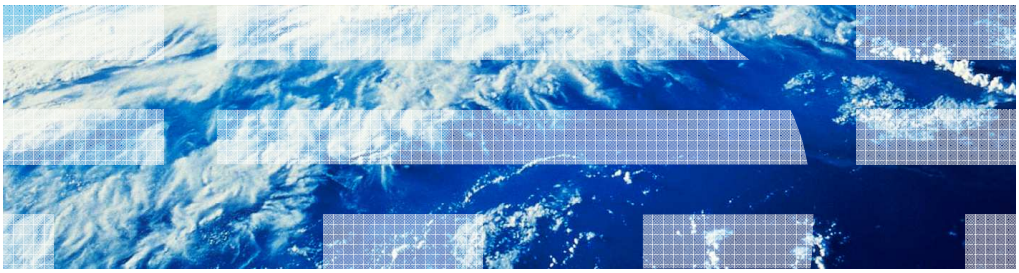
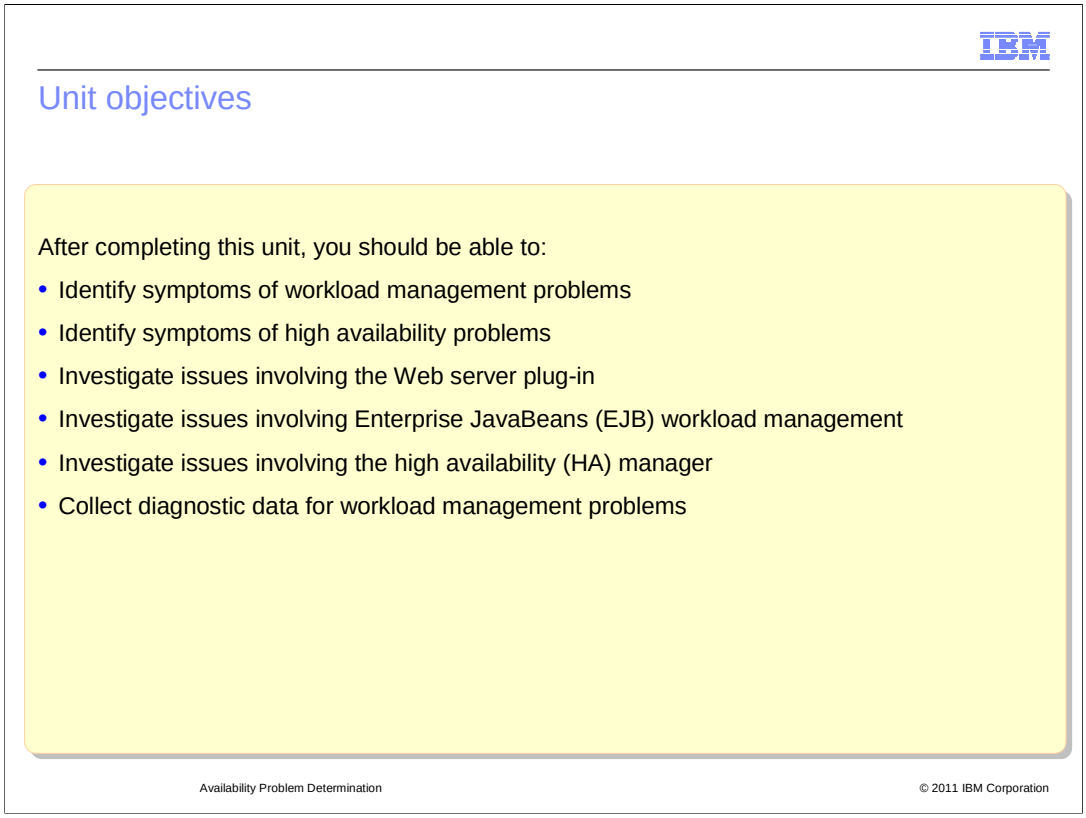
Workload management and high availability problem determination



WebSphere software

This unit describes how to troubleshoot workload management and high availability problems in WebSphere Application Server V7.



After you complete this unit, you will be able to identify problems with workload management and high availability, as well as collect diagnostic data to help with your problem determination. You will also learn how to investigate issues involving the Web server plug-in.

WLM overview

- Three types of Workload Management (WLM) in WebSphere Application Server:
  - Web server plug-In WLM
    - Balances HTTP requests between cluster members
  - Enterprise JavaBean (EJB) WLM
    - Balances WLM enabled RMI/IIOP requests between cluster members
  - Message engine WLM
    - Distributes the load among messaging engines when a cluster is a member of a service integration bus (SIBus)
    - Not covered in this unit

-There are three types of Workload Management (WLM) in WebSphere Application Server: Web server plug-In WLM which balances HTTP requests between cluster members, Enterprise JavaBean (EJB) WLM which balances WLM enabled RMI/IIOP requests between cluster members, Message engine WLM which distributes the load among messaging engines when a cluster is a member of a service integration bus (SIBus).

Workload management routing logic

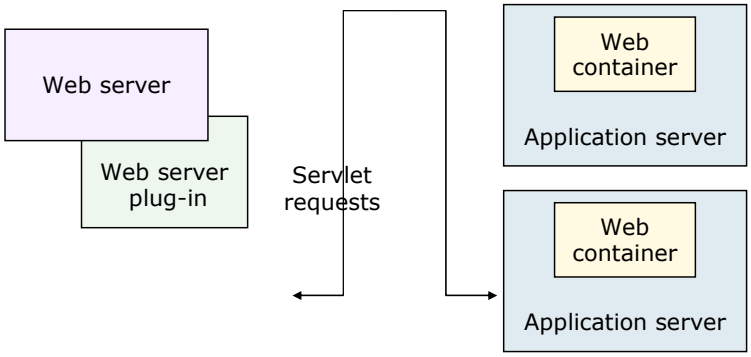
- Routing is based on weights associated with cluster members.
  - Round robin algorithm used when weights are equal
  - Weights can be modified to send more requests to a particular cluster member or members
  - More information about WLM routing from the V7.0 Information Center:
  - **http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/crun_srvgrp.html**
- If the EJB client is on the same physical box as the cluster member, the “Prefer Local” setting will ensure that all requests from the client go to the local cluster member.

Routing is based on weights associated with the cluster members. If all cluster members have identical weights, the plug-in sends equal requests to all members of the cluster, which is known as Round Robin.

Weights can be modified to send more requests to a particular cluster member or members. You can use the administrative console to specify a weight for a cluster member.

Note: If the client is on the same physical box as the cluster, the “Prefer Local” setting will ensure that all requests from the client go to the local cluster member.

Web server plug-in workload management



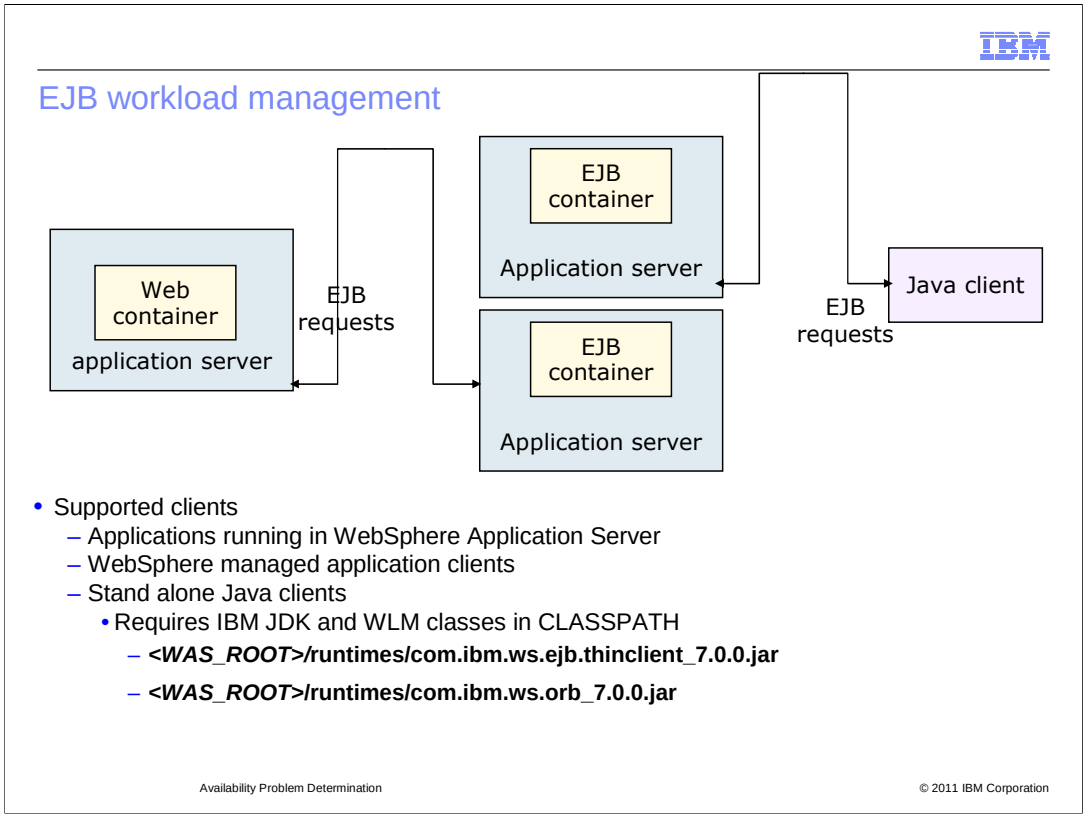
- Supported clients
  - Browsers
  - Applications that make remote HTTP and HTTPS calls
  - Web service clients

Web server plug-in is the “glue” between a Web server and WebSphere Application Server. The primary responsibility of the plug-in is to forward HTTP and HTTPS requests from the Web server to the WebSphere Application Server.

Web server plug-in routing settings

- Routing configuration settings in plugin-cfg.xml
  - LoadBalanceWeight (Integer)
    - Specifies the weight associated with the server when the plug-in does weighted round robin load balancing
  - LoadBalance
    - Round robin (default) or random routing
  - IgnoreAffinityRequests (0 or 1)
    - Prevents server affinity when using round robin routing
  - RetryInterval (Integer)
    - Length of time that should elapse from the time that a server is marked down to the time that the plug-in will retry a connection
  - PrimaryServers
    - Specifies a list of servers to which the plug-in routes requests for this cluster
  - BackupServers
    - Specifies a list of servers to which requests should be sent to if all servers specified in the primary servers list are unavailable

Most aspects of the web server plug-in are configuration. For example, using the LoadBalanceWeight parameter you can influence the way the load balancer distributes load when using the Round Robin algorithm. Perhaps you aren't satisfied with the Round Robin algorithm and would rather utilize the Random algorithm. If that is the case you could change the value of the LoadBalance parameter. Lastly, you might want to specify a customized RetryInterval value in order to reduce the amount of time that the plug-in will wait before retrying a connection to a downstream server.



Workload management for EJB containers can be performed by configuring the web container and EJB containers on separate application servers. Multiple application servers can be clustered with the EJB containers, enabling the distribution of enterprise bean requests between EJB containers on different application servers. EJB WLM balances WLM enabled RMI/IIOP requests between clients and clusters.

In this configuration, EJB client requests are routed to available EJB containers in a round robin fashion based on assigned server weights. The EJB clients can be servlets operating within a Web container, stand-alone Java programs using RMI/IIOP, or other EJBs.

EJB workload management enabled calls

- Types of application calls balanced by EJB WLM
  - JNDI Lookups
  - EJB creates
  - EJB business methods
  - EJB removes
- Example

```
ctx = new InitialContext(env);
Object obj = ctx.lookup("ejb/ejbs/WLMTestHome";
WLMTestHome home = (WLMTestHome)
   PortableRemoteObject.narrow(obj), WLMTestHome.class);
WLMTest bean = home.create();
String servername = bean.whichServer();
bean.remove()
```


When dealing with EJB communications, JNDI lookups, EJB create and remove operations, and EJB method invocations are all subject to WebSphere workload management.

EJB WLM dependent services

- HA manager service
  - Provides a specialized messaging mechanism (bulletin board) that enables processes to exchange information about their current state.
  - Each process sends or posts information related to its current state, and can register to be notified when the state of the other processes changes.
  - The WLM component uses this mechanism to build and maintain routing table information.
  - Routing tables built and maintained using this mechanism are highly available.
- Distribution and Consistency Services (DCS)
  - Runs on every process in the cell by default
  - Provides fast interconnects between core group members
  - Used by HA manager component to share bulletin board data

The WebSphere High Availability manager (HA Manager) provides a specialized messaging mechanism that enables processes to exchange information about their current state. This mechanism is commonly referred to as the bulletin board.

The WLM component uses this mechanism to build and maintain routing table information. Each process sends or posts information related to its current state, and can register to be notified when the state of the other processes changes.

Distribution and Consistency Services (DCS) provide the underlying group services framework for the HA Manager such that each application server process knows the health and status of JVMs and singleton services. It basically provides view synchronous services to the HA Manager.

Common WLM problems

- Uneven routing
  - Requests not load balanced between cluster members
- No requests sent to a particular cluster member
  - Target application server is running but not receiving requests
- WLM related exceptions in application server logs
  - NO_IMPLMENT errors
  - No available target
  - No cluster data available
  - Forward limit reached, retry limit reached

Some common WLM problems are uneven routing, lack of requests sent to a particular cluster member, and exceptions in the application server logs.

Diagnosing HTTP routing problems

- Check cluster member entries in **ServerCluster** stanzas in the plugin-cfg.xml.
  - Ensure all cluster members are listed with correct hosts and ports.
- Confirm **LoadBalance** setting set to expected algorithm for the cluster.
  - Round robin is the default value.
- Check **LoadBalanceWeight** value for each server.
  - Should match value defined in the cluster definition in the administrative console.
  - Look for servers with **LoadBalanceWeight** set to 0.
- Confirm server affinity settings are set to the expected values.
  - Determines whether a request is “pinned” to a particular server in the cluster

To diagnose HTTP routing issues, one should first examine the current configuration of the web server plug-in giving consideration to the values of the LoadBalance and LoadBalanceWeight parameters along with the ServerCluster stanzas. Confirm that they all contain the expected values before continuing on with the assumption that you are, in fact, seeing incorrect routing behavior.

Tracing HTTP WLM routing problems

1. Edit the **plugin-cfg.xml** file and change Loglevel to **Trace** in the Log stanza.

2. Restart the web Server.

3. Review the Plug-In trace log and look for the URI in question.

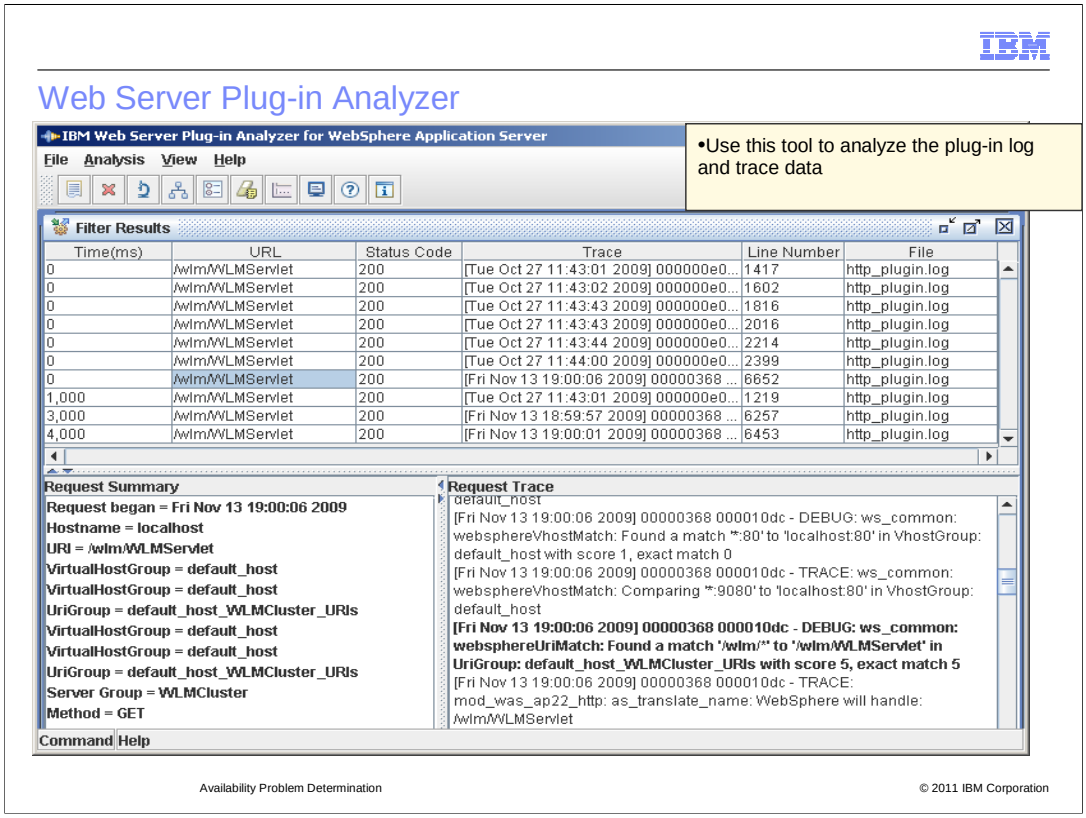
- [Mon Jul 27 12:22:28 2009] 000011f0 00000d1c - DETAIL: ws_common: websphereShouldHandleRequest: trying to match a route for: vhost='myhost1.austin.ibm.com'; uri='/myapp/myservlet'

1. Follow the thread ID (3rd Column) and look for the **websphereFindTransport** entry to determine which server the request was sent.

- [Mon Jul 27 12:22:28 2009] 000011f0 00000d1c - DETAIL: ws_common: **websphereFindTransport**: Setting the transport(case 2): myhost1.austin.ibm.com on port 9080

1. Continue checking requests for the same URI. Look for connection timeouts or other errors that can indicate why the requests are not being routed evenly

The web server plug-in can be configured to provide detailed trace data for troubleshooting. To enable the trace, locate the LogLevel parameter in the plug-in configuration file and set the value to “Trace”.



The IBM web Server Plug-in Analyzer for WebSphere Application Server helps discover potential problems with trace and configuration files during use of WebSphere Application Server.

The tool parses both plug-in configuration and corresponding trace files and then applies pattern recognition algorithms in order to alert users of possible inconsistencies.

The tool provides a list of HTTP return codes, URI and graphical presentations of available clusters, and server topologies from the configuration and trace files.

The tool is available via the IBM Support Assistant (ISA) workbench, a freely downloadable application available from the IBM Support site.

Diagnosing EJB routing problems

- Check the configured Weights for each cluster member under **Servers > Clusters > cluster_name > Cluster Members**.
  - Weights determine routing distribution.
- Check for a static routing table in the cluster config directory on the target EJB cell.
  - Static routing file location
  - **`<PROFILE_ROOT>/config/cells/cell_name/clusters/cluster_name/ cluster_name.wsrttbl`**
  - Static routing will use predefined weights that are not updated using bulletin board data.
  - You can route requests to stopped servers.
  - WLM balancing can be uneven if weights were not equal in the static file.
- Check the Prefer Local setting for the cluster.
  - Prefer Local will route EJB requests to the local host where the client is running if possible.
  - If workload to the client is not balanced, this can cause uneven distribution to EJB application

Diagnosing WlM problems with EJBs is similar to diagnosting HTTP WLM problems. First, assure that you have configured EJB WLM to distribute load as required by checking the configured weightings set for each cluster member. To verify this, in the administration console, browse to Servers > Clusters > <cluster_name> > Cluster Members and review the weights for each cluster member.

If static EJB routing is in use, verify the configuration by examining the routing table file located in the <PROFILE_ROOT>/config/cells/cell_name/clusters/<cluster_name>/ directory. The file name is generally the name of the cluster with a .wsrttbl file extension.

Also, check the Prefer Local setting for the cluster. Prefer Local will route EJB requests to the local host where the client is running if possible.

Tracing EJB WLM routing problems (1 of 2)

1. Gather ORB/WLM tracing on **client** where routing problem occurs (**ORBRas=all:WLM*=all**).
2. Find all of the three parameter **getConnection()** lines in the trace (example below).
3. Determine if the host and port (red) for each similar EJB method call (blue) are rotating between cluster members.

- [3/27/08 11:01:34:671 CDT] 00000036 ORBRas 3 com.ibm.ws.orbimpl.transport.WSTransport getConnection(Profile, ClientDelegate, operationName) WebContainer : 0 method entry: **host=myhost1.austin.ibm.com port=1329**
clientDelegate=IOR:00bdbdbd00000022524d493a656a62732e574c4d546573743a303030303030303030303030303030303000bdbd0000000100000000000001dc000102bd0000001662756c6c69732e61757374696e2e69626d2e636f6d0023900000006f4a4d4249000000126ec064d2373730656563303634643264326363300000002400000004b49454a50020155b702bd0b74657374436c757374657203454a420000002dacac000200010122000000574c4d546573745f636c757374657223574c4d454a422e6a617223574c4d54657374bd0000000c000000010000001400bdbdbd0501000100000000000101000000000049424d0a0000000800bd000115000002000000260000000020002bdbd49424d040000000500502010 2bdbdbd0000001f0000000400bd0003000000200000000400bd000149424d0b0000007f00000004000c62756c6c697343656c6c3031000b74657374436c7573746572000000124c924c8001562756c6c69732e61757374696e2e69626d2e636f6d23900000003700000000020000b434c55535445524e414d45000b74657374436c7573746572000843454c4c4e414d45000c62756c6c697343656c6c303100000000bd000000030000002000bdbdbd0000001662756c6c69732e61757374696e2e69626d2e636f6d00239049424d040000000500502010 2bdbdbd0000001f0000000400bd000300000020000000400bd000100000025000000400bd0003 **operationName=whichServer**

To enable tracing to debug EJB WLM routing problems you can use a trace string of ORBRas=all:WLM*=all

An example of the output is shown on this slide.

Tracing EJB WLM routing problems (2 of 2)

- Example of proper routing between cluster members (clientDelegate IOR removed)
- [3/27/08 11:02:50:703 CDT] 00000036 ORBRas 3 com.ibm.ws.orbimpl.transport.WSTransport getConnection(Profile, ClientDelegate, operationName) **host=myhost1.austin.ibm.com port=1329** operationName=whichServer
- [3/27/08 11:02:50:734 CDT] 00000036 ORBRas 3 com.ibm.ws.orbimpl.transport.WSTransport getConnection(Profile, ClientDelegate, operationName) **host=myhost2.austin.ibm.com port=1327** operationName=whichServer
- [3/27/08 11:02:50:796 CDT] 00000036 ORBRas 3 com.ibm.ws.orbimpl.transport.WSTransport getConnection(Profile, ClientDelegate, operationName) **host=myhost1.austin.ibm.com port=1329** operationName=whichServer
- [3/27/08 11:02:50:828 CDT] 00000036 ORBRas 3 com.ibm.ws.orbimpl.transport.WSTransport getConnection(Profile, ClientDelegate, operationName) **host=myhost2.austin.ibm.com port=1327** operationName=whichServer
- [3/27/08 11:02:50:859 CDT] 00000036 ORBRas 3 com.ibm.ws.orbimpl.transport.WSTransport getConnection(Profile, ClientDelegate, operationName) **host=myhost1.austin.ibm.com port=1329** operationName=whichServer
- [3/27/08 11:02:50:890 CDT] 00000036 ORBRas 3 com.ibm.ws.orbimpl.transport.WSTransport getConnection(Profile, ClientDelegate, operationName) **host=myhost2.austin.ibm.com port=1327** operationName=whichServer

The example trace shown here demonstrates a symmetric balance between two EJB hosts.

WLM trace points (1 of 2)

- Look for keyword “Unexpected”
  - Trace point to indicate things that should not be happening
- Look for **popServerForInvocation()**
  - Selected cluster member target printed during trace exit
  - Can be used to confirm WLM routing
-
- [3/27/08 11:02:03:765 CDT] 00000036 SelectionMana < **popServerForInvocation** Exit {CELLNAME=bullisCell01 , CLUSTERNAME=testCluster} : [[[[ProcessDescriptionImpl#1688364194{**CELLNAME=Cell01, MEMBERNAME=clustermember1, NODENAME=Node01**}]{}] available:0 reachable:0 leaf:true version:]nil:nil]

Using a few eye-catchers, one can quickly skim the traces or indications of problems. For example, the term “unexpected” in the trace will point to things that should generally not be happening. Additionally, if you look for the term popServerForInvocation, you can easily locate when the WLM component selects a cluster member to service an EJB request.

WLM trace points (2 of 2)

- Look for **setObservedWeight()**
  - Used to decrement weight of cluster member during request routing.
  - Check to ensure maximum observed weight is n – 1 of the configured value for the cluster member.

- [3/27/08 11:02:51:078 CDT] 00000036 RouterMediato > setObservedWeight Entry
- {CELLNAME=Cell01, CLUSTERNAME=testCluster}
- {CELLNAME=Cell01, MEMBERNAME=clustermember1,NODENAME=Node01} 1
- [3/27/08 11:02:51:078 CDT] 00000036 RouterMediato 3 **ObservedWeight: 1**
- [3/27/08 11:02:51:078 CDT] 00000036 RouterMediato < setObservedWeight Exit

If you look for the term setObservedWeight(), you will be able to see the WLM component decrement the weight of a cluster member as it services a request. Check to ensure maximum observed weight is n-1 of the configured value for the cluster member.

Routing pattern problems

- Consider these code snippets:

  ctx = new InitialContext(env);

  WLMTestHome home = (WLMTestHome) PortableRemoteObject.narrow(ctx.lookup("ejb/ejbs/WLMTestHome"), WLMTestHome.class);

  WLMTest bean = home.create();

  String servername = bean.whichServer();

  bean.remove()
- There are four WLM calls in the code above (JNDI Lookup, EJB Create, EJB Method, EJB Remove).
- If there were two members in the cluster with the same weight (or multiples of the same weight), the following routing can occur:

  Member1 – JNDI Lookup

  Member2 – EJB Create

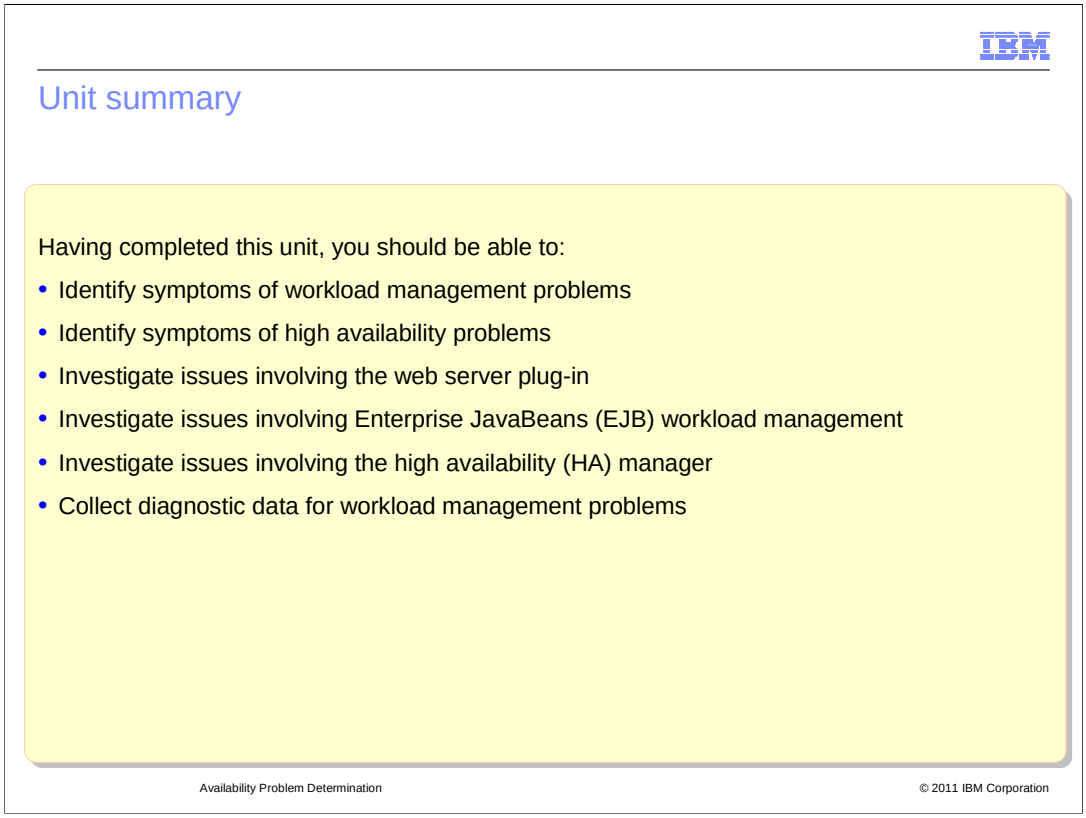
  Member1 - EJB Method

  Member2 - EJB Remove
- If this same rotation continued, the EJB method call would always occur on Member1.
- Solution:
  - Ensure that the number of WLMable requests is not the same as the number of cluster members.
  - Alternatively, cache the EJB create and only perform the EJB method call during each request

Sometimes perceived routing problems are actually a product of the EJB-related operations occuring in the application code. For example, consider the sample code show on this slide and assume that there are only two servers in the EJB cluster. If the routing weights were equal or multiples of the same weight, cluster member one can always end up servicing the EJB method whereas member two would only service the lighter weight EJB create and remove operations.

If this same rotation continued, the EJB method call would always occur on Member1.

The solution for this scenario would be to ensure that the number of workload manageable requests is not the same as the number of cluster members. Alternatively, cache the EJB create and only perform the EJB method call during each request.



Having completed this unit, you will be able to identify problems with workload management and high availability, and collect diagnostic data to help with your problem determination. You will also learn how to investigate issues involving the web server plug-in and EJB WLM.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WA5721G18_WorkloadManagement_edited1.PPT

This module is also available in PDF format at: ../WA5721G18_WorkloadManagement_edited1.pdf

Availability Problem Determination

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.
Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.