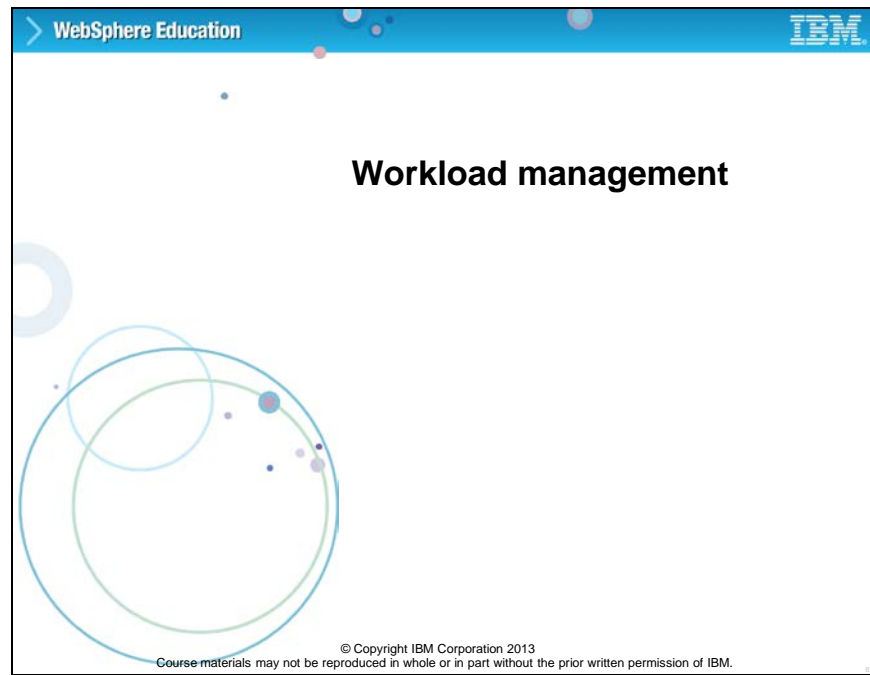
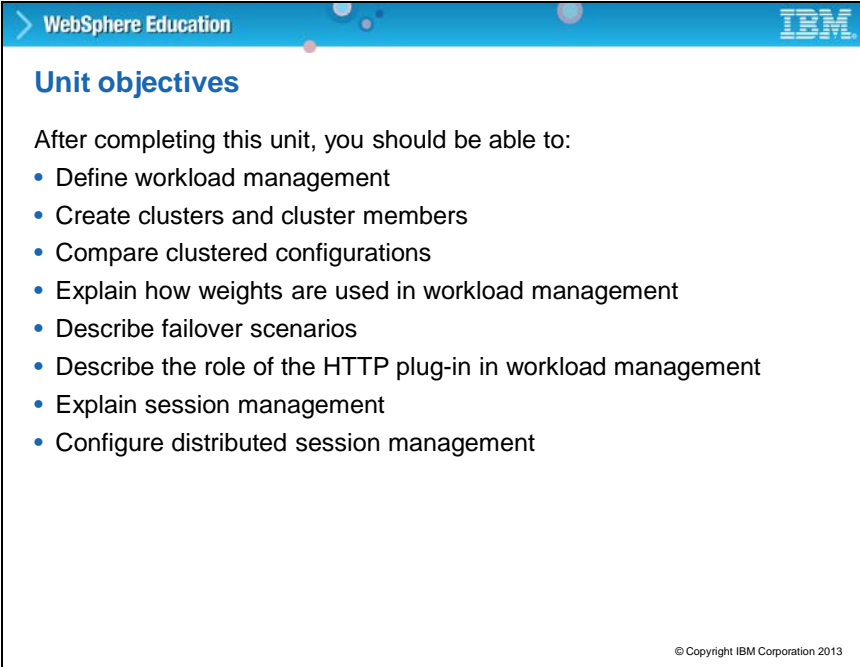


Slide 1



Workload management. In this unit, you learn about workload management concepts in a Network Deployment configuration.

Slide 2



The slide is titled 'Unit objectives' and is part of a 'WebSphere Education' presentation, as indicated by the header. It lists eight objectives for the unit. The IBM logo is visible in the top right corner of the slide frame. At the bottom right, there is a small copyright notice: '© Copyright IBM Corporation 2013'.

Unit objectives

After completing this unit, you should be able to:

- Define workload management
- Create clusters and cluster members
- Compare clustered configurations
- Explain how weights are used in workload management
- Describe failover scenarios
- Describe the role of the HTTP plug-in in workload management
- Explain session management
- Configure distributed session management

© Copyright IBM Corporation 2013

After completing this unit, you should be able to:

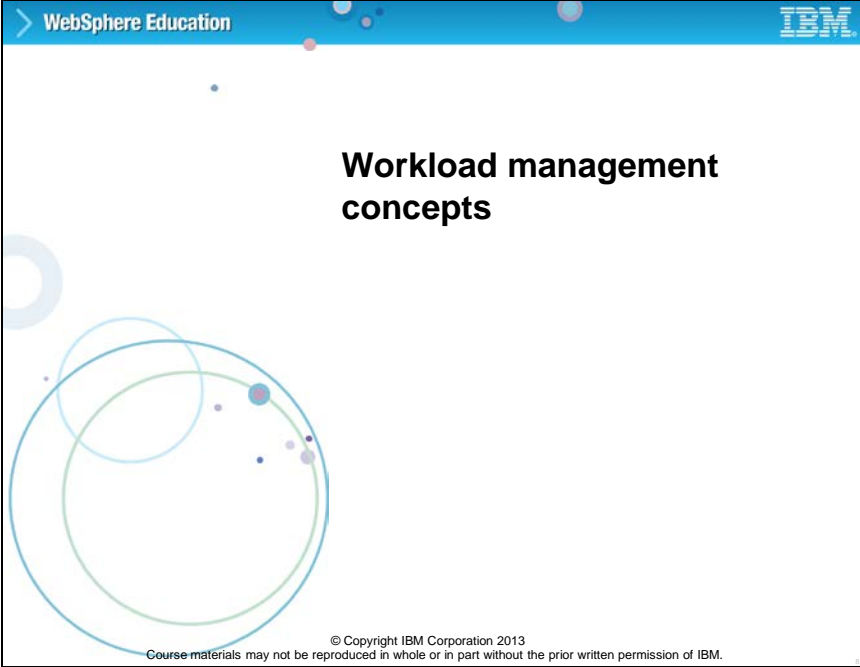
- Define workload management
- Create clusters and cluster members
- Compare clustered configurations
- Explain how weights are used in workload management
- Describe failover scenarios
- Describe the role of the HTTP plug-in in workload management
- Explain session management
- Configure distributed session management

Slide 3

The slide is titled "WebSphere Education" in the top left corner and features the IBM logo in the top right corner. The main content is under the heading "Topics" and lists five bullet points: "• Workload management concepts", "• Clusters and cluster members", "• Routing concepts and session affinity", "• Failover", and "• Session persistence". A small copyright notice "© Copyright IBM Corporation 2013" is located in the bottom right corner of the slide.

This unit is divided into five topics.

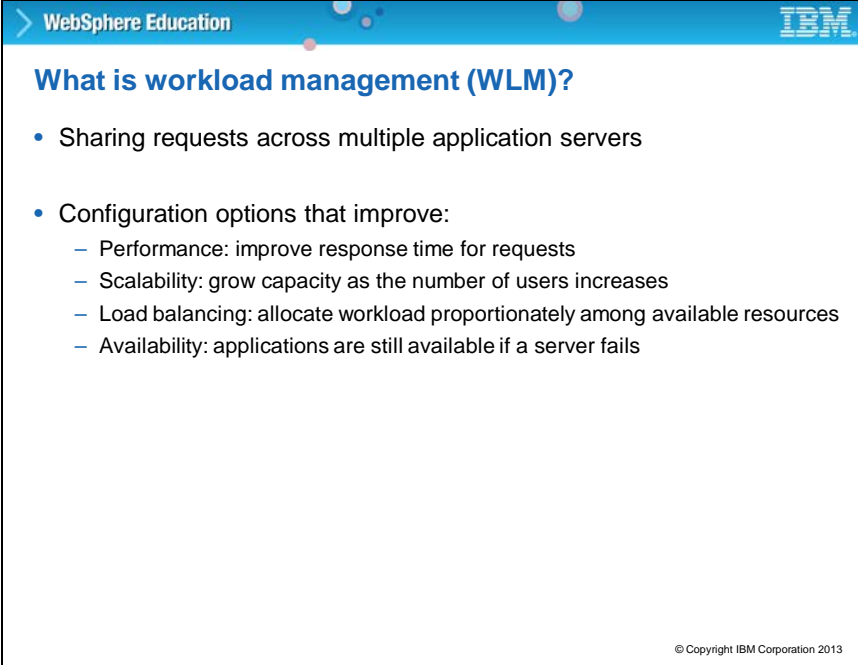
Slide 4



The slide features a blue header bar with the text 'WebSphere Education' on the left and the 'IBM' logo on the right. The main title 'Workload management concepts' is centered in a bold, black font. Below the title, there is a decorative graphic consisting of several overlapping circles in light blue and green, with small colored dots scattered around them. At the bottom of the slide, there is a small copyright notice: '© Copyright IBM Corporation 2013. Course materials may not be reproduced in whole or in part without the prior written permission of IBM.'

Topic: Workload management concepts. This topic contains introductory workload management concepts.

Slide 5



The slide is titled "What is workload management (WLM)?" and is part of a "WebSphere Education" presentation. It features a blue header with the IBM logo. The content is a bulleted list describing WLM as sharing requests across multiple application servers and listing configuration options that improve performance, scalability, load balancing, and availability. A copyright notice for IBM Corporation 2013 is at the bottom right.

- Sharing requests across multiple application servers
- Configuration options that improve:
 - Performance: improve response time for requests
 - Scalability: grow capacity as the number of users increases
 - Load balancing: allocate workload proportionately among available resources
 - Availability: applications are still available if a server fails

© Copyright IBM Corporation 2013

There are numerous potential definitions for the term *workload management*. Maintaining high levels of access to information across heterogeneous environments, without compromising a quality user experience, can challenge any organization. Within the context of the WebSphere Application Server, workload management is meant to spread the work between different hosts. This feature can provide for better performance, scalability, load balancing, and availability.

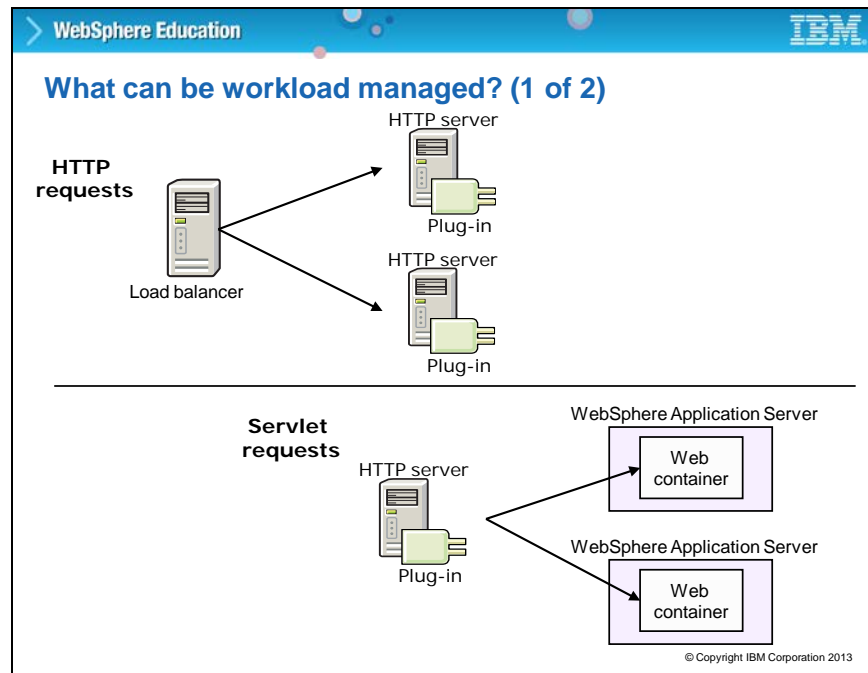
Creating more throughput within a cell cannot be done by installing one application on multiple application servers. This situation can possibly create conflicts with the namespace and the plug-in. Clustering is a fundamental approach for accomplishing high availability. Clusters allow for the same application server to have multiple copies within a single cell. Clustering application servers provides workload management and failover for applications that are installed on the application server cluster.

In addition, other clustering techniques can be used for WebSphere end-to-end system high availability. The WebSphere system can include a database, an LDAP directory server, firewalls, a Caching Proxy, one or more load balancers, HTTP servers, and application servers.

Even if you make all of the processes highly available, the WebSphere system might still fail because of data availability. Without data availability, the WebSphere system cannot do any

meaningful work. Data access is an important part of most applications, especially transactional applications.

Slide 6

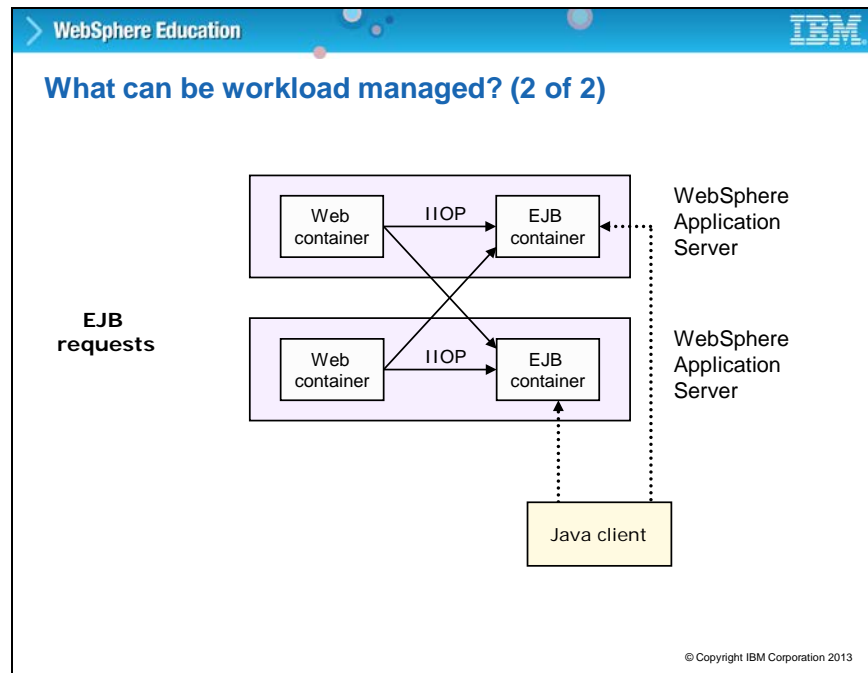


Workload management is also a procedure for improving performance, scalability, and reliability of an application. It provides failover when servers are not available. WebSphere uses workload management to send requests to other members of the cluster.

Within a standard application server topology, there are frequently three points where workload management occurs. The first point is where a load balancer spreads the work among numerous web servers. The load balancer component provides both, scalability and high availability for the web servers. The workload is dispatched to the web servers in the cluster that is based on a load balancing mechanism that is usually known as IP spraying. The load balancer intercepts the HTTP requests and redirects them to the appropriate computer in the cluster, which is based on weights and availability.

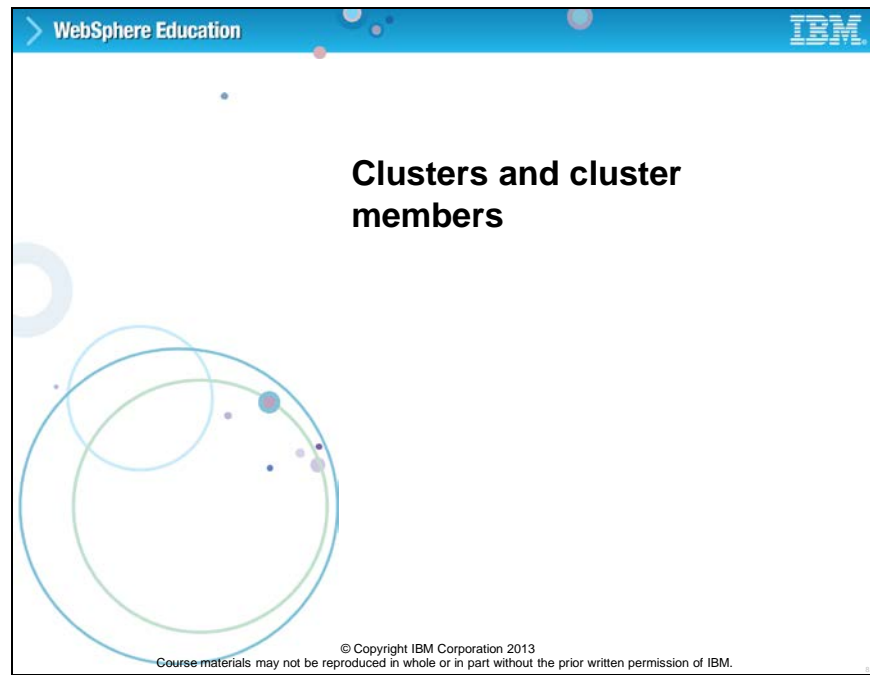
The second point is where the web server plug-in spreads work among numerous application servers, or more specifically, web containers. When an HTTP request reaches the HTTP server, a decision must be made. The HTTP server might handle some requests for static content. Requests for dynamic content or some static content are passed to a web container that is running in an application server. The web server plug-in decides whether the request should be handled or passed to WebSphere.

Slide 7



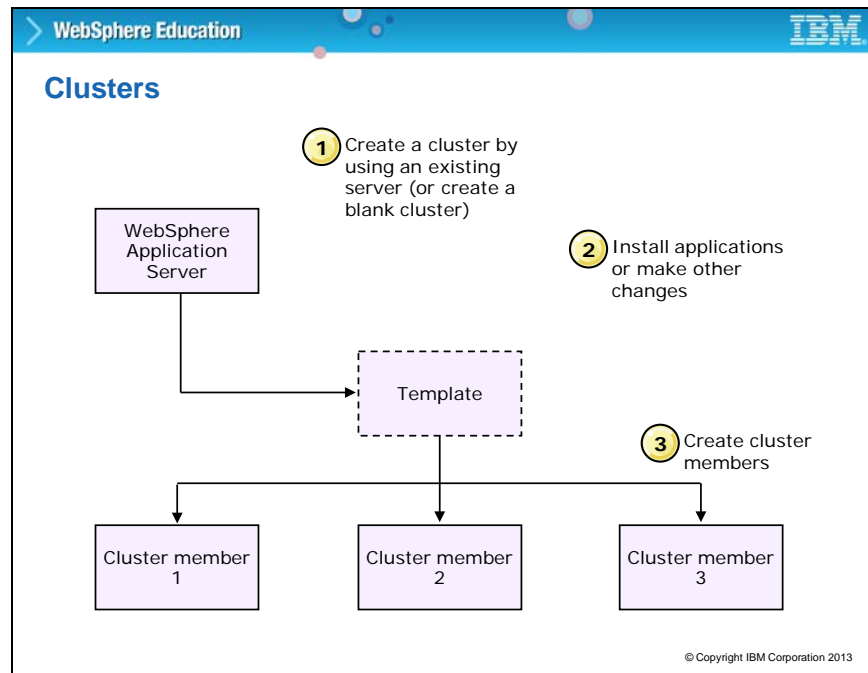
The third point where work can be spread is between the EJB clients and the EJB containers. When an EJB client makes calls from the web container or client container or from outside, the EJB container handles the request in one of the clustered application servers. This third point does not usually occur because the EJB client, in most cases the web container, is typically in the same JVM as the EJB container. That topology forces all traffic to stay within the same JVM. The other cases, which are not so common, occur when a web container is placed in a different JVM, or when a stand-alone Java client is being used.

Slide 8



Topic: Clusters and cluster members. In this topic, you learn how to create clusters and add cluster members.

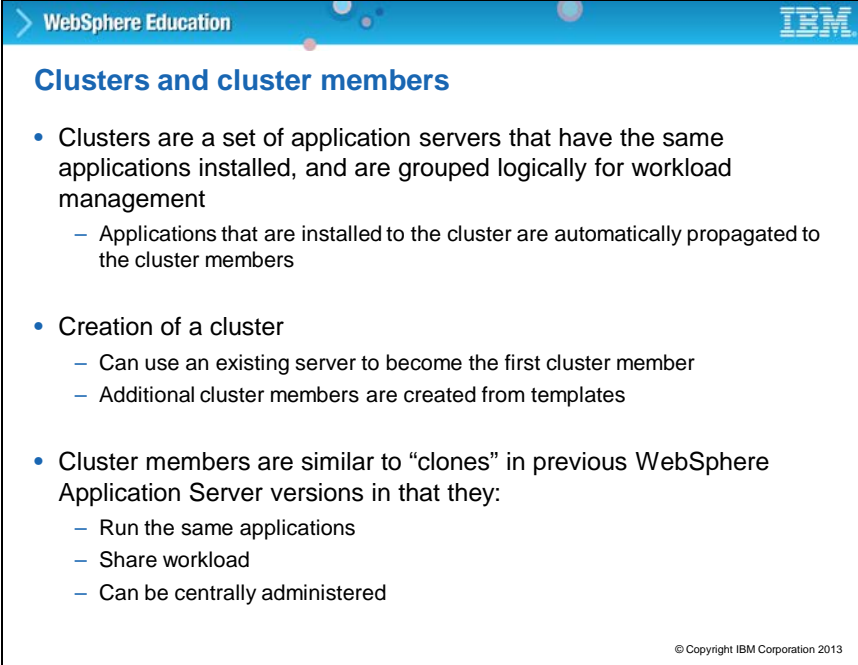
Slide 9



IBM WebSphere Application Server Network Deployment workload management optimizes the distribution of incoming requests between application servers that are able to handle a client request. WebSphere workload management is based on application server clusters that contain multiple application servers, so-called cluster members. A cluster is a set of application servers that are managed together and participate in workload management. Application servers that participate in a cluster can be on the same node or on different nodes. An application that is deployed to a cluster runs on all cluster members concurrently. The workload is distributed based on weights that are assigned to each cluster member. Thus more powerful computers receive more requests than smaller systems.

There are many ways to create a cluster, but the basic idea is illustrated on the slide. You can create a cluster by using an existing server that you already configured, or you can create an empty cluster as a blank template. You can install applications or change this template, and make copies from it, which in turn become new cluster members. There is also the possibility of adding more members to an existing cluster later on.

Slide 10



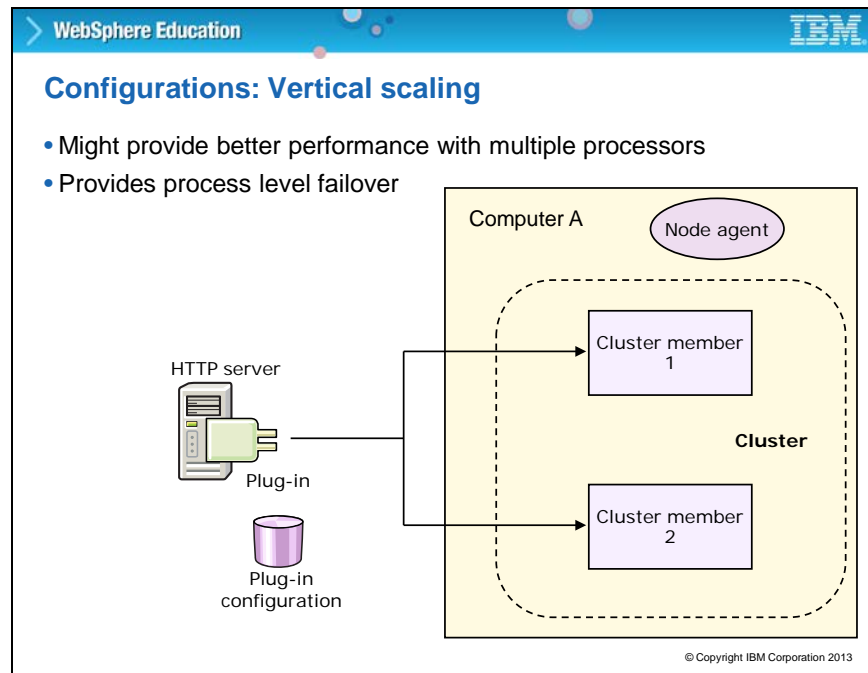
The slide is titled "Clusters and cluster members" and is part of a WebSphere Education presentation. It contains a bulleted list of information about clusters. The IBM logo is in the top right corner, and the copyright notice "© Copyright IBM Corporation 2013" is in the bottom right corner.

- Clusters are a set of application servers that have the same applications installed, and are grouped logically for workload management
 - Applications that are installed to the cluster are automatically propagated to the cluster members
- Creation of a cluster
 - Can use an existing server to become the first cluster member
 - Additional cluster members are created from templates
- Cluster members are similar to “clones” in previous WebSphere Application Server versions in that they:
 - Run the same applications
 - Share workload
 - Can be centrally administered

A cluster is a set of application servers, all of which have the same applications that are installed, and are grouped logically for workload management. Members of the same cluster are essentially replicas of one another.

You can create a blank cluster and then install applications on it, or create a cluster that is based on an existing application server. At any point, cluster members can be added to it. Any additional applications or configuration changes can be made to the cluster, and those changes are pushed out to all of the cluster members.

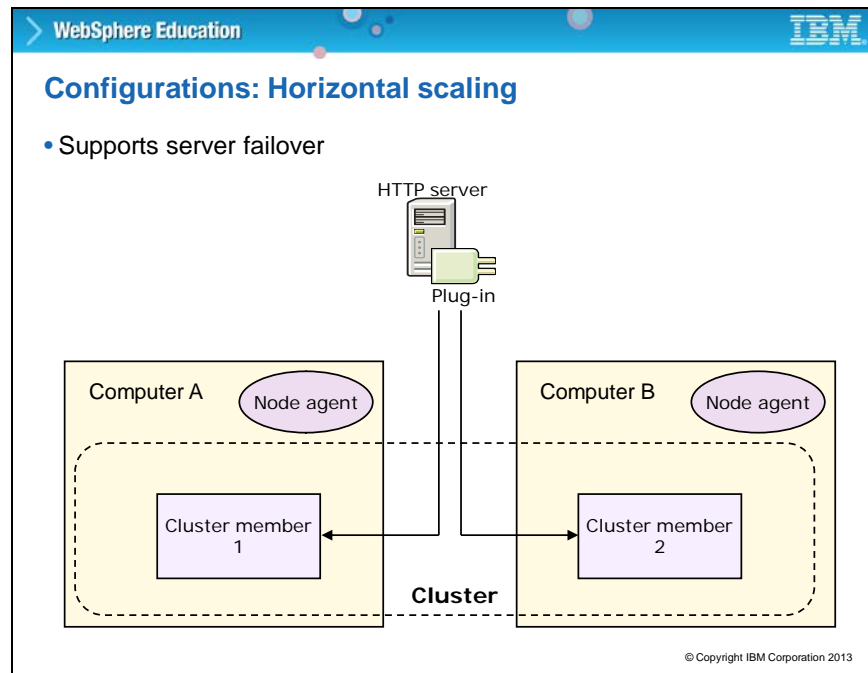
Cluster members are required to have identical application components, but they can be sized differently in terms of weight, heap size, and other environmental factors. You must be careful though not to change anything that might result in different application behavior on each cluster member.



Clustering is an effective way to configure vertical and horizontal scaling of application servers. Vertical scaling, which is pictured on the slide, is when you put multiple cluster members on the same computer. Vertical scaling might allow the processing power of the system to be more efficiently allocated. Even if a single JVM can fully use the processing power of the computer, you might still want to have more than one cluster member on the computer. For other reasons, such as using vertical clustering for process availability.

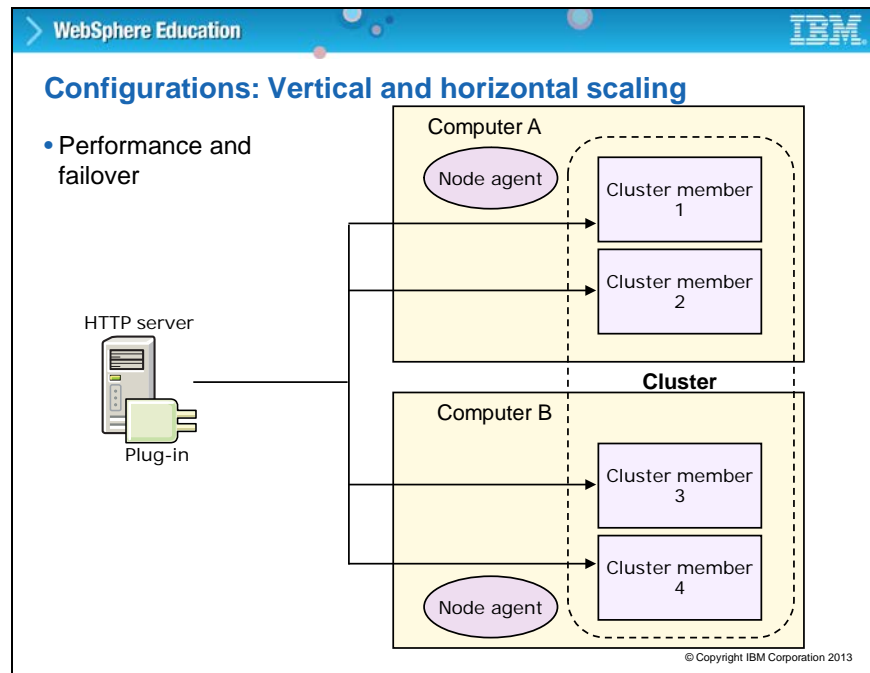
This model might provide better performance on a computer with multiple processors and provides failover at the process level. However, it does not increase throughput when the computer resources are used fully.

Slide 12



Horizontal scaling is when cluster members are placed on different computers. In other words, the cluster spans multiple computers. Horizontal scaling allows a single WebSphere application to run on several computers while still presenting a single system image, making the most effective use of the resources of a distributed computing environment. Horizontal scaling is especially effective in environments that contain many smaller, less powerful computers. Client requests that overwhelm a single computer can be distributed over several computers in the system.

Failover is another important benefit of horizontal scaling. If a computer is unavailable, its workload can be routed to other computers that contain cluster members.



By combining both vertical and horizontal scaling, you can optimize performance and support failover at multiple levels.

Slide 14

WebSphere Education **IBM**

Creating a cluster (1 of 4)

- In console, select **Servers > Clusters > WebSphere application server clusters** and click **New**
- Enter cluster name
- Check options
 - Prefer local
 - Configure HTTP session memory-to-memory replication
- Click **Next**

Create a new cluster

Create a new cluster

→ **Step 1: Enter basic cluster information**

Step 2: Create first cluster member

Step 3: Create additional cluster members

Step 4: Summary

Enter basic cluster information

* Cluster name
PlantsCluster

☒ Prefer local. Specifies whether enterprise bean requests will be routed to the node on which the client resides when possible.

☒ Configure HTTP session memory-to-memory replication

Next **Cancel**

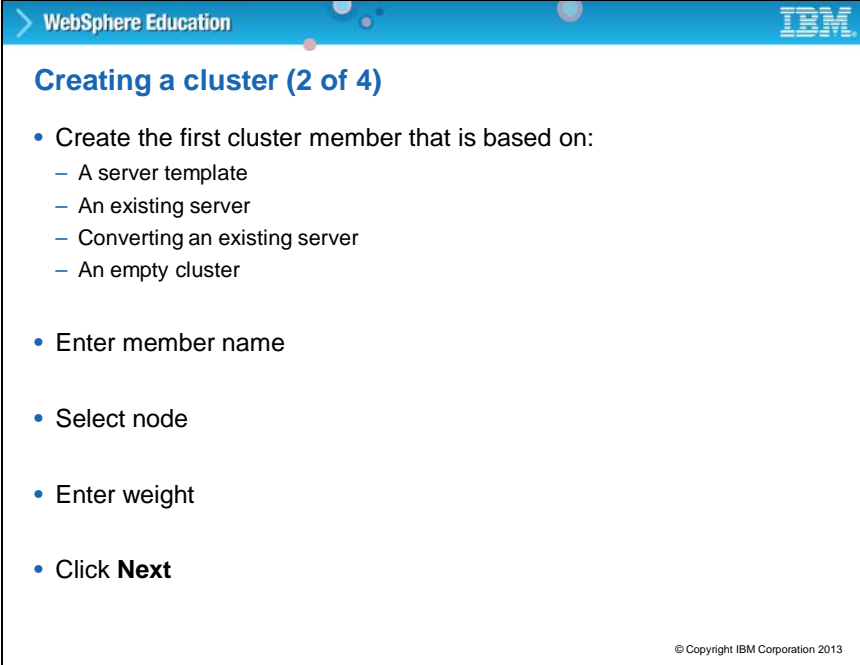
© Copyright IBM Corporation 2013

You can create clusters by using the administrative console. Select **Clusters > WebSphere application server clusters > New**. Enter the cluster name and check the appropriate options. You can choose the “prefer local” option. This setting indicates that a request to an EJB should be routed to an EJB on the local node if available. Prefer local is the default setting and generally results in better performance.

You can choose to configure HTTP session memory-to-memory replication. WebSphere Application Server supports session replication to another application server instance. In this mode, sessions can replicate to one or more application server instances to address HTTP session single point of failure. When you create a cluster, you can choose to create a replication domain for the cluster. The replication domain is given the same name as the cluster and is configured with the default settings for a replication domain. When the default settings are in effect, a single replica is created for each piece of data and encryption is disabled. Also, the web container for each cluster member is configured for memory-to-memory replication.

If you do not configure HTTP session memory-to-memory replication now, you can configure it later.

Slide 15



The slide is titled "WebSphere Education" and "IBM". The main heading is "Creating a cluster (2 of 4)". The content is a bulleted list of steps for creating a cluster member.

- Create the first cluster member that is based on:
 - A server template
 - An existing server
 - Converting an existing server
 - An empty cluster
- Enter member name
- Select node
- Enter weight
- Click **Next**

© Copyright IBM Corporation 2013

When creating a cluster, several options can be specified. These options include what to base the new cluster on, for example, a template, an existing application server, or converting an existing application server. You must also specify the cluster member name, the weight, the node, and other parameters.

WebSphere Education **IBM**

Creating a cluster (3 of 4)

→ **Step 2: Create first cluster member**
 Step 3: Create additional cluster members
 Step 4: Summary

The first cluster member determines the server settings for the cluster members. A server configuration template is created from the first member and stored as part of the cluster data. Additional cluster members are copied from this template.

* Member name

Select node

* Weight (0..100)

☒ Generate unique HTTP ports

Select how the server resources are promoted in the cluster.

Select basis for first cluster member:

- ☐ Create the member using an application server template.
- ☐ Create the member using an existing application server as a template.
- ☒ Create the member by converting an existing application server.
- ☐ None. Create an empty cluster.

© Copyright IBM Corporation 2013

When creating a cluster, you have a few options on the first cluster member. The first application server added to the cluster acts as a template for subsequent servers. You can create a new cluster member that is based on a template. Or, it is possible to select an existing application server as the template for the cluster without adding that application server into the new cluster. In this case, the chosen application server is used only as a template, and is not affected in any way by the cluster creation. You can create the first cluster member that is based on an existing server or create an empty cluster. After you create the cluster, all other cluster members are then created based on the configuration of the first cluster member.

In this example, the option to create the first cluster member that is based on an existing server is selected.

Slide 17

WebSphere Education
IBM

Creating a cluster (4 of 4)

1. Enter Member name
2. Select node
3. Set weight
4. Check options:
 - Generate unique HTTP ports (default)
5. Click **Add Member**
6. Repeat to create other cluster members
7. Click **Next**
8. Click **Finish** on Summary screen

Create additional cluster members

Enter information about this new cluster member, and click Add Member to add this cluster member to the member list. A server configuration template is created from the first member, and stored as part of the cluster data. Additional cluster members are copied from this template.

* Member name
server2

Select node
was85hostNode02 (ND 8.5.5.0)

* Weight
2 (0..100)

☒ Generate unique HTTP ports

Add Member

Use the Edit function to modify the properties of a cluster member in this list. Use the Delete function to remove a cluster member from this list. You are not allowed to edit or remove the first cluster member.

Edit
Delete

Select	Member name	Nodes	Version	Weight
<input type="checkbox"/>	server1	was85hostNode01	ND 8.5.5.0	2
Total 1				

© Copyright IBM Corporation 2013

After you add the first cluster member, you can create more cluster members to add to the cluster.

In this step, other parameters, such as the cluster member name, are specified. You must select the node on which to create the cluster member. Other options include generate unique HTTP ports. The option generates unique port numbers for every transport that is defined in the source server. The server that is created does not have transports that conflict with the original server or any other servers that are defined on the same node.

You must also indicate a weight for the server. The weight value controls the amount of work that is directed to the application server. If the weight value for this server is greater than the weight values that are assigned to other servers in the cluster, then this server receives a larger share of the workload. The weight value represents a relative proportion of the workload that is assigned to a particular application server. The value can range from 0 to 20.

Next, click **Add Member** to add the member to the cluster. You can continue to add more cluster members. When completed, click **Next**.

Slide 18

WebSphere Education **IBM**

Installing enterprise applications to a cluster

1. Select a cluster as the target

Enterprise Applications > PlantsByWebSphere > Manage Modules

Manage Modules

Specify targets such as application servers or clusters of application servers with which to install the application. Modules can be installed on the same application server or distributed across multiple servers as targets that serve as routers for requests to this application. The requests are then generated, based on the applications that are routed through.

Clusters and servers:

WebSphere:cell=was8host01Cell01,cluster=PlantsCluster
 WebSphere:cell=was8host01Cell01,node=ihsnode,server=webserver01

3. Click Apply

Apply

Remove Update Remove File Export File

Select	Module	URI	Module Type	Server
<input type="checkbox"/>	PlantsByWebSphere	PlantsByWebSphere.war,WEB-INF/web.xml	Web Module	WebSphere:cell=was8host01Cell01,cluster=PlantsCluster

2. Map to web servers, if applicable

OK Cancel

© Copyright IBM Corporation 2013

You can create a cluster from an existing server that already has the applications you want installed on it, in which case any cluster members created would also have those applications. You can also install new applications onto a cluster. Installing an application onto a cluster is similar to installing an application onto a single server, except that the application is installed onto each member of the cluster.

This screen capture displays how to install an application onto a cluster by using the administrative console. You must map the modules to a particular target or targets. In this example, select a cluster and a web server as the target, and then click **Apply**.

Slide 19

WebSphere Education
IBM

Controlling a cluster

- Start the cluster
 - Start starts all servers together
 - Ripplestart starts servers one at a time
- Status
 - Started: all servers are started
 - Partial Start: some servers are started
 - Stopped: all the servers are stopped
 - Partial Stop: some servers are stopped

The screenshot shows the WebSphere Administration Console interface. At the top, there are buttons for 'New...', 'Delete', 'Start', 'Stop', 'Ripplestart', and 'ImmediateStop'. Below these are icons for cluster management. A table lists resources, with 'PlantsCluster' highlighted. The status column shows a red 'X' icon, indicating a problem or error. The total number of resources is 1.

Select	Name	Status
<input type="checkbox"/>	PlantsCluster	

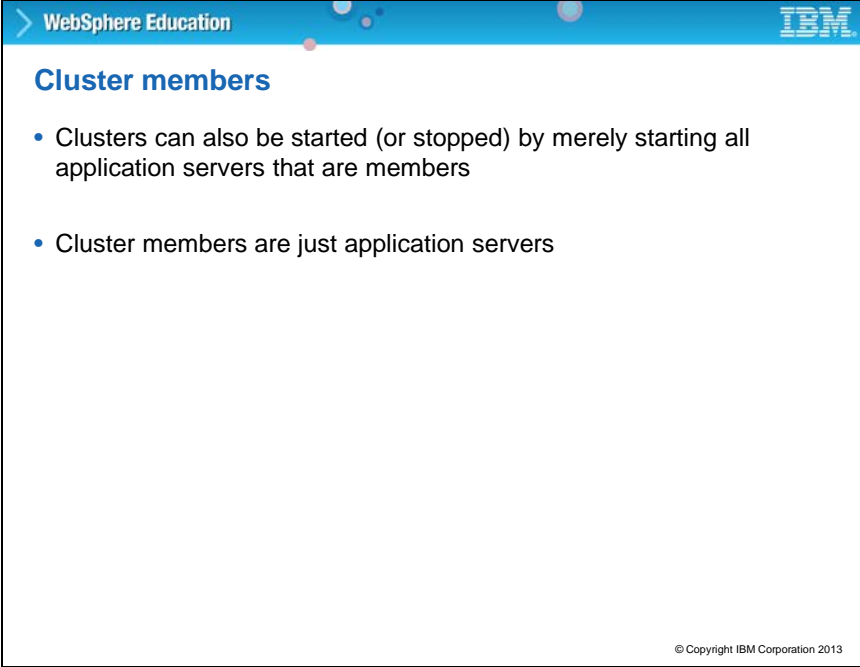
Total 1

© Copyright IBM Corporation 2013

Some special commands are provided for managing a cluster. You can start and stop a cluster much like you start and stop a single server. However, you can choose to ripplestart a cluster. This action causes each member of the cluster to start one at a time. During a ripplestart, a cluster might have a status of being partially started. A partially started icon indicates this status. The cluster can also have a status of partially stopped.

Another option is Immediate Stop, which stops the server, but bypasses the normal server quiesce process, which enables in-flight requests to complete before shutting down the entire server process. This shutdown mode is faster than the normal server stop processing, but some application clients can receive exceptions. Stop quiesces the application server and stops it. In-flight requests are allowed to complete.

Slide 20



The slide is titled "Cluster members" and is part of a "WebSphere Education" presentation. It features a blue header with the IBM logo. The content consists of two bullet points: "Clusters can also be started (or stopped) by merely starting all application servers that are members" and "Cluster members are just application servers". A copyright notice "© Copyright IBM Corporation 2013" is located in the bottom right corner.

> WebSphere Education **IBM**

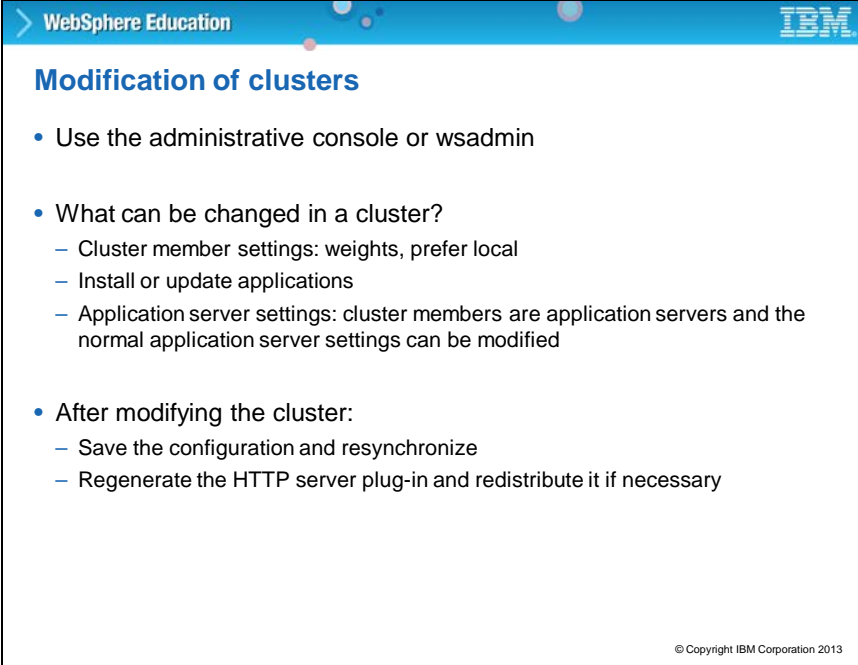
Cluster members

- Clusters can also be started (or stopped) by merely starting all application servers that are members
- Cluster members are just application servers

© Copyright IBM Corporation 2013

Cluster members can be started individually by starting the application servers directly. It is possible to modify some of the individual cluster member configurations, including their weights by using the administrative console.

Slide 21



The slide is titled "Modification of clusters" and is part of a WebSphere Education presentation. It contains a bulleted list of instructions for modifying a cluster. The list includes using the administrative console or wsadmin, identifying what can be changed (cluster member settings, installing/updating applications, and application server settings), and the steps to take after modification (saving configuration and resynchronizing, and regenerating the HTTP server plug-in). The IBM logo is in the top right corner, and a copyright notice is at the bottom right.

- Use the administrative console or wsadmin
- What can be changed in a cluster?
 - Cluster member settings: weights, prefer local
 - Install or update applications
 - Application server settings: cluster members are application servers and the normal application server settings can be modified
- After modifying the cluster:
 - Save the configuration and resynchronize
 - Regenerate the HTTP server plug-in and redistribute it if necessary

© Copyright IBM Corporation 2013

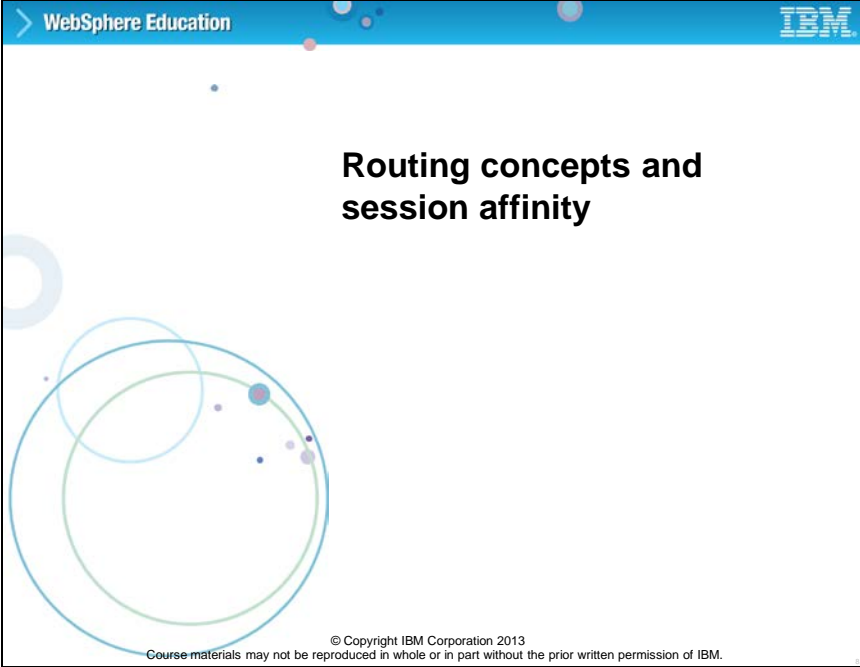
Individual cluster member settings include weight and prefer local. Applications can be installed or updated on a cluster. Changes to application server settings must be made to each cluster member individually. After changing a cluster member, you must save the configuration and resynchronize.

The plug-in configuration file must be regenerated and propagated to the web servers when there are changes to your WebSphere configuration that affect how requests are routed from the web server to the application server. These changes include:

- Installing an application
- Creating or changing a virtual host
- Creating a server
- Modifying HTTP transport settings
- Creating or altering a cluster

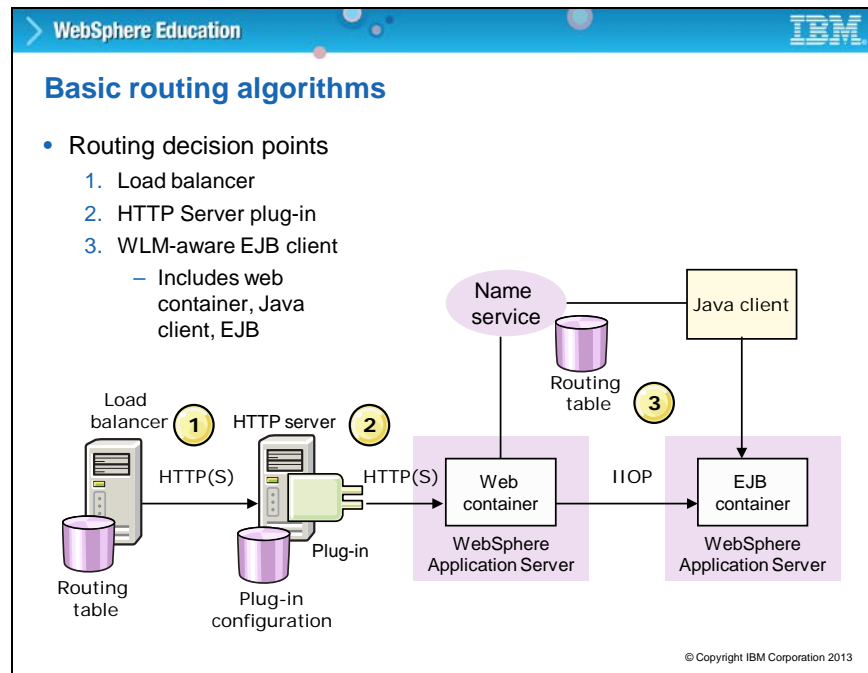
You must also regenerate the HTTP server plug-in and redistribute it if necessary. The plug-in file can be regenerated manually by using the administration tools. You can also set up the plug-in properties of the web server to enable automatic generation of the file whenever a relevant configuration change is made.

Slide 22



The slide features a blue header bar with the text 'WebSphere Education' on the left and the 'IBM' logo on the right. The main title 'Routing concepts and session affinity' is centered in a bold, black font. Below the title, there is a decorative graphic consisting of several overlapping circles in light blue and green, with small colored dots scattered around them. At the bottom of the slide, there is a small copyright notice: '© Copyright IBM Corporation 2013. Course materials may not be reproduced in whole or in part without the prior written permission of IBM.'

Topic: Routing concepts and session affinity. In this topic, you learn about routing concepts and session affinity.



This diagram illustrates a basic routing algorithm with workload management decision points. An IP sprayer component, such as the Edge Components load balancer or a network appliance, can be used to complete load balancing and workload management functions for incoming HTTP requests. This diagram uses the load balancer component. A routing decision table, which the load balancer stores internally, is configured by using the load balancer tool. There are many intelligent routing options available.

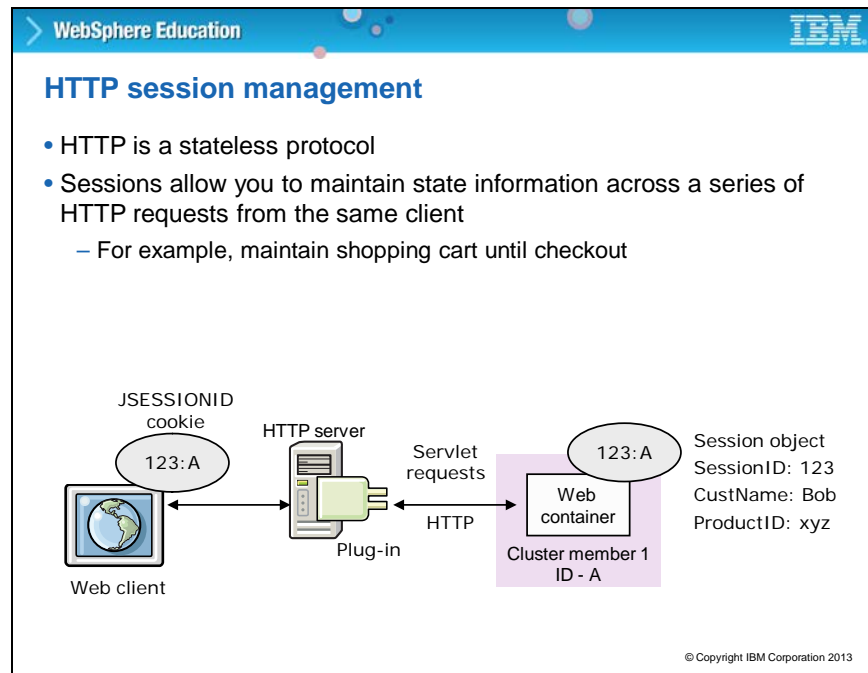
When an HTTP request reaches the HTTP server, a decision must be made. In the HTTP server, the plug-in configuration file defines routing. This file is configured by using the administrative console or wsadmin commands. WebSphere provides the weighted round robin and random load balancing options. With weighted round robin, routing is based on the weight that is associated with the cluster members. If all cluster members have identical weights, the plug-in sends equal requests to all members of the cluster, assuming no strong affinity configurations. If the weights are scaled in the range from 0 to 20, the plug-in routes requests to those cluster members with the higher weight value more often. No requests are sent to cluster members with a weight of zero unless no other servers are available. The default routing algorithm is weighted round robin. With random, the plug-in picks a member of the cluster randomly.

For a workload management-aware application server, the routing table is supplied by the name service, and configured by using the administrative console, or wsadmin commands. In this

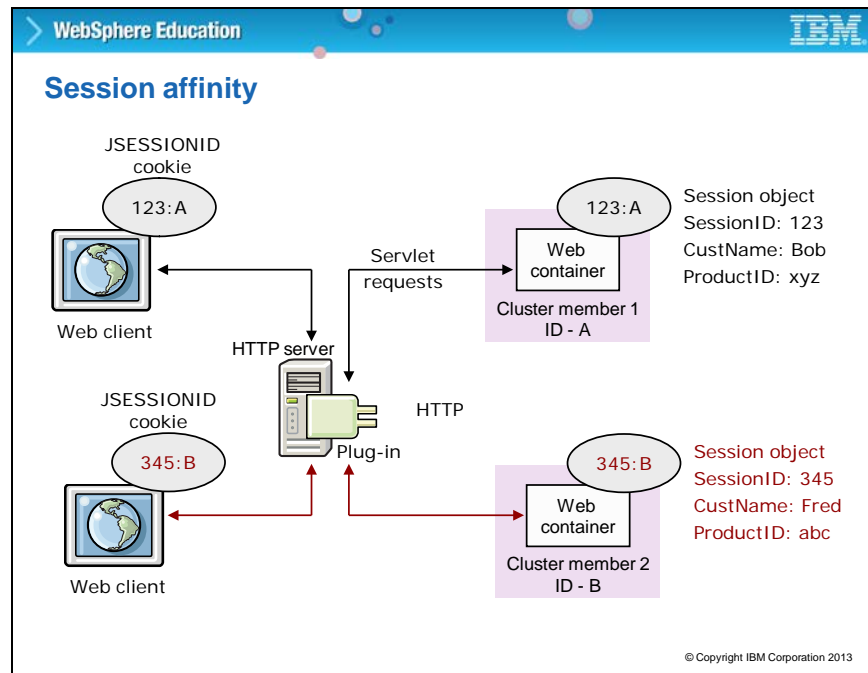
configuration, EJB client requests are routed to available EJB servers in a round-robin fashion that is based on assigned server weights. The EJB clients can be servlets that operate within a web container, stand-alone Java programs by using RMI/IIOP, or other EJB beans. The server weighted round-robin routing policy ensures a distribution that is based on the set of server weights that are assigned to the members of a cluster.

You can also choose that EJB requests are routed preferably to the same host as the host of the requesting EJB client with the prefer local option. In this case, only cluster members on that host are chosen by using the round-robin weighted method. Cluster members on remote host are chosen only if a local server is not available.

Slide 24



HTTP is a stateless protocol. Unless you have only a single application server or your application is stateless, maintaining state between HTTP client requests also plays a factor in determining your configuration. Sessions allow you to maintain state information across a series of HTTP requests from the same client (for example, maintaining your shopping cart until checkout). The Java servlet specification defines a session management process for web applications. The session manager stores session information, and sends the client a unique clone ID and session ID through either a cookie in the HTTP header, or URL rewriting in a parameter on links or forms. Many sites choose cookie support to pass the user's identifier between WebSphere and the user. WebSphere Application Server session support generates a unique session ID for each user and returns this ID to the user's browser with a cookie. The default name for the session management cookie is JSESSIONID.



Session affinity is where the load distribution facility recognizes the existence of a session and attempts to direct all requests within that session to the same server. Affinity requests are requests that contain a JSESSIONID. This diagram shows how the JSESSIONID cookie is used to establish session affinity between a web client and a specific cluster member. A running application on a server can retain state information for a user's session in memory. However, other servers in the cluster might not have this information. It might be necessary to rely on session affinity to ensure that the client requests go to the same application server that holds the associated state information.

Each request that is coming into the web server is passed through the plug-in, which uses its configuration information to determine whether the request should be routed to. The plug-in always attempts to route a request that contains session information to the application server that processed the previous requests. The HTTP server plug-in routes subsequent servlet requests consistently to the same application server after the session is created, by using a clone ID passed with the session ID in a cookie or URL.

JSESSIONID cookie

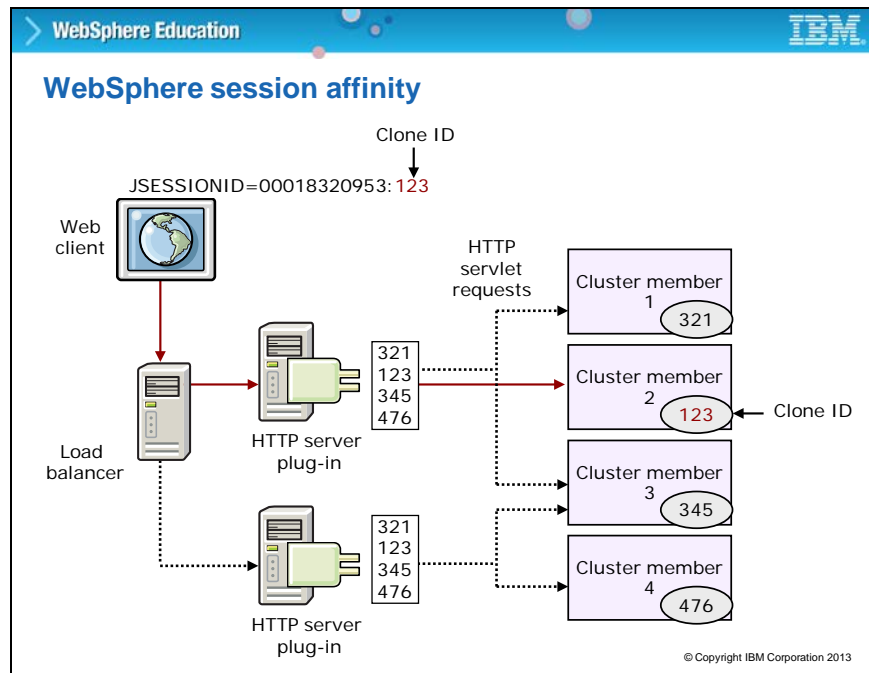
- The JSESSIONID cookie is used to help manage sessions
- The plug-in uses the data to find which clone has affinity
- The web container uses it to find the right http session object

The screenshot shows a 'Cookies' window with a search for 'localhost'. A table lists cookies for 'localhost', including 'JSESSIONID'. The details for the 'JSESSIONID' cookie are shown below the table. The 'Content' field contains '000hmgpKC7Vr7N9BvpfWMSbAz4:F827NN3JR'. Annotations with arrows point to the parts of the cookie value: '000' is labeled 'Epoch number', 'hmgpKC7Vr7N9BvpfWMSbAz4' is labeled 'HTTP session ID', and 'F827NN3JR' is labeled 'Clone ID'.

© Copyright IBM Corporation 2013

The JSESSIONID cookie includes three main parts: the HTTP session ID, the clone ID, and the epoch number. The clone ID allows the plug-in to identify which cluster member holds the session object. The HTTP session object allows the application server to find the object that is specific to that particular request. And finally, the epoch number allows the web container to make sure that the cached HTTP session object is not stale.

The format of the cookie can change in certain cases. For example, if memory-to-memory session replication was used, the format would be entirely different. Also, if the session is failed over to another web container, there can be a list of clone IDs instead of just one.



This diagram shows what happens with WebSphere session affinity. The session cookie is generated. It allows affinity on the first hit by using the clone ID. The plug-in knows how to route the request to the correct clone ID. It can be turned off by removing the clone IDs from plug-in file.

WebSphere Education

Plug-in

- Based on the data in the `plugin-cfg.xml` file, the plug-in is able to route sticky requests to the correct application server
 - If the application server is unavailable, the request is rerouted to the next application server
 - This new application server now has affinity
 - The session information that is held within the web container does not fail over unless session failover is configured
 - An unavailable application server is marked as down for a certain amount of time (default is 2 minutes)
 - After that time, the server is tried again
 - The `plugin-cfg.xml` file is checked for updates every 60 seconds so new application servers are automatically brought into the active list

```

graph LR
    HTTP[HTTP server] --> PlugIn[Plug-in]
    PlugIn --> WS1[WebSphere Application Server]
    PlugIn --> WS2[WebSphere Application Server]
    PlugIn --- Config[(Plug-in configuration)]
            
```

© Copyright IBM Corporation 2013

Based on the data in the plug-in configuration file, the plug-in is able to route sticky requests to the correct application server. If the application server is unavailable, the request is rerouted to the next application server. This new application server now has affinity. The session information that is held within the web container does not failover unless session failover is configured. An unavailable application server is marked as down for a certain amount of time. The default time is 2 minutes. After that time, the server is tried again. The plug-in configuration file is checked for updates every 60 seconds. So new application servers are automatically brought into the active list.

WebSphere Education
IBM

Plugin-cfg.xml (1 of 2)

```

<?xml version="1.0" encoding="ISO-8859-1"?><!-- HTTP server plugin config file for the webserver was7host01Cell01.ihsnode.webserver01 generated
on 2009.02.19 at 01:08:34 AM EST-->
<Config ASDisableNagle="false" AcceptAllContent="false" AppServerPortPreference="WebserverPort" ChunkedResponse="false"
FIPSEnable="false" IISDisableNagle="false" IISPluginPriority="High" IgnoreDNSFailures="false" RefreshInterval="60" ResponseChunkSize="64"
VHostMatchingCompat="false">
  <Log LogLevel="Error" Name="c:\Program Files\IBM\HTTPServer\Plugins\logs\webserver01\http_plugin.log"/>
  <Property Name="ESIEnable" Value="true"/>
  <Property Name="ESIMaxCacheSize" Value="1024"/>
  <Property Name="ESIInvalidationMonitor" Value="false"/>
  <Property Name="ESIEnableToPassCookies" Value="false"/>
  <Property Name="PluginInstallRoot" Value="c:\Program Files\IBM\HTTPServer\Plugins\"/>
  <VirtualHostGroup Name="default_host">
    <VirtualHost Name="*:*:9080"/>
    <VirtualHost Name="*:*:9443"/>
    <VirtualHost Name="*:*:5060"/>
    <VirtualHost Name="*:*:5061"/>
    <VirtualHost Name="*:*:443"/>
    <VirtualHost Name="*:*:9081"/>
    <VirtualHost Name="*:*:9444"/>
  </VirtualHostGroup>
  <ServerCluster CloneSeparatorChange="false" GetDWLMTTable="false" IgnoreAffinityRequests="true" LoadBalance="Round Robin"
  PostSize="64" PostSizeLimit="-1" RemoveSpecialHeaders="true" RetryInterval="60">
    <Server CloneID="13u6hqm8" ConnectTimeout="0" ExtendedHandshake="false" LoadBalanceWeight="2" MaxConnections="-1"
    Name="was7host01Node01_server1" ServerIOTimeout="0" WaitForContinue="false">
      <Transport Hostname="was7host01" Port="9080" Protocol="http">
        <Property Name="keyring" Value="c:\Program Files\IBM\HTTPServer\Plugins\config\webserver01\plugin-key.kdb"/>
        <Property Name="stashfile" Value="c:\Program Files\IBM\HTTPServer\Plugins\config\webserver01\plugin-key.sth"/>
      </Transport>
    </Server>
    <Server CloneID="13u6hr5pv" ConnectTimeout="0" ExtendedHandshake="false" LoadBalanceWeight="2" MaxConnections="-1"
    Name="was7host01Node02_server2" ServerIOTimeout="0" WaitForContinue="false">
      <Transport Hostname="was7host01" Port="9081" Protocol="http">
        <Property Name="keyring" Value="c:\Program Files\IBM\HTTPServer\Plugins\config\webserver01\plugin-key.kdb"/>
        <Property Name="stashfile" Value="c:\Program Files\IBM\HTTPServer\Plugins\config\webserver01\plugin-key.sth"/>
      </Transport>
    </Server>
  </ServerCluster>

```

© Copyright IBM Corporation 2013

This example shows the contents of a plug-in configuration file. The CloneID is highlighted. You examine the plug-in configuration file in detail in the lab exercises.

WebSphere Education
IBM

Plugin-cfg.xml (2 of 2)

```

<Property Name="stashfile" Value="c:\Program Files\IBM\HTTPServer\Plugins\config\webserver01\plugin-key.sth"/>
</Transport>
<Server>
<PrimaryServers>
<Server Name="was7host01Node01_server1"/>
<Server Name="was7host01Node02_server2"/>
</PrimaryServers>
</ServerCluster>
<UriGroup Name="default_host_TradeCluster_URIs">
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/invl"/>
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/snoop"/>
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/hello?"/>
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/hitcount"/>
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/jsp"/>
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/jsw"/>
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/j_security_check"/>
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/ibm_security_logout"/>
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/servlet"/>
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/Trade/web"/>
</UriGroup>
<Route ServerCluster="TradeCluster" UriGroup="default_host_TradeCluster_URIs" VirtualHostGroup="default_host"/>
<RequestMetrics armEnabled="false" loggingEnabled="false" rmEnabled="false" traceLevel="HOPS">
<filters enable="false" type="URI">
<filterValues enable="false" value="/snoop"/>
<filterValues enable="false" value="/hitcount"/>
</filters>
<filters enable="false" type="SOURCE_IP">
<filterValues enable="false" value="255 255 255 255"/>
<filterValues enable="false" value="254 254 254 254"/>
</filters>
<filters enable="false" type="JMS">
<filterValues enable="false" value="destination=aaa"/>
</filters>
<filters enable="false" type="WEB_SERVICES">
<filterValues enable="false" value="wsdlPort=aaa:op=bbb:namespace=ccc"/>
</filters>
</RequestMetrics>
</Config>

```

© Copyright IBM Corporation 2013

The example shows the portion of the contents of a plug-in configuration file that contains JSESSIONID cookie information.

WebSphere Education

IBM

Interpreting the plugin-cfg.xml file (1 of 4)

Find a UriGroup that has a regular expression matching the request

```
<Config ...>
...
<UriGroup Name="default_host_PlantsCluster_URIs">
...
  <Uri Name="/PlantsByWebSphere/*" .../>
...
<Route UriGroup="default_host_PlantsCluster_URIs"
      ServerCluster="PlantsCluster"/>
...
<ServerCluster Name="PlantsCluster" LoadBalance="Round Robin" ... >
  <Server CloneID="17shqbbbrq" LoadBalanceWeight="2" ...>
    <Transport Hostname="was85host" Port="9080" Protocol="http"/>
    <Transport Hostname="was85host" Port="9443" Protocol="https">
      ...
  <Server CloneID="17shqbcf9" LoadBalanceWeight="2" ...>
    <Transport Hostname="was85host" Port="9081" Protocol="http"/>
    <Transport Hostname="was85host" Port="9445" Protocol="https">
      ...
  
```

© Copyright IBM Corporation 2013

To determine where the plug-in can route a request, first find a Uri xml element whose name attribute contains a regular expression that matches the request. Note the UriGroup xml element that the Uri is a part of.

Slide 32

WebSphere Education

IBM

Interpreting the plugin-cfg.xml file (2 of 4)


Find the Route element that maps to the UriGroup name

```
<Config ...>
  ...
  <UriGroup Name="default_host_PlantsCluster_URIs">
    ...
    <Uri Name="/PlantsByWebSphere/*" .../>
    ...
  <Route UriGroup="default_host_PlantsCluster_URIs"
        ServerCluster="PlantsCluster"/>
  ...
  <ServerCluster Name="PlantsCluster" LoadBalance="Round Robin" ... >
    <Server CloneID="17shqbbrq" LoadBalanceWeight="2" ...>
      <Transport Hostname="was85host" Port="9080" Protocol="http"/>
      <Transport Hostname="was85host" Port="9443" Protocol="https">
        ...
      <Server CloneID="17shqbcf9" LoadBalanceWeight="2" ...>
        <Transport Hostname="was85host" Port="9081" Protocol="http"/>
        <Transport Hostname="was85host" Port="9445" Protocol="https">
          ...
        
```

© Copyright IBM Corporation 2013

Next, find a Route xml element whose UriGroup attribute matches the name of the UriGroup.

Slide 33

WebSphere Education 

Interpreting the plugin-cfg.xml file (3 of 4)

Find the ServerCluster element that maps to the Route's ServerCluster

```
<Config ...>
...
<UriGroup Name="default_host_PlantsCluster_URIs">
...
  <Uri Name="/PlantsByWebSphere/*" .../>
...
<Route UriGroup="default_host_PlantsCluster_URIs"
      ServerCluster="PlantsCluster" />
...
<ServerCluster Name="PlantsCluster" LoadBalance="Round Robin" >
  <Server CloneID="17shqbbrq" LoadBalanceWeight="2" ...>
    <Transport Hostname="was85host" Port="9080" Protocol="http"/>
    <Transport Hostname="was85host" Port="9443" Protocol="https">
      ...
  <Server CloneID="17shqbcf9" LoadBalanceWeight="2" ...>
    <Transport Hostname="was85host" Port="9081" Protocol="http"/>
    <Transport Hostname="was85host" Port="9445" Protocol="https">
```

© Copyright IBM Corporation 2013

Find the value of the ServerCluster attribute of the Route xml element; then find a ServerCluster xml element with the same name.

WebSphere Education
IBM

Interpreting the plugin-cfg.xml file (4 of 4)

Server selected based on algorithm, affinity, and protocol

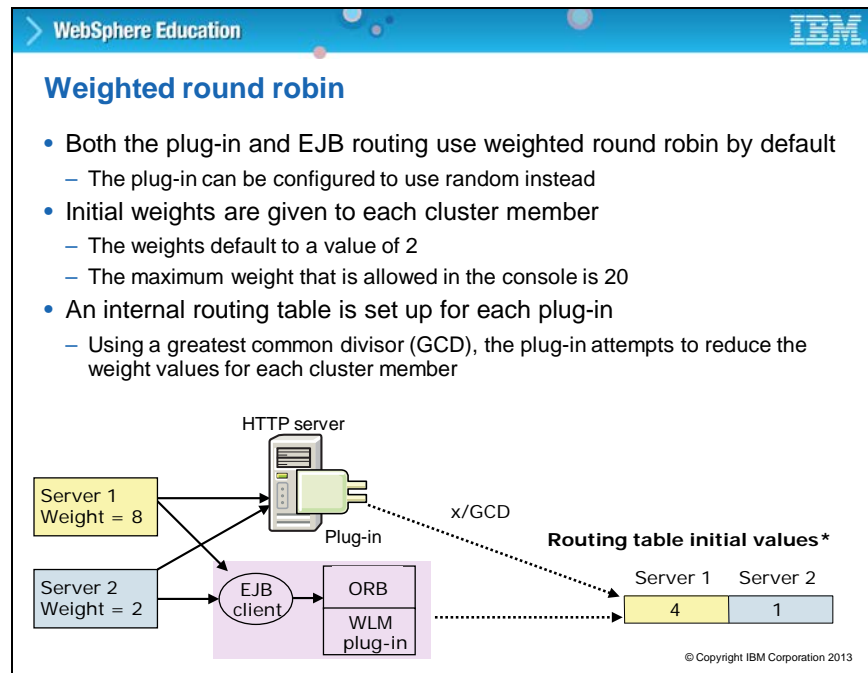
```

...
<UriGroup Name="default_host_PlantsCluster_URIs">
    ...
    <Uri Name="/PlantsByWebSphere/*" .../>
</UriGroup>
...
<Route UriGroup="default_host_PlantsCluster_URIs"
      ServerCluster="PlantsCluster" />
...
<ServerCluster Name="PlantsCluster" LoadBalance="Round Robin" ...
  <Server CloneID="17shqbbbrq" LoadBalanceWeight="2" ...>
    <Transport Hostname="was85host" Port="9080" Protocol="http"/>
    <Transport Hostname="was85host" Port="9443" Protocol="https">
      ...
  <Server CloneID="17shqbcf9" LoadBalanceWeight="2" ...>
    <Transport Hostname="was85host" Port="9081" Protocol="http"/>
    <Transport Hostname="was85host" Port="9444" Protocol="https">

```

© Copyright IBM Corporation 2013

The ServerCluster xml element contains the possible application server host and port information that the request can be routed to.



The default routing option for both the plug-in and the ORB is weighted round robin. Each server is assigned a weight, which indicates the amount of requests the server gets. The weight value that the plug-in uses are reduced by dividing all values by the greatest common denominator. These new values are then inserted into a routing table that is used to track the distribution between the available servers. The ORB uses the actual weights for routing.

WebSphere Education
IBM

Weighted routing example with no affinity

	Server 1	Server 2
Start:	4	1
Request 1	3	1
Request 2	3	0
Request 3	2	0
Request 4	1	0
Request 5	0	0
Reset:	4	1

As soon as all values ≤ 0 , a reset is done

Reset adds initial values of 4, 1

- Example 1: only new hits (no session affinity)
- As requests come into the plug-in, round robin is used to distribute the hits
- Each request decrements the value in the internal table
- As soon as a server has a count of zero, no new hits are sent to that server
- As soon as all servers have a value of zero, the values are all reset
- Results:
 - server1: 80%
 - server2: 20%

© Copyright IBM Corporation 2013

In this example, there is no affinity. As requests come into the plug-in, weighted round-robin is used to distribute requests to cluster members. Each request to a server causes its weight to decrement by one. When a server has a weight value of zero, no new requests are sent to that server. When the weight values of all the cluster members reach zero, the weights are reset. In this example, server 1 has a weight of 4, and server 2 has a weight of 1. The result is that server 1 receives 80 percent of the requests, and server 2 receives 20 percent.

Slide 37

WebSphere Education
IBM

Weighted routing example with affinity

- Example 2: combination of hits
- Sticky hits are those hits that have affinity
- Property **IgnoreAffinityRequests** (default = true) causes the internal table to **not decrement the count for sticky hits**
- As requests come into the plug-in, weighted round robin is used
- Each non-sticky request decrements the value in the table
- Sticky hits are routed to the server for which they have affinity
- As soon as a server has a count of zero, no new hits are sent to that server
- As soon as all servers have a value of zero, the values are all reset
- Results:
 - server1: around 80%
 - server2: around 20%

	Server 1	Server 2
Start:	4	1
Request 1	3	1
Request 2*	3	1
Request 3	3	0
Request 4**	3	0
Request 5	2	0
Request 6	1	0
Request 7**	1	0
Request 8	0	0
Reset:	4	1

Forced hits due to sticky sessions:
Server 1: *
Server 2: **

© Copyright IBM Corporation 2013

If you select weighted round robin for the load balancing option and leave the IgnoreAffinityRequests property set to its default value of true, the affinity requests do not lower the weight. This behavior might cause an uneven distribution of requests across the servers in environments that use session affinity. Setting the IgnoreAffinityRequests property to false causes the weight value to be lowered every time an affinity request is received which results in a more balanced round robin environment.

In this example, there is affinity and the property IgnoreAffinityRequests is set to true. In this case, the result is also 80:20.

WebSphere Education

Weighted routing example with counting affinity

- Example 3: combination of hits
- Setting **IgnoreAffinityRequests** to false
- As requests come into the plug-in, weighted round robin is used
- All requests decrement the weight values in the table
 - Sticky hits are routed to the server for which they have affinity
 - Weighted round robin is used to route non-sticky hits
- As soon as a server has a count of zero, no new hits are sent to that server
- As soon as all servers have a value of zero or less, the values are all reset
 - A reset adds the modified server weights (4 and 1) to the servers repeatedly until all servers are greater than zero
- Results (can vary):
 - server1: around 80%
 - server2: around 20%

	Server 1	Server 2
Start:	4	1
Request 1	3	1
Request 2*	2	1
Request 3	2	0
Request 4**	2	-1
Request 5	1	-1
Request 6**	1	-2
Request 7	0	-2
Reset:	4	-1

Forced hits due to sticky sessions:
Server 1: *
Server 2: **

© Copyright IBM Corporation 2013

In this example, where the `IgnoreAffinityRequests` property is set to false, all requests, even sticky ones, cause the weight value of the server to decrement. In this case, sticky requests are routed to the server for which they have affinity, even if the server weight is zero. This setting can cause the weight value to become a negative number. When all cluster members have a weight value of zero or less, the original weight value is added until all the server weights are greater than zero.

Slide 39

WebSphere Education

IBM

Routing alternative: Random

- An alternative algorithm to weighted round robin is random
 - Sticky hits still go to the appropriate application server
 - New hits are randomly distributed

Web servers

[Web servers](#) > [webserver01](#) > [Plug-in properties](#) > [Request routing](#)

Use this page to configure request routing properties for a web server plug-in. requests the web server routes to application servers.

Configuration

[Request routing](#)

Load balancing option

Round Robin

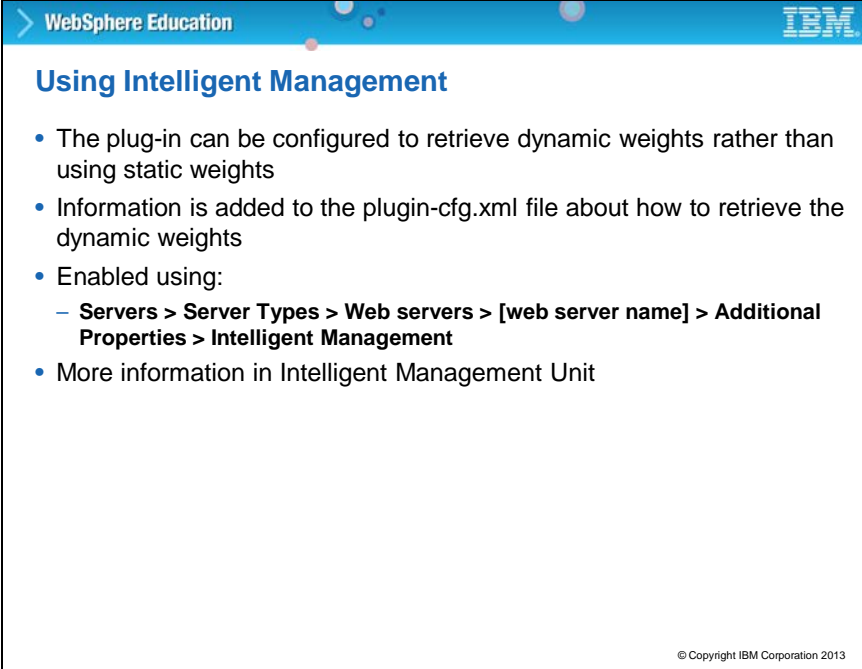
Round Robin

* Random


60 seconds

© Copyright IBM Corporation 2013

An alternative to the weighted round robin algorithm is the random algorithm. This attribute can be set in the console under plug-in properties. If you select random for the load balancing option property, affinity requests are still sent to the same server, but new requests are routed randomly. The value that is specified for the weight property is ignored.



The slide is titled "Using Intelligent Management" and is part of a "WebSphere Education" presentation. It contains a bulleted list of instructions for configuring intelligent management. The list includes: the plug-in can be configured for dynamic weights; information is added to the plugin-cfg.xml file; and a specific navigation path in the management console: Servers > Server Types > Web servers > [web server name] > Additional Properties > Intelligent Management. It also refers to the Intelligent Management Unit for more information. A small copyright notice for IBM Corporation 2013 is at the bottom right.

WebSphere Education 

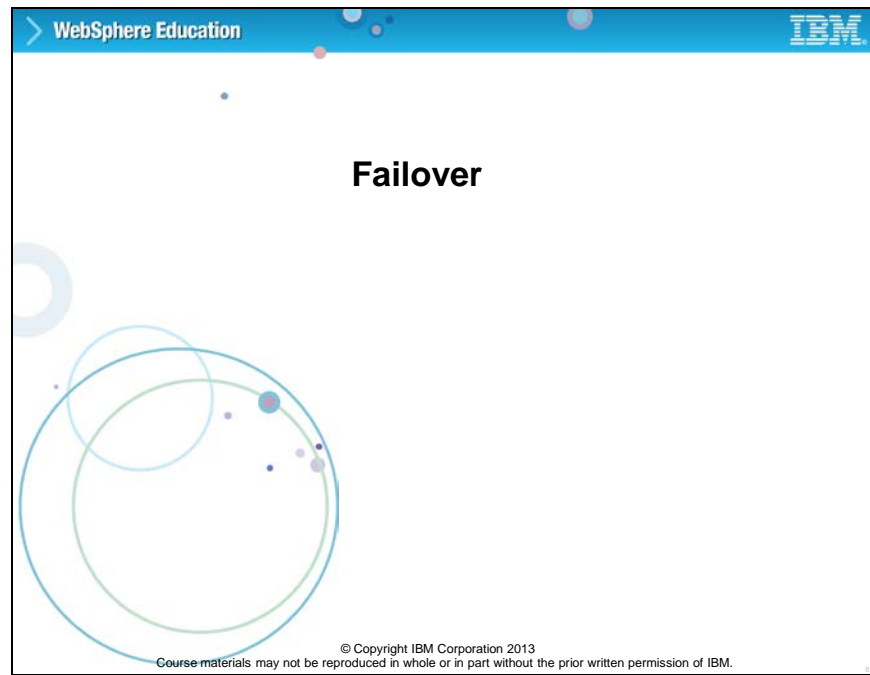
Using Intelligent Management

- The plug-in can be configured to retrieve dynamic weights rather than using static weights
- Information is added to the plugin-cfg.xml file about how to retrieve the dynamic weights
- Enabled using:
 - **Servers > Server Types > Web servers > [web server name] > Additional Properties > Intelligent Management**
- More information in Intelligent Management Unit

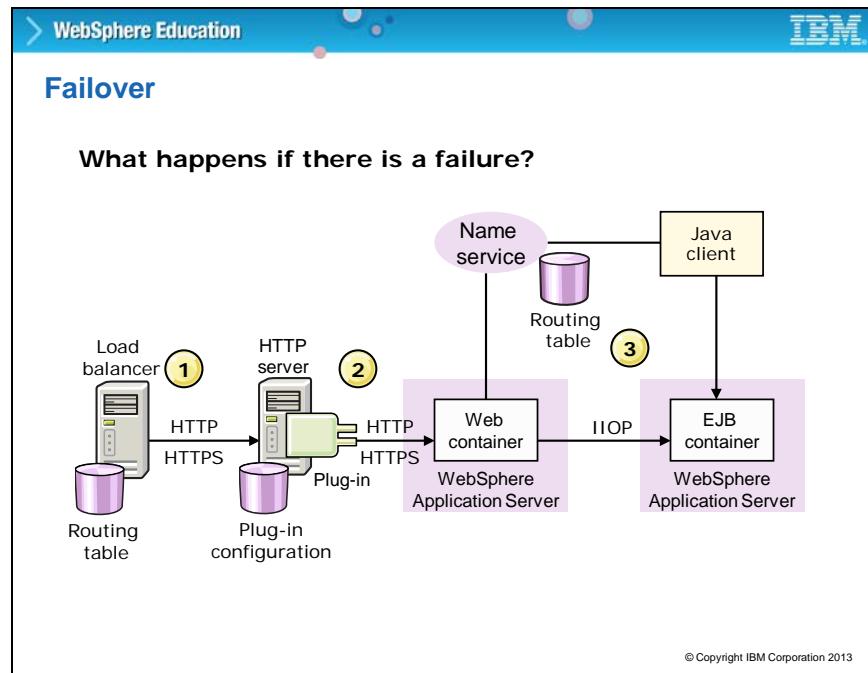
© Copyright IBM Corporation 2013

Rather than using the round robin or random algorithm, you can also configure the plug-in to use intelligent management features of WebSphere Network Deployment. With Intelligent Management, dynamic weights, which are based on how servers are performing, are used to make routing decisions. More information on this topic is contained in the Intelligent Management unit.

Slide 41

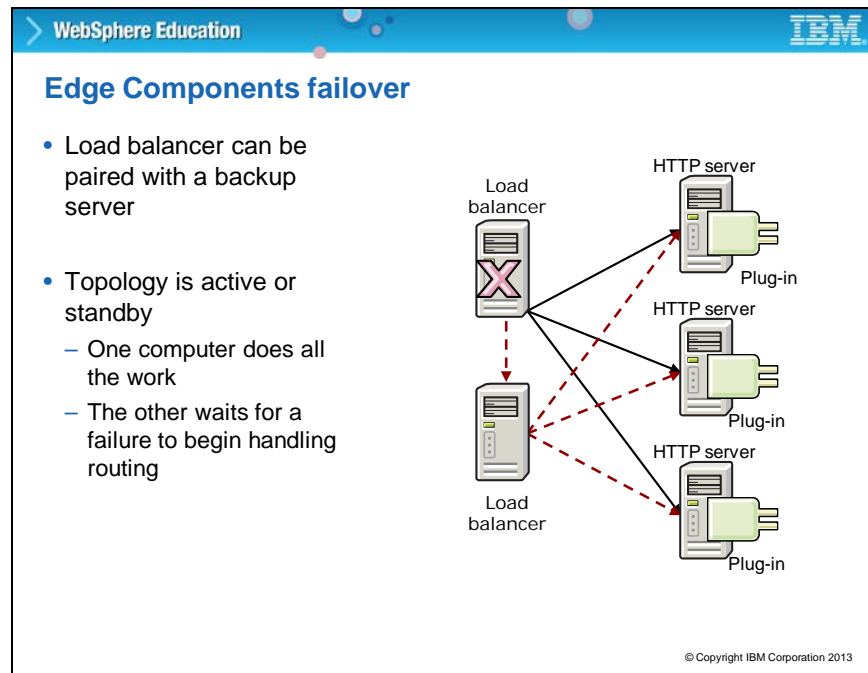


Topic: Failover. In this topic, you learn about failover concepts in a highly available environment.



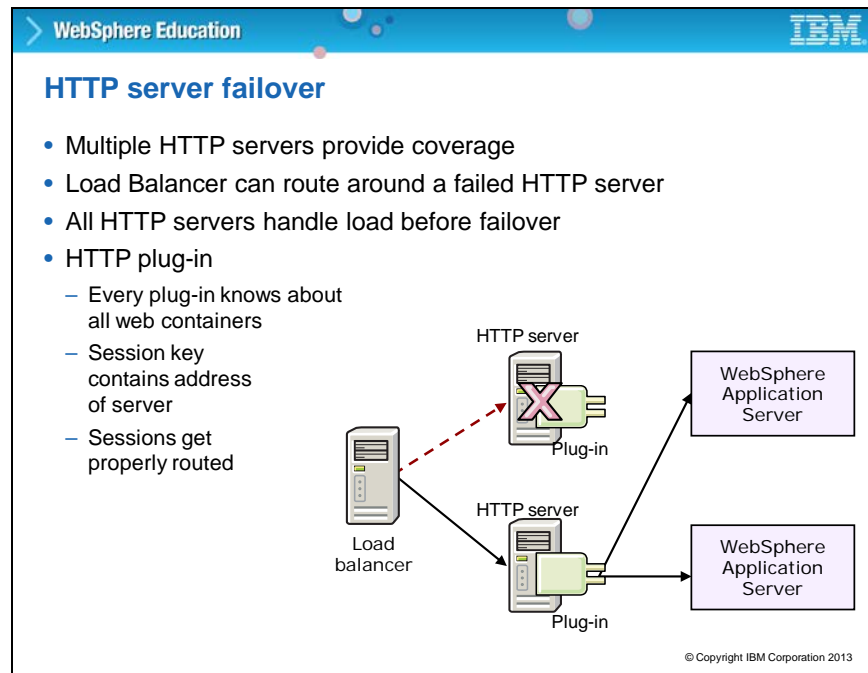
The proposition to have multiple servers naturally leads to the potential for the system to provide failover. That is, if any one computer or server in the system fails for any reason, the system should continue to operate with the remaining servers. The load balancing property should ensure that the client load gets redistributed to the remaining servers, each of which takes on a proportionally higher percentage of the total load. Such an arrangement assumes that the system is designed with some degree of overcapacity so that the remaining servers are indeed sufficient to process the total expected client load.

The next few slides cover what happens during failures at different points in the request flow.



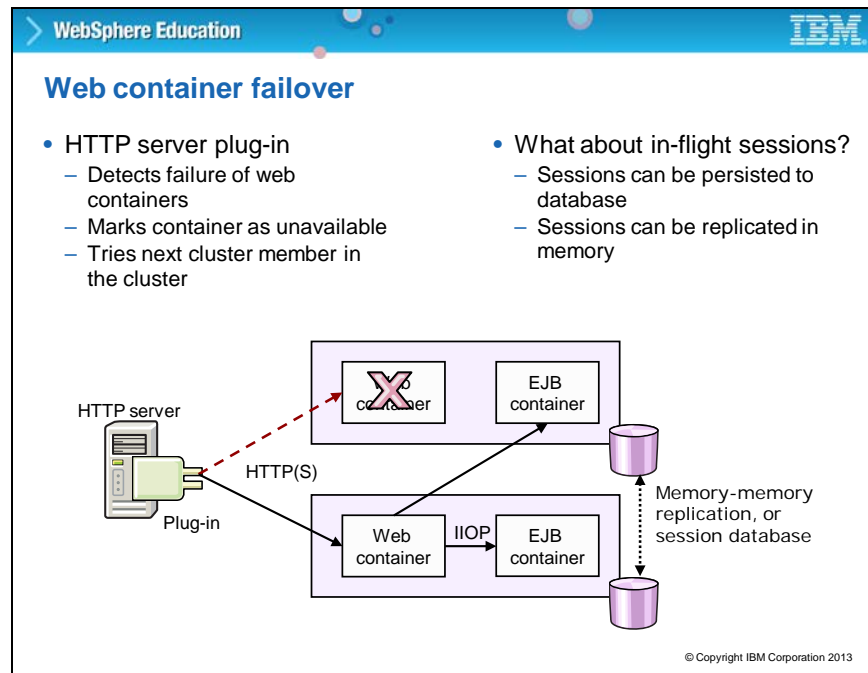
Being the entry point into your WebSphere system, it is important that your load balancer is highly available. High availability for the load balancer can be achieved by setting up a second load balancer server with the same configuration. If a load balancer fails, a standby load balancer can take over.

Slide 44



Perhaps a mechanism of providing request distribution and failover for incoming HTTP requests is required. Without this mechanism, the web server becomes a single point of failure, and scalability is limited to the size of the hardware that is used to host the web server. A likely solution is to employ an IP sprayer, such as the load balancer. All requests are routed to the load balancer, which in turn sprays them among the members of the web server cluster. **If there is** a failure of one of the web servers, the load balancer stops directing work to the failed server.

If a web server fails, any other web server can deal with the request since each plug-in is aware of all of the downstream application servers. If there is affinity to a specific application server, the request still ends up going to the correct application server (based on the JSESSIONID cookie) even if a web server is unavailable.



Each request that is coming into the web server is passed through the plug-in, which uses its configuration information to determine whether the request should be routed. If a plug-in detects that a web container is unavailable, it marks that container as down and does not make more attempts to communicate with it for 2 minutes. After 2 minutes, it attempts to communicate with that server. If it is still unavailable, it is marked down for 2 minutes again, and the process continues.

This algorithm allows each plug-in to dynamically deal with web containers that either become unexpectedly unavailable or are taken offline. At the same time, when new cluster members are added and the plugin-cfg.xml file is generated and propagated, the plug-in automatically knows about the new containers and dynamically adds them to the pool.

The question to ask then is what about sessions? If the server that contains the session is not available to the plug-in when it forwards the request, then the plug-in can route the request to another server. Many web applications use the simplest form of session management, which is the in-memory, local session cache. The local session cache keeps session information in memory and local to the computer and the application server where the session information was first created. Local session management does not share user session information with other clustered computers.

Slide 46

WebSphere Education
IBM

EJB container failover

- Client code and ORB plug-in can route to next available cluster member
- Failure occurs before any work is initiated on the cluster member:
 - ORB automatically reroutes EJB request
 - If no other cluster member available, throws NO_IMPLEMENT exception
- Failure occurs after EJB method call initiated work:
 - System exceptions are thrown
 - Client must determine appropriate recovery action
 - Reissue request or rollback transaction
 - If NO_IMPLEMENT exception is thrown, no recovery is possible

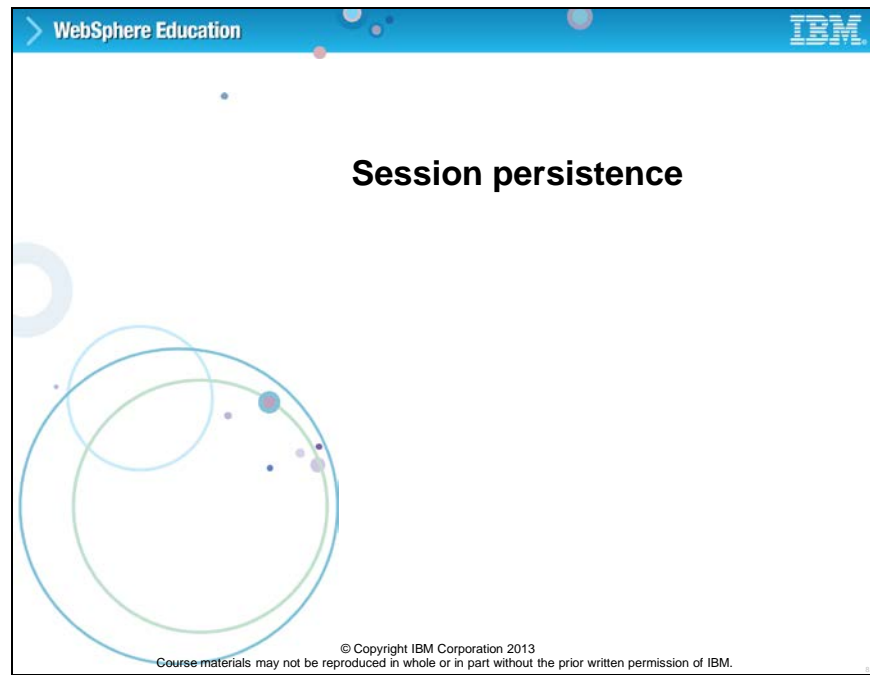
```

graph LR
    subgraph Cluster1 [ ]
        WC1[Web container] -- IIOP --> EC1[EJB container]
    end
    subgraph Cluster2 [ ]
        WC2[Web container] -- IIOP --> EC2[EJB container]
    end
    EC1 -- IIOP --> EC2
    JC[Java client] --> WC1
    WC1 -.-> EC1
    WC2 -.-> EC2
    EC1 -.-> EC2
  
```

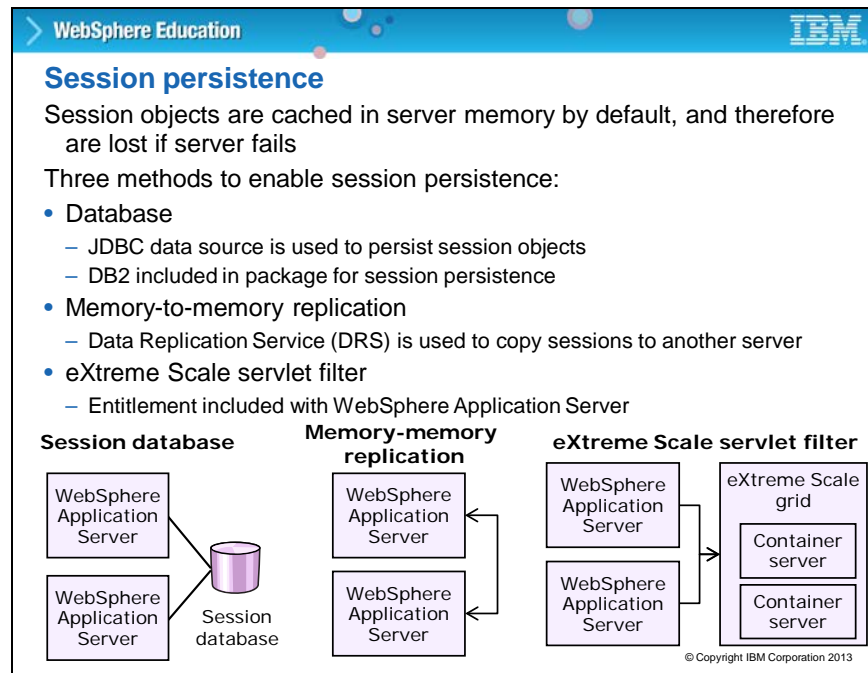
© Copyright IBM Corporation 2013

When an EJB container fails, the failover depends on the initial type of failure. If the error was in reaching the EJB container, then WebSphere fails over to another possible EJB container. However, if the initial call was successful, but the response timed out, then the programmer must try again.

Slide 47



Topic: Session persistence. In this topic, you learn how to persist sessions.



By default, WebSphere places session objects in memory. However, you have the option of enabling persistent session management, which instructs WebSphere to place session objects in a persistent store. If your application cannot tolerate loss of session information, configure session persistence. The three choices are database, memory-to-memory, WebSphere eXtreme Scale.

Using the database approach, all sessions are stored in a session database. The database approach has greater cost, but provides persistence to a database. Memory-to-memory replication uses the data replication service to replicate data across many application servers in a cluster without using a database. Separate threads handle replication within an existing application server process. In this mode, sessions can replicate to address HTTP session single point of failure (SPOF) and eliminates the effort of maintaining a replication database. Using the memory-to-memory approach can still fail if all copies of the session are lost. The WebSphere eXtreme Scale solution is similar to memory-to-memory, except that an eXtreme Scale infrastructure is used to replicate sessions, rather than the data replication service.

A number of settings can be used to affect the balance between failover and performance. This topic is described in upcoming slides.

The slide is titled "Session configuration: Memory-to-memory" and is part of a WebSphere Education presentation. It contains a list of steps for configuring memory-to-memory session replication and two screenshots of the administrative console.

- Configuring memory-to-memory session replication
 - Found under **WebSphere application servers** > **<servername>** > **Session management** > **Distributed environment settings**
 - Select **Memory-to-memory replication**
 - Configure the replication domain

The first screenshot, titled "General Properties", shows the "Distributed sessions" section with three radio buttons: "None", "Database (Supported for Web container only.)", and "Memory-to-memory replication". The "Memory-to-memory replication" option is selected.

The second screenshot, also titled "General Properties", shows the "Replication domain" dropdown menu set to "PlantsCluster" and the "Replication mode" dropdown menu set to "Both client and server". At the bottom are buttons for "Apply", "OK", "Reset", and "Cancel".

© Copyright IBM Corporation 2013

With memory-to-memory replication, there are different replication modes, which can be configuring by using the administrative console.

Server mode stores backup copies of other WebSphere Application Server sessions, but does not send out copies of any session data that is created in that particular server.

Client mode broadcasts or sends out copies of the sessions it owns and does not receive backup copies of sessions from other servers.

Both mode simultaneously broadcasts or sends out copies of the sessions it owns and acts as a backup table for sessions of other WebSphere Application Server instances.

Slide 50

WebSphere Education
IBM

Session configuration: Replication domains

Creating replication domains:

- Created during cluster creation
- Or **Environment > Replication Domains > New**

To configure replication domains after creation:

- Through the administrative console, select the Replication domain under **Environment > Replication domains**
- Select the replication domain
- Select the number of replicas
 - Single replica
 - Entire domain
 - Specified

Replication domains > PlantsCluster
 Use this page to configure the replication properties that are used by

Configuration

General Properties
 Name
 PlantsCluster
 * Request timeout
 5 seconds
 Number of replicas
☒ Single replica
☐ Entire Domain
☐ Specify
 Number of replicas

 Apply OK Reset Cancel

© Copyright IBM Corporation 2013

Replication domains are created during cluster creation or later. Replication domains can be found in the administrative console under **Environment > Replication Domains > New**. Select the replication domain, which is named after the cluster, and select the number of replicas to manage.

A single replica means that for every session object, one backup copy is stored on a different server for failover. It is also possible to back up the sessions on every member of the domain. This practice does not scale as well because every member must store every other members session objects. However, it means that the chance for a successful failover is higher. Depending on the needs of the application and the tolerance for failure, it is possible to tune the number of members that back up every session object.

Slide 51

WebSphere Education **IBM**

Database persistence configuration

[Application servers](#) > [server1](#) > [Session management](#) > [Distributed environment settings](#) > [Database settings](#)

Use this page to specify your database settings.

Configuration

General Properties

* Datasource JNDI name:
jdbc/Sessions

User ID:
db2admin

Password:

DB2 row size:
ROW_SIZE_4KB

Table space name:

☐ Use multi row schema

Apply OK Reset Cancel

1. Create database in DBMS
2. Create data source: **Resources > JDBC > Data sources**
3. Select **Database** in **Distributed environment settings** page
 - Found under **WebSphere Application Servers > <servername> > Session management**
4. Configure database settings

© Copyright IBM Corporation 2013

Database persistence configuration requires a data source. In distributed environments, the session table is created automatically when you define the data source for the session management database. This screen capture shows the settings in the administrative console for database replication and data source configuration.

The default row size is 4 KB, a single row. However, if you want to use a page (row) size greater than 4 KB, you **must** create the table space manually. Using multi-row sessions becomes important if the size of the session object exceeds the size for a row. If the multi-row session enabled, the session manager breaks the session data across multiple rows as needed. This method allows WebSphere Application Server to support large session objects. It also provides a more efficient mechanism for storing and retrieving session contents under certain circumstances.

WebSphere Education
IBM

Tuning session persistence

- Session persistence can be tuned to favor performance or failover
- Accesses through **WebSphere Application Servers > <servername> > Session management > Distributed environment settings > Tuning parameter**

Configuration

General Properties

Tuning level:


☐ Very high (optimize for performance)
Write frequency Time based: 300 seconds
Write contents Only updated attributes
Schedule sessions cleanup: true

☐ High
Write frequency Time based: 300 seconds
Write contents All session attributes
Schedule sessions cleanup: false

☐ Medium
Write frequency End of servlet service
Write contents Only updated attributes
Schedule sessions cleanup: false



☐ Low (optimize for failover)
Write frequency End of servlet service
Write contents All session attributes
Schedule sessions cleanup: false

☒ [Custom settings](#)
Write frequency Time based: 10 seconds
Write contents Only updated attributes
Schedule sessions cleanup: false

Apply

Reset
Cancel

© Copyright IBM Corporation 2013

Session persistence can be optimized for failover or for performance. These settings allow the administrator to choose greater failover ability (at the price of performance) or better performance (at the price of failover ability).



eXtreme Scale persistence configuration

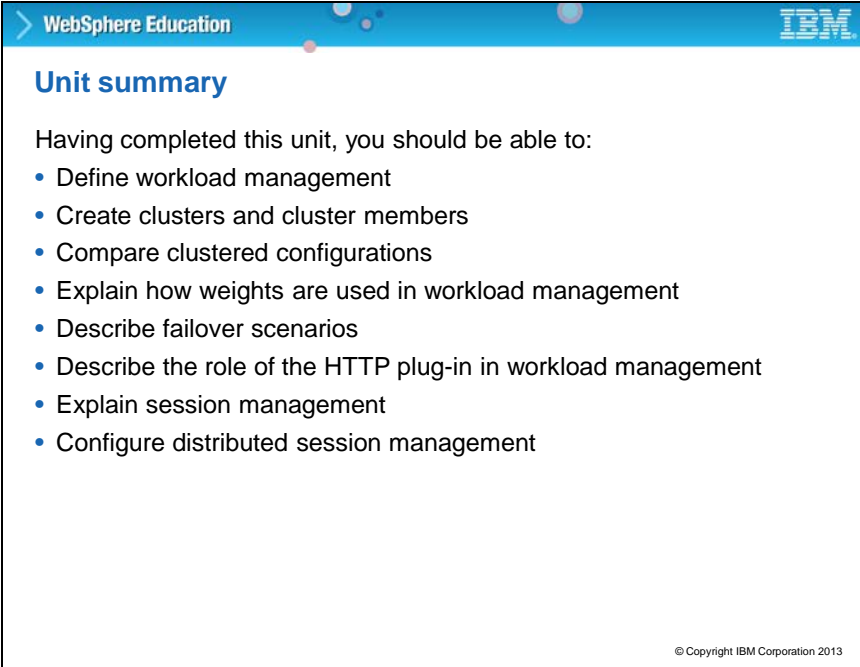
- Create splicer.properties file
 - Provides eXtreme Scale configuration details
- “Splice” HTTP servlet filter into a Web application
 - Run script

```
addObjectGridFilter.[bat|sh]
<location of ear/war file>
<location of splicer.properties file>
```
- Deploy application


© Copyright IBM Corporation 2013

When you use WebSphere eXtreme Scale for session persistence, you add a servlet filter to the application. The servlet filter handles the session management requests.

Slide 54



The slide is titled 'Unit summary' and is part of a WebSphere Education presentation. It lists eight learning objectives for the unit. The slide has a blue header with 'WebSphere Education' and the IBM logo. The text is in a sans-serif font.

WebSphere Education 

Unit summary

Having completed this unit, you should be able to:

- Define workload management
- Create clusters and cluster members
- Compare clustered configurations
- Explain how weights are used in workload management
- Describe failover scenarios
- Describe the role of the HTTP plug-in in workload management
- Explain session management
- Configure distributed session management

© Copyright IBM Corporation 2013

You completed this unit.

Having completed this unit, you should be able to:

- Define workload management
- Create clusters and cluster members
- Compare clustered configurations
- Explain how weights are used in workload management
- Describe failover scenarios
- Describe the role of the HTTP plug-in in workload management
- Explain session management
- Configure distributed session management