On Demand Business solutions
White paper

WebSphere® software

Increasing IT responsiveness and efficiency with IBM WebSphere Extended Deployment.

November 2004

Contents

Introduction

In today's on demand business environment, your organization has to maintain — and continually improve — the quality of service that it provides, while minimizing IT expenditures. You have to respond quickly to customer and market demand. And your IT infrastructure must be able to keep pace as your business needs change — with a minimum of human intervention. By tightening the link between the technologies that run your business and your overall business goals, you can use your IT infrastructure to help you implement your business strategies more efficiently.

Building robust function into your IT infrastructure can help you reduce IT management complexities to make better use of your existing assets. Improve your organization's ability to adapt to changing customer and trading partner demands on the fly — and be prepared for future growth. Incorporating sense-and-respond capabilities into your infrastructure can increase efficiencies and let you shift valuable IT and human resources to higher-value work.

IBM has a long history of providing leading-edge middleware that addresses high availability and reliability — without sacrificing the consistent, predictable performance you need to maintain competitive advantage. IBM WebSphere® Extended Deployment, Version 5.1 delivers on this promise — by adding to the already robust enterprise-class capabilities of IBM WebSphere Application Server Network Deployment, Version 5.1. With its enhanced function, WebSphere Extended Deployment can help you maximize the value to your organization's on demand operating environment.

An on demand operating environment spans integration, virtualization and automation. Robust integration capabilities are a core component of the WebSphere portfolio. IBM clients across the globe use IBM WebSphere Business Integration software from IBM to integrate their back-end data stores and to provide enterprise-services integration using Web services, process choreography and messaging technologies. It is in the virtualization and automation areas that WebSphere Extended Deployment adds its value.

WebSphere Extended Deployment is built on a virtualized infrastructure that extends the traditional concepts of Java™ 2 Platform, Enterprise Edition (J2EE) resources and applications, as well as their relationships with one another. This new infrastructure facilitates the ability of WebSphere Extended Deployment to automate operations in an optimal and repeatable fashion. With its automation capabilities, WebSphere Extended Deployment can help you reduce your total cost of ownership (TCO) and provide a more stable, predictable and reliable operating environment.

This white paper provides a technical 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 leverage WebSphere Extended Deployment software to increase business flexibility and reduce IT complexity.

An IT infrastructure that dynamically and reliably adapts to changing business demands

By extending the capabilities of WebSphere Application Server Network Deployment, WebSphere Extended Deployment can help you optimize the utilization and management of your deployments and enhance the quality of service of your business-critical applications.

As Figure 1 illustrates, WebSphere Extended Deployment capabilities can help you handle the IT scalability and performance challenges of on demand operations. Leveraging the principles and concepts of proven IBM systems and years of IBM research and client experience, WebSphere Extended Deployment enables:

- Dynamic operations
  Allows 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
  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
  Enhances the quality of service of business-critical applications to support near-linear scalability for high-end transaction processing, helping you improve customer-service levels.

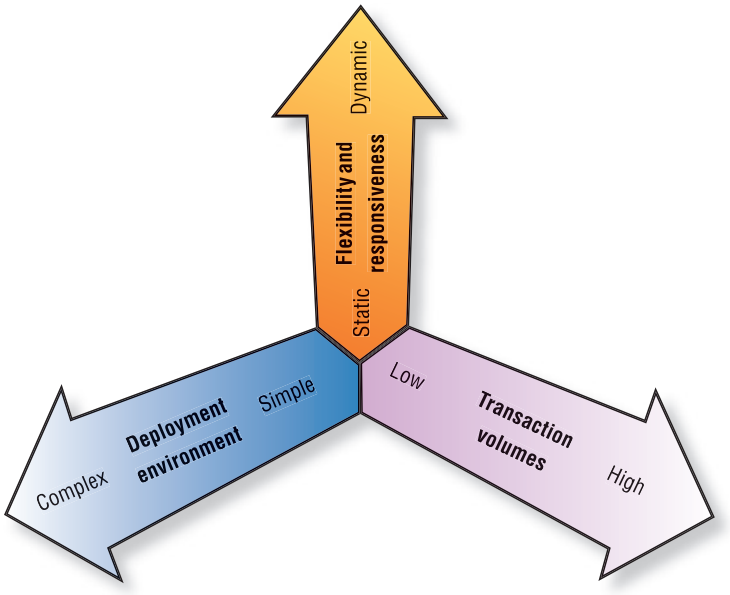

Figure 1. WebSphere Extended Deployment is designed to provide on demand responsiveness, simplified administration and high-performance enhancements.

Dynamic operations capabilities help increase responsiveness and flexibility

WebSphere Extended Deployment is designed to deliver *dynamic operations* through two key capabilities – the virtualization of WebSphere environments and the introduction of a goals-directed infrastructure. A virtualized WebSphere environment allows you to grow your solution as business needs dictate through the dynamic allocation of WebSphere resources. WebSphere Extended Deployment implements a virtualized environment by creating

pools of resources that can be shared among applications, thereby optimizing utilization and simplifying overall deployment. As resources are needed for expected (or unexpected) spikes in workload demand, application resources can be allocated to where they're needed most. This enables better usage of the computing resources that you already own and might allow you to run more applications on the machines that you already have in place.

Share resources among applications to optimize utilization and simplify deployment

WebSphere Extended Deployment redefines the relationship between traditional J2EE constructs. Instead of deploying applications directly onto a server, you can map an application into a resource pool. This application can then be deployed on some subset of servers within that pool according to your configured business goals. The WebSphere Extended Deployment virtualized infrastructure is predicated on two new constructs: *node groups* (which represent resource pools) and *dynamic clusters*.

Node groups

In WebSphere Extended Deployment, the relationship between applications and the nodes on which they can be run is expressed as an intermediate construct called a *node group*. In concrete terms, a node group is nothing more than a set of machines. In a more abstract sense, a node group is a pool of machines with some common set of capabilities and properties (such as connectivity to a given network or the capability to connect to a certain type of database). These characteristics aren't explicitly defined; node group attributes are purely implicit in the WebSphere Extended Deployment design.

Within a node group, one or more dynamic clusters are created. 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.

Because a node group's set of common capabilities and properties is required by some suite of applications, a node group defines the collection of machines able to run a given application. Because the administrator now understands what is implied by participation in a given node group, he or she can ensure that applications are deployed into node groups where they can be accommodated. The resources in a given node group are dynamically allocated according to load and policy to deliver better resource utilization, leading to cost savings. Implementing virtualization using node groups breaks the tie between application clusters and machines, and enables them to be shared among applications, optimizing resource utilization and simplifying overall deployment.

Dynamic clusters

The process of deploying an application in WebSphere Extended Deployment begins with choosing or defining a node group that satisfies the application's requirements, and continues with the creation of a *dynamic cluster*. A dynamic cluster is a container construct that extends its static counterpart in WebSphere Application Server Network Deployment – the *static cluster*. Dynamic clusters are associated with a single node group. Applications (.ear files) are then deployed into a dynamic cluster in much the same fashion as they are deployed into WebSphere Application Server Network Deployment static clusters. However, WebSphere Extended Deployment supports autonomic expansion and contraction of a dynamic cluster within its parent node group. Thus, periodic spikes in demand for an application result in a corresponding increase in that application's resource for processing requests. The strategy for increasing these resources is dictated by operational policies that reflect business goals.

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-operational policies that are tied to business goals. Application performance is optimized according to these policies that reflect service-level goals and relative importance to the organization. Simply put, you can state what applications are important to you,

and these applications will get the highest-priority access to your WebSphere resources at the right time. This can help you ensure, for example, that your business-critical transactions get the best quality of service.

Operational policy is an explicit configurable edict through which operating goals and guidelines are fed into a WebSphere environment. Through operational policy, WebSphere Extended Deployment introduces the capability to designate the business importance of different request types. Among other things, this can help balance your resources optimally in a controlled manner during periods of overutilization. WebSphere Extended Deployment support of operational policy is manifested in two new constructs: *transaction classes* and *service policies*.

Transaction classes and service policies

Transaction classes define some logical request type. They are manifested as lists of Uniform Resource Identifier (URI) filters. For example, administrators can define a transaction class entitled *purchase*, which represents all the buy actions in an enterprise. An administrator assigns a business importance to a transaction class by mapping it into a service policy.

A *service policy* consists of a performance goal and a business-importance assessment. In times of low demand, operational policy is of little interest, because all requests can be serviced immediately. Performance goals become useful and interesting when demand exceeds computing power. When this occurs, service policies of low importance begin to miss their performance goal, while mission-critical requests maintain a high quality of service, thus achieving graceful and controlled degradation. As shown is Figure 2, four service policies have been defined—*platinum*, *gold*, *silver* and *bronze*—with corresponding service levels of *highest*, *high*, *medium* and *low*. The response-time goals for these service policies have also been set based on their relative importance to the organization.

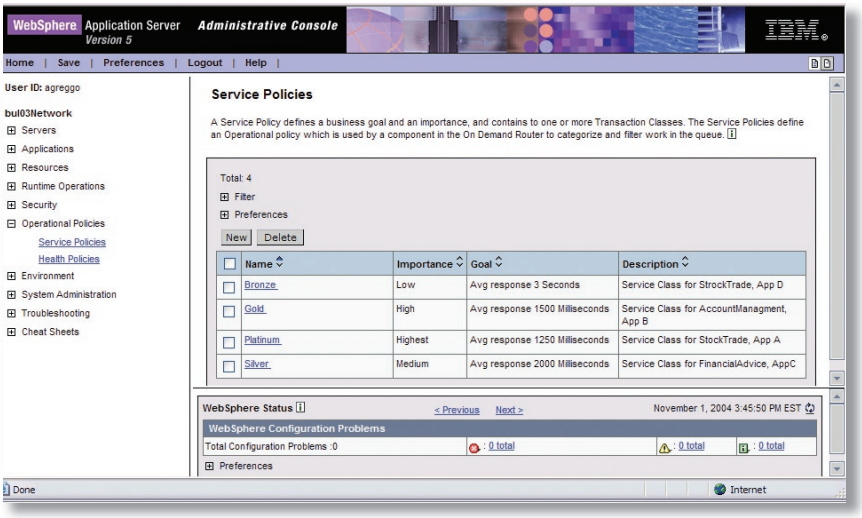

Figure 2. Defining service policies in the administrative console

The categorization of requests into service policies could be accomplished by mapping URIs directly into service policies. However, this would introduce problems in the areas of reporting and visualization, because statistics would only be available in the scope of a service policy or URI. The transaction class offers a convenient way to facilitate reporting statistics, because it represents some logical grouping of requests that is specific to each enterprise. In this respect, WebSphere Extended Deployment offers customizable visualization and reporting of performance statistics. Classification of incoming requests (into transaction classes) is performed in the on demand router, as is queuing of requests according to service policy. Later in this white paper, the section, *The on demand router component*, goes into more detail on these processes.

Quantify performance requirements to help ensure adequate resource provisioning

WebSphere Extended Deployment currently supports two types of performance goals: *average response time* and *discretionary response time*. An average response time goal is fairly self-explanatory. It requires a specified time quantity that represents the highest average response time. When the computed average response time for an application exceeds that of its service policy, autonomic managers take action to mitigate the problem. Discretionary performance goals simply command a best effort from the WebSphere Extended Deployment run time to service incoming requests. No absolute response time goal is associated with this type of performance goal, however.

Performance goals as an early-warning mechanism

Many unfavorable circumstances, including overutilization, constrained computing resources or error conditions, could possibly impact your ability to meet performance goals. When response time increases, actions must be taken to correct the situation. In this respect, WebSphere Extended Deployment operational-policy facilities can act as an early-warning mechanism. If there are problems with reaching performance goals and an administrator knows that incoming traffic is not high enough to cause those problems, he or she is compelled to investigate error conditions, memory leaks or other unfavorable circumstances as potential causes.

Focus on your business strategy—not the technology behind it

Earlier, this white paper introduced the WebSphere Extended Deployment virtualized infrastructure as a basis for its on demand capabilities. This section discusses the capability enabled by that infrastructure: automated enterprise orchestration, or *automation*. Automation brings with it a host of new topological entities, including the on demand router component and several WebSphere Extended Deployment autonomic managers. IBM Tivoli® Intelligent Orchestrator and IBM Tivoli Provisioning Manager (optional, available separately) can be used in enterprise-wide scenarios. The on demand router component controls the flow of work into a WebSphere Extended Deployment topology, the autonomic managers perform autonomic deployment of applications to server instances, and Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager handle the provisioning of resources between resource pools.

The on demand router component

The *on demand router* is a component that sits in the front of the WebSphere Extended Deployment implementation. 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 operational policy. In optimizing queue lengths and dispatch rates, several factors are considered, including request concurrency (per node group), operational policy, service-policy weights and load balancing. Figure 3 illustrates the on demand router architecture.

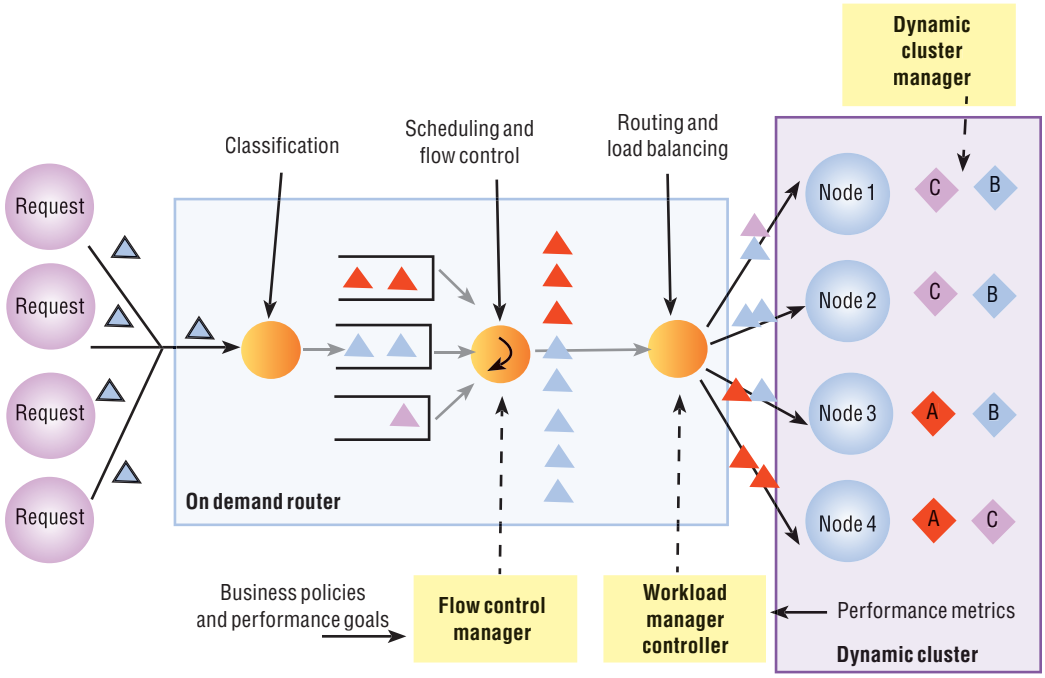

Figure 3. On demand router architecture

Classification of requests

The purpose of the on demand router is to accept incoming requests and to distribute these requests to the WebSphere Extended Deployment back end in an intelligent manner that reflects configured business goals. This process is dependent on the characterization of requests so that each request's relative business importance can be compared. The characterization begins with the classification of requests into a finite set of request types, or transaction classes. As explained in the section of this white paper entitled *Differentiate application service levels according to your business requirements*, transaction classes correspond to a list of URIs. Thus, classification of requests is equal to pattern matching on the set of these URI lists.

Class-based queuing of work

As soon as a request has been mapped to a transaction class, it is linked to that transaction class's service policy. The request is then placed into the queue that corresponds to its service policy; each service policy is assigned a separate queue. An autonomic manager occasionally adjusts dispatching weights of each queue to achieve business goals based on measurements of arrival rates and service times.

Limiting concurrency

To protect against the prospect of an overloaded enterprise, the on demand router limits the number of requests being serviced concurrently in the WebSphere Extended Deployment back end. Concurrency limits are computed based on configured maximum thread-pool sizes. When these limits are reached, requests begin queuing up in the on demand router until demand subsides. If queues reach a (configurable) maximum length, subsequent requests can result in a message returned to the client indicating that the server is too busy to process the incoming request.

Introduce autonomic capabilities into your infrastructure at your own pace

WebSphere Extended Deployment autonomic capabilities are delivered in a set of components known as *autonomic managers*. These components monitor performance and health statistics through a series of sensors, and turn various internal control knobs to optimize system performance. This section takes a look at each of the autonomic managers in a WebSphere Extended Deployment topology.

Flow control manager

The *flow control manager* oversees the dispatch of requests from the set of queues in the on demand router. It uses a weighted round-robin scheduling algorithm. The flow control manager modifies the weights of queues to align flow control with business goals. After work has been dispatched by the flow control manager, it is passed to the dynamic workload manager.

Dynamic workload manager

The *dynamic workload manager* handles load balancing of work across an enterprise back end by maintaining a table of servers to which it is delivering work. In this table, each server is dynamically assigned a weight corresponding to its relative capacity to perform work. In WebSphere Extended Deployment terminology, the dynamic workload manager maintains a list of active server instances for each dynamic cluster, and assigns each a routing weight according to observed performance trends. Requests are then routed to candidate server instances to balance workloads on the nodes within a dynamic cluster.

Dynamic cluster managers

Dynamic cluster managers exist in each node agent process in a WebSphere Extended Deployment topology, with only one instance being active in a node group at any given time. The primary responsibility of this autonomic manager is to control an application's footprint within a node group. Specifically, dynamic cluster managers dictate the location and cardinality of active instances in a given dynamic cluster. As demand increases or decreases, a given dynamic cluster is expanded or contracted, according to operational policy. A dynamic cluster can be deployed on any subset of the nodes in a node group except an empty set.

Scale beyond your defined application server environment

When demand distribution within a WebSphere Extended Deployment enterprise shifts between node groups, the ability to compensate extends beyond the boundaries of WebSphere Extended Deployment autonomic managers. By exploiting Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager, a WebSphere Extended Deployment topology can acquire more physical computing resources from outside a node group.

Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager are IBM's standard tools for provisioning. In the general case, Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager monitor and manage a set of disparate resource pools (such as WebSphere Application Server, IBM WebSphere MQ, SAP, e-mail) to help ensure processing power (in the form of machines, logical partitions [LPARs] or processors) is allocated according to business goals. In the specific application as the WebSphere Extended Deployment provisioning engine, Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager are provided with information that they can use to compute optimum allocations of available resources within the scope of a WebSphere Extended Deployment topology. They then modify distributions of resources among candidate workloads.

Extended manageability helps simplify IT management while maintaining administrator control

It can be difficult to visualize and manage complex IT environments where tens of 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 run-time environment. WebSphere Extended Deployment extends the administration console to allow operators to see at a glance what is happening in their infrastructures and the relative health of the components. It also enhances the existing WebSphere Application Server administrative console by charting application performance against business goals. Alerts that notify you when intervention is required to deliver on business goals help decrease human-intensive monitoring and management.

Stay informed about how your infrastructure is running

To maintain ease-of-use in a product that lends itself to the management of complex deployments in an optionally fully automated fashion, WebSphere Extended Deployment provides enhanced manageability features. These features include a visual console that provides a graphical representation of a dynamic WebSphere Extended Deployment topology, reporting of performance statistics and implementation of administrative operations. These visualization functions are delivered as an extension to the WebSphere Application Server administrative console.

The WebSphere Extended Deployment administrative console contains several tools that help you visualize the inner workings of a WebSphere Extended Deployment topology, so you can remain well-informed about the activities taking place within your environment. Operational views represent an intuitive, central distribution point of information pertaining to health, performance and, potentially, autonomic decisions.

Run-time topology

One such view is the *run-time topology* view, which is a depiction of the momentary state of a WebSphere Extended Deployment environment (see Figure 4). This view refreshes on a configurable interval to provide updated information. The run-time topology contains many useful bits of information, including:

- *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)*

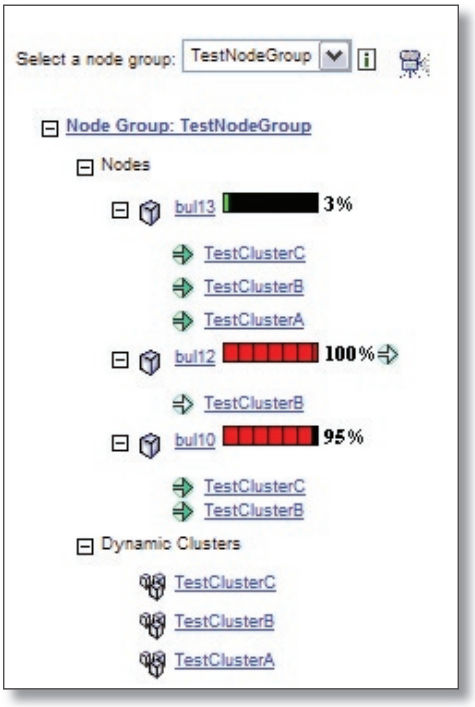

Figure 4. Run-time topology view

Charting

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

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

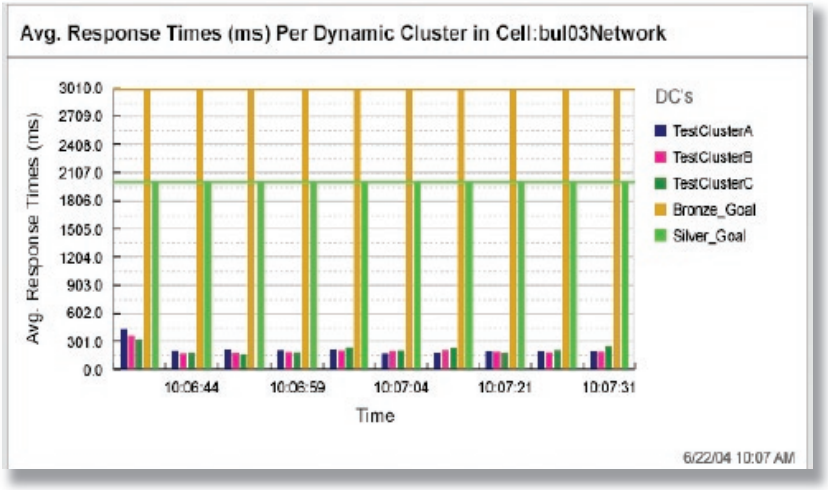

Figure 5. Chart view

Charts can be constructed 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 proxy (on demand router) perspectives.

The WebSphere Extended Deployment charting facility also offers a brief historical log of statistical data. As new data points are added to the right side of a chart, old data is displayed until it scrolls off the left side of the chart. Of course, the length of the history visible on the chart at any point in time depends on the number of data points per sampling period, and the type of chart being viewed.

Run-time map

WebSphere Extended Deployment provides an innovative visualization technique for displaying hierarchical data called a *treemap*. It is simply a rectangle that is recursively subdivided into smaller rectangles, each of which represents the collection of nodes at some level in the tree of data being depicted. The significance of each rectangle's size and color is purpose-specific.

As Figure 6 demonstrates, from a WebSphere Extended Deployment perspective, the top-level rectangle represents a cell. This rectangle is subdivided into rectangles that represent node groups in that cell. Each node group rectangle is subdivided into rectangles representing dynamic clusters, and so on. The size and color of rectangles correlate to the magnitude of some statistic, depending on which treemap is being viewed. Color is typically used to represent health or goal attainment, whereas size typically represents a quantity, such as concurrent requests, number of server instances and so on. By default, the treemap displays the entire WebSphere Extended Deployment topology. However, you can drill down to more fine-grained scopes if you wish by simply clicking on a nonroot rectangle in the treemap.

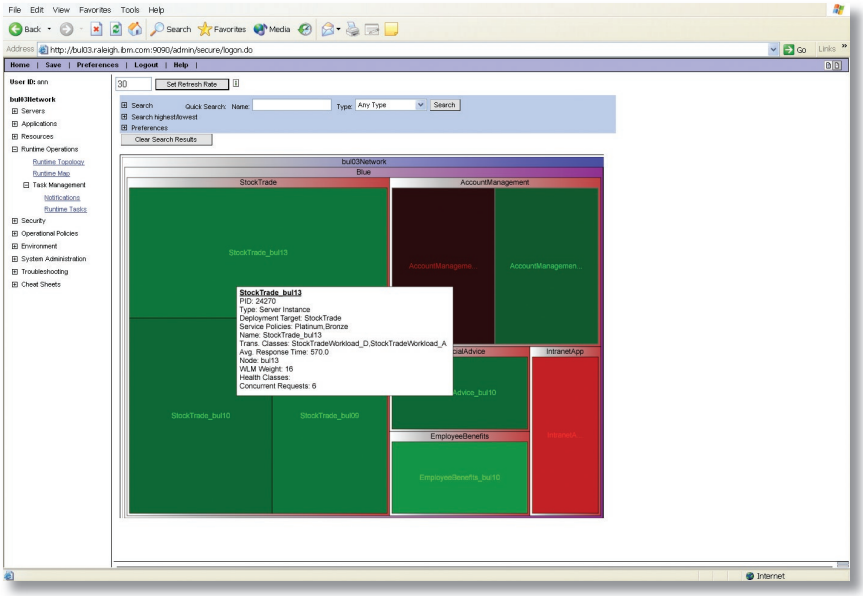

Figure 6. Treemap

The WebSphere Extended Deployment treemap facility can be of particular interest to administrators of very large topologies, because it is well-suited to depicting large sets of data in a concise manner. For instance, the usage of a set of 1000 applications could be quickly observed by viewing a treemap of dynamic clusters and comparing the sizes of rectangles. You can view goal attainment of a set of many service policies by observing the color of rectangles in a map of service policies.

Run-time maps provide a robust search capability, which allows you to single out a subset of the data in an entire map (such as all application server instances in a single node), highlight the top-10-performing applications based on response time, or select the dynamic clusters that have the five lowest, concurrent request values.

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 *on demand* mode, the action plan runs and a simple notification is presented to the administrator. In *supervisory* mode, the administrator is presented with the action plan, and prompted for approval. In *manual* mode, WebSphere Extended Deployment presents a plan, and you can either follow or ignore the advice.

Over time, as the administrator grows more familiar with WebSphere Extended Deployment and its behavior, such decisions and corresponding actions can happen automatically. With the three modes of operation provided by WebSphere Extended Deployment, you can introduce autonomic capabilities into your IT infrastructure in a controlled and gradual way.

Unplanned events

Events that are not assigned an action plan are displayed to the administrator as a warning to let him or her 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.

Get a clear view of the inner workings of your infrastructure components

WebSphere Extended Deployment provides insight into the inner workings of various run-time components. These views facilitate debugging of problems in a WebSphere Extended Deployment environment and give you a glimpse of what is happening in your topology. The current set of proposed views can enable the visualization of:

- *On demand router queue contents*
- *Application placement decisions*
- *Dynamic workload-management routing tables and weights*
- *Statistical data presented in the operational views*
- *WebSphere Extended Deployment internal configuration data*

High-performance computing is 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, workload management and autonomic high-availability management.

The *WebSphere 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 properly 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 WebSphere partition facility—and potentially resulting in significant throughput improvements. The following section describes the partitioning pattern and how WebSphere Extended Deployment enables this pattern through the WebSphere partition facility.

Partitioning pattern

The *partitioning pattern* addresses bottlenecks that occur in high-volume online transaction processing (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.

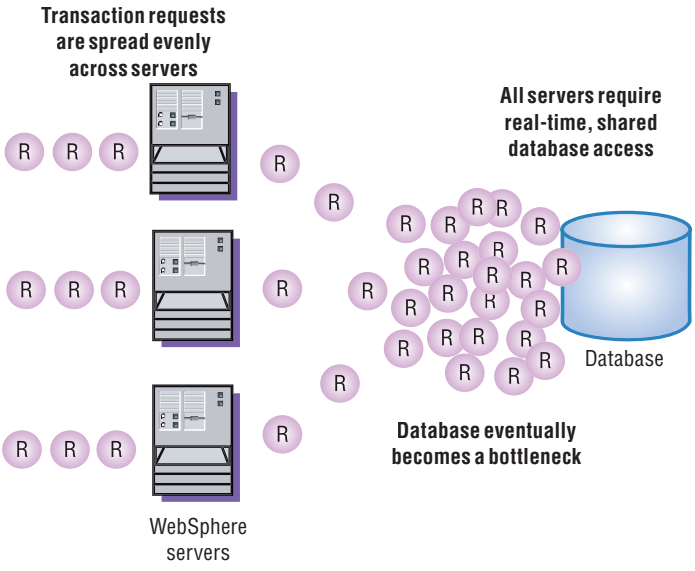

Figure 7. A conventional distributed environment

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 five 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	• WebSphere partitioning facility
Partition-aware workload management	• WebSphere Extended Deployment through the on demand router (for HTTP) • Enterprise JavaBeans (EJB) client through Internet • Inter-ORB Protocol (IIOP) routing • Java Message Service (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 partitioning facility management bean

Table 1. Five key elements to the partitioning pattern

The following section describes the elements of the partitioning pattern and how they can be used in WebSphere Extended Deployment to achieve significant improvement in J2EE application performance and scalability.

Partitioning

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 these benefits can be achieved. The WebSphere partition facility lets you create and manage partitions in WebSphere Extended Deployment. A WebSphere partition facility partition can be described in several ways. In its simplest form, a *partition* is a list of labels that can be created by applications (or metadata found in declarations within applications) 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).

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

- `/stockapp/stocks/ibm/*`
- `/stockapp/stocks/xyz/*`
- `/stockapp/stocks/abc/*`

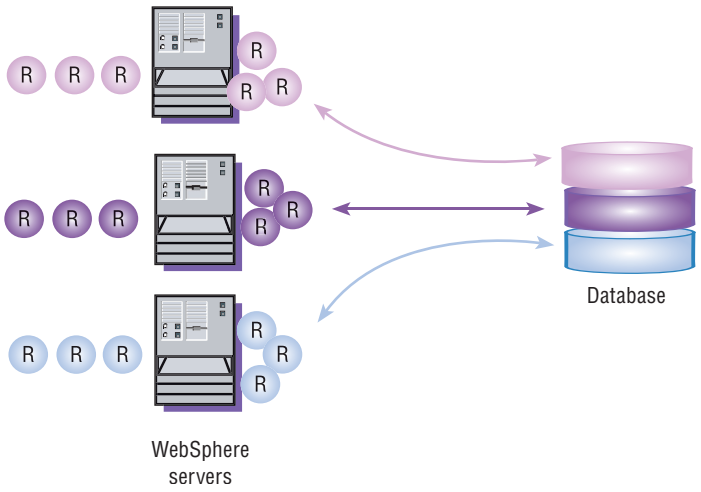

Figure 8. Application partitioning using the WebSphere partition facility

Application partitioning using the WebSphere partition facility

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 WebSphere partition facility component assigns each partition to a server process in the WebSphere Application Server cluster.

Partitions then are the foundation of the partitioning pattern. The next section describes what to do with the partitions.

Partition-aware workload management

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 partition 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 that each is associated with. 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 WebSphere 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 WebSphere partition facility, the affinity process can be controlled in a more precise manner.

- JMS pull model
  Although the WebSphere 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 only subscribe 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 very powerful capability if it is properly leveraged. An application can leverage these capabilities to achieve a new level of performance and scalability. 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 exploiting partitions are *caching*, *batching* and *singleton services*.

- Caching
  An application can leverage 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. Partitioning can be used 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.

By leveraging partitions (such as caching and batching), there is a potential for significant performance improvements in applications that are typically difficult to scale (such as applications with high write-to-read ratios).

Recover from failures quickly

Leveraging partitions can significantly improve performance. However, a partition could represent a single point of failure if preventative steps aren't taken. For example, in the case of leveraging 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 WebSphere partition facility is implemented as a highly available service.

The WebSphere 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. The WebSphere 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 ensure that a system is running optimally, you sometimes have to move partitions to rebalance the load on servers. For example, if the sporting goods category was experiencing a high volume of requests, it might be useful to move it to a more capable server. The WebSphere 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, are typically something that can be planned and done under program or operator control.

Summary

WebSphere Extended Deployment includes add-on features that extend WebSphere Application Server Network Deployment in the areas of scalability, usability and adaptability. These extended capabilities facilitate management of larger and more complex deployments, as well as extending the current limits of transaction volume. You can exploit the new WebSphere Extended Deployment virtualized environment to gain the autonomic computing capabilities you need to carry you into the on demand era — potentially translating to lower 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 translates to higher levels of adaptability to changing usage patterns.

To help ease your organization into the frontier of on demand business, 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, thereby enhancing 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/

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

Produced in the United States of America
11-04

G224-9126-00