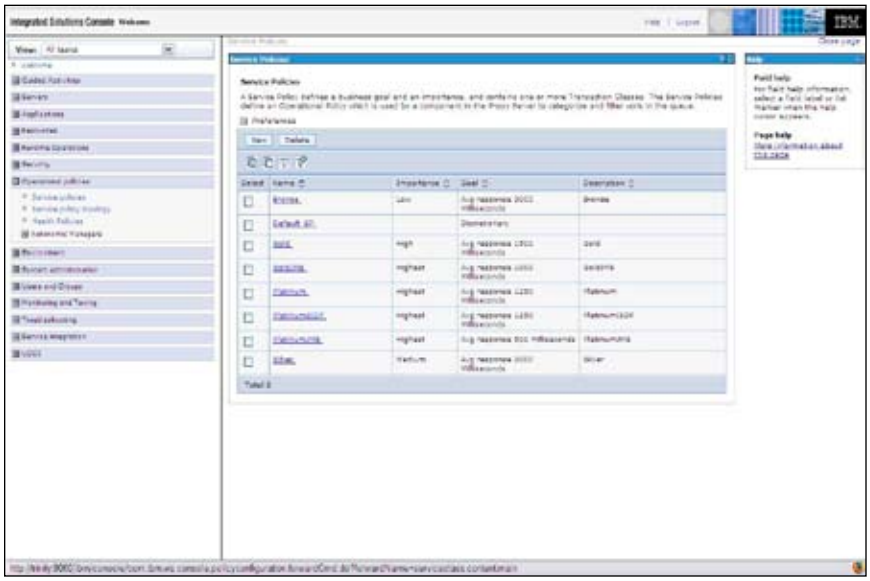

Figure 2. Defining service policies in the administrative console

Introduce autonomic capabilities into your infrastructure at your own pace

WebSphere Extended Deployment uses a virtualized infrastructure as a basis for its on demand capabilities. This section focuses on automated enterprise orchestration, or self-managing automation delivered with WebSphere Extended Deployment. Automation brings with it a host of new topological entities, including the on demand router and several WebSphere Extended Deployment autonomic managers.

The on demand router

The *on demand router* is a component that sits in the front of the WebSphere Extended Deployment implementation and acts as a proxy server. It represents the entry point into a WebSphere Extended Deployment topology and controls the flow of requests into the back end. Specifically, the on demand router handles the queuing and dispatching of requests according to the business goals assigned to the work.

The on demand router, when deployed with WebSphere Extended Deployment on z/OS, provides enhanced routing capabilities in cross-logical partition (LPAR) routing. Also, the on demand router on the z/OS platform provides policy-based traffic routing and load-distribution capabilities similar to z/OS Workload Manager. An on demand router that is not running on z/OS is also able to recognize the capacity of a z/OS node and use the power of such a node by directing the appropriate amount of traffic to it. Among other things, WebSphere Extended Deployment and the base z/OS platform provide service-policy integration. This function enables WebSphere Extended Deployment to combine the capabilities of its powerful transaction classification engine with the workload management capabilities of the z/OS platform to deliver optimal resource utilization. (See Figure 3.)

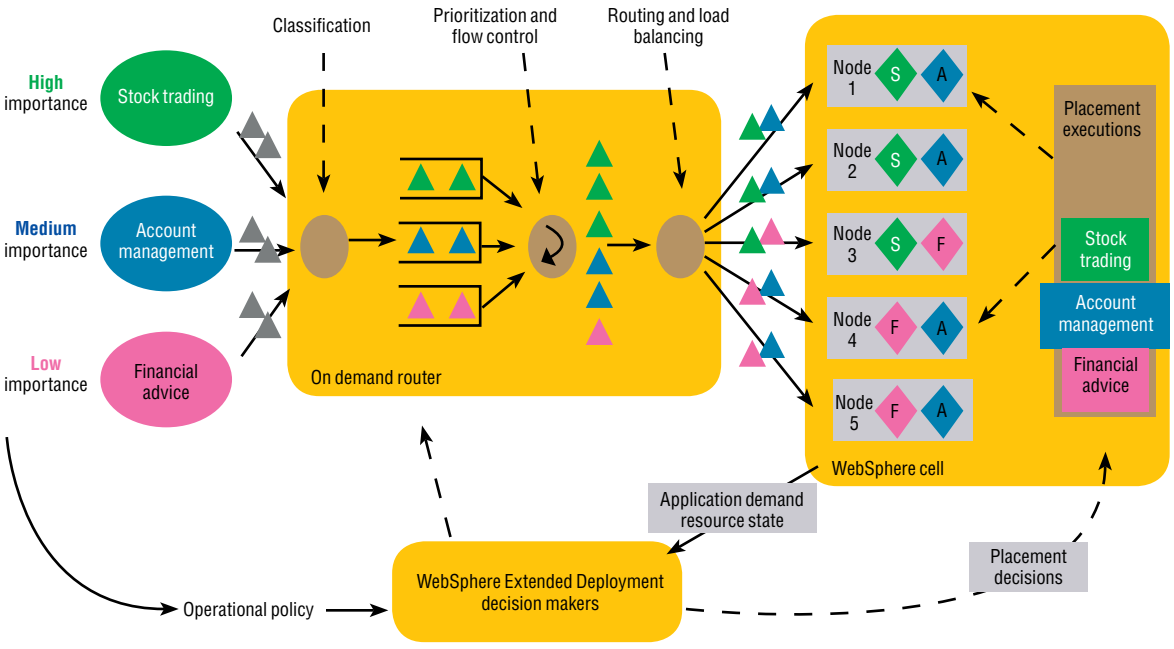

Figure 3. On demand router architecture

Dynamic workload manager

The *dynamic workload manager* is responsible for maintaining the load-balancing weights used by the on demand router. The dynamic workload manager modifies server weights by constantly monitoring and analyzing response times and utilization rates of the servers in the enterprise. The primary goal of the dynamic workload manager is to even out the average service times among the servers in a given cluster; there is also a secondary goal of evening out the weights if and when the processor utilizations are all low (at which time uneven weighs are unlikely to have any positive effect anyway). This function enables the autonomic infrastructure to help ensure a fair distribution of incoming work to servers in the enterprise that is proportional to their capacity.

Autonomic request flow manager

The automomic request flow manager (ARFM) includes the gateway (part of the on demand router) and work profiler, as well as the ARFM controller. Together, these components manage the flow of requests from the on demand router queues to the back-end nodes. ARFM acts as a faucet to control how many or how few requests make it to the back end for implementation. It decides which requests are dispatched and when, whereas the dynamic workload manager decides where. In addition, the ARFM controller continuously monitors the performance of each service class and modifies queue weights for the on demand router as needed to help ensure workload balance. Each ARFM gateway also handles processor-overload protection by limiting concurrency, using a limit determined autonomically by the ARFM controller. A cell's ARFM controller also computes the processing requirements that are inputs to the application placement controller.

Application placement controller

A single application placement controller exists in the cell, and its sole purpose is to determine the location and cardinality of active application-server instances in a given dynamic cluster. As demand changes for a given dynamic cluster, that dynamic cluster is expanded or contracted according to service policy.

Health policy managers

You can protect your system resources by enabling WebSphere Extended Deployment to be constantly alert to a series of preexisting software conditions, and when those conditions are discovered, to automatically take action. The preexisting health policies that come with WebSphere Extended Deployment enable it to act upon application server conditions, such as excessive memory consumption, excessive throughput, applications not meeting their goals or applications running for some length of time that might have caused problems or unwanted conditions to creep into the application with extended use. These policies free you from having to locate problems in the environment, and provide a proactive approach to avoiding issues by acting upon them before they become problematic.

Functional enhancements deliver improved efficiency and flexibility

Beginning with Version 6.0.2 of WebSphere Extended Deployment, you can support installations on both WebSphere Application Server Network Deployment, Version 6.0.2 and Version 6.1. Also, the product delivers new function for dynamic operations that takes advantage of its robust autonomic capabilities. The primary focal areas for enhancements in operations optimization revolve around removing application restrictions, allowing for more granular classification of workloads and improving the consumption of the node groups.

On demand application start

On demand WebSphere Extended Deployment, Version 6.0.2 removes configuration restrictions set in the initial release of WebSphere Extended Deployment, where each application deployed to a dynamic cluster had to run at least one instance of the application, regardless of whether demand existed. The default maximum size for the application deployment was determined by the size of the node group, indicating that the application could grow to consume the resources of the entire node group.

WebSphere Extended Deployment, Version 6.0.2 also includes the on demand application start-deployment option (introduced in Version 6), which enables you to deploy a dynamic cluster, but not immediately run it. This capability implies that a cluster is configured and ready for use, but not actually running until there is a demand for it. When a request for that cluster is received by the on demand router, the cluster is started automatically on an available node. If after some period of time the cluster has not been used, WebSphere Extended Deployment shuts it down to make the resources it is consuming available to other clusters in the system. This mode of operation is useful if you are running a set of infrequently used applications because it enables these applications to be serviceable without requiring them to continuously consume resources.

Vertical stacking

The second dynamic operations enhancement delivered in WebSphere Extended Deployment, Version 6.0.2 removes the restriction that at most a dynamic cluster could have one instance running on a given machine (node) at a given time. This restriction was predicated on the assumption that a given application in that dynamic cluster could drive the processors on that machine to 100 percent. In other words, it assumed the application was scalable, when in fact that might not always be the case. As a result, WebSphere Extended Deployment, Version 6.0.2 includes a feature called *vertical stacking*, which enables multiple instances of an application server to be started on the same physical node to provide greater throughput on that application, and ultimately more efficient consumption of node-group resources. For optimal resource utilization, however, it is recommended that vertical stacking be used only in environments hosting applications with scalability issues, such as synchronization bottlenecks, that would prevent the application from scaling with increased traffic.

Enhanced classification of requests to include variables other than URI

All requests with the same URI are not equal. Often it is desirable to apply different routing and service policies to a request based upon a query parameter, an HTTP header or cookie value. WebSphere Extended Deployment, Version 6.0.2 provides a unified method for performing this task. For example, requests to browse stock quotes could be associated with the "bronze" service policy whereas requests to buy or sell stock could be assigned the "gold" service policy, where the gold service policy is configured to give a higher quality of service than bronze.

Furthermore, when fronted by a security proxy such as IBM Tivoli® WebSEAL, you can assign routing and service policies based on the identity of the client. In particular, Tivoli WebSeal is configured to pass the user's identity in an HTTP header, and the on demand router uses this information to assign the appropriate routing and service policies. For example, consider a scenario in which two versions of a single application are deployed simultaneously to two different clusters, where version 1 is for a production cluster and version 2 for a test cluster. In this scenario, it is possible to send John Customer's request to the production cluster while sending Joe Tester's to the test cluster. Or, as another example, you might classify a customer who is a member of the "BigSpenders" group to have a higher quality of service than a customer who only browses and never buys.

Not only is classification more granular, but you can now use it to classify requests that are routed to non-WebSphere servers. This capability enables WebSphere Extended Deployment to apply routing and service policies to all types of incoming requests, regardless of the servers to which they will be routed.

Session rebalancing

The third dynamic operations enhancement delivered in WebSphere Extended Deployment, Version 6.0.2 enables you to normalize workloads across available servers in your enterprise by dynamically and actively balancing the distribution of HTTP sessions among application servers.

WebSphere Application Server routes HTTP messages based on HTTP session affinity to application servers. After a session is established in a particular application server, subsequent requests that belong to the same session are routed to the application server that has the established session affinity.

Session affinities to various application servers in a cluster tend to balance out for a cluster in the long term, and generally any imbalances that occur are short-lived and tolerable. However, you can use session rebalancing to help some of the shorter-term imbalances that can occur, without modifying the performance benefits of session affinities.

The dynamic workload manager knows about the number and location of sessions. It also tracks usage of various servers in the form of server weights. Based on these pieces of information, it arrives at a decision to actively relocate HTTP sessions from overutilized servers to relatively underutilized ones. You can use session rebalancing if you use distributed sessions and track your sessions with cookies.

Optional automation to get more out of your IT environment

IBM Tivoli Intelligent Orchestrator and IBM Tivoli Provisioning Manager (optional, available separately) can be used in enterprise-wide scenarios for supported platforms. When integrated with Tivoli Intelligent Orchestrator, a WebSphere Extended Deployment topology can obtain more physical computing resources from outside a node group.

When demand distribution within a WebSphere Extended Deployment enterprise shifts between node groups, the ability to compensate extends beyond the boundaries of the autonomic managers in WebSphere Extended Deployment. When you integrate with Tivoli Intelligent Orchestrator, the size of the WebSphere Extended Deployment node groups can be automatically extended based on fluctuating demand. You can share servers among multiple application environments with multiple WebSphere Application Server cells or environments that are not using WebSphere software.

Tivoli Intelligent Orchestrator monitors and manages a set of different resource pools, such as IBM WebSphere Application Server, IBM WebSphere MQ, service access point (SAP) and e-mail. In this way, the product helps ensure that the processing power of these resource pools in the form of machines, LPARs, processors and so on, is allocated in accordance with your business goals. With the WebSphere Extended Deployment provisioning engine, Tivoli Intelligent Orchestrator receives information that it can use to compute allocations of available resources within the scope of a WebSphere Extended Deployment topology. It then allocates physical resources among the candidate node groups.

The autonomic provisioning for WebSphere Extended Deployment can be performed on two levels: autonomic decisions based on processing objectives, and autonomic decisions based on end-to-end performance goals.

Supporting heterogeneous IT environments

WebSphere Extended Deployment business-flexibility capabilities embrace real-world IT environments. As a result, WebSphere Extended Deployment, Version 6.0.2 includes support for the management of application servers other than WebSphere Application Server. This support includes older versions of or stand-alone WebSphere Application Server, BEA, Tomcat, JBoss, Microsoft® .NET, Web servers and so on.

Mixed-server environments

IBM WebSphere Extended Deployment for Mixed Server Environments provides a remote monitoring agent that you can install on non-WebSphere Extended Deployment server nodes to retrieve additional system information (such as processor utilization) and forward it to on demand routers for dynamic workload distribution. By using on demand routers to route traffic to other vendor server processes, you can build a single workload routing layer to handle all HTTP traffic flowing into that set of server processes. The operational policies and visualization features of IBM WebSphere Extended Deployment, Version 6.0.2 support this workload-routing layer. With this quality-of-service extension, you can attain the benefit of autonomic workload management and service policies based on workload prioritization even when your back-end server processes are not WebSphere Extended Deployment server processes.

Generic server clusters

To enable an on demand router to route to other vendor server processes, you can configure them into a generic server cluster. A *generic server cluster* is a collection of transport end points that can be used as a target for an on demand router routing rule. In the generic server cluster, you configure the server processes and nodes, including their connection protocol (HTTP or HTTPS), host names and ports. The servers corresponding to the end points in a generic server cluster can be any server-process type, that is, WebSphere Application Server processes, foreign server processes or generic server processes.

Generic server processes can be stopped and started by a WebSphere Application Server node agent, whereas foreign server processes are usually not stopped and started by a node agent. All server node types can run the remote agent that is provided by WebSphere Extended Deployment for Mixed Server Environments to allow for dynamic workload management based on processor utilization.

Routing to other vendor servers and older versions of WebSphere Application Server

Normally, the on demand router supports automatic routing to nodes in target cells that have a deployment manager with WebSphere Extended Deployment installed. To route to nodes that do not have WebSphere Extended Deployment installed, a generic server cluster must be explicitly configured. In addition, you need to manually configure service policies and transaction classes for applications that are deployed on these generic server clusters, including the mapping of URIs (or URI patterns) to transaction classes. The on demand router can also support session affinity with HTTP sessions when routing to all types of other vendor servers. To accomplish this, it allows for the configuration of active and passive affinity for various server types.

Through such a mixed-server environment installation, you can also use WebSphere Extended Deployment to route traffic to WebSphere Application Server, Version 5.x and other older versions of the WebSphere Application Server. Such routing is simpler to configure compared with other vendor servers, because the on demand router tier automatically recognizes and honors the affinity information format in the incoming request. (See Figure 4.)

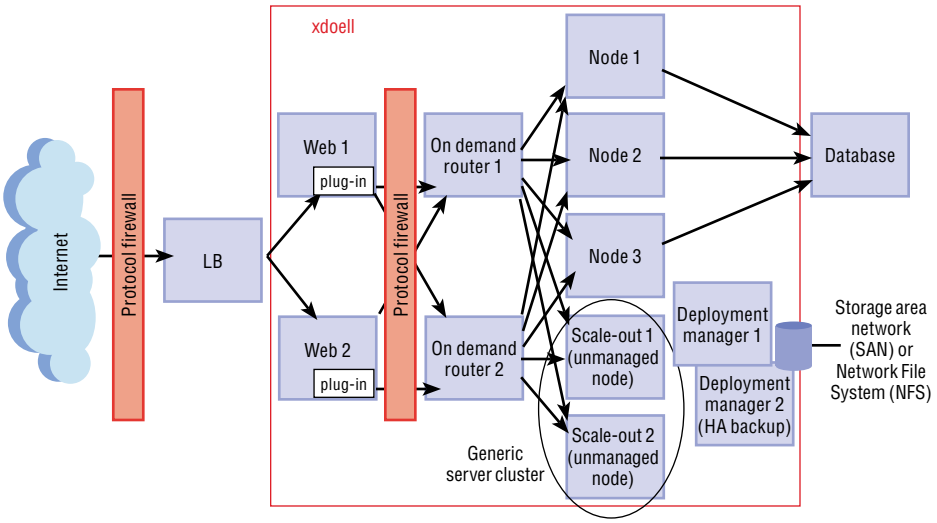

Figure 4. Sample deployment depicting the use of a generic server cluster for non-WebSphere Application Server (or unmanaged node) routing

Although using such a topology enables you to get several benefits of WebSphere Extended Deployment in a non-WebSphere Application Server environment, none of the application placement and policy-based health-monitoring features are available by using WebSphere Extended Deployment for Mixed Server Environments.

Extended manageability helps simplify IT management while maintaining administrator control

It can be difficult to visualize and manage complex IT environments where many applications are deployed on hundreds of application servers. Although the WebSphere Application Server administration console provides excellent built-in capabilities, the special needs of very complex deployments require an aggregated, meaningful view of the application runtime environment. WebSphere Extended Deployment extends the administration console to allow operators to assess their environment at a glance.

The WebSphere Extended Deployment administrative console contains several tools that help you visualize the inner workings of a WebSphere Extended Deployment topology, so that you can remain well informed about the activities taking place within your environment.

Runtime topology

One such view is the runtime topology view, which is a depiction of the momentary state of a WebSphere Extended Deployment environment. This view is refreshed on a configurable interval to provide updated information. The runtime topology contains many useful pieces of information, including:

- *Unhealthy servers*
- *Application editions*
- *Long-running workloads*
- *Application-provisioning activity*
- *Deployment of dynamic-cluster instances*
- *Processor usage (per node)*
- *Node-to-node group memberships*
- *Dynamic cluster-to-node group memberships*
- *Dynamic workload-management weight (per application-server instance)*
- *Process identification (per application-server instance)*
- *Data-logging capabilities*

Charting

WebSphere Extended Deployment charting presents administrators with customizable graphs of runtime data observed throughout a WebSphere Extended Deployment environment (see Figure 5). This view is refreshed on the same configurable interval as the runtime topology view. With WebSphere Extended Deployment, you can chart a wide variety of statistics in different styles of graphs. Supported statistics include:

- *Capacity*
- *Average response time*
- *Concurrent requests*
- *Average throughput*
- *Average queue wait time*
- *Average service time*
- *Average queue length*
- *Average drop rate*

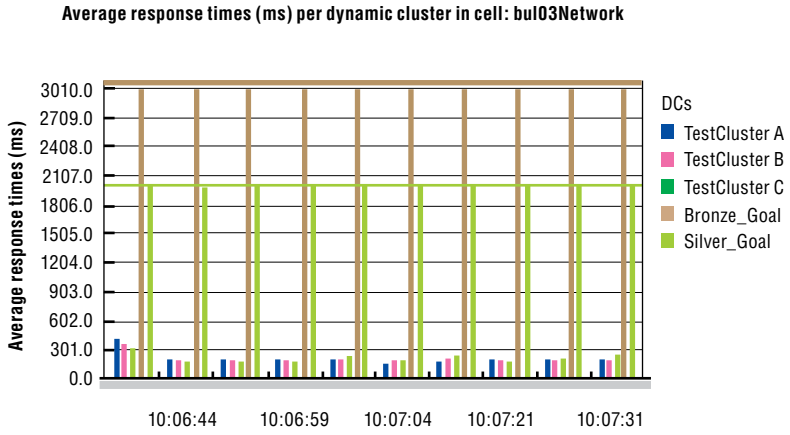

Figure 5. Chart view

You can construct charts from several different perspectives as well, which gives you flexibility in the scope of the statistics you observe. WebSphere Extended Deployment supports charting from cell, node group, dynamic cluster, service policy, transaction class, J2EE module and on demand router perspectives.

Know when intervention is required

WebSphere Extended Deployment extends the administrative console to notify you of decisions made by autonomic managers. Notifications can represent either planned or unplanned events.

Planned events

Planned events are those (expected) events for which the WebSphere Extended Deployment run time has an action plan. An example would be an average response time breaching its configured limit, which might trigger an increased dynamic-cluster footprint. Depending on the configured level of automation, these events could be presented to the administrator in one of several ways. If WebSphere Extended Deployment is operating in automatic mode, the action plan is implemented and a simple notification is presented to the administrator. In supervised mode, the administrator is presented with the action plan, and prompted for approval.

Unplanned events
Events that are not assigned an action plan are displayed as warnings to let you know that something unexpected has happened. It is then up to you to develop a plan to correct the situation, if it is indeed problematic.

Enable interruption-free application version updates
Application availability is a critical business imperative for many companies. Downtime of business applications for any reason, including service, has increasingly negative business impacts. The reality of doing business today requires advanced application management for versioning of existing applications, due to serviceability needs of applications. The advanced manageability enabled by WebSphere Extended Deployment is designed to reduce the complexity of IT environments.

WebSphere Application Server currently offers multiple options for updating applications, including full and partial replacement, but these capabilities can still interrupt application rollout. In some cases, application requests might be lost. Also, the existing application update methodologies can be destructive, which means that new application artifacts replace the preexisting ones, without any back-out capability. These features can be crucial to supporting production applications.

Managing production applications requires support for scenarios that the current version of WebSphere Application Server does not address, such as validation of an application update before putting it into production and the hosting of multiple versions of an application for different customer constituencies, including regular customers and pilot customers. User ingenuity has helped overcome these limitations in some cases.

WebSphere Extended Deployment, Version 6.0.2 helps minimize the need for user interference in providing application updates by providing a function called the *application edition manager*. This feature provides key application-management capabilities that enable you to access and update applications in production, without affecting downtime – helping to enhance the value of your WebSphere environment by enabling core function where it is most needed.

The application edition manager provides advanced application-management capabilities that focus on five functional requirements:

- *The application-versioning capability makes it possible to distinguish one version of an application in the production environment from another.*
- *Virtually interruption-free application rollout enables a smooth transition from one edition of an application to another without a loss of service. This capability means that all application requests can be serviced during the rollout—and none are lost.*
- *You can reverse an application update without fear of losing in-transit work.*
- *Application validation before deploying to a production environment is supported using validation mode, a special case of activation that enables you to perform final preproduction testing of an application edition in the actual production environment with a selected set of users (see Figure 6).*
- *Support for concurrent versions of applications enables support for complicated rollout strategies, such as piloting and branch upgrade.*

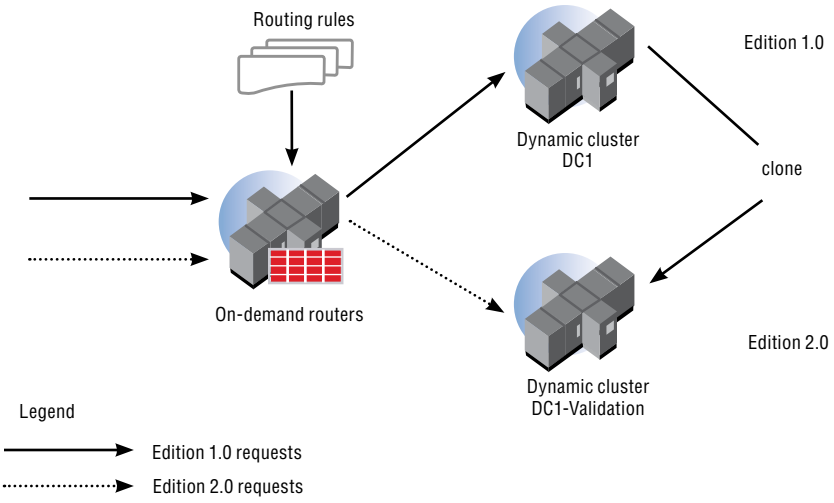

Figure 6. Using the application edition manager to test preproduction rollout in validation mode

High-performance computing designed to reliably support high-volume transaction requirements

To reliably support ultra-high-end transaction-processing requirements within a unified WebSphere environment, WebSphere Extended Deployment provides dynamic application partitioning and repartitioning, high-end caching, object caching, workload management and autonomic high-availability management. The product delivers extreme computing through two key functions described in the following sections: the *partition facility* and the *ObjectGrid*.

Partition facility: Enabling the development of partitioned J2EE applications

The partition facility provides an essential capability required to achieve the next level in performance, scalability and availability in J2EE applications. During the past five years of WebSphere Application Server development, a series of design patterns have emerged that, when correctly implemented, have proven to significantly alter the performance playing field. The first design pattern that WebSphere Extended Deployment addresses is the partitioning pattern, which introduces the concept of application-specific partitions implemented using the partition facility – and potentially resulting in significant throughput improvements that can increase performance by as much as a factor of 10. The following paragraphs describe the partitioning pattern and how WebSphere Extended Deployment enables this pattern through the partition facility.

The partitioning pattern addresses bottlenecks that occur in high-volume OLTP applications that intensively read and write data to databases and require the utmost in data consistency and availability. Examples of such systems include trading, banking, reservation and online auctioning systems. Today's J2EE servers have been optimized for read-mostly environments like commerce systems, where data-caching capabilities can be used to offload the back-end database systems. However, as systems observe increased data-write (such as database insert, delete, create and update) ratios, these caching systems start to break down because of the ever-strenuous task of maintaining consistency between the cache and database (see Figure 7). These schemes often quickly reach a point of diminishing returns and are better off sending all traffic back to the database and allowing the database to manage consistency. This strategy frequently leads to large and costly database configurations, often running on large symmetric multiprocessor (SMP) machines. In ultra-high-volume environments, these database servers inevitably become a cost and performance bottleneck.

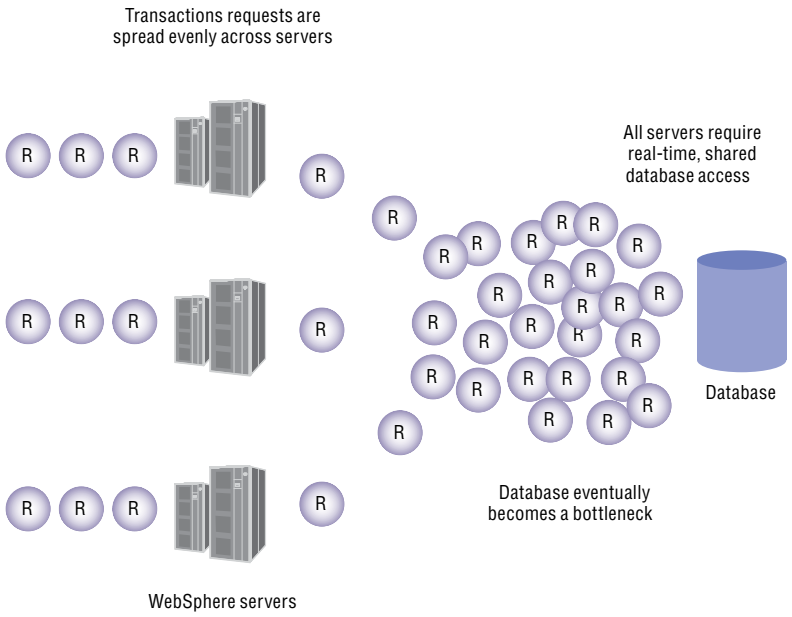

Figure 7. A conventional distributed environment

The partitioning pattern aims to offload the database by enabling the application server tier to act as a buffer. It also makes interactions with the database more productive. There are four key elements to the partitioning pattern. Table 1 outlines these elements and describes how they are manifested in WebSphere Extended Deployment.

Element	WebSphere capability
Partitioning-aware workload management	• WebSphere Extended Deployment through the on demand router (for HTTP) • Enterprise JavaBeans (EJB) client through the Internet • IIOP routing • JMS through pull model
Leveraging partitions	• WebSphere dynamic-caching service • Java Database Connectivity (JDBC) batching
Highly available partitions	High-availability manager for highly available partitions
Rebalancing partitions	WebSphere partition facility management beans

Table 1. Four key elements to the partitioning pattern

As the name suggests, partitioning is the essential element in the partitioning pattern. Although partitioning can't help you improve performance and availability on its own, it can establish the foundation upon which you can achieve these benefits.

A partition-facility partition can be described in several ways. In its simplest form, a partition is a list of labels that applications (or metadata found in declarations within applications) can create whenever they are required. For example, an application can render a set of abstract partition names to represent categories of items being bid upon during auctions: sporting goods, automobiles, toys and antiques (see Figure 8).

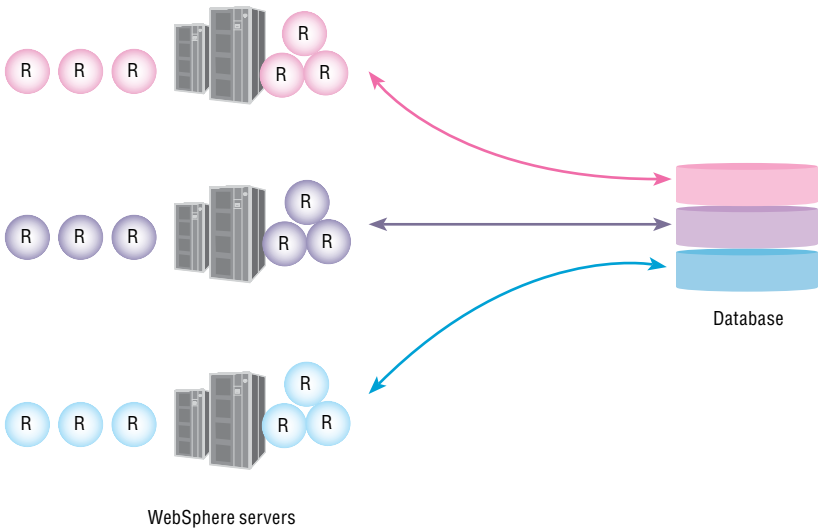

Figure 8. Application partitioning using the WebSphere partition facility

Partitions can also be expressed as a set of HTTP request URLs:

- */stock_app/stocks/ibm/**
- */stock_app/stocks/xyz/**
- */stock_app/stocks/abc/**

You can programmatically calculate partitions using an application-provided hashing function. For example, an application can apply a hash function to an arbitrary token (such as an HTTP header or cookie) to derive a number corresponding to a partition. When an application uses one of the aforementioned methods to create its partitions (typically at application startup or during a repartition operation), the partition-facility component assigns each partition to a server process in the WebSphere Application Server cluster.

The first step toward making partitions productive is to associate requests (of potentially varied protocols, such as HTTP, IIOP and messaging) with a partition. In this case, a request is classified as having an association with a participation, and the work is delivered to the server process where that specific partition has been assigned. For example, in the case of an online auction, if a bidder is bidding on an item in the sporting goods category, the bid request is routed to the server-process associated with the sporting goods partition.

In WebSphere Extended Deployment, there are several supported ways to route a request to partitions:

On demand router for HTTP

The on demand router component of WebSphere Extended Deployment routes HTTP requests to the partition with which each is associated. A deployment policy can be included in a WebSphere Extended Deployment application that specifies partitions and the rules by which HTTP requests are mapped to these partitions. At application startup, the partition facility determines the partitions and assigns them to active server processes in the WebSphere Application Server cluster. The partitions-to-server process assignments are sent to the on demand routers, where the information is used to classify and route incoming requests to the corresponding partitions.

EJB clients for IIOP

Remote EJB clients can be generated to be partition-aware. When these EJB call remote methods are running in a WebSphere Application Server cluster, a similar classification process occurs in the EJB stub where workload management and routing occurs. If the method call is associated with a partition, requests to that remote method are routed to the server process corresponding to the partition. This behavior is similar to the manner in which work is routed to stateful session EJB in clustered environments. However, with the partition facility, you can control the affinity process in a more precise manner.

JMS pull model

Although the partition facility doesn't include explicit support for messaging protocols, you can create a publish-and-subscribe-based messaging application so that publish-and-subscribe topics correspond to partitions. In this case, a server process would subscribe only to topics corresponding to its partition name. With this approach, a server can pull messages off a queue that corresponds to its partition.

Leveraging partitions to maximize performance and scalability

The ability to define application partitions and route messages to these partitions can be a powerful capability if it is used correctly. You can take advantage of these capabilities to achieve a new level of performance and scalability for your applications. With partitions and routing, you can set up areas of the server where you know work can be performed that will not occur anywhere else in the WebSphere cluster. The three most popular means of using partitions are caching, caching batching and singleton services.

Caching

An application can take advantage of the exclusive nature of a partition by aggressively caching data. For example, in an equity trading system, the system is designed to match buy orders against sell orders and to complete trade transactions. You can use partitioning to divide data by stock ticker symbol, allowing more-intelligent routing of trading requests and more aggressive caching on servers.

Batching

In the trading-system example, new high bids for a given item can be queued in local memory and then sent to the server in a batch, all at once. In a high-volume environment, such batching can produce significant savings in remote interactions with the database.

Singleton services

A partition can be the point in a WebSphere Application Server cluster where a particular function or service of single cardinality resides. In this case, the service is called a *singleton* because it is the only place in the cluster where this function is running. Singletons allow applications to be broken up and run across the cluster.

Recover from failures quickly

Using partitions can significantly improve performance. However, a partition could represent a single point of failure if preventive steps aren't taken. For example, in the case of using a partition for data caching, if a partition fails and there is no backup, a considerable amount of cached data can be lost. The same is true when running a singleton service. If the partition goes down, that service can be lost. It is for these reasons that the partition facility is implemented as a highly available service.

The partition facility uses the new high-availability manager in WebSphere Application Server, which actively monitors processes, services and partitions within a WebSphere Application Server cluster. Each server process in a WebSphere Application Server cluster maintains an awareness of the availability of the other processes in the cluster. If a server process goes down, the surviving processes work together to try to recover the process, its services and its partitions. For example, if the server process containing the partition for a given online auction category fails, the surviving servers in the cluster can detect this failure and elect a new server process to be the host of this category. Also, the partition facility and the high-availability manager component are designed to recover from a failure in a short period of time.

Rebalance partitions to help ensure optimum load balancing

To help ensure that a system is running optimally, you sometimes have to move partitions to rebalance the load on servers. For example, if the partition hosting the sporting goods category is experiencing a high volume of requests, it might be useful to move it to a more capable server. The partition facility includes a Java Management Extension (JMX) MBean to allow such operations. The result of rebalancing a partition is similar to a server-process failure. Rebalancing operations, however, is typically something that can be planned and done under program or operator control.

Create a caching fabric that enables application data to be shared among multiple clients

Whereas the partitioning facility approaches the development of applications into highly available targeted partitions to enable intelligent caching, the ObjectGrid provides the foundation for a distributed caching framework. In the former, the performance gains are confined to the partitioned application, whereas in the latter, performance gains are shared by any artifact requiring access to the cached data.

The *ObjectGrid* is a state-of-the-art distributed memory system for persisting application state to memory-based storage that enables applications to store object-based data in maps. It's scalable from an in-memory-only manager to a multitier distributed, replicated, partitionable grid holding gigabytes of data. The ObjectGrid is also the first component to be offered in a fully componentized form that can work with or without WebSphere Application Server and works on WebSphere Application Server, Version 5.0.2 or later. The ObjectGrid also works on other vendors' application servers, and is designed to integrate into an existing environment, regardless of whether it is based on IBM technology.

Transactional and remote data access

Changes to data stored in the ObjectGrid are made transactionally using atomicity, consistency, isolation and durability (ACID) properties. You can index and retrieve the objects once stored in the ObjectGrid using either primary keys or simple queries on indexed object attributes. The committed data can be stored locally in the same process as the application or it can be pushed remotely to an ObjectGrid server. Multiple processes can connect to a single ObjectGrid server and access shared object data stored there.

Data locking

ObjectGrid supports many different locking and isolation models. You can partition the server side using a function on the key, which enables very large data sets to be stored on the server side by striping the data over several Java Virtual Machines (JVMs). A strict partitioning model is used to help ensure scalability. Many JVMs can participate in this server-side farm, enabling you to store many gigabytes of data. Processes can incorporate a local or near cache in this client-server model, which also provides much-improved performance for the subset that is cached locally. You can use optimistic locking to detect stale data in this near cache. ObjectGrid processes can also operate concurrently in a peer-to-peer mode where changes made in the ObjectGrid in a process can be pushed to its peers using a publish-subscribe messaging system such as a JMS provider. The changes can be pushed in the form of the actual data that changed or the set of records to invalidate.

Relational database caching

You can plug the ObjectGrid into most object relational mappers to act as a second-level cache and integrate it into most popular open-source frameworks. The ObjectGrid includes application programming interfaces (APIs) or plug points to enable applications to integrate very tightly to the management of the data, which is a key differentiator to relational databases in terms of software-design philosophy. Relational databases do not integrate tightly with client applications. The ObjectGrid enables applications to receive various events from the ObjectGrid run time as they occur in the same application process. As an option, you can back the ObjectGrid by any persistent back end, such as berkeley db, a relational database or an SOA data service. This capability enables the ObjectGrid to pull data that is not currently cached automatically from the back end and push changed data to the back end. Many types of data used by an application need to be persistent, but not at the quality of service offered by a database — and not at the cost that single-database quality of service entails.

Data replication

Databases currently offer only a single quality of service, such as disk hardened. The ObjectGrid enables you to store this data in a memory replicated grid with variable persistence levels. This capability enables you to choose the persistence level actually required and then achieve it rather than be boxed into a single level. The grid can be synchronously or asynchronously replicated to one or more replicas, whether the replicas are in the same geographic zone or a remote zone. Applications running in backup sites can view replication data streams in from the primaries and react to this data — something that is impossible or very difficult to do given performance constraints using conventional storage access network (SAN) or database replication. Also, application queries can be routed to replicas rather than the primary database automatically, enabling better scalability at the cost of some data staleness.

This flexibility makes the ObjectGrid compatible with many types of application data:

- *HTTP sessions*
- *Conversational state used by an application (building an order up before it's actually purchased)*
- *Reference data*
- *Intermediate data, such as lists of unmatched buy or sell orders on a stock exchange*
- *Real-time derived data*

Data can be pulled or preloaded into an ObjectGrid farm, and then the ObjectGrid can receive change events and calculate derivative data in real time. Figure 9 represents an example topology.

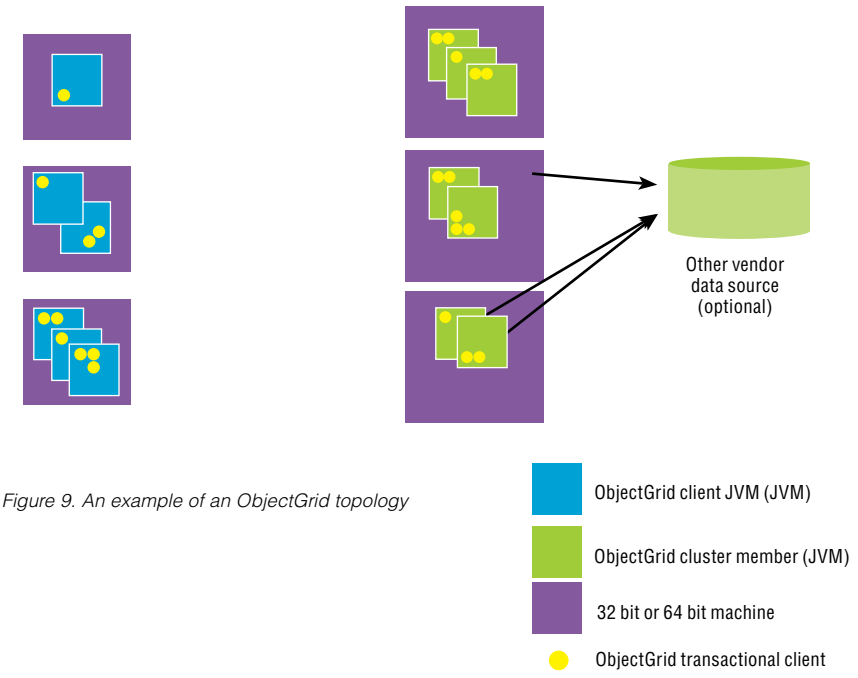

Figure 9. An example of an ObjectGrid topology

Business flexibility is key to supporting real IT business environments

WebSphere Application Server and J2EE technology have traditionally focused on transactional applications. These applications are usually built to handle large volumes of relatively lightweight requests. Many other types of business applications can benefit greatly from the managed environment provided by WebSphere Application Server, but these applications cannot be forced into the transactional application paradigm.

The term *business flexibility* is used to refer to an infrastructure that integrates long-running and transactional work. The business flexibility delivered in WebSphere Extended Deployment, Version 6.0.2 extends the WebSphere platform to support long-running applications and enables you to effectively deploy these applications alongside transactional applications on a single infrastructure. With this release, WebSphere Extended Deployment supports two styles of long-running workloads: compute-intensive and transactional batch. The primary distinction between the two styles lies in the programming model provided to the application by the underlying application container.

Compute-intensive application support

A compute-intensive application expresses its units of work using an extension to the CommonJ WorkManager programming model. Essentially, a compute-intensive job is a Java application that receives a thread of implementation and is expected to perform long-running processor- or data-intensive processing. Compute-intensive applications are fairly free-form in their programming structure and have full access to all of the facilities of WebSphere Application Server and J2EE technology.

Compute-intensive applications are packaged as ordinary J2EE applications. During deployment, the application is flagged as a long-running application rather than a normal J2EE application. This flag triggers the special processing by the long-running placement and balancer components.

Batch-application support

A *batch application* is a transactional record-oriented application that processes a large volume of data. Batch jobs differ from compute-intensive jobs in that they have a more regular structure and therefore can benefit from more container management.

Most batch jobs implement a batch loop that obtains input data from one or more input streams and drives business logic based on the input data. The batch job optionally produces data that is put into an output stream that can be processed by other batch jobs. The batch programs within WebSphere Extended Deployment implement this batch loop using a set of interfaces that are provided by this batch support. Figure 10 depicts the flow of a typical batch job.

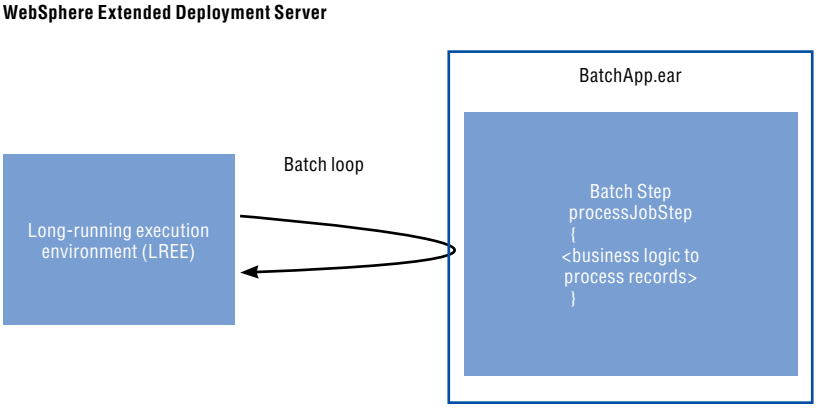

Figure 10. The execution environment invokes the processJobStep() defined on the batch step in a batch loop.

The batch support in WebSphere Extended Deployment, Version 6.0.2 provides an EJB technology-based programming model for creating batch programs. This programming model enables the batch job to reuse existing J2EE artifacts that are also used by a corresponding OLTP application. The batch container in WebSphere Extended Deployment provides all of the transaction management, job management, check pointing and job recovery. The programmer has to deal only with writing the logic to access the data in the data stream and processing logic for a single record within that stream.

Removing the need for redundant resources

Because of the nature of long-running work, it is almost always undesirable to run transactional work and long-running work concurrently on the same processor. The balancer component in WebSphere Extended Deployment helps you avoid this situation by examining the state of the cell and determining when nodes should be switched between transactional and long-running work.

Long-running scheduler

The long-running scheduler receives job submission and management requests. It persists job information in an external job database and is responsible for selecting where and when jobs can run. As jobs are completed, the scheduler updates the job information in the database. The long-running scheduler provides interfaces through the following mechanisms:

- *Stateless session EJB*
- *Web service*
- *Command-line interface (CLI)*
- *Administrative console*

The long-running scheduler is visible primarily to the administrator and operator, for whom it will be the primary mechanism for dealing with long-running work in WebSphere Extended Deployment.

Execution environment

The execution environment is responsible for the mechanics of dispatching and implementing the long-running work on the machine where the work is going to run. This feature is of primary interest to the application developer; the user has limited interaction with this component.

Long-running placement and balancer

The administrator specifies the service policies that govern the placement and balancer functions and, in some cases, approves their decisions. Except for these indirect interactions and seeing the results of their operation, the user is unaware of the placement and balancer functions.

Dynamic placement of long-running work

The long-running placement component makes decisions about where and when to start and stop instances of long-running applications. It is analogous to the placement function currently provided by WebSphere Extended Deployment, Version 5.1 for transactional applications. The long-running placement function considers the current workload on the system and the service policies defined for long-running work to determine where and when long-running applications should be started and stopped. Like the transactional application-placement function, the long-running placement function can operate in manual, supervised or autonomic mode.

Scale-out systems administration

WebSphere Extended Deployment is focused on delivering enhanced quality of service beyond what is available in WebSphere Application Server Network Deployment. Scale-out administration introduces key improvements to WebSphere Extended Deployment systems management by increasing availability of the WebSphere administrative function and improving administrative error correction through configuring the repository checkpoint and restore mechanism.

High-availability support for the deployment manager

Introduced in WebSphere Extended Deployment, Version 6, the high-availability (HA) deployment manager function improves the product's systems-management capabilities by increasing the availability of the WebSphere Application Server administrative function. This capability is important in environments that have significant reliance on automated operations, including application deployment and server monitoring. The HA deployment manager helps eliminate the deployment manager as a single point of failure for cell administration through the use of a hot-standby model for availability.

In an HA deployment-manager environment, two or more deployment managers are configured to exist as peers. One is considered *active*, also known as *primary*, and hosts the cell's administrative function, whereas the others are backups in standby mode. If the active manager fails, then a standby takes over and is designated the new active deployment manager. Each deployment manager is installed and configured to run on a different physical or logical machine. The deployment managers don't have to be hosted on homogenous operating platforms, although similar platforms are recommended. Each deployment manager shares the same instance of the master configuration repository and workspace area, which must be located on a shared file system.

All administrative operations are performed through the elected active deployment manager. The on demand router is configured with the communication end points for the administrative console, the wsadmin command-line client and scripting. The on demand router recognizes which deployment-manager instance is active and routes all administrative communication to that instance. The standby deployment manager is fully initialized and ready to do work but cannot be used for administration because the administrative function isn't designated to support multiple concurrent server processes writing to the same configuration. Therefore, the standby rejects any login and JMX requests. However, if the active deployment manager is stopped or fails, the HA deployment-manager component recognizes the loss of the active deployment manager and dynamically switches the standby into active mode so that it can take over for the lost deployment manager. The actives and standbys share work spaces. When a deployment manager takeover occurs, work is not lost. (See Figure 11.)

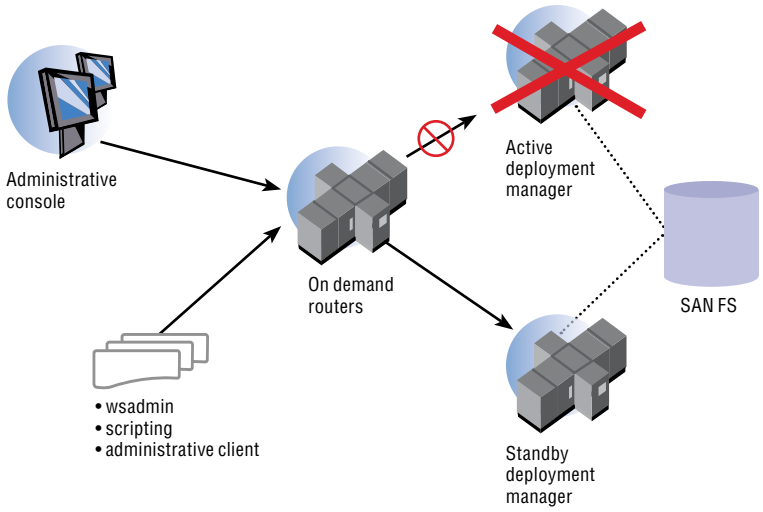

Figure 11. Using an HA deployment manager to prevent a single point of failure

Repository checkpointing and restoration

System administrators recognize that a high percentage of unplanned outages are due to operator errors. A common source of operator error is a bad configuration change. The ability to quickly undo a bad configuration change is critical to minimizing the outage window. The WebSphere Extended Deployment repository checkpoint and restore function enables you to create backup copies of files from the master configuration repository. You can use these backups to restore the configuration to a previous state in case an administrator makes configuration changes that cause operational problems. The benefit of the repository checkpoint and restore function is that it helps reduce recovery time for problems caused by configuration changes.

Full and *delta* are the two checkpoint types supported (see Figure 12). A *full checkpoint* is a complete copy of the entire configuration repository that you can create manually at your discretion. This checkpoint is useful to take a snapshot of a known working configuration to establish a baseline. *Delta checkpoints* are created automatically by the system each time a configuration change is made. As the name implies, a delta checkpoint is not a full copy of the configuration, but rather, is a subset made up of a before-image snapshot of the individual configuration files modified by a discrete configuration change. A configuration save marks the end of a discrete configuration change. You can restore delta checkpoints in the reverse order of their creation to achieve a multilevel undo capability, much like clicking ctrl-z in a word processor.

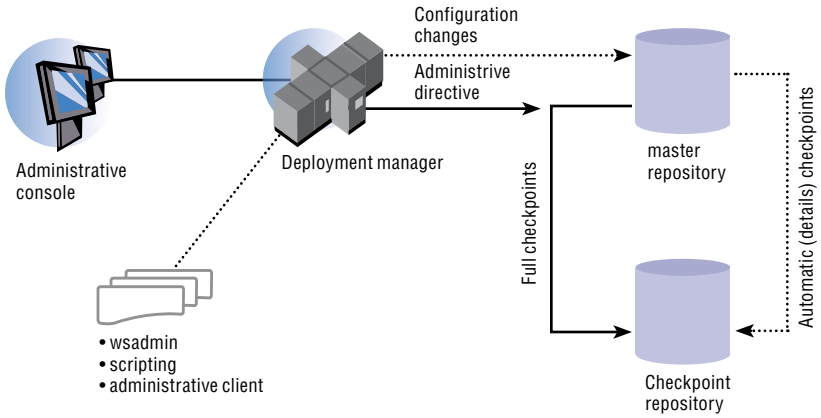

Figure 12. Full and delta checkpointing

Configuration externals are provided to enable or disable automatic (delta) checkpoints. The number of automatic checkpoints saved is also configurable. When the specified limit is reached, the next automatic checkpoint is saved and the oldest discarded. Checkpoints are stored in the file system. You can also configure the location for storing checkpoints, enabling them to be stored on a file system different from the one containing the master configuration repository, to facilitate disaster recovery.

WebSphere Extended Deployment and other stack products

WebSphere Extended Deployment, Version 6.0.2 integrates well with other stack products such as WebSphere Portal,* WebSphere Commerce* and WebSphere Process Server. The benefits that WebSphere Extended Deployment brings to these stack products are standard for any J2EE application – including traffic shaping, dynamic resource allocation by starting and stopping new instances of application servers, health monitoring and visualization enhancements, among others.

WebSphere Portal

You can deploy on demand routers to route traffic dynamically to the appropriate WebSphere Portal nodes based upon available resources. Other autonomic controllers also gather statistics that are useful for intelligent health management and monitoring. IBM WebSphere Portal, Version 5.1 enables you to create and manage virtual portal servers, without having to reinstall WebSphere Portal software over and over again. You can establish these virtual portals by partitioning a single WebSphere Portal installation into independent, logical servers. WebSphere Extended Deployment supports the classification of requests based upon the virtual portals, which enables WebSphere Extended Deployment to provide different qualities of service for different virtual portals.

WebSphere Commerce

WebSphere Extended Deployment can differentiate between incoming HTTP traffic to the enterprise through finely granular classification using URIs, request headers and query parameters, among others. When WebSphere Commerce applications are hosted in a WebSphere Extended Deployment environment, traffic coming into a WebSphere Commerce application can be characterized in this fashion and processed in accordance with its importance and goals associated with its service policies. For example, store IDs in WebSphere Commerce applications are received in the form of query parameters, and hence applications hosting different businesses can be handled with appropriate quality-of-service levels. Also, customers purchasing items from businesses using WebSphere Commerce applications running WebSphere Extended Deployment can be treated with elevated importance compared with customers who that are browsing a catalog, enhancing the user experience and helping you win customer loyalty.

WebSphere Extended Deployment also supports running alongside other stack products such as WebSphere Process Server and WebSphere Enterprise Service Bus. Service policies can be configured in such a composite environment for HTTP, IIOP and JMS traffic entering the system. The WebSphere Extended Deployment product can classify and prioritize requests coming into the system and assign them the appropriate importance level based on the type of traffic it has been configured with. All of the autonomic controllers in WebSphere Extended Deployment can help ensure optimal resource utilization when such a composite system is subject to unpredictable traffic spikes that cause resources to become constrained.

Summary

WebSphere Extended Deployment, Version 6.0.2 includes features that enhance the scalability, usability and adaptability of your WebSphere environment. Robust capabilities facilitate management of larger and more complex deployments, and extend the transaction-volume limits. You can use WebSphere Extended Deployment business flexibility features to gain the autonomic computing capabilities you need to keep pace in today's marketplace – with the potential to deliver lower total cost of ownership (TCO) for your company. Because computing resources are more intelligently managed, you can take advantage of dissimilar usage patterns between different types of work. As a result, autonomic computing capabilities translate to higher levels of adaptability to changing usage patterns.

Also, WebSphere Extended Deployment provides extensive visualization and serviceability graphical interfaces. These tools can offer you a valuable view into WebSphere Extended Deployment to give you confidence as you increase the level of automation of your enterprise. These views can also report events taking place in a WebSphere Extended Deployment topology, and can detail performance and health data. All of these functions are combined in a central control panel to help you keep tabs on your IT investment, to help enhance overall usability.

For more information

To learn more about IBM WebSphere Extended Deployment, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/software/webservers/appserv/extend/

To join the Global WebSphere Community, visit:

www.websphere.org

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
12-06

* The following restrictions apply to various components in WebSphere Extended Deployment for z/OS, Version 6.0.2:

- WebSphere Extended Deployment for z/OS, Version 6.0.2 supports only one servant region (SR) server configuration. If the application server is configured with more than one SR, the WPF will not behave as documented.
- WebSphere partition facility-workload managed (WPF-WLM) IIOP routing is not supported.
- The long-running scheduler can be hosted in only a single servant server.
- The ObjectGrid cannot be hosted in server mode inside a WebSphere Application Server for z/OS server.
- IIOP and JMS work classes are not supported for the z/OS platform.
- Proxy data-source function is not supported on the z/OS platform.
- WebSphere Extended Deployment, Version 6.0.2 supports deployment with WebSphere Portal, Version 5.1.0.4.
- WebSphere Extended Deployment, Version 6.0.2 supports deployment with WebSphere Commerce, Version 5.6.1.1.

WSW11278-USEN-00