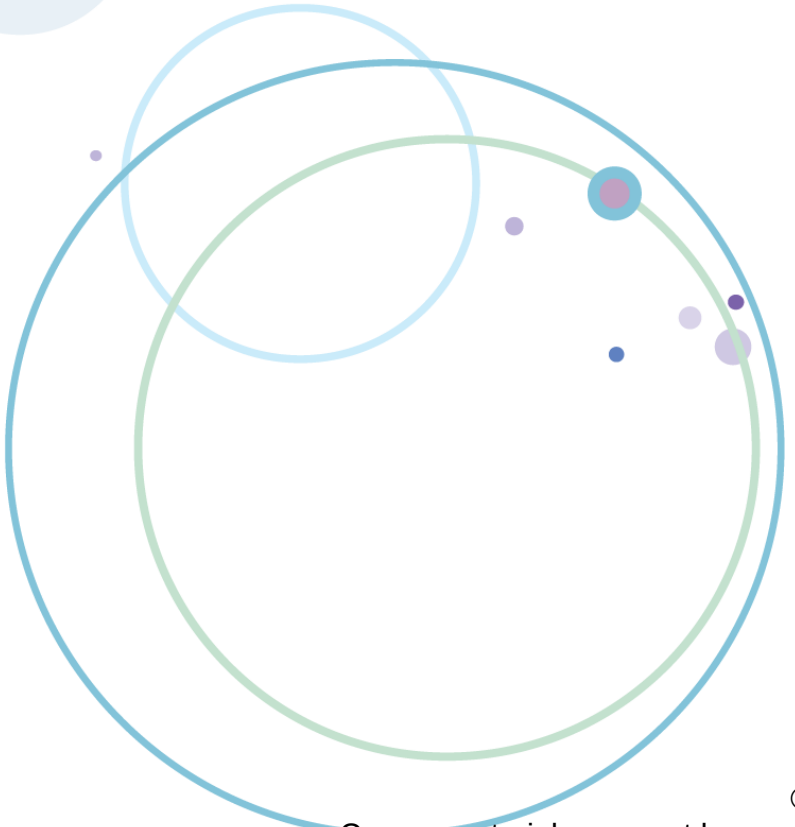
Workload management



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

Topics

- Workload management concepts
- Clusters and cluster members
- Routing concepts and session affinity
- Failover
- Session persistence

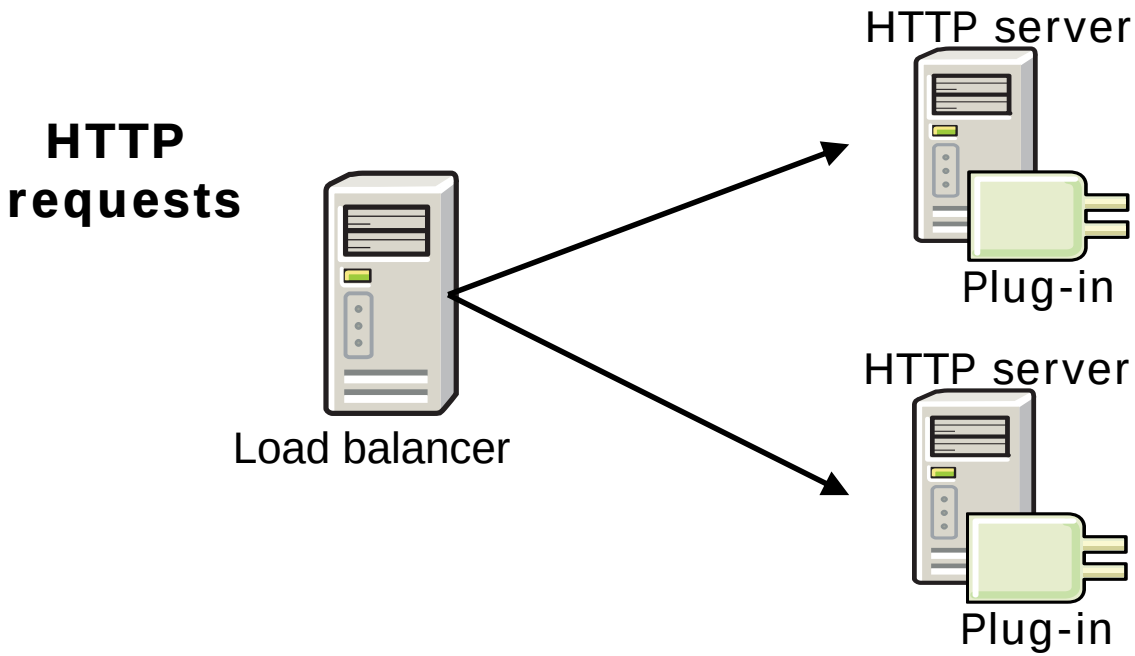
Workload management concepts

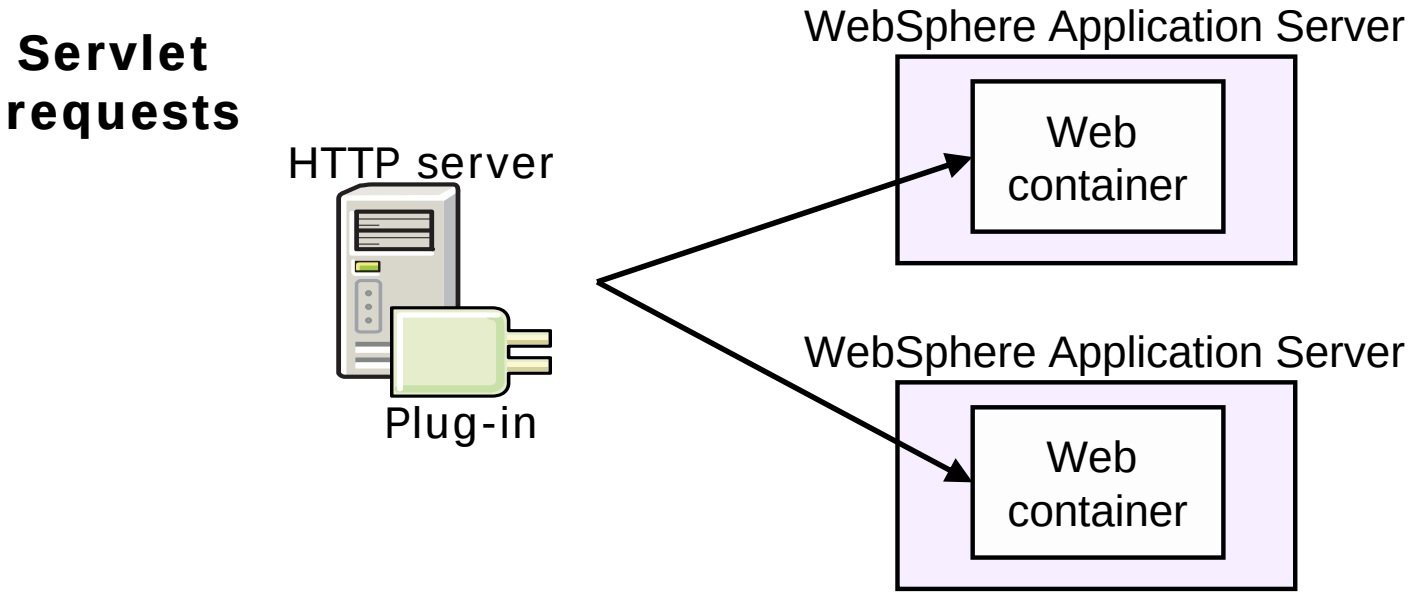
What is workload management (WLM)?

- 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

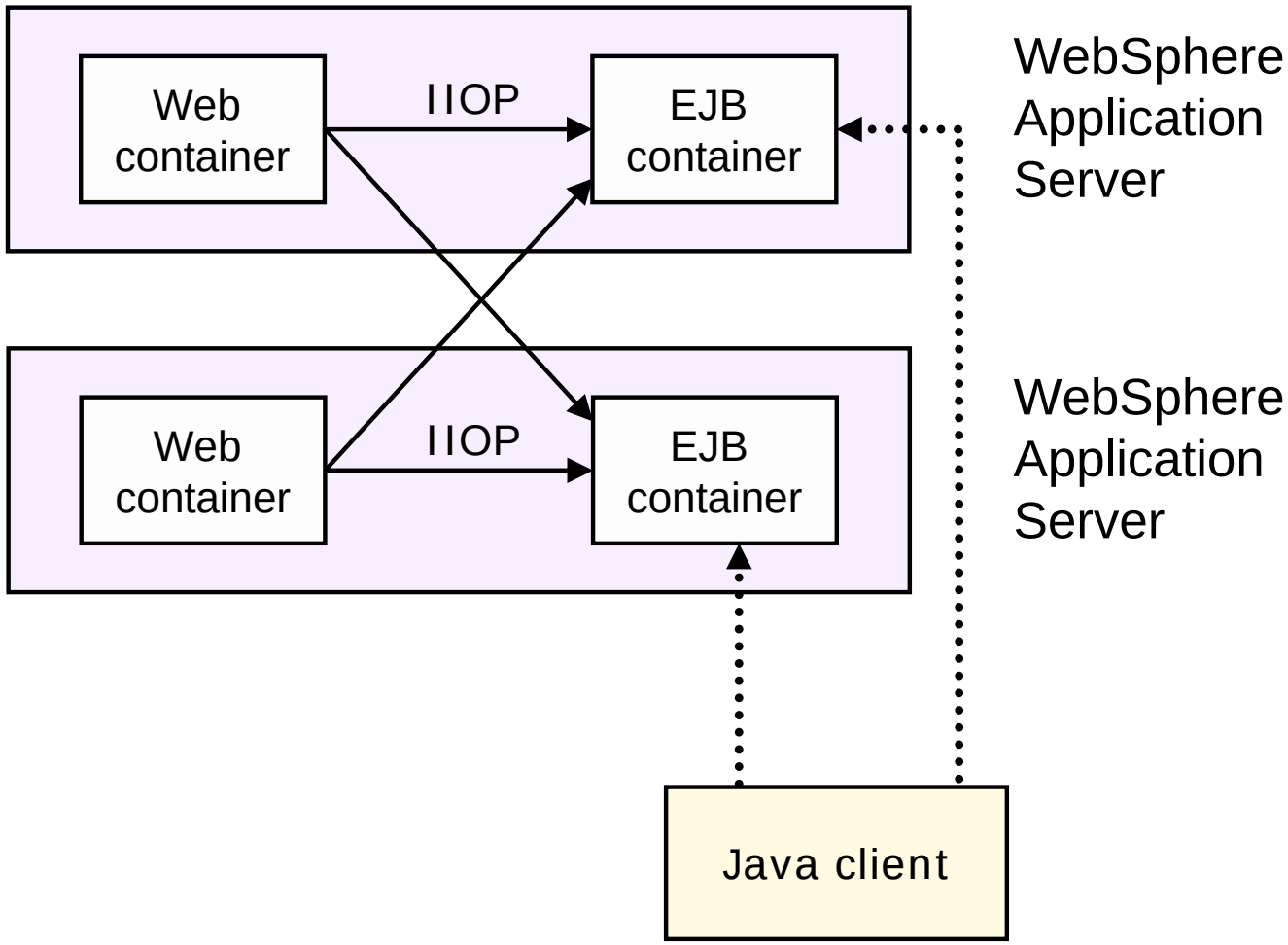
What can be workload managed? (1 of 2)





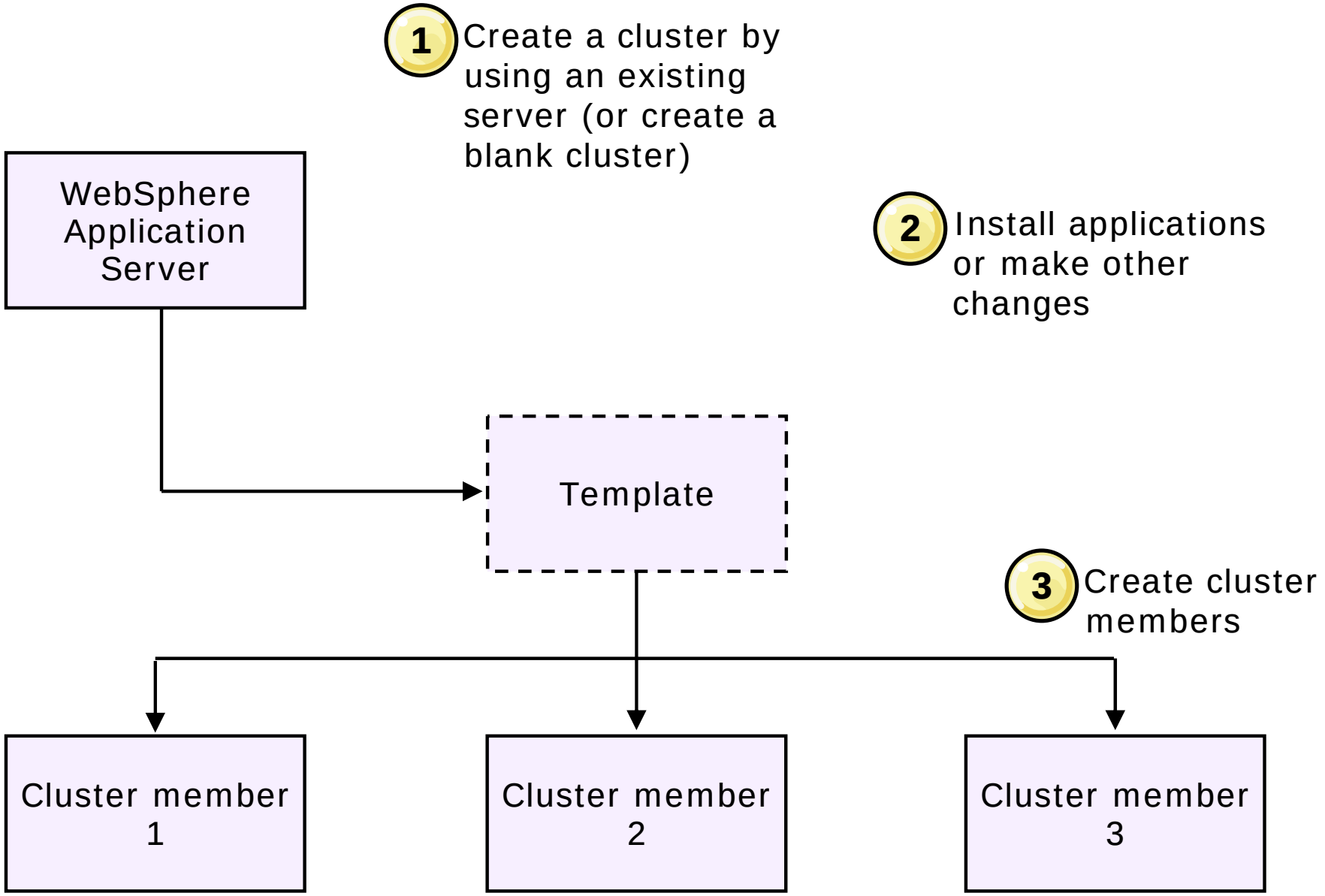
What can be workload managed? (2 of 2)

EJB requests



Clusters and cluster members

Clusters

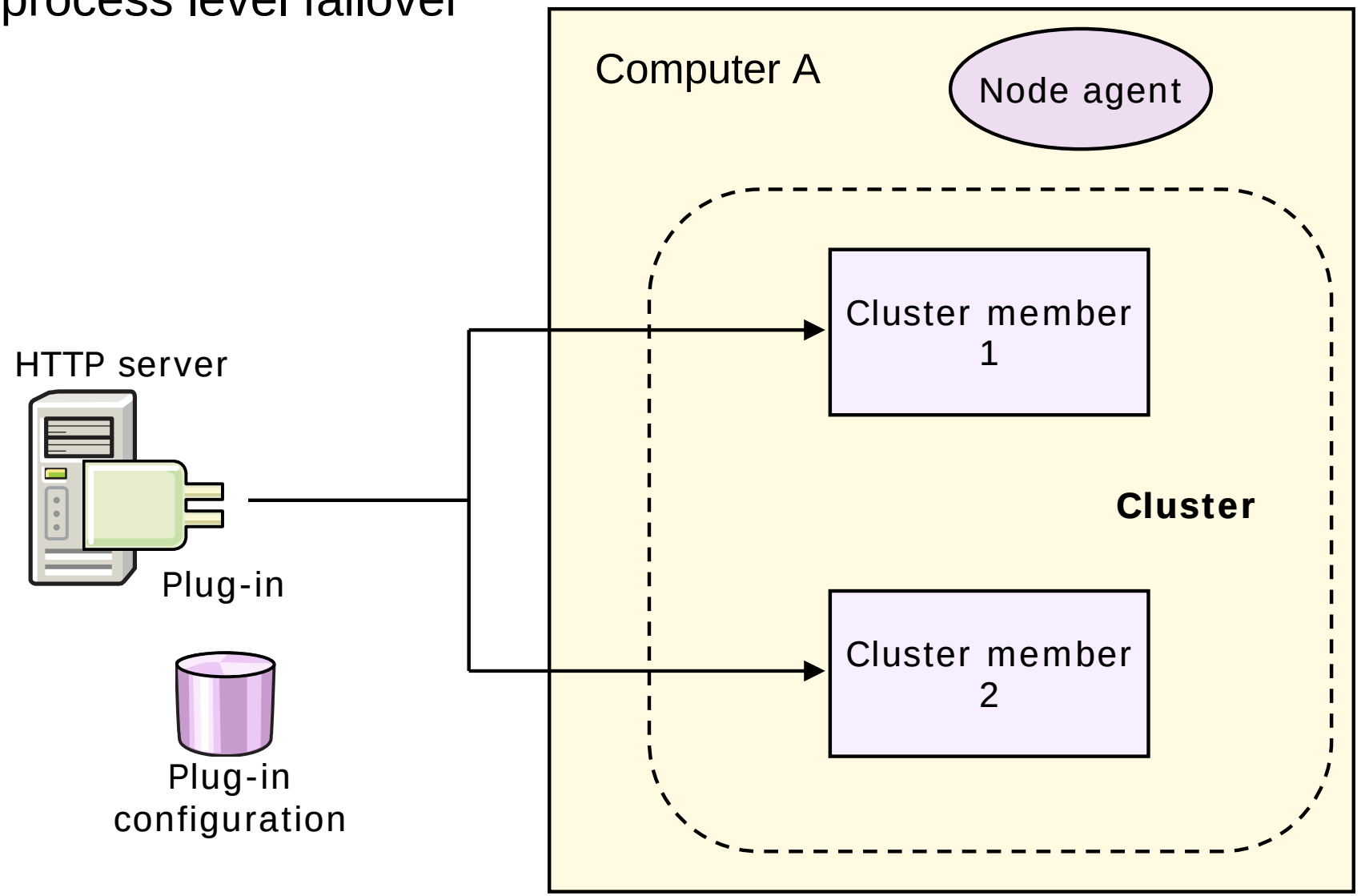


Clusters and cluster members

- 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

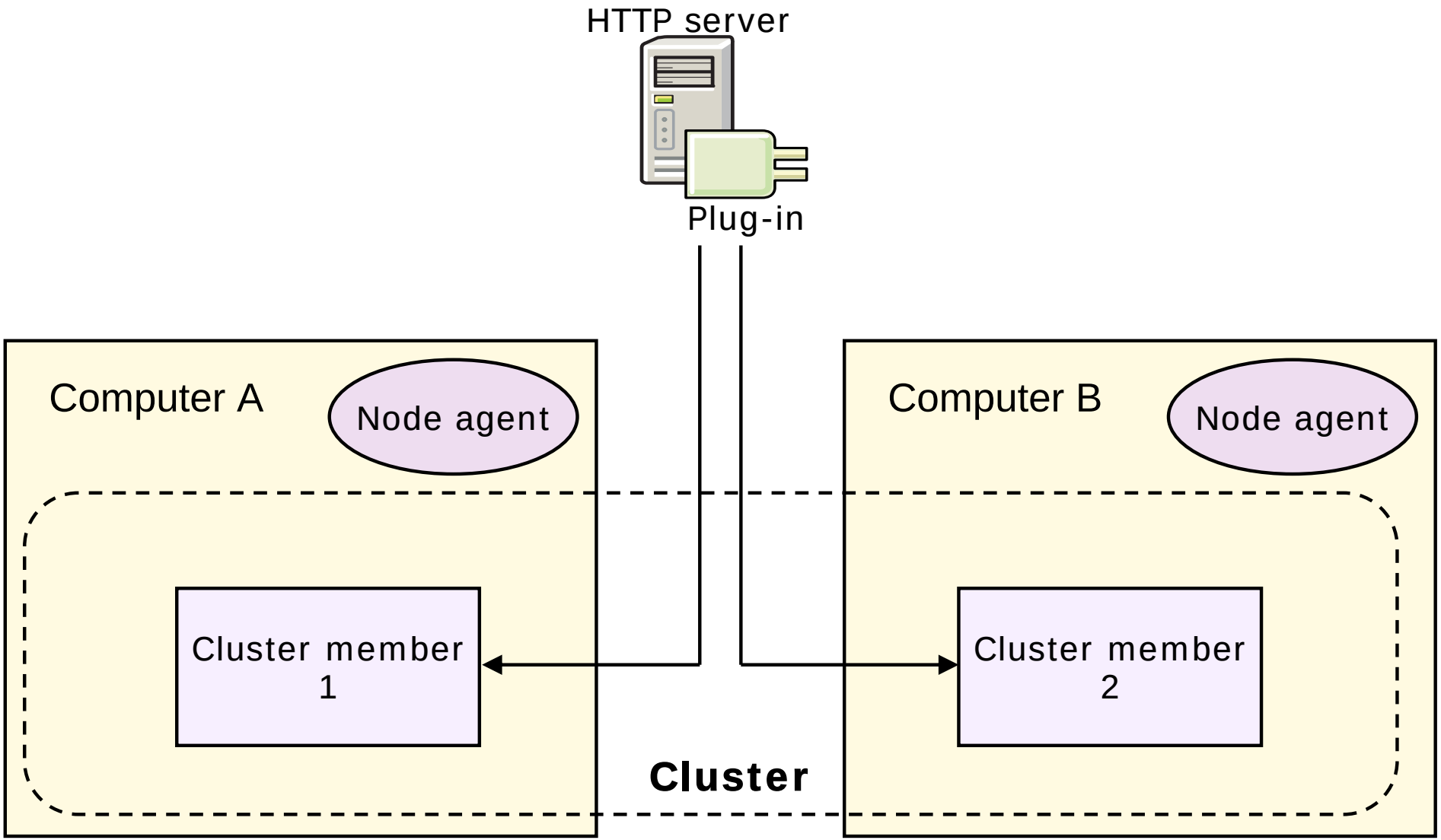
Configurations: Vertical scaling

- Might provide better performance with multiple processors
- Provides process level failover



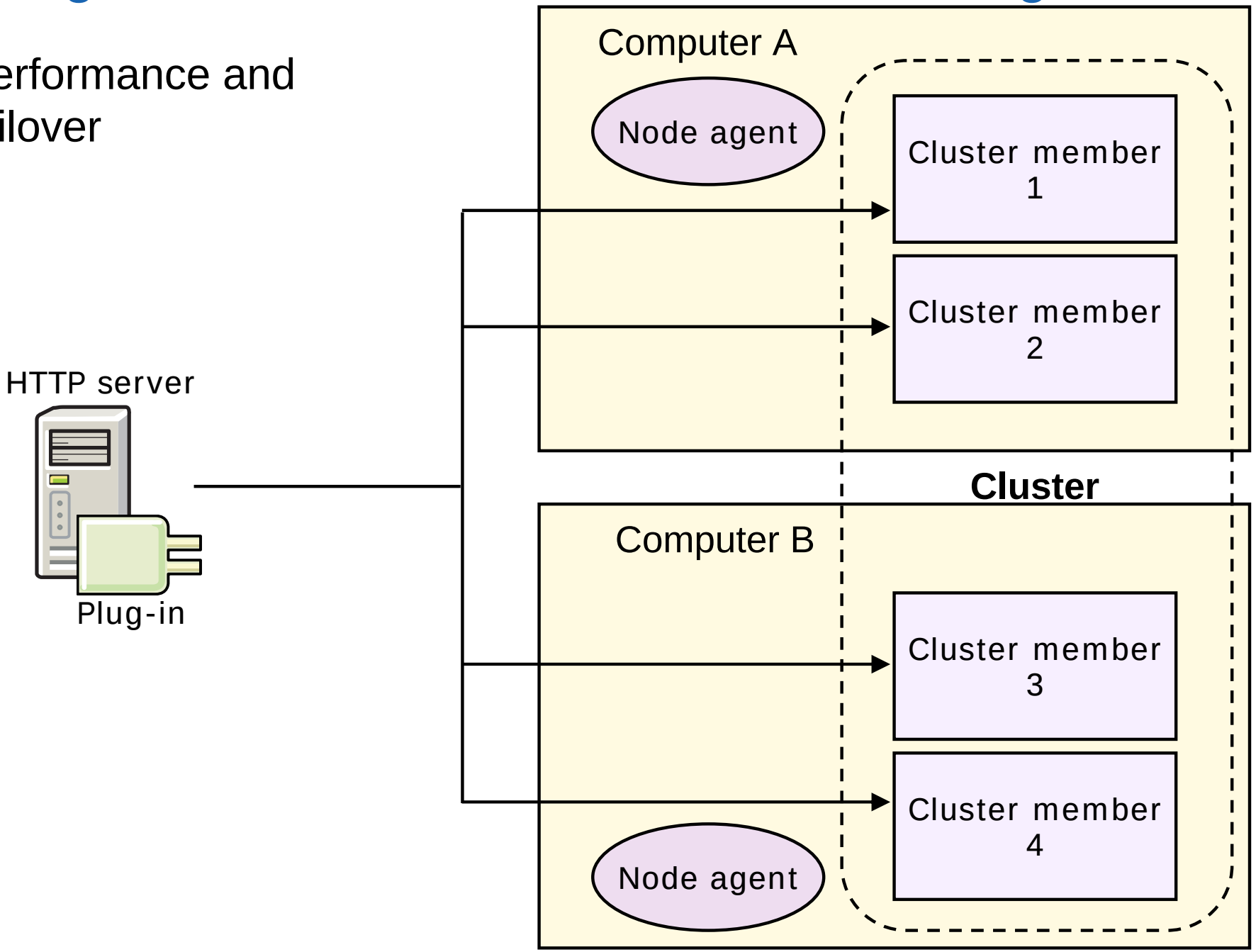
Configurations: Horizontal scaling

- Supports server failover



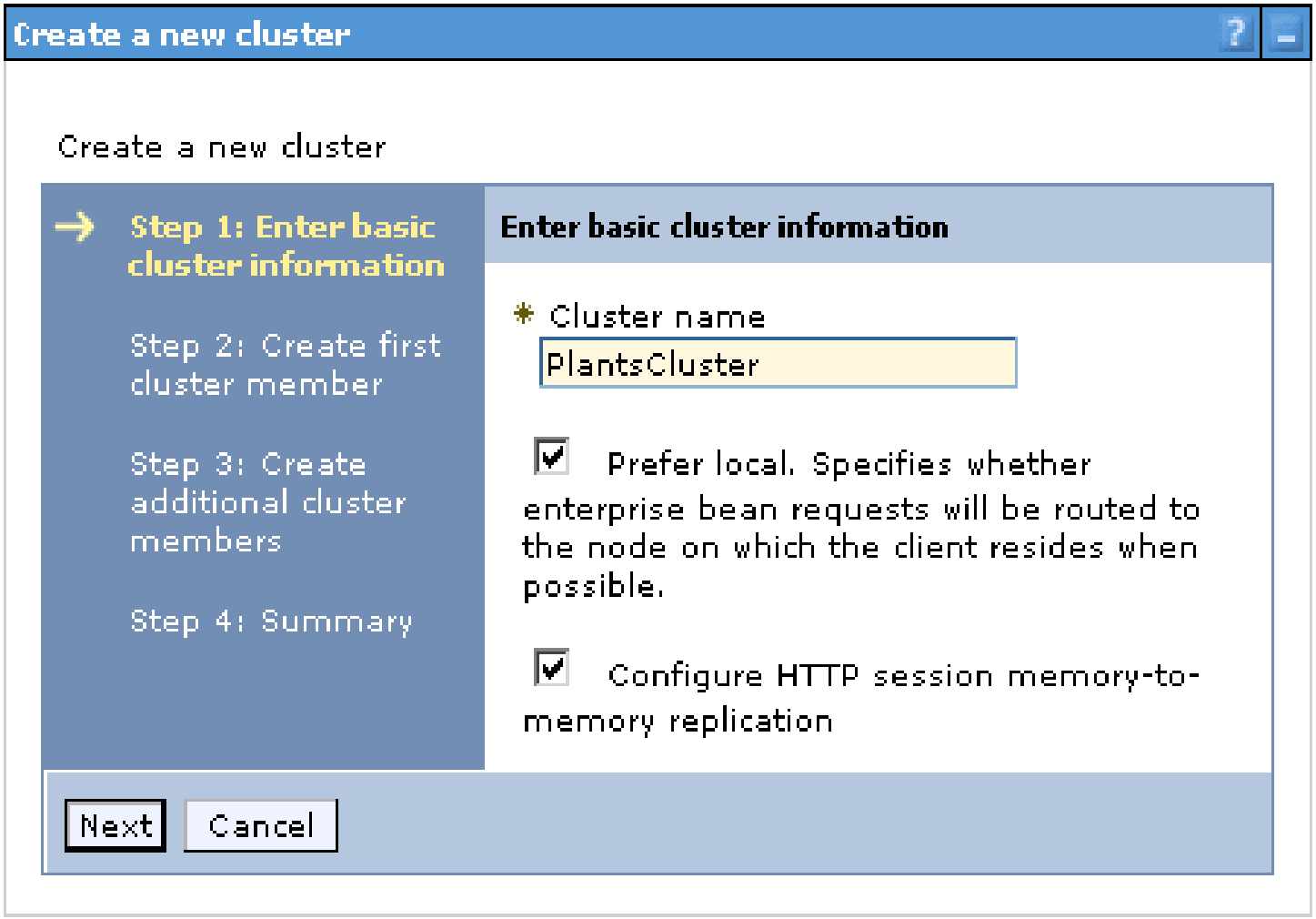
Configurations: Vertical and horizontal scaling

- Performance and failover



Creating a cluster (1 of 4)

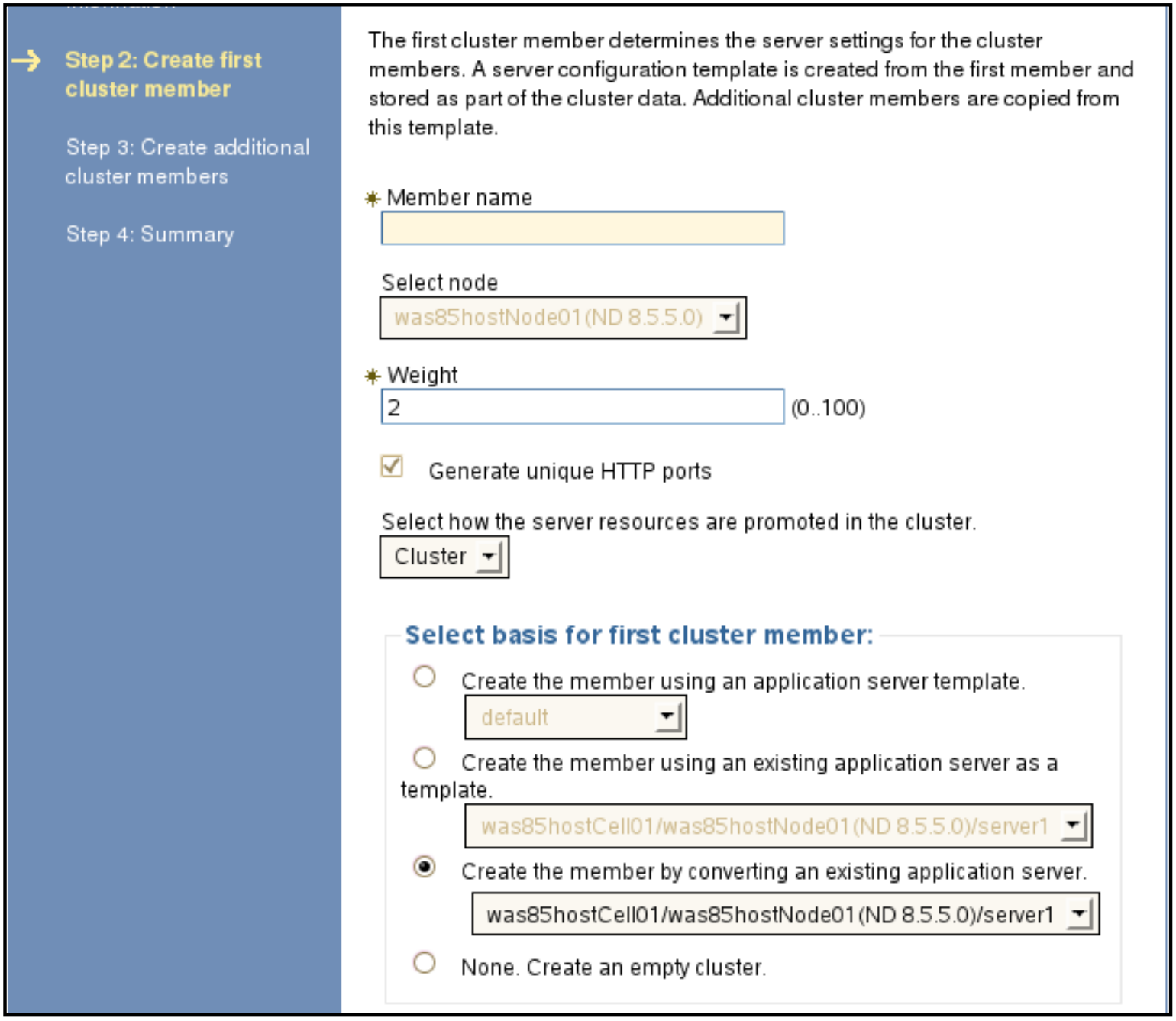
- In console, select **Servers > Clusters > WebSphere application server clusters** and click **New**
- Enter cluster name
- Check options
  - Prefer local
  - Configure HTTP session memory-to-memory replication
- Click **Next**



Creating a cluster (2 of 4)

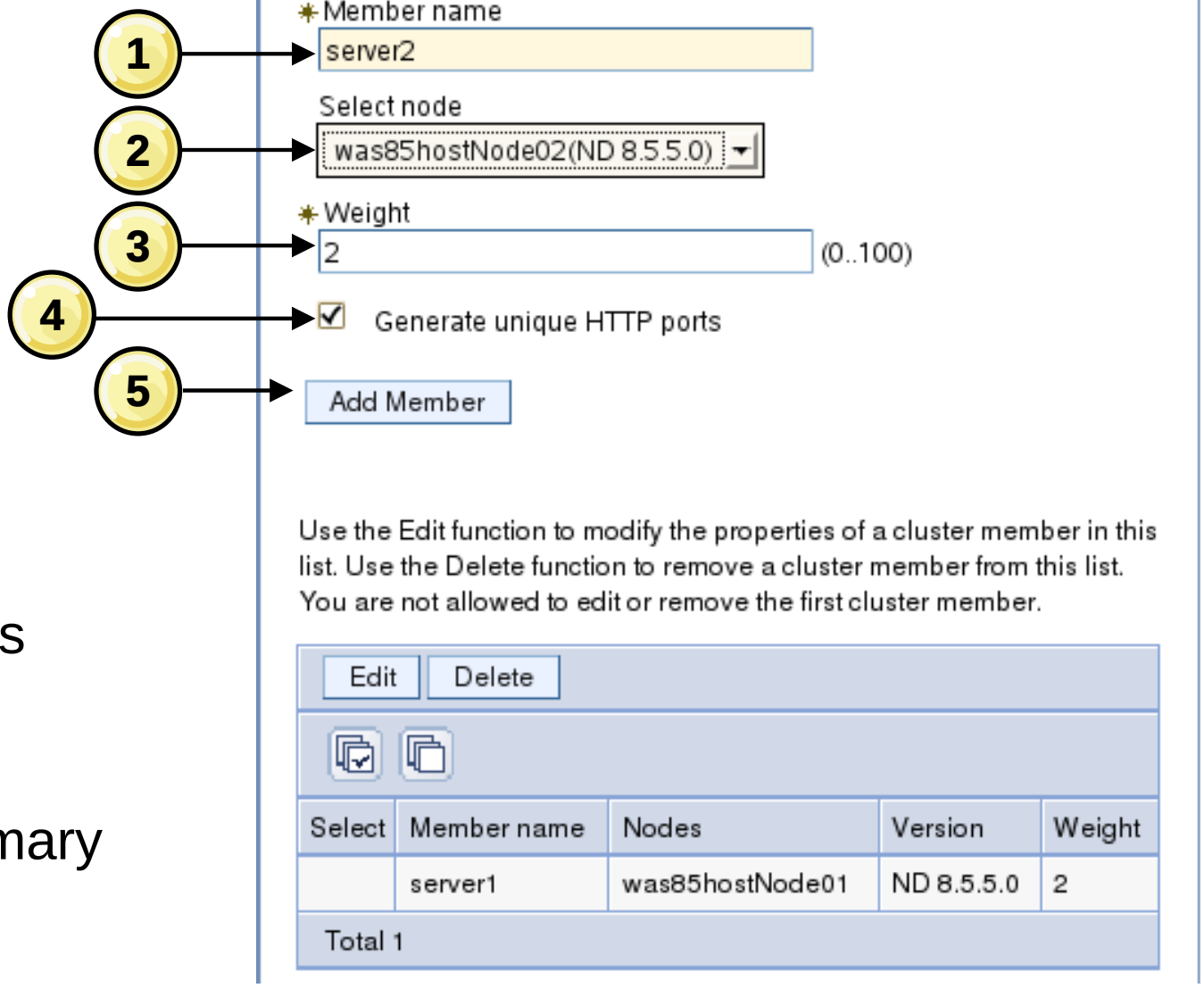
- Create the first cluster member that is based on:
  - A server template
  - An existing server
  - Converting an existing server
  - An empty cluster
- Enter member name
- Select node
- Enter weight
- Click **Next**

Creating a cluster (3 of 4)

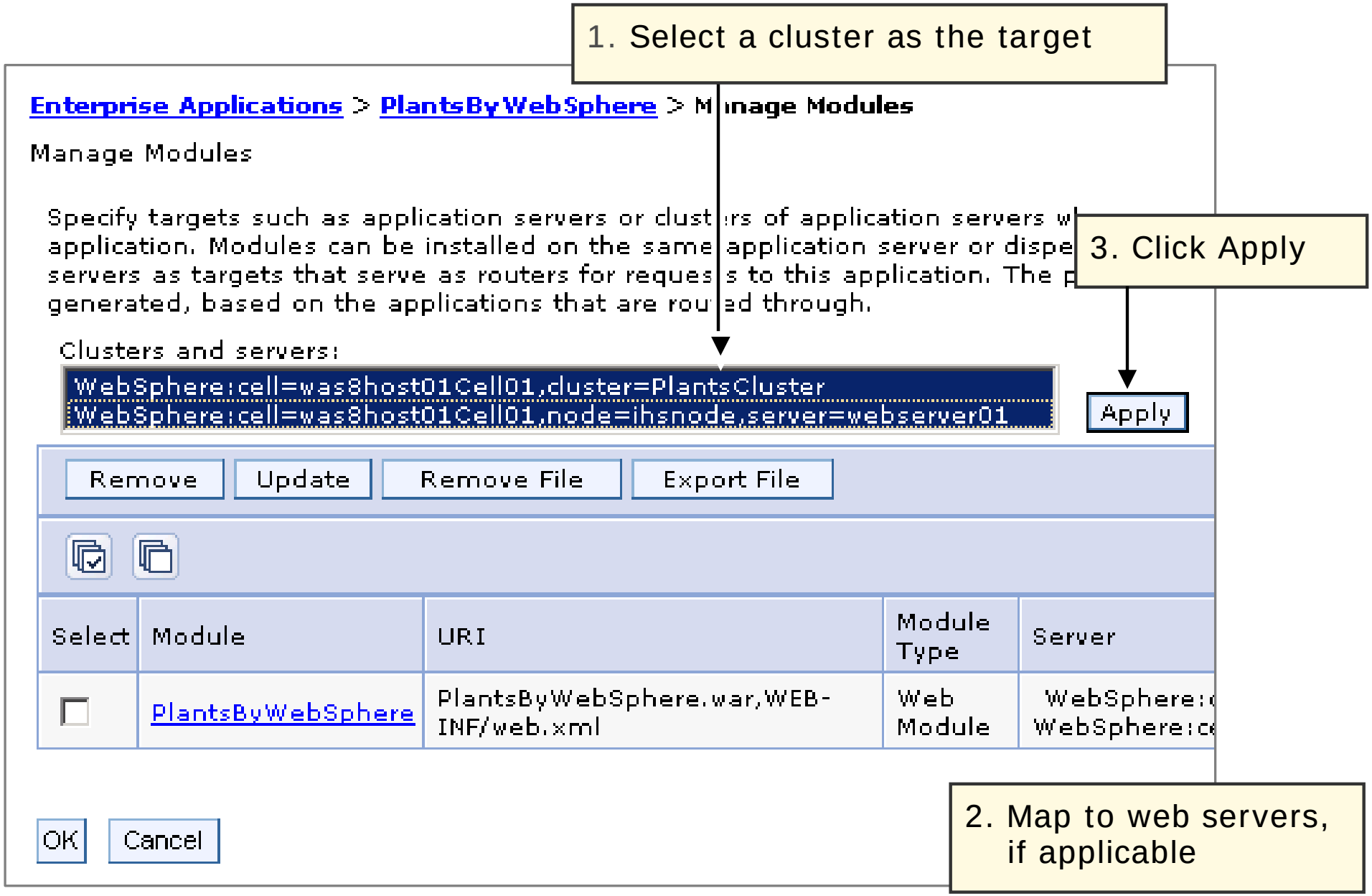


Creating a cluster (4 of 4)

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

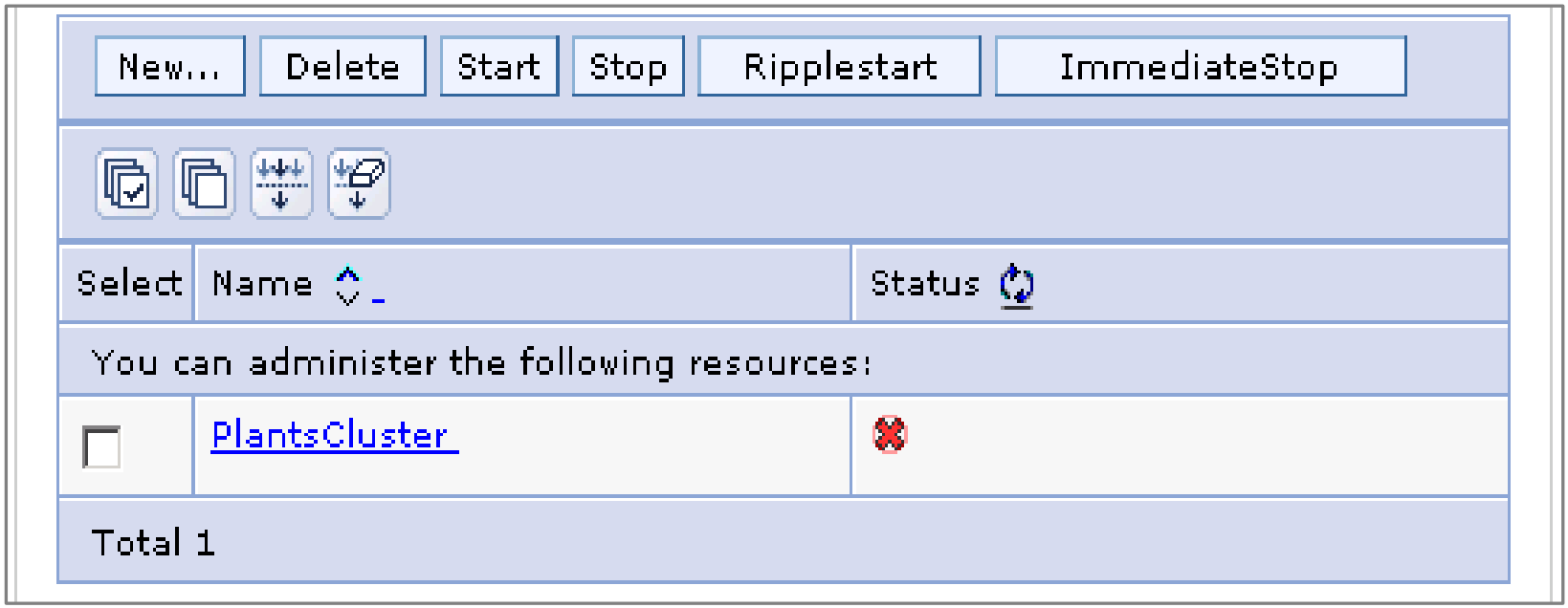


Installing enterprise applications to a cluster



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



Cluster members

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

- Cluster members are just application servers

Modification of clusters

- Use the administrative console or wsadmin

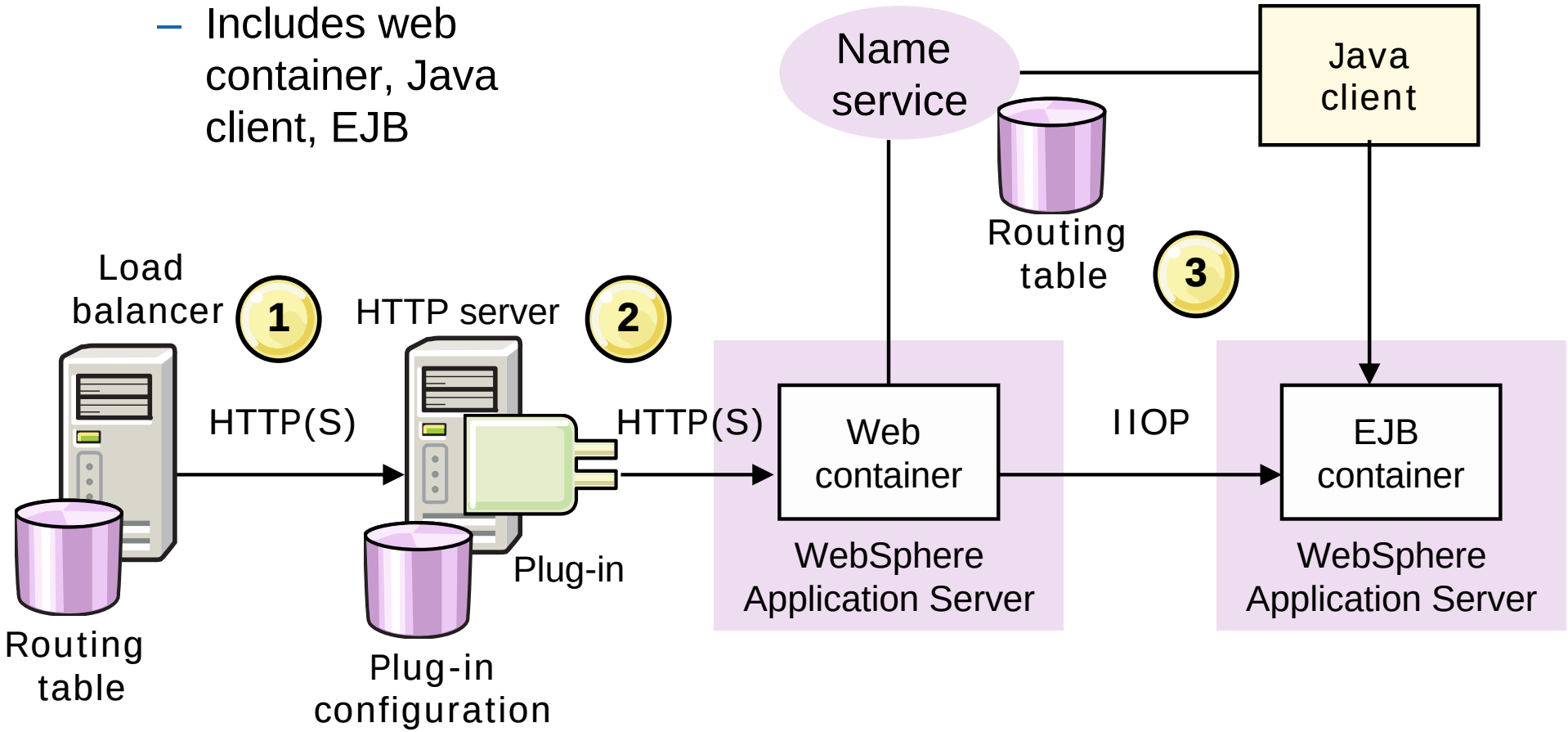
- What can be changed in a cluster?
  - Cluster member settings: weights, prefer local
  - Install or update applications
  - Application server settings: cluster members are application servers and the normal application server settings can be modified

- After modifying the cluster:
  - Save the configuration and resynchronize
  - Regenerate the HTTP server plug-in and redistribute it if necessary

Routing concepts and session affinity

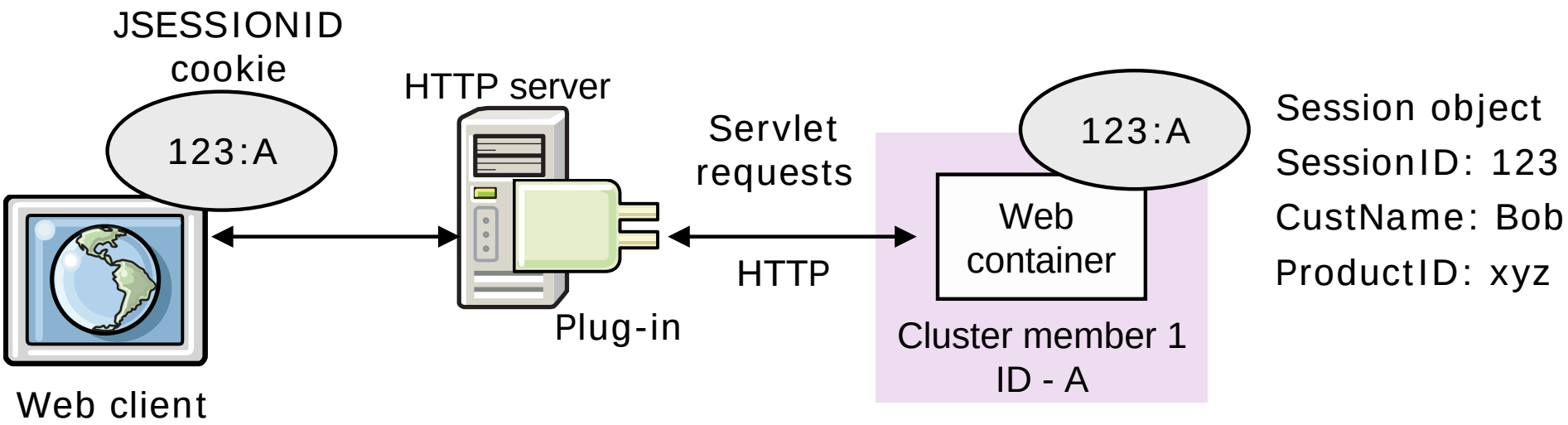
Basic routing algorithms

- Routing decision points
  1. Load balancer
  2. HTTP Server plug-in
  3. WLM-aware EJB client
     - Includes web container, Java client, EJB

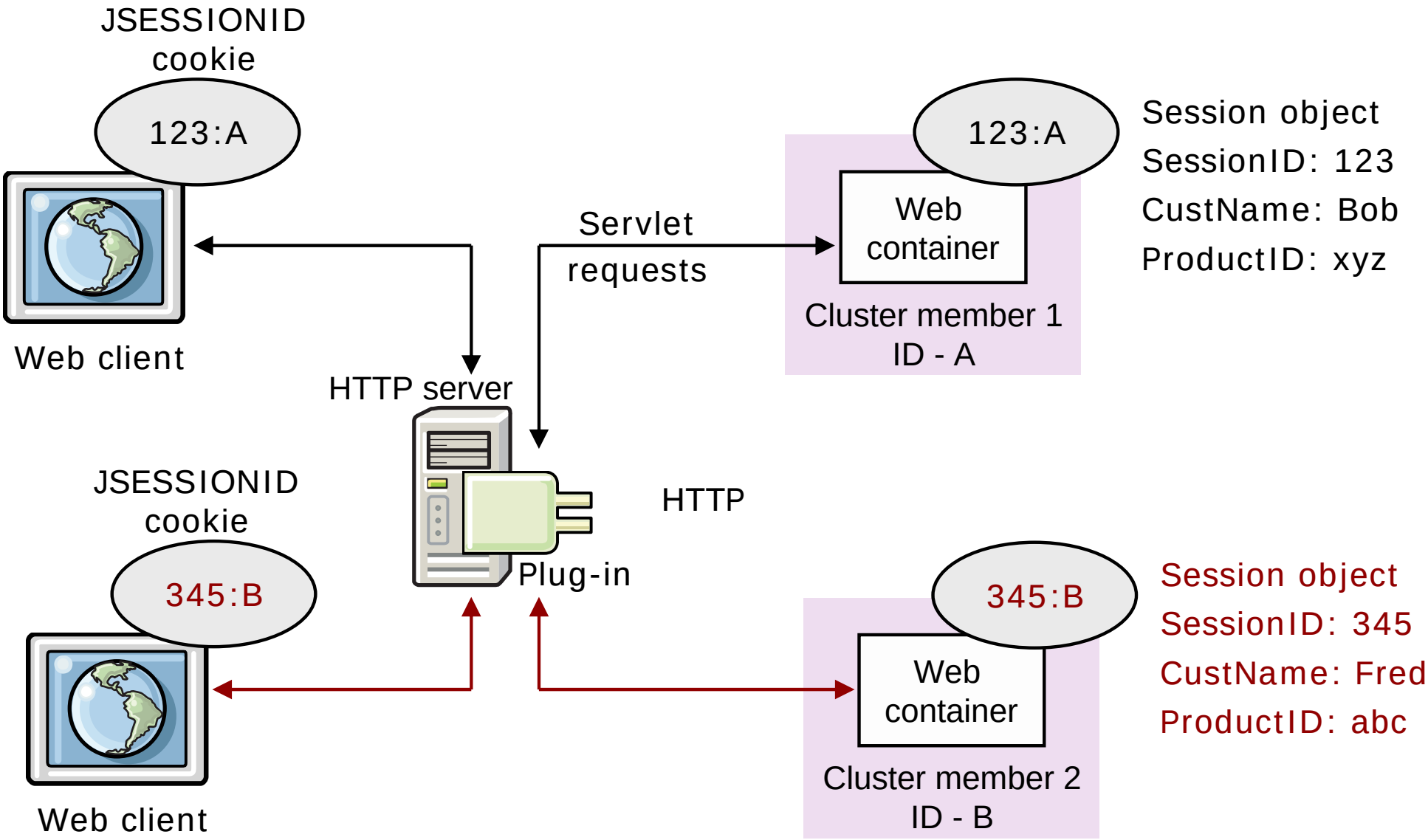


HTTP session management

- HTTP is a stateless protocol
- Sessions allow you to maintain state information across a series of HTTP requests from the same client
  - For example, maintain shopping cart until checkout

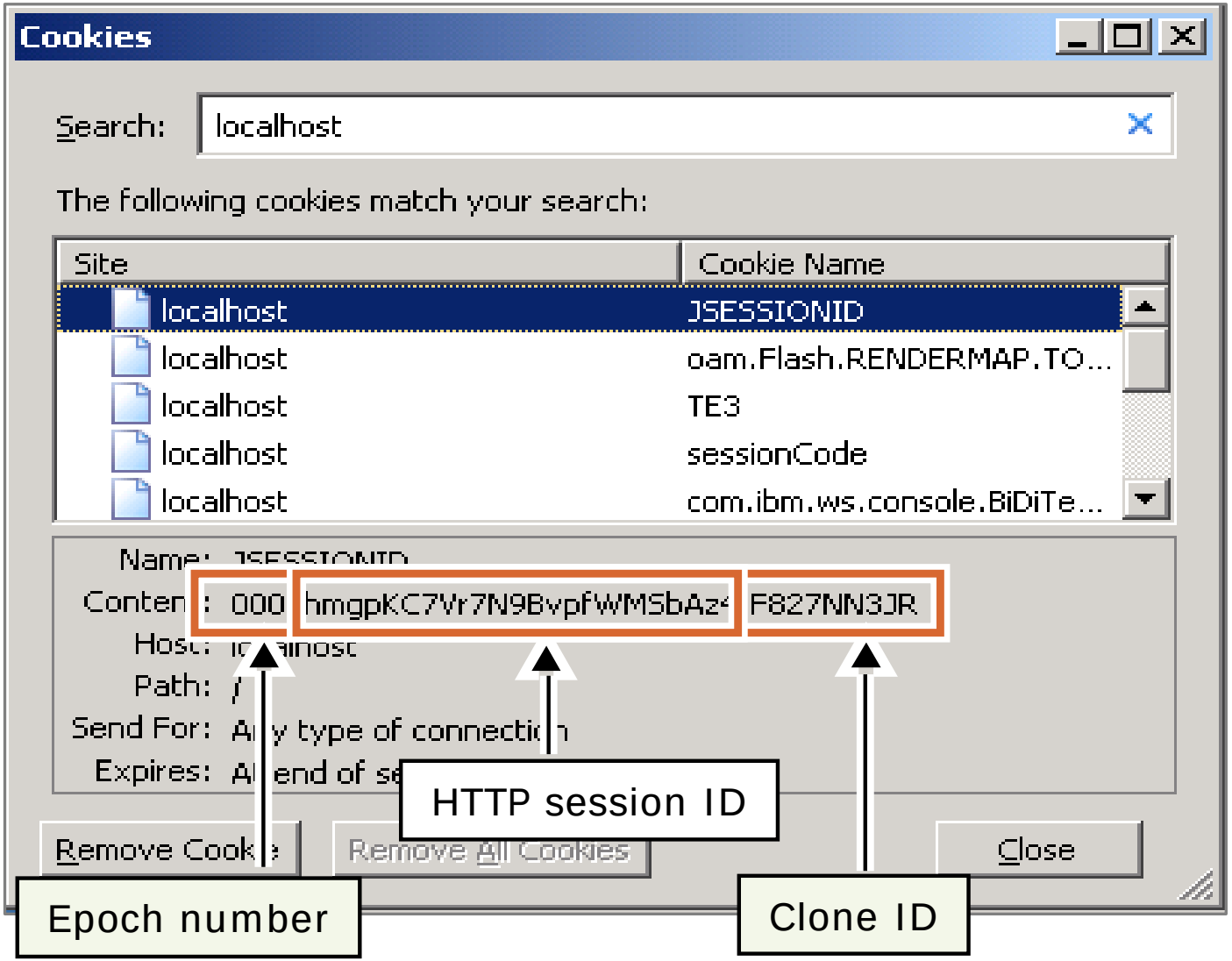


Session affinity

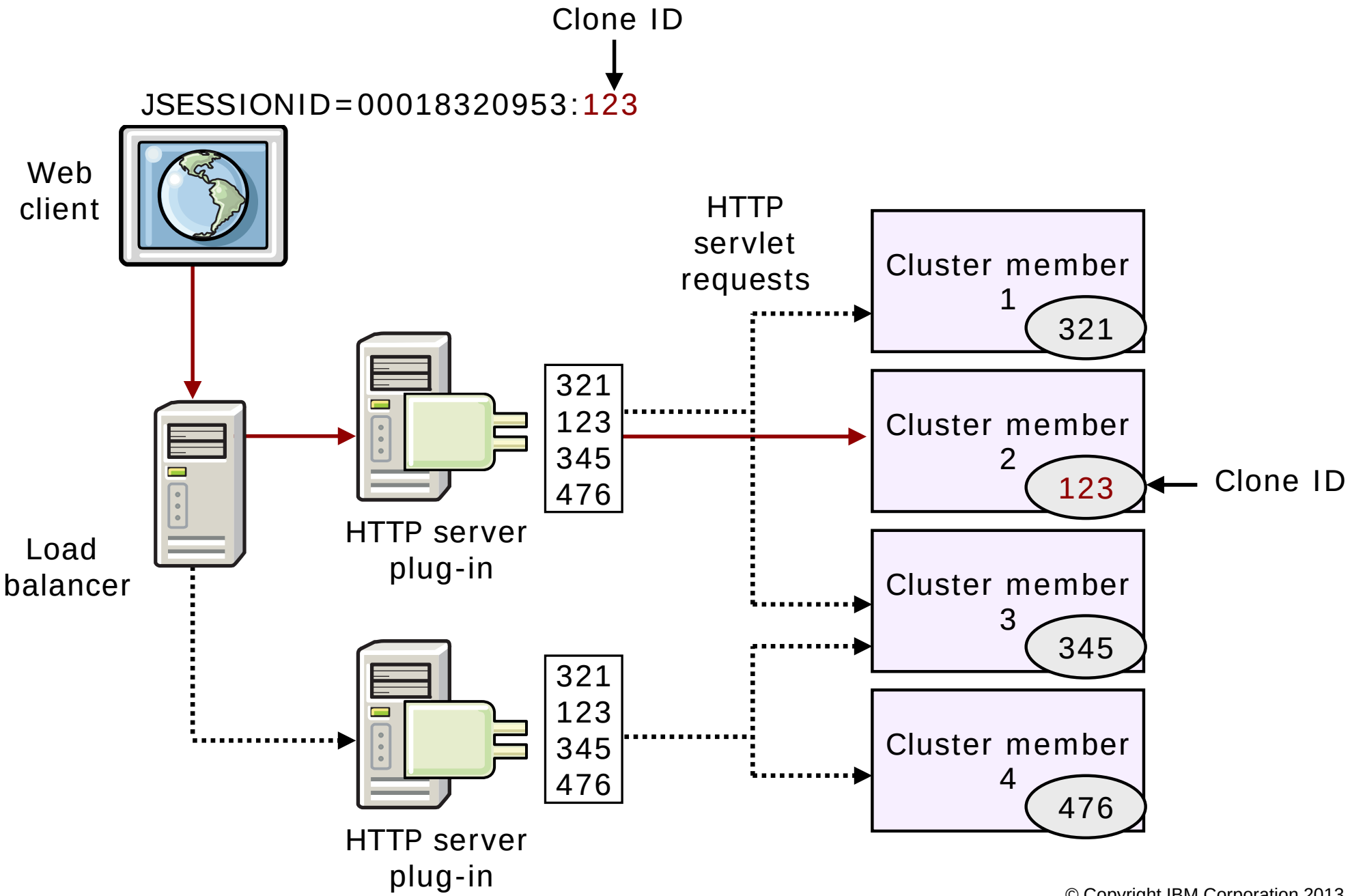


JSESSIONID cookie

- 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

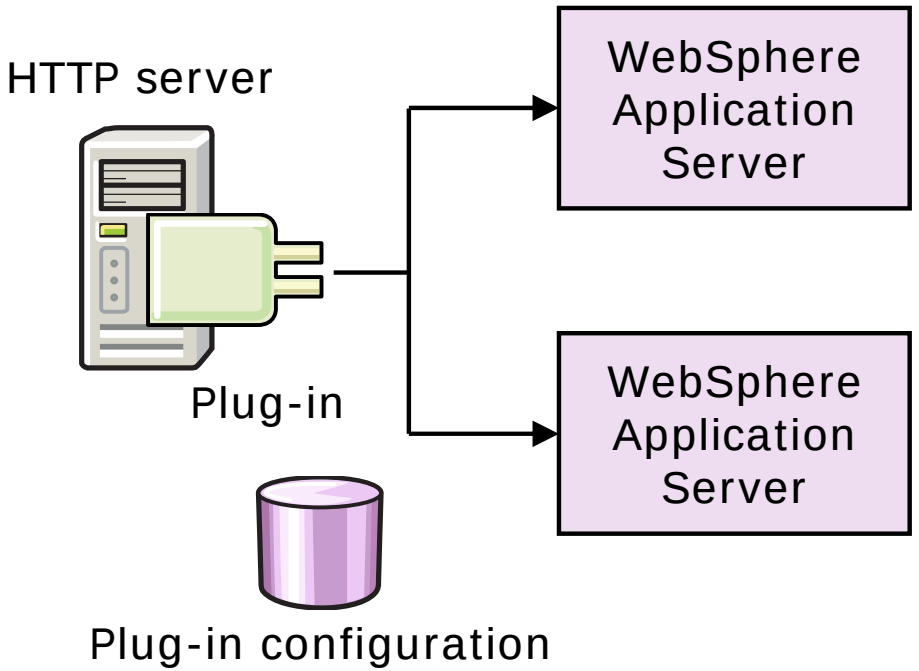


WebSphere session affinity



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



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="*:80"/>
    <VirtualHost Name="*:9443"/>
    <VirtualHost Name="*:5060"/>
    <VirtualHost Name="*:5061"/>
    <VirtualHost Name="*:443"/>
    <VirtualHost Name="*:9081"/>
    <VirtualHost Name="*:9444"/>
  </VirtualHostGroup>
  <ServerCluster CloneSeparatorChange="false" GetDWLMTable="false" IgnoreAffinityRequests="true" LoadBalance="Round Robin"
                 e="64" PostSizeLimit="-1" RemoveSpecialHeaders="true" RetryInterval="60">
    <Server CloneID="13u6hqmf8" ConnectTimeout="0" ExtendedHandshake="false" LoadBalanceWeight="2" MaxConnections="-1"
            ServerIOTimeout="0" WaitForContinue="false">
      <Transport Hostname="was7host01" Port="9080" Protocol="http"/>
      <Transport Hostname="was7host01" Port="9443" Protocol="https">
        <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"/>
      <Transport Hostname="was7host01" Port="9444" Protocol="https">
        <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>
```

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="/ivt/*"/>
    <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="*.jsv"/>
    <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>
```

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="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">
```

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">
```

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">
```

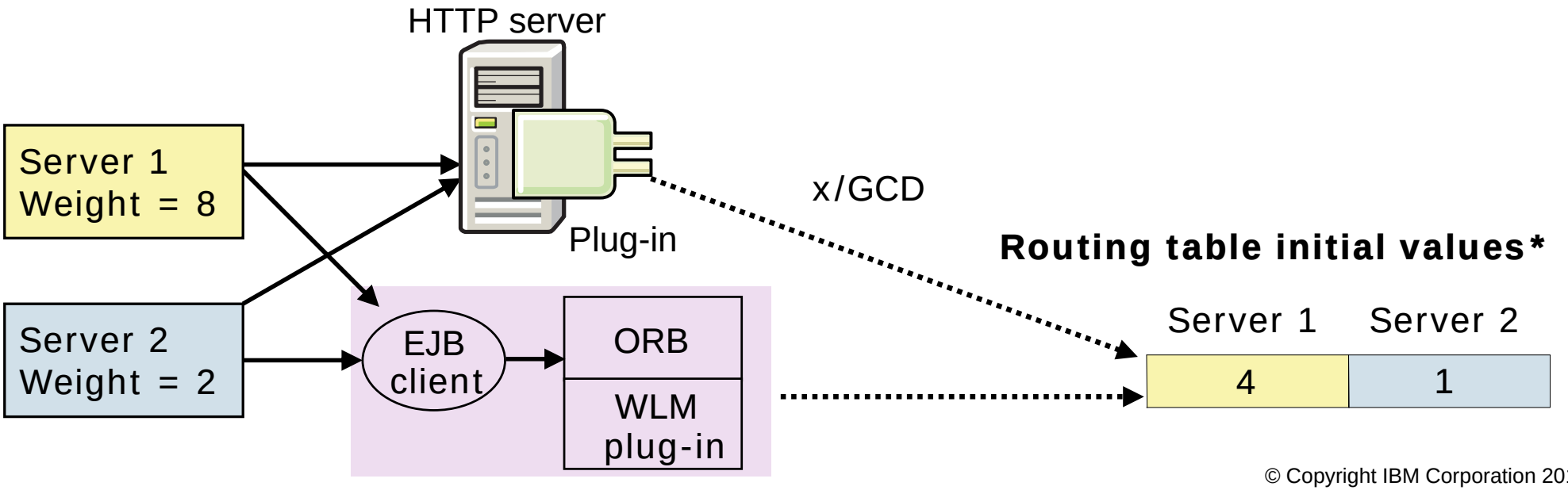
Interpreting the plugin-cfg.xml file (4 of 4)

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="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="9444" Protocol="https">
```

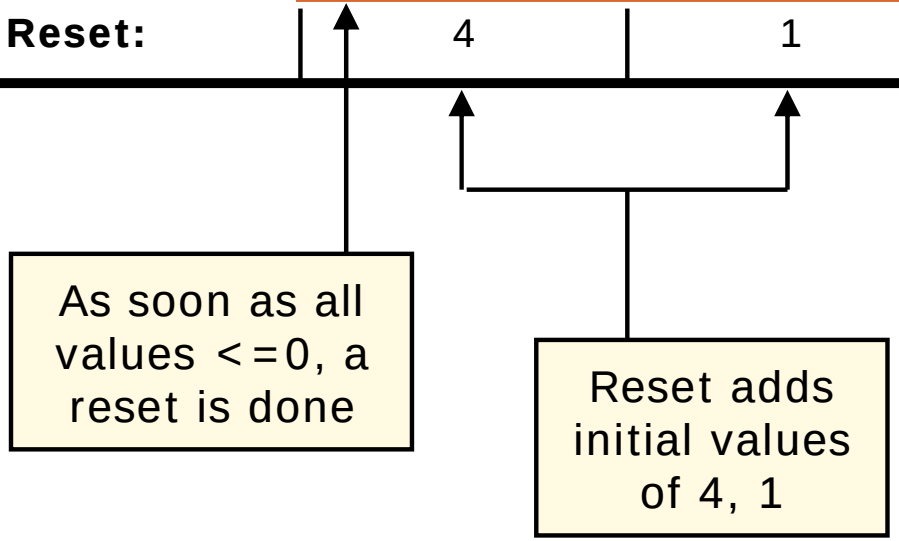
Weighted round robin

- Both the plug-in and EJB routing use weighted round robin by default
  - The plug-in can be configured to use random instead
- Initial weights are given to each cluster member
  - The weights default to a value of 2
  - The maximum weight that is allowed in the console is 20
- An internal routing table is set up for each plug-in
  - Using a greatest common divisor (GCD), the plug-in attempts to reduce the weight values for each cluster member



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



- 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%

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: **

Weighted routing example with counting affinity

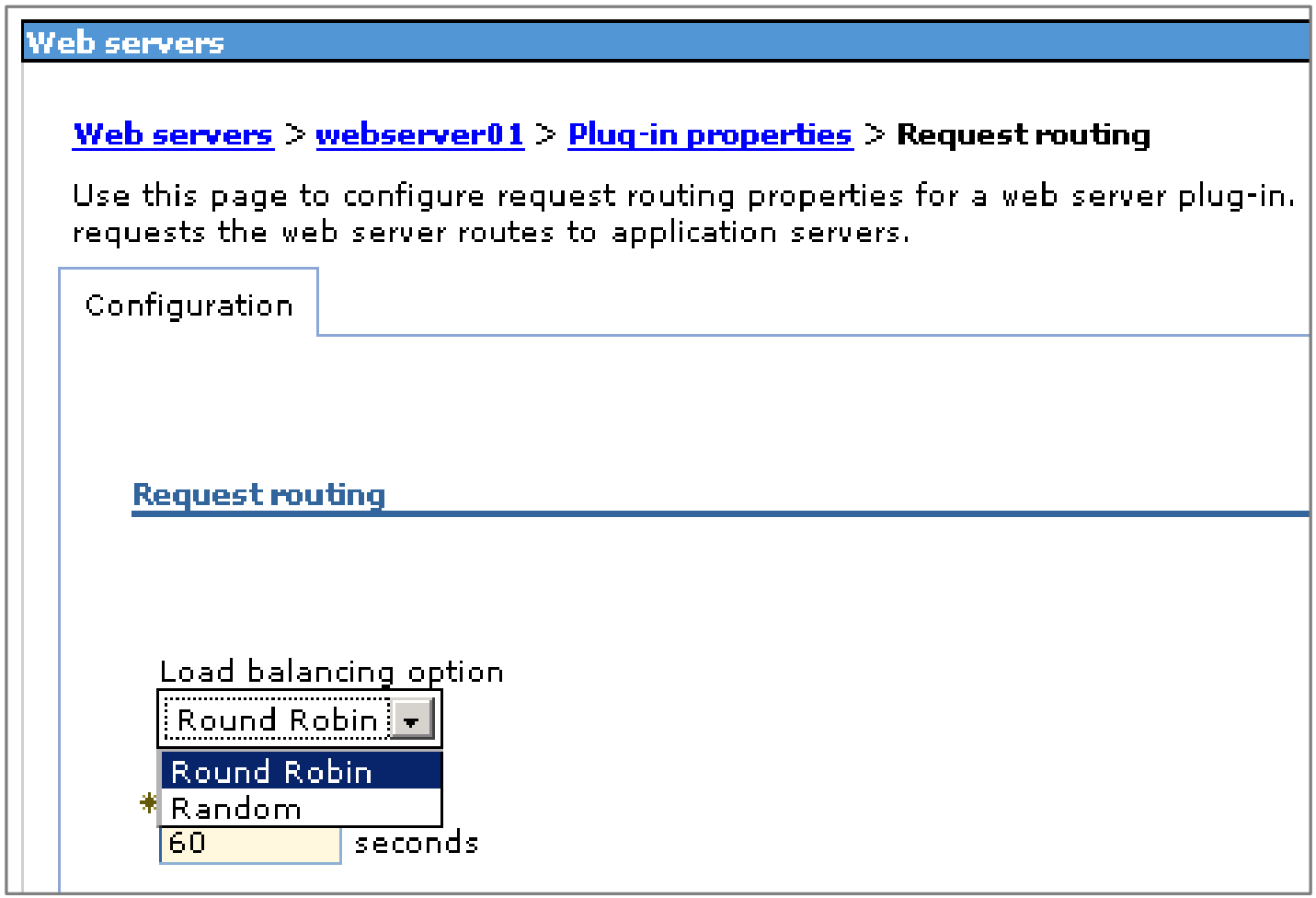
- Example 3: combination of hits
- Setting **IgnoreAffinityRequests** to false
- As requests come into the plug-in, weighted round robin is used
- All requests decrement the weight values in the table
  - Sticky hits are routed to the server for which they have affinity
  - Weighted round robin is used 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: **

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



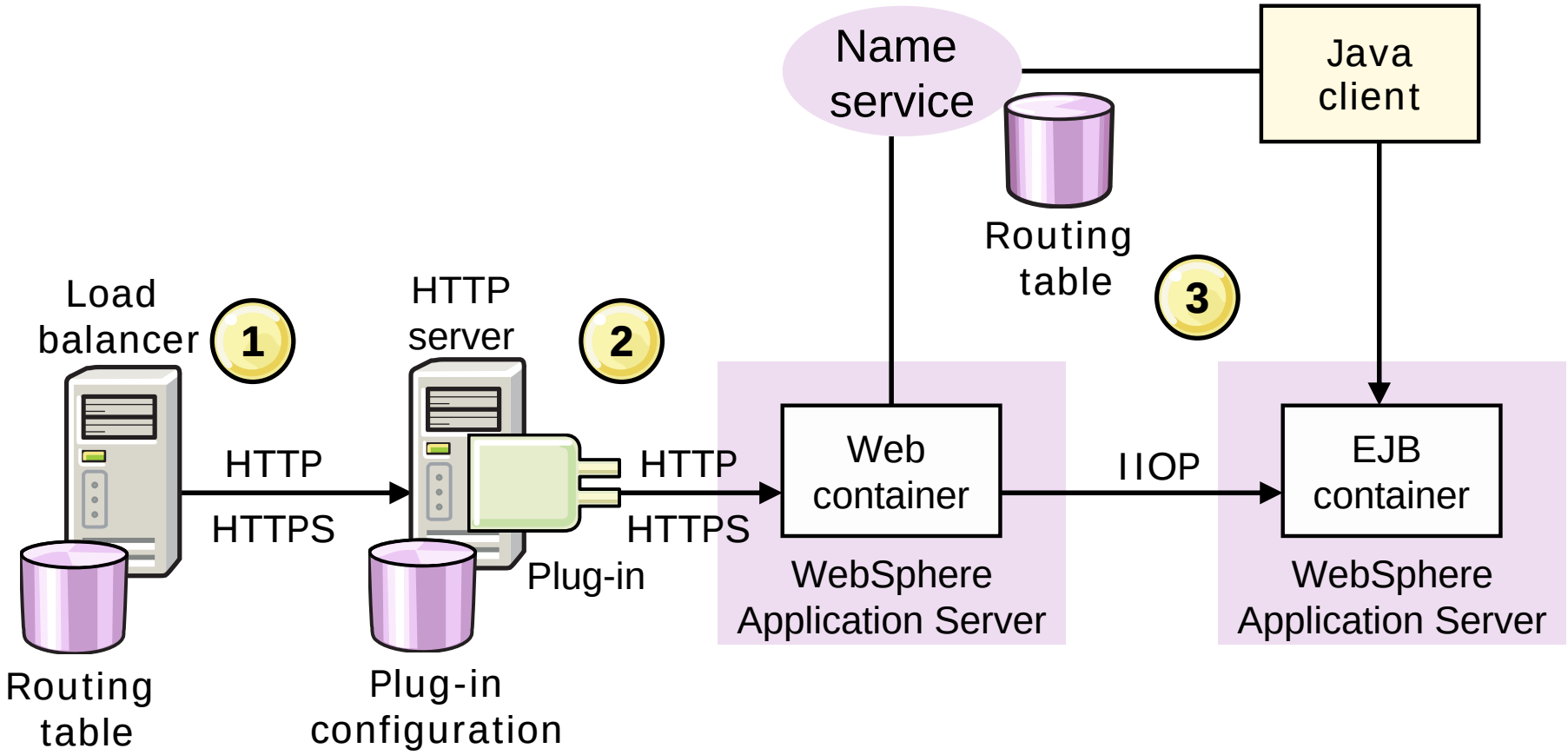
Using Intelligent Management

- 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

Failover

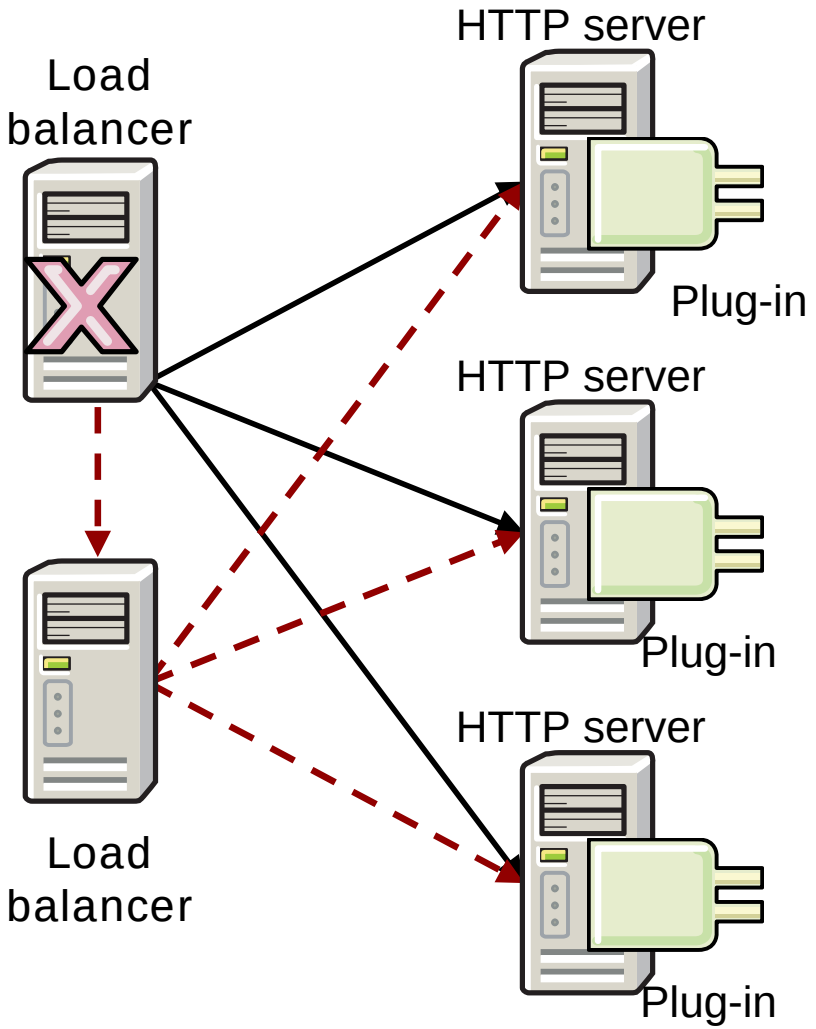
Failover

What happens if there is a failure?



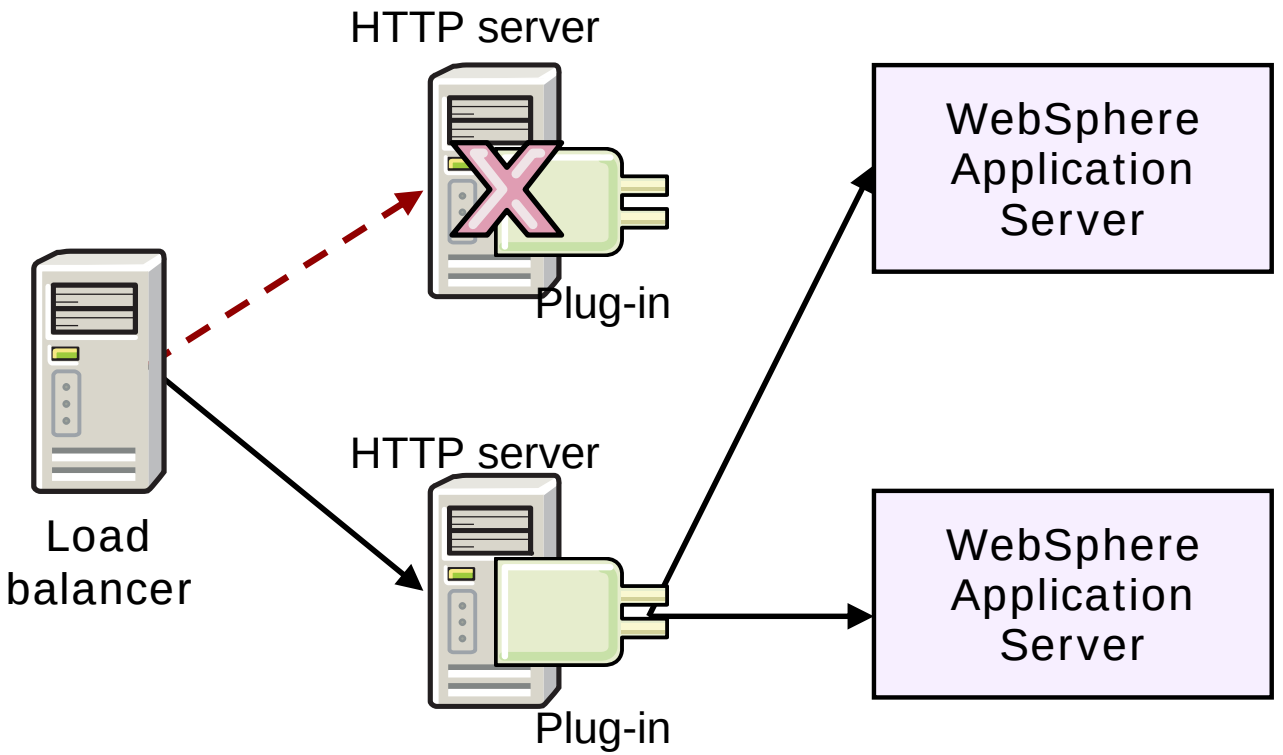
Edge Components failover

- Load balancer can be paired with a backup server
- Topology is active or standby
  - One computer does all the work
  - The other waits for a failure to begin handling routing



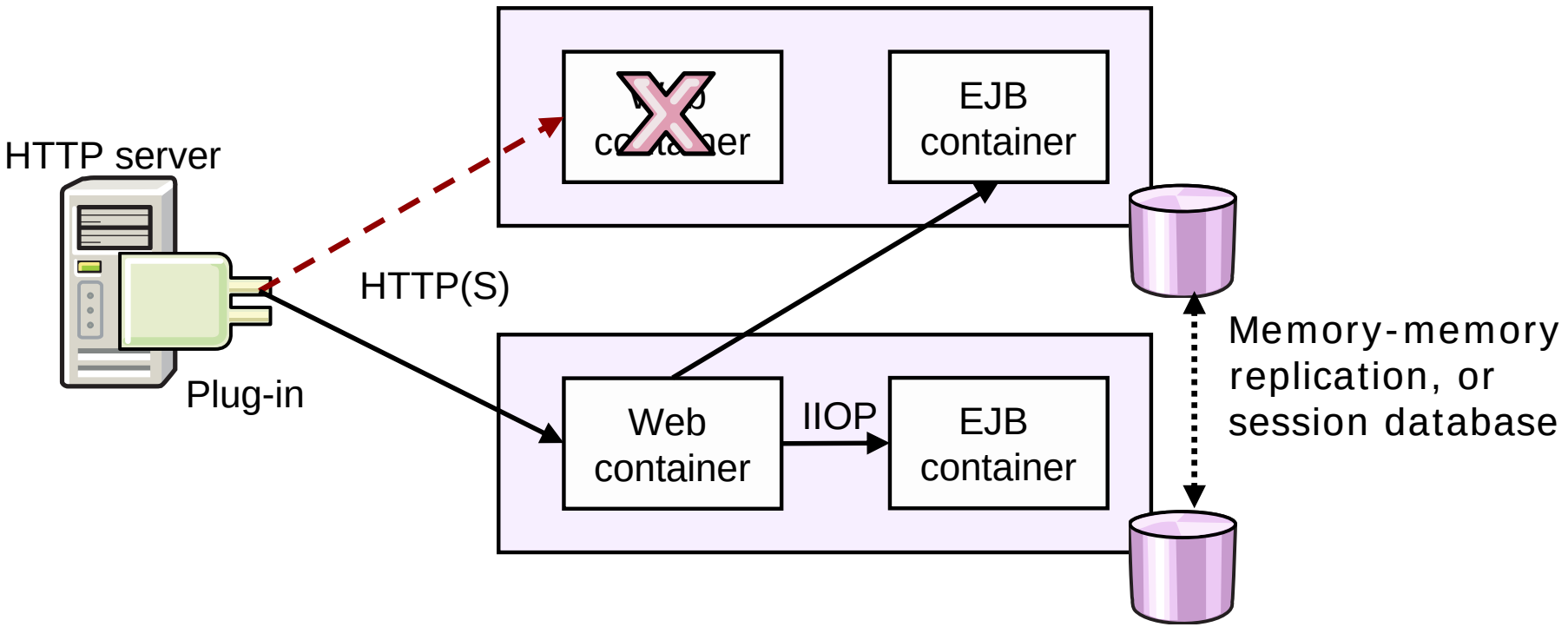
HTTP server failover

- Multiple HTTP servers provide coverage
- Load Balancer can route around a failed HTTP server
- All HTTP servers handle load before failover
- HTTP plug-in
  - Every plug-in knows about all web containers
  - Session key contains address of server
  - Sessions get properly routed



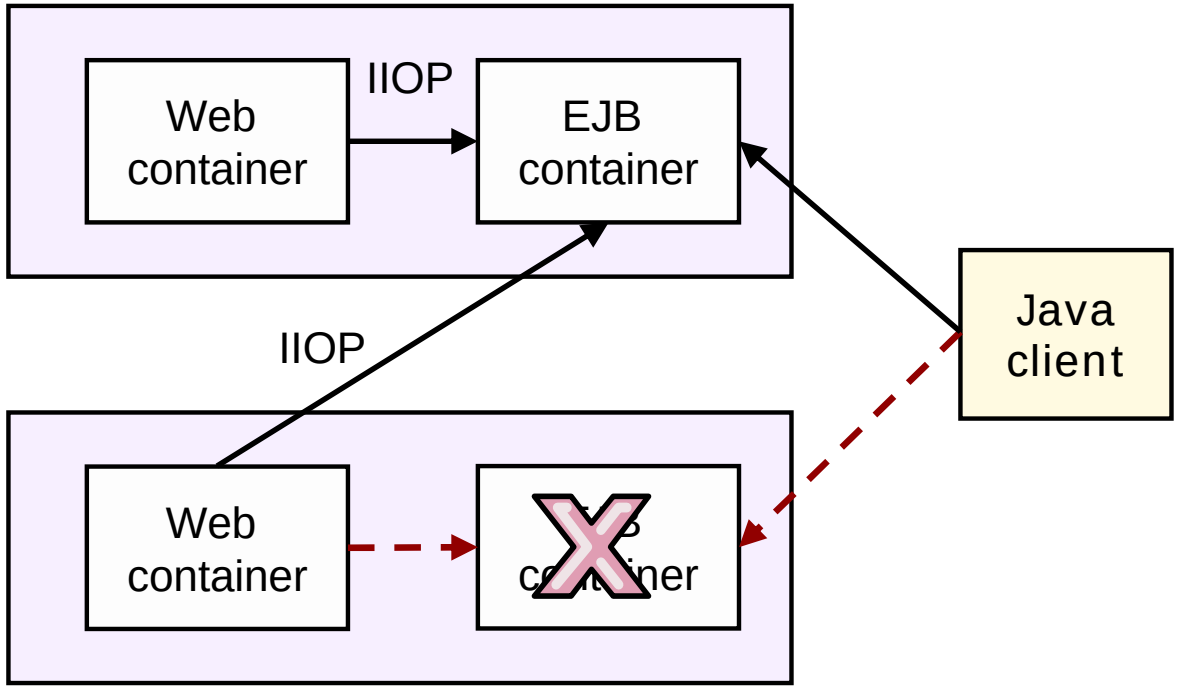
Web container failover

- HTTP server plug-in
  - Detects failure of web containers
  - Marks container as unavailable
  - Tries next cluster member in the cluster
- What about in-flight sessions?
  - Sessions can be persisted to database
  - Sessions can be replicated in memory

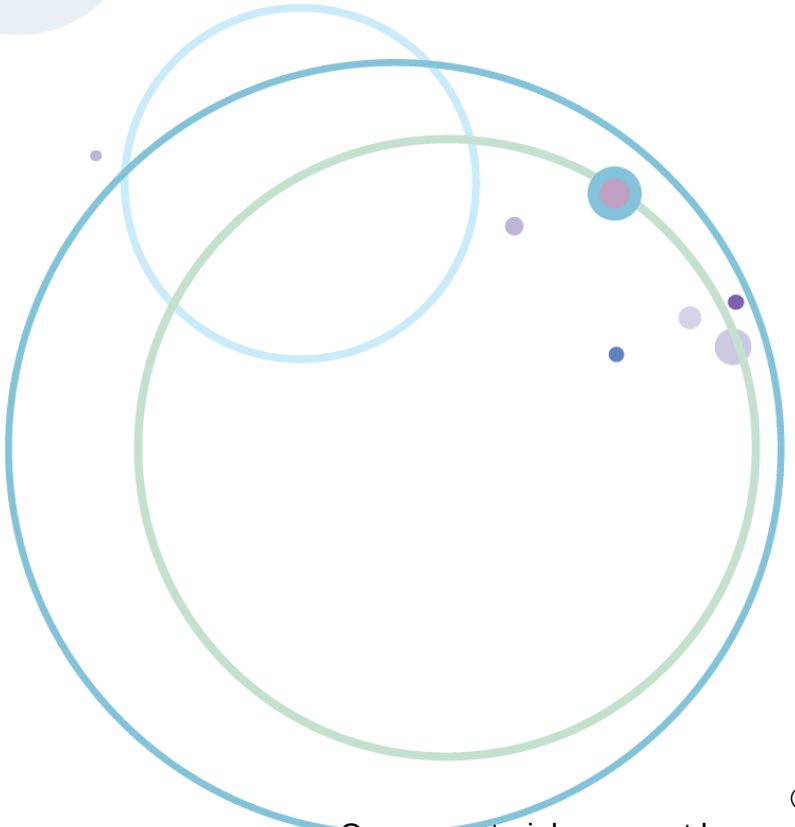


EJB container failover

- 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



Session persistence

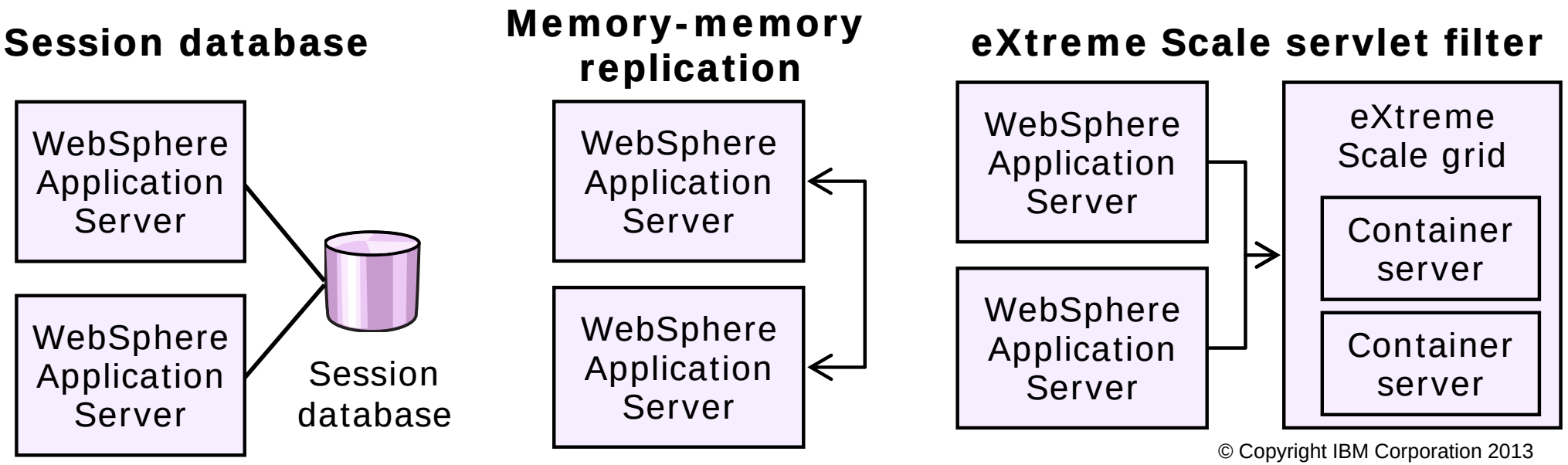


Session persistence

Session objects are cached in server memory by default, and therefore are lost if server fails

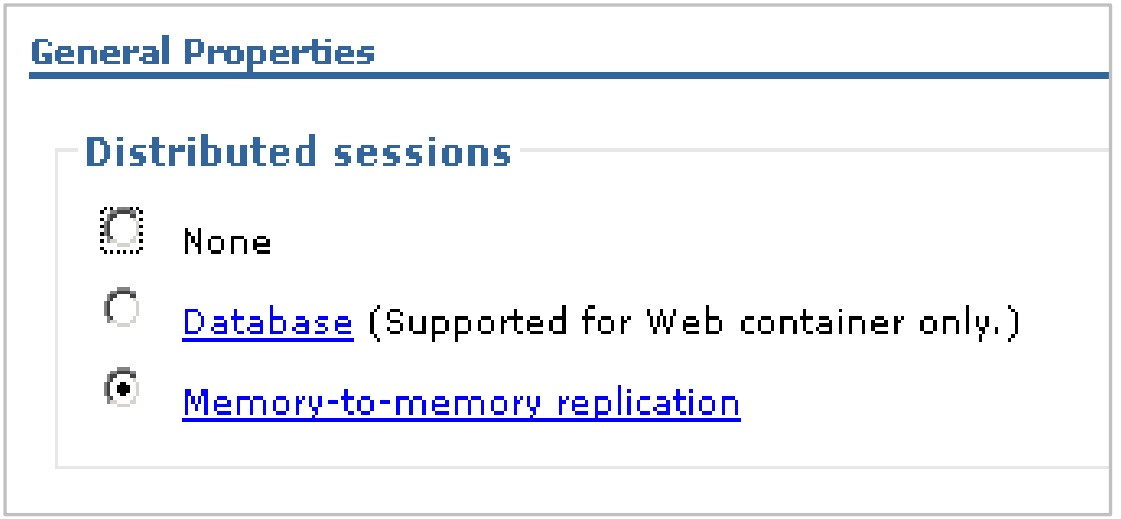
Three methods to enable session persistence:

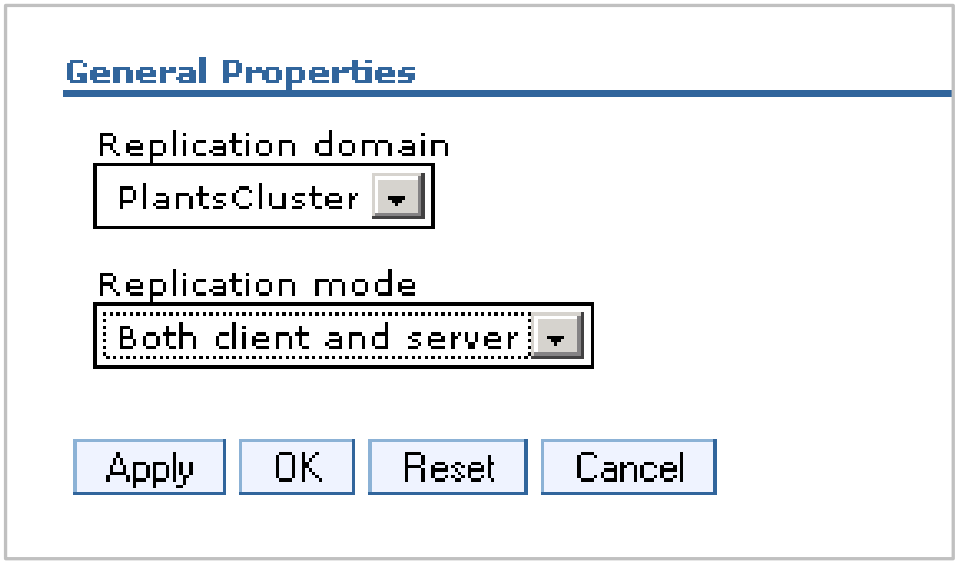
- Database
  - JDBC data source is used to persist session objects
  - DB2 included in package for session persistence
- Memory-to-memory replication
  - Data Replication Service (DRS) is used to copy sessions to another server
- eXtreme Scale servlet filter
  - Entitlement included with WebSphere Application Server



Session configuration: Memory-to-memory

- 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





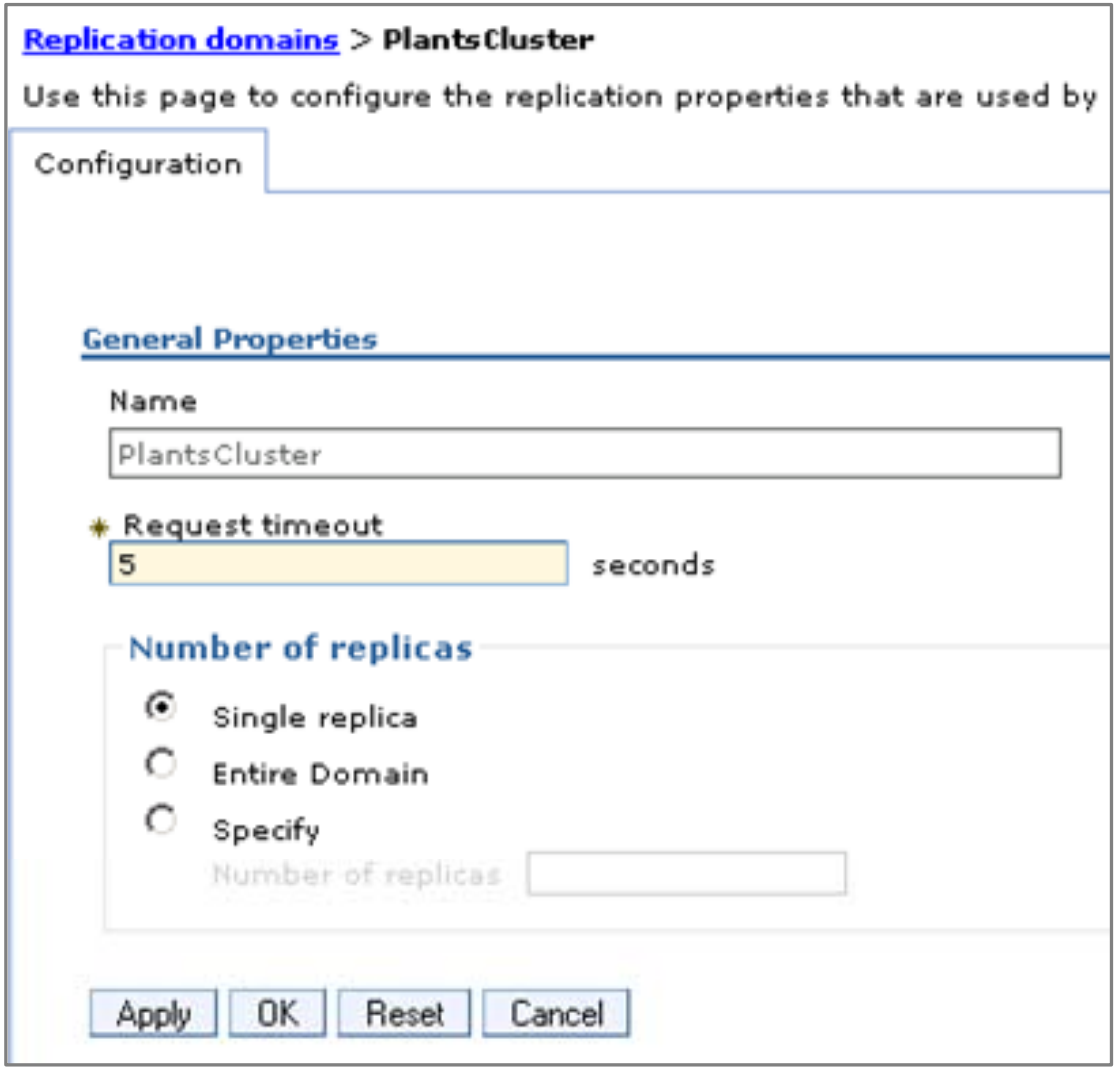
Session configuration: Replication domains

Creating replication domains:

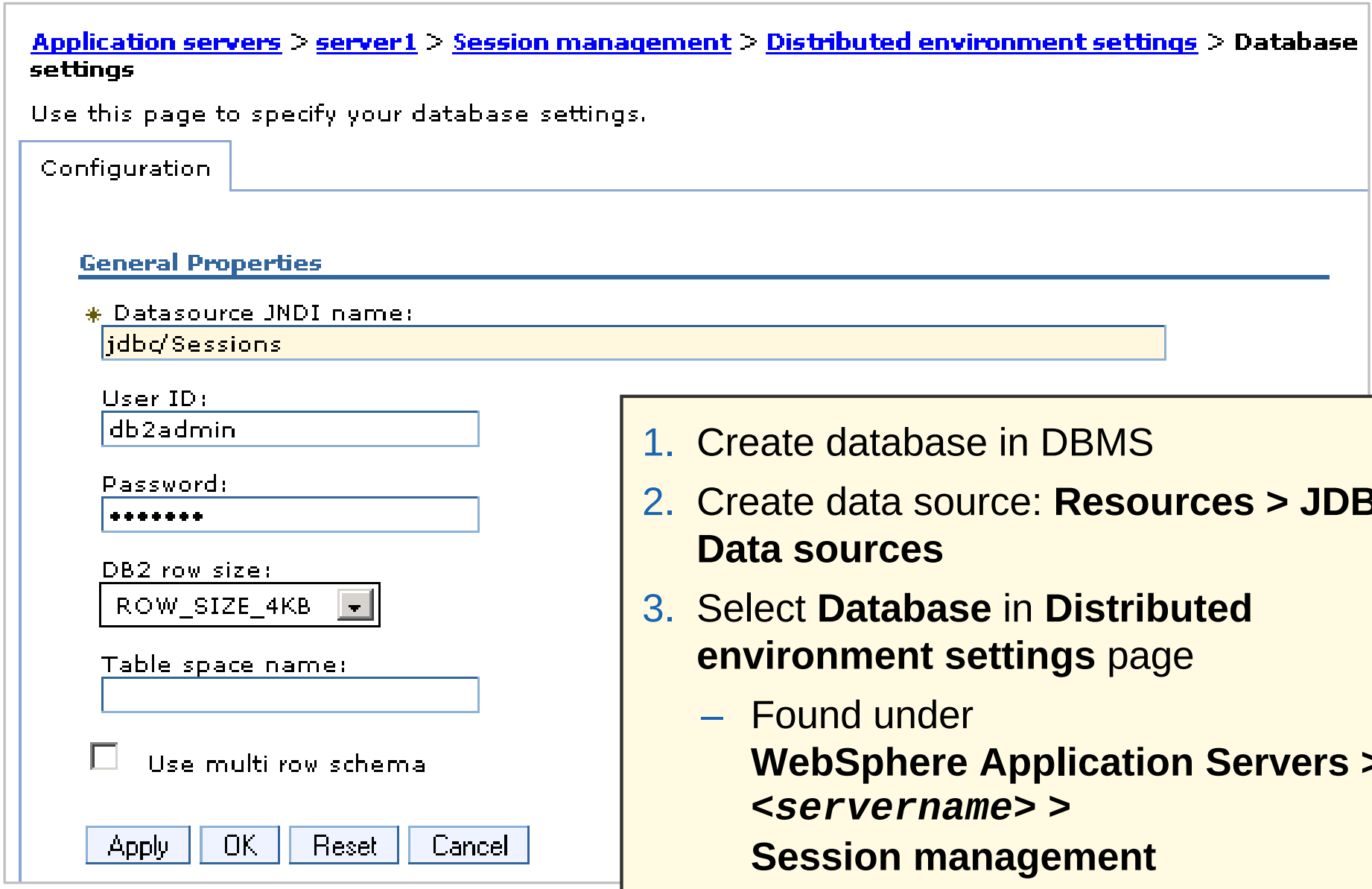
- Created during cluster creation
- Or **Environment > Replication Domains > New**

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



Database persistence configuration

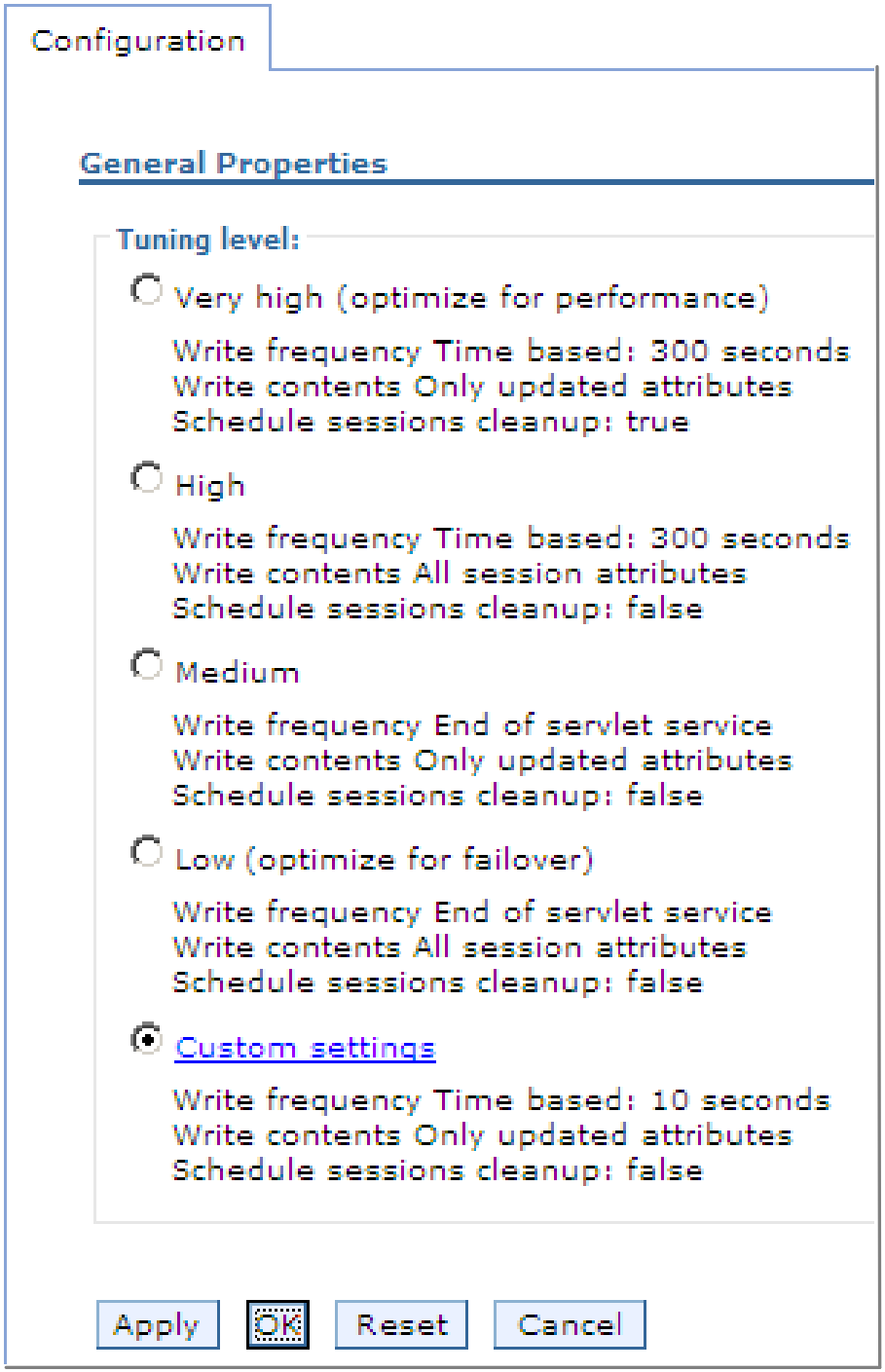


1. Create database in DBMS
2. Create data source: **Resources > JDBC > Data sources**
3. Select **Database** in **Distributed environment settings** page
   - Found under **WebSphere Application Servers > <*servername*> > Session management**
4. Configure database settings

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**



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

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

Checkpoint questions

1. A WebSphere cluster member is what type of process?
   A. An application server
   B. A web server
   C. An edge server
   D. A proxy server

2. The creation of a cluster can be based on which of the following?
   A. An application
   B. An application server
   C. An enterprise application
   D. An application manager

3. Having session affinity means that session information is not lost during failover.
   A. True
   B. False

Checkpoint answers

1. A WebSphere cluster member is what type of process?
   A. An application server

2. The creation of a cluster can be based on which of the following?
   B. An application server

3. Having session affinity means that session information is not lost during failover.
   B. False

Exercise 10

Clustering and workload management

Exercise objectives

After completing this exercise, you should be able to:

- Create a cluster and add cluster members
- Map modules to clusters and web servers
- Test load balancing and failover between two cluster members
- Configure a data replication domain for session management