



Workload Management / High Availability

WebSphere. software



 e-business software

Updated 3/08/2003

© 2002, 2003 IBM Corporation

Objectives

- **Define Workload Management**
- **Describe Clusters & Cluster Members**
- **Examine the Topology**
- **Discuss Weighted Servers**
- **Determine Failover Scenarios**
- **Revisit the HTTP Plug-in**
- **Present Problem Determination**

In this module, we will begin with a simple definition of what Workload Management is, then look at the flow of requests through the WebSphere topology. We will look at several configuration options, see how the new weighted routing algorithm directs traffic, and finish up with a short look at some possible Problem Determination guidelines.

What is Workload Management (WLM) ?

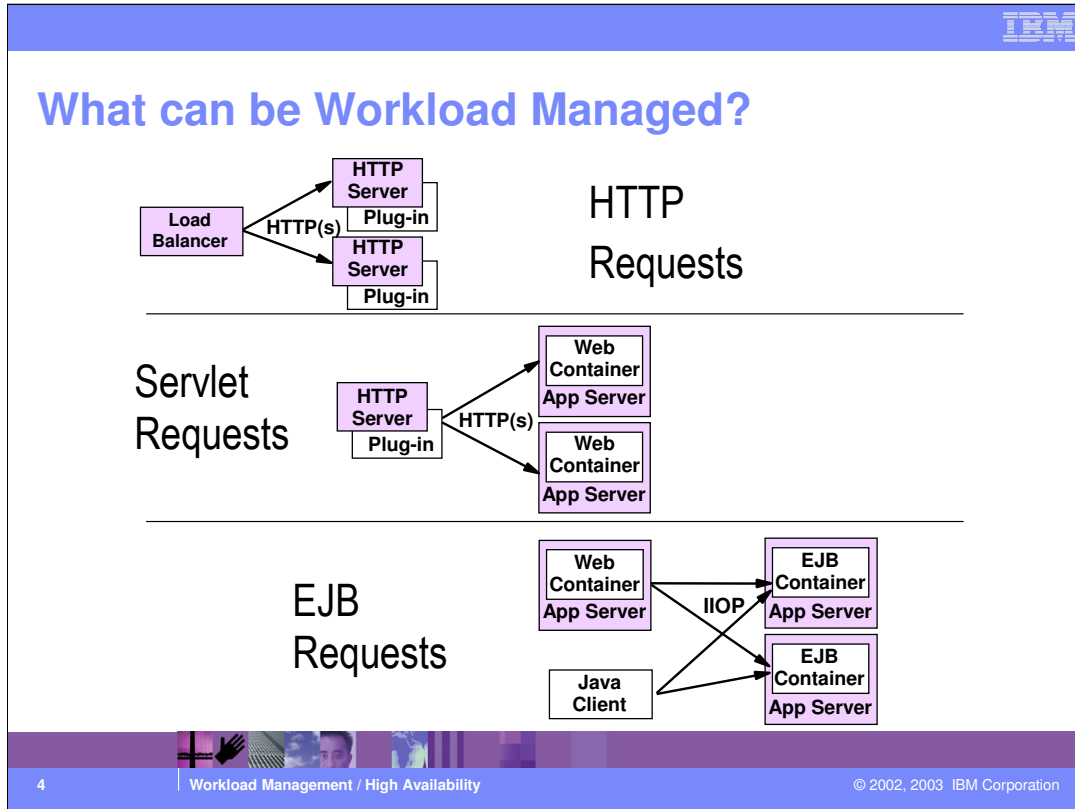
- **Sharing requests across multiple Application Servers**
- **Configuration options that improve**
 - Scalability - serve more users
 - Load balancing - allocate workload proportionately among available resources
 - Availability - system runs if server fails

Workload management optimizes the distribution of client processing requests.

Incoming work requests are distributed to the application servers, enterprise beans, servlets, and other objects that can most effectively process the requests.

Workload management also provides failover when servers are not available, improving application availability.

In the WebSphere Application Server environment, workload management is implemented by using Clusters of application servers.



Three types of requests can be workload managed in WebSphere v5.0.

HTTP Requests can be shared across multiple HTTP Servers.

This requires a TCP/IP sprayer to take the incoming requests and distribute them. There are both hardware and software products available to spray TCP/IP requests. Network Dispatcher is a software solution that is part of the WebSphere Edge Server. Network Dispatcher applies intelligent load balancing to HTTP requests.

Servlet Requests can be shared across multiple Web Containers.


The WebSphere Plug-in to the HTTP Server distributes Servlet requests.

Web Containers can be configured on the same machine or multiple machines.

EJB Requests can be shared across multiple EJB Containers.

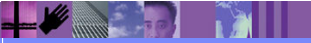
The Workload Management Plug-in to the Object Request Broker (ORB) distributes EJB requests.

EJB requests can come from Servlets, Java client applications, or other EJBs.



WLM - What's New In V5.0

- **HTTP Requests**
 - Handled as in 3.x and 4.0.x - Edge Components
- **Servlet Requests**
 - Cluster Replaces Model/Clone & Server Group/Clone
 - 1 to N Clusters per WebSphere cell
 - Primary/Backup Server Lists for HTTP Server Plug-in
 - Improves HTTP Session Failover Routing
 - Server Weighted Round Robin Routing
 - Replaces Random and Round Robin for HTTP & EJB WLM
 - Plugin-cfg.xml is the source for routing table
- **EJB Requests**
 - Location Service Daemon is the source for routing table



5 | Workload Management / High Availability | © 2002, 2003 IBM Corporation

As mentioned on the last foil, HTTP Requests are workload managed externally to WebSphere Application servers.

For Servlet requests, 'Clusters' replace the Model-and-clone topology of version 3 and the similar Server Group-and-clone topology of version 4. You can configure multiple Clusters in a cell, multiple cluster members within a cluster, as logic dictates for a given scenario. The HTTP server plug-in reads a list of servers it can route to from the plugin-cfg.xml file. In the unlikely event that ALL the servers fail to respond, the plug-in also has a back-up server list to route to.

Each of those servers also has an associated weight, which we will discuss momentarily. The routing option is a weighted round robin. Previously, we had two options - random and round-robin. After a few years in production environments, no one was able to come up with a scenario where one was preferable over the other, but the difference in capacity of machines pointed out a need for a weighted routing algorithm, so the decision was made to abandon the prior routing schemes in favor of this one.

EJB Requests can also be routed among EJB containers. The Location Service Daemon process is responsible for the routing table, which can have entries for servers in other clusters.

Cluster Management

▪ Definition of a Cluster

- Clusters are a set of application servers having the same applications installed, and grouped logically for Workload Management
- Cluster members are similar to 'Clones' in that they:
 - run the same applications
 - share workload
 - can be centrally administered
- Cluster members are Different from 'Clones' in that:
 - Changes to the template are NOT propagated to the cluster members
 - Changes to cluster members cannot disassociate the cluster members

By default, you can only install one copy of the application server binaries on a machine. Once those binaries are installed, you can have multiple application servers configured - the data needed for each additional server is stored in several XML files, and uses up about 50 K of disk space.

Several application servers can run on a single machine - but there is no requirement that they all be in the same cluster. Clustering is a logical grouping, not a physical one. All members of a cluster are nearly identical 'clones' of a common ancestor.

Clarifying Clusters - What's the Difference?


- **WebSphere 3.5 Models and 4.0 Server Groups were Active Templates**
 - Changes to the template were propagated to the Clones
 - Installing applications required manual updating of application binaries
- **WebSphere 5 uses 'passive template' concept**
 - Once a cluster member has been created, no changes to the template are propagated
 - Installing applications - binaries are copied to the nodes during synchronization

Since there may be some confusion with the shift from Clones to Cluster members, let's clarify a few points.

In WebSphere version 3, you could make changes to a clone that would sever the relationship to the model. In version 4.0, you could edit some fields on the clone, but not the ones that had to be tied to the template. But if you edited the template, those changes would be propagated to the clones. In 3.5 and earlier, the template was called a 'model'. In 4.0, the template was called a 'server group'.


In WebSphere 5.0, you can no longer change so much about a cluster member that it ceases to be a cluster member. You are not limited as to what properties of that cluster member you can change. The cluster member is an application server. What makes it a cluster member is internal - when you define a cluster member, you can copy an existing application server, and thereby start with all the settings of that server. Then you can change the cluster member however you can, and it will always be a cluster member. There is no way to make a cluster member be no longer a cluster member, except to delete it entirely.

Applications may be installed to an application server, OR to a cluster. However, if an application server is a member of a cluster, you cannot install applications directly to it as if it were a stand-alone. Another difference is that the synchronization process propagates application binaries as needed, so the administrator does not need to manually move EAR files to remote nodes.



Modification of Clusters

- **Done in the Admin console**
- **Changes propagated at next synchronization after Save**
- **What can be changed in a Cluster?**
 - Change server weights
 - Toggle Prefer Local setting
 - Install applications
 - Cluster members are still Application Servers, and can be manipulated as such
- **Remember to regenerate HTTP Server plug-in after changes**




8 | Workload Management / High Availability | © 2002, 2003 IBM Corporation

Syntax for generating the plug-in from the command line: `genplugincfg.bat -cell.name <NetworkDeploymentCell>`

Making modifications to cluster members AS CLUSTER MEMBERS (that is, under 'clusters' in the navigation panel) is somewhat limited - You can install applications or modify applications, change server weightings for the routing table, or change the Prefer Local setting. However, you can make any change you wish to a cluster member as an application server.


If you change the routing options, remember to manually regenerate the plug-in to the HTTP server, using the `genplugincfg` tool, or the admin console. You do not need to stop the HTTP server - the changes will be loaded when the next `ReloadInterval` timer expires.

There is no warning message in the Admin Console to alert the user of the need to update the plug-in.



Creating a Cluster

- **Start with an existing server configuration**
 - that server may become the first cluster member
- **Additional servers are created from templates**
 - i.e., copies of existing servers
- **Servers-> Clusters -> New**



9 | Workload Management / High Availability | © 2002, 2003 IBM Corporation

Let's start with a few simple screen shots to walk through the process of creating a cluster. In the Network Deployment admin console, we click on Servers, then Clusters. In the Work Area panel, we'll click on 'New'.

As in earlier releases, the first step is getting a template, or pattern, for the servers we will create in this cluster. Making the original server part of the cluster is an architectural decision - The default is no, unlike 4.0, where there was no option, and the first server was in the server group whether you wanted it there or not. If you need to delete a cluster for some reason, it might be nice to have the template available. But if you do NOT add it to the cluster, make sure you set it's initial state to 'Stopped', so that it doesn't handle requests that were not intended for it.

Creating a Cluster (continued)

- **Check options**
 - Prefer Local (Default)
 - Create Replication Domain
- **Select the 'First member' (Optional)**

→ Step 1: Enter Basic Cluster Information

Cluster name:	<input type="text" value="MyCluster"/>
Prefer local:	<input checked="" type="checkbox"/> Prefer local enabled
Internal replication domain:	<input checked="" type="checkbox"/> Create Replication Domain for this cluster
Existing server:	<input checked="" type="radio"/> Do not include an existing server in this cluster <input type="radio"/> Select an existing server to add to this cluster
	Choose a server from this list: <input type="text" value="SystemBNetwork/SystemB/server1"/>
	Weight: <input type="text" value="2"/>
	<input type="checkbox"/> Create Replication Entry in this Server

Add Servers to Cluster

- **Fill in Name**
 - this server will be created on the selected node
- **Assign load balancing weight**
- **Generate Unique HTTP ports (default)**
- **Select to generate replication entry (optional)**
- **Select template -**
 - existing application server or
 - pre-defined template
- **Click "Apply" to add to the list**
- **Repeat**

Step 2: Create New Clustered Servers

Enter information about the new server below, and then use the Ap cluster. Use the Edit button to edit the properties of a server already

Name:

Select Node:

Weight:

Http Ports Generate Unique Http Ports

Replication entry: Create Replication Entry in this Server

Select template:

 Default application server template

 Existing application server

11 | Workload Management / High Availability | © 2002, 2003 IBM Corporation

Here we create new servers from the templates available. Give it a name, a server weight, and select which template you are going to copy. Each time you click Apply, the servers will be added to the list. If you click Edit, the server you have configured will be copied back to the upper panel for you to make changes and hit Apply again.

The 'Select Template' option is only available until you select one; you cannot select a second template for a cluster member within the cluster.

You can create Cluster Members on any node in your topology. The weighting you give each will determine how much of the workload will be routed to that cluster member.

The 'Unique' in Generate Unique HTTP Ports is specifically unique on the node where it will run, not unique within the cell. Two cluster members can have the same port number, as long as they reside on different physical boxes.

Replication Entry is for Memory to Memory session replication. This can be created automatically when creating a cluster, or manually at any time.

Creating a Cluster - Finish

- **Click "Finish"**
- **Save**
 - Changes will be propagated with the next synchronization, unless you check the box to Synchronize changes with Nodes
- **Regenerate the HTTP server plug-in**
 - Remember to manually edit Stashfile and Keyring paths as per the release notes
 - Changes will be effective at the next Reload Interval , or when you re-start the http server

Review the cluster members and click Finish. Save your changes.

Finally, regenerate the HTTP server plug-in, edit if needed, and copy it to the location where the HTTP server expects to find it. Note that the Release Notes may be a little confusing at this point – Fixing the path for the Stashfile and Keyring entries is listed as needed for IBM HTTP Server 2.0, but in fact is also necessary for IBM HTTP Server 1.3.26.

Installing / Updating Applications to a Cluster

- **Same steps as Installing to base server except:**
 - select a Cluster as the target, rather than select a server
- **Application files (binaries and configuration files) are copied at next synchronization**
 - Behavior can be changed in the console
- **Servers may be ripple-started**
 - Restarts cluster members one at a time

Updating applications to a cluster is done in the same manner as updating applications to a stand-alone server.

Basic WLM Request Routing

```

    graph TD
      LB[Load Balancer] -- HTTP(s) --> HS[HTTP Server Plug-in]
      HS -- HTTP(s) --> WCA[Web Container App Server]
      WCA -- IIOP --> EJA[EJB Container App Server]
      subgraph RD [Routing Decision Points]
        LB
        HS
        WCA
        EJA
      end
  
```

- **Load Balancer**
 - Routing decision table stored internally
 - Configurable with NAdmin tool
 - Multiple intelligent routing options
- **HTTP Server Plug-in**
 - Routing table part of plugin-cfg.xml
 - Configured with Admin web app or wsadmin scripting tool
- **WLM-aware Client**
 - Includes Web Container, Java client, EJB
 - Routing table supplied by LSD
 - Configured with Admin web app or wsadmin scripting tool
 - Options:
 - Prefer Local - yes or no

14 | Workload Management / High Availability | © 2002, 2003 IBM Corporation

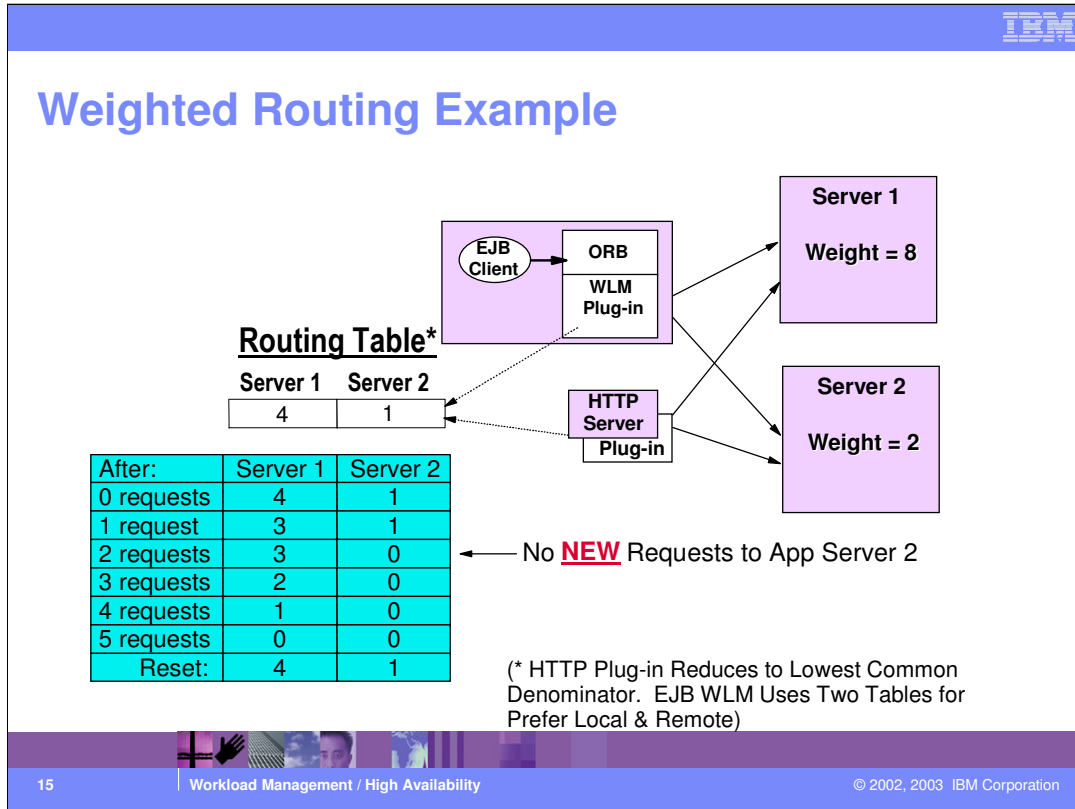
Now we'll address request routing.

Let's first look at the Fair Weather scenario - Assuming that everything works. We'll address failover scenarios in a few slides.

Edge Component's Load Balancer is an IP sprayer that makes intelligent load balancing decisions. Using the NAdmin tool, you can set it up to route to your HTTP servers based on Round Robin, Statistical Round Robin, Best, Custom Advisor, or Content Based Routing.

Once the request arrives at an HTTP server, the routing is Weighted Round Robin - the only configuration option is how much 'weight' to give each server. The routing information, the list of available servers and their weights, is ultimately stored by the Deployment Manager. This configuration is used to create the plugin-cfg.xml file for the HTTP server plug-in to read.

WLM-aware Clients include the Web Container, and stand-alone Java clients and EJBs running from WebSphere containers. The Location Service Daemon runs in the Node Agent and is responsible for providing clients with the routing table for EJB Containers. Again, the information comes from the Deployment Manager, and is configured in the Admin Console. Server weights and the Prefer Local are the configurables.



Weighted routing is fundamentally the same concept for HTTP requests, or for EJB client requests, so the two are combined on this slide.

When the HTTP Server plug-in is generated, servlet request routing weights are written into the plugin-cfg.xml file, which the HTTP server will reload at configurable intervals.

There is a distinct Routing Table for each cluster. When a client requests an IOR for an EJB, the Location Service Daemon returns the IOR and a copy of the routing table. The client uses two in-memory copies of the table - one static, one dynamic. The static copy is used only to cache a local copy of the weights.

The table illustrated here is the dynamic one - it has two entries, Server 1 and Server 2. The initial values are 4 and 1. The first new request will be sent to Server 1, and the counter for Server 1 will be decremented by one. The next request will be routed to Server 2, and the count for server 2 will be decremented. After that, Server 2's values is zero, so no new requests will be sent to Server 2 until Server 1's value has decremented to zero. At that point, the table will be reset and things start over.

Basically, the routing is Round Robin for all servers with non-zero table values.

The HTTP Server plug-in behaves a little differently from the EJB routing in one respect, though... If you set the values to 8 and 2, the HTTP Plug-in will 'normalize' the table, and behave as if you had used 4 and 1. EJB routing uses exactly what you

Weighted Routing - Mechanics

- **HTTP Plug-in and the ORB in the EJB client has a Routing table for each cluster**
- **Routing Table Decrement on each New request**
- **No New Requests to the App Server, once its Routing entry reaches Zero, except when overridden by:**
 - Affinity (Transaction, HTTPSession)
 - In Process (Handled by ORB)
 - Prefer Local
- **When All Servers Reach Zero, Table is Reset**
- **Suggested Best Practice**
 - Utilize Low Values to Avoid Load Variations
 - Plug-in Will Use Least Common Denominator to Minimize Variation

One thing to emphasize is that only NEW requests are subject to the weighted routing - requests that have sessions already in progress will be sent to the server that started the session. Either session Affinity or transaction affinity will override the routing.

There are actually two tables passed when Prefer Local is set - only the 'Local' table is used, unless all the servers in the Local table have failed. At that point, the other table comes into play.

Since the difference between the values in the tables is handled sequentially, it is a good idea to use small numbers for the weights.

The picture here is expected to change a little for Fixpack 1. All requests, not just new requests, will decrement the table. And rather than resetting the table, the original values will be added to whatever values are there when all the server values become non-positive. This algorithm better handles sessions that have a large variation in the number of subsequent requests, and allows a saturated server some breathing space.

Weighted Routing Configuration

- **Get/Set Weights with JMX or Admin Console**
- **HTTP Server Plug-in**
 - LoadBalanceWeight Attribute Added to Plugin-cfg.xml
 - established when adding app server to cluster
 - can be edited in the Admin Console
 - Changes Effective at Plug-in Refresh - no need to restart the HTTP Server
- **EJB Container clients**
 - Network Deployment - Changes Require Application Server Restart

Weighted routing is configured on the cluster member. Those weights are available to both the HTTP Server - by way of the plug-in - and the Location Service Daemon that runs in the Node Agent process.

The weights show up as the LoadBalanceWeight value in the plugin-cfg.xml file. Since the HTTP server routinely reloads the plugin-cfg.xml at intervals, the changes will get picked up automatically.

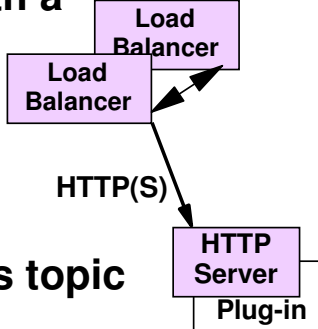
EJB Container clients get the weights in tables sent back after an IOR request.

Failover Scenarios

IBM

Edge Server Failover

- **Load Balancer can be paired with a backup machine**
- **Topology is 'Active/Standby'**
 - One machine does all the work
 - The other waits for a failure to begin handling routing
- **More detail in Edge Components topic**



The diagram illustrates an Active/Standby topology for load balancers. It shows two boxes labeled 'Load Balancer' at the top. A double-headed arrow connects them, indicating a heartbeat or synchronization. Below the left 'Load Balancer' box, an arrow labeled 'HTTP(S)' points to a box labeled 'HTTP Server Plug-in'.

19 | Workload Management / High Availability | © 2002, 2003 IBM Corporation

Now let's look at what happens when things are not running so smoothly. Failover is one reason to implement workload management; it effects all the places where routing decisions are made.

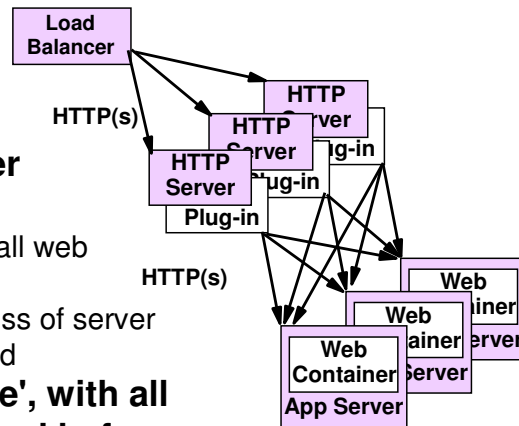
The first routing decision is made outside the WebSphere Application Server's footprint - in the Load Balancer (or other IP spraying technology).

Edge Component's Load Balancer can be configured on two machines - they share a heartbeat, and if that heartbeat fails, the backup server becomes the primary and handles the load.

Having eliminated that as a Single Point of Failure, let's move next to the HTTP server

HTTP Server Failover

- **Multiple HTTP servers provide coverage**
- **Edge Server can route around failed HTTP server**
- **HTTP Plug-in**
 - Every plug-in knows about all web containers
 - Session key contains address of server
 - Sessions get properly routed
- **Topology is 'Active/Active', with all HTTP servers handling load before failover**



Typically, a production environment will have multiple HTTP servers; each of those HTTP servers will route to multiple WebSphere Application Server instances.

Each plug-in knows about all the servers; it has a server list and a back-up server list. If all the servers in the server list are unavailable, it will route to the backup list. (This must be manually edited into the plugin-cfg.xml file.)

If any HTTP server fails, the Load Balancer will simply route around it. The plug-in reads the cloneID from the session key, and can route the request to its originating server.

Web/Servlet Container Failover

- **HTTP Server Plug-in**
 - Detects Failure
 - Marks Container as unavailable
 - Tries next Cluster member in the Cluster
- **What about In-flight sessions?**
 - Sessions may be persisted to database or replicated in memory
 - More details in HTTP Session Management lecture

21 | Workload Management / High Availability | © 2002, 2003 IBM Corporation

What happens if a server process dies? The HTTP server notes the failure and marks that application server as unavailable, then routes the request to the next cluster member.

Sessions already in progress will have a server ID for that failed server; the HTTP server routes them to the next server... What about the session? We can handle that two ways. Session Persistence to a Database, or internal messaging of session information.

Database session persistence functions largely as it did in version 4.0.

WebSphere Internal Messaging is new in 5.0, and we will see the details in the HTTP Session module. One point to note here is that since the routing algorithm is a round robin algorithm, if a server fails, it's requests will always go to the SAME 'next' server. This allows us to use the configuration where servers get memory information from only one other server... The 'Buddy' System. This reduces the amount of memory replication necessary in a cluster.

One path through the GUI to session configuration settings is:

Servers->Application Servers -> <Server Name>->Web Container->Session Management->Distributed Environment Settings

Then either:

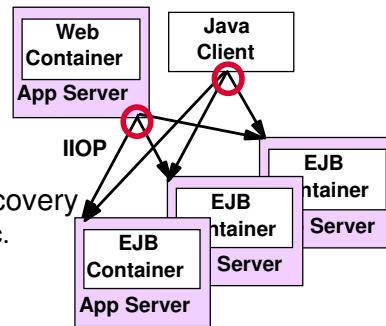
->Database

or

->Memory to Memory Replication

EJB Container Failover

- **Client code and ORB plug-in can route to next available cluster member**
- **Failure occurs before any work was initiated on the cluster member:**
 - ORB automatically re-routes 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 needs to determine appropriate recovery
 - Re-issue request, rollback transaction, etc.
 - If "NO_IMPLEMENT" exception thrown, no recovery is possible



22

Workload Management / High Availability

© 2002, 2003 IBM Corporation

Finally, it is possible that an EJB Container may fail. More likely that the web container and the EJB container will fail together, since they are part of the same application server process. But not all requests to the EJB Container go through the Web Container - Other EJBs and stand-alone Java clients can go directly to the container.

We need to differentiate between two possible failure scenarios.

In the first case, if the EJB request fails before any work was actually initiated on the target cluster member, the WLM-aware client ORB can transparently re-route the EJB request to another available cluster member. If no other members are available, then a `org.omg.CORBA.NO_IMPLEMENT` exception is thrown.

If the failure occurs after the EJB request was accepted and some work was initiated on the cluster member, then there is no possibility of a transparent failover. Depending on the circumstances, the container will throw one of a number of system exceptions.

It's up to the logic of the specific application to determine whether the operations that were in progress are recoverable in some way. Recovery actions may include reissuing the request -and in that case the WLM-aware ORB will route it to a cluster member that is alive. Or, the client may decide to rollback the transaction, if one is active.

Node Agent Failover

- **5.0 clients need one Node Agent up for first call to Location Service Daemon**
 - These clients can use a list of LSDs if multiple nodes are available in the topology
 - The IOR contains the list of LSD's host and ports
- **Interoperability**
 - 3.x and 4.x clients will work only when the specific Node Agent they connect to is alive

In Version 5, any LSD available in the WebSphere topology can provide the correct IOR for an EJB lookup. In other words, a Version 5 Client can rely on any Node Agent to get hold of the IOR it needs. All IORs include a list of alternate LSDs.

V4 and V3 clients connect to a specific node agent and do not know how to use the alternate LSD IIOP addresses - they need a specific node agent to be up and running in order to work.

Node Agent Failover (continued)

- **WebSphere-aware clients get "special treatment"**
 - WebSphere private information in the WLM TaggedComponent
 - Includes a complete description of the cluster's topology
 - Secure and non-secure transport ports for all cluster members are included and can be used directly by the client ORB

WebSphere client ORBs get a full blown description of the complete topology of a cluster when they get hold of an IOR. Not only would they be provided with a list of alternate LSDs, but they would also get a data structure that includes the indication of all the ports (secure and non-secure) that are available on each one of the cluster members - this makes it possible for the client ORB to failover without recontacting the LSD.

Non-WebSphere client ORBs would not be able to take advantage of this feature - they need to recontact an LSD in order to get an alternate IOR to failover to.

Support for Third-party Client ORB

- **WebSphere supplies "indirect IOR" to client ORB for EJB WLM**
 - According to part of the GIOP 1.4 specification
 - Indirect IOR points to the Node Agent
 - Node Agent used to get "direct IOR" pointing to actual cluster member
- **Failover Support**
 - Third-party ORB compliant with GIOP 1.4
 - Request fails, ORB gets new IOR via indirect IOR
 - Other ORBs
 - Less transparent recovery
 - Likely to need to regain access to the object via create() or findBy...()

WebSphere supports interoperability with non-IBM ORBs also at the Workload Management level.

WebSphere complies with a portion of CORBA's GIOP 1.4 specification which implies that an indirect IOR can be provided to a requesting client ORB. The indirect IOR can be used by the client ORB to get the direct IOR which points to a specific cluster member. The Indirect IOR is the IOR of a WebSphere Node Agent.

If the client ORB complies with this part of GIOP 1.4, recovery from EJB failures is pretty simple. A new direct IOR can be requested by the client ORB to the indirect IOR.

In case of a non compliant ORB, recovery is more complex and exposed to a greater degree to the client program - which may be required to regain full access to the EJB that failed through a create() or find operation.

Deployment Manager failover

- **Deployment Manager failover not addressed in Network Deployment**
- **Node Agents have copies of all configuration information**
- **Consequence of Deployment Manager failure:**
 - Unable to broadcast configuration changes to Node Agents
 - Admin Console unavailable
 - wsadmin unavailable
 - (unless manually directed to specific server)
 - In short, no changes to the cell configuration

What about a Deployment Manager failure?

The Deployment Manager does not handle client requests – it only handles the configuration repository. Since the configuration information is replicated to the Node Agents, and the Node Agents are fail-over-able, Network Deployment does not address Deployment Manager failover.

Other than the fact that Node Agents will not be notified of failed servers on other nodes, the impact of a downed server is minimal. Oh, you'll want to restart it... But while it is down, no customer requests will fail in-flight.

Problem Determination

What could go wrong?

- **Look in the logs first**
 - Log may point to component to trace
- **HTTP Plug-in needs to be current**
- **HTTP Server will not start if invalid entries in plugin-cfg.xml**
 - From a command prompt, launch apache.exe. If there is a problem with the plug-in, the error messages on the screen are more helpful than the terse summaries returned by Windows

Problem Determination is always an important part of the big picture. Knowing how requests are routed, and knowing where they are supposed to go will be a large part of figuring out what went wrong.

Troubleshooting comes in two parts - initial configuration, and Things That Break Later. For the initial configuration, the most common reason for failure is typographical errors. A period where a colon should go in the Transport Host configuration will cause a server to not listen.

Synchronization problems can cause your newly created server to not show up on the node. Yet you can manipulate the master configuration in the console, change things and save your changes. A server status is 'unavailable' if the Deployment Manager cannot talk to the Node Agent for any reason. Often it is because we forgot to start the node agent after a re-boot or similar event.

What could go wrong? (continued)

If the problem appears to be here . . .	Trace this component
HTTP Sessions	com.ibm.ws.webcontainer.httpsession.*=all=enabled
Memory to Memory Replication (WebSphere Internal Messaging)	com.ibm.ws.runtime.component.SystemMessageServerImpl=all=enabled
Workload Management	WLM=all=enabled
Synchronization of files	com.ibm.ws.management.sync.* = all = enabled

The table on this slide offers some tracing suggestions. Whenever you get an error, check the logs first - often the Log Analyzer tool will suggest the answer, or just looking at the error will prompt something in your memory. If the answer is not obvious, turn on trace. Trace the component that was identified in the log.

Summary

- **Defined Workload Management**
- **Described Clusters and Cluster Members**
- **Examined the Topology**
- **Weighted Server routing topology**
- **Reviewed Failover scenarios**
- **Visited the HTTP Plug-in briefly**
- **Listed Problem determination steps**

In summary, we reviewed a basic definition of Workload Management. Then we looked at the topology, and saw where request routing decisions are made. We talked about failover, and how various components are configured to avoid a Single Point of Failure (SPOF).

Finally, we offer some problem determination suggestions.

