IBM WebSphere Web Multi-Platform Configuration

Workload Management
Objectives

- Discuss workload management.
- Discuss workload management options in WebSphere.
  - Discuss performance implications of an option.
  - Discuss failover implications of an option.
- Discuss session management implications.
- Discuss how the different options work together.
A simple definition is the process of spreading multiple requests for work over the resources that can do the work.

Workload management (WLM) is a procedure for improving performance, scalability, and reliability of an application.

- One of the problems with the term "Workload Management" is that it means something different to each person.
- The desired end result is improved performance and reliability.
Why Use WLM?

- Maximize use of resources
- Maximize throughput
- Minimize risk of having single points of failure

- Maximize use of resources: If an application cannot make full use of a machine resources, WLM techniques can help make more efficient use of the processing power.
- Maximize throughput: Workload management techniques can greatly increase the number of requests an application can serve concurrently.
- Minimize risks of single points of failure: WLM techniques build a level of redundancy into your configuration. If one copy fails, there are other copies that can process requests.
How Do I Manage My Workload?

- Servlet clustering
- Enterprise JavaBeans (EJBs)
- TCP/IP spraying with SecureWay Network Dispatcher
This is the architecture used in the day one labs.

The HTTP server and WAS exist on a single physical node and point to a data source.
Models and Clones
Model: a template for creating additional, nearly identical copies of an object, such as an application server or servlet engine.

Clone: a copy of the model.

- After you clone a resource, modifying the model automatically propagates the same changes to all of the clones.

- Best practice: In most cases you get better performance by creating a model of the entire application server with the web application and EBJ containers defined.
Objects That Can Be Modeled/Cloned

- Application Server
- EJB Container
- Enterprise Bean
- Servlet Engine
- Web Application
- Servlet
Creating Models and Clones

- Configure your application.
- When the application is fully and correctly configured, make a model of the application.
- Select a routing option.
- Make clones of the application from the model.

- Best practice: make the original application server a clone.
- The routing option controls how the workload is shared between clones.
- Models exist in a single WebSphere Domain.
Routing options are only used for EJB workload management:
- Round Robin Prefer Local
- Random Prefer Local
- Round Robin
- Random

Server failures are handled transparently (unless within a transaction).

When you create your model, you select routing options based on your configuration and performance requirements. The following list briefly describes these options:

- **Round Robin Prefer Local**: requests are routed sequentially with preference given to clones on the same physical node. This option maximizes performance by taking advantage of the speed of in-process calls.
- **Random Prefer Local**: requests are routed randomly to clones with preference given to clones on the same physical node.
- **Round Robin**: requests are routed sequentially to the clones in the server group.
- **Random**: requests are routed to any clone in the server group with no preference to a particular node, nor in any particular sequence.
Prefer Local Routing Option

- Clones are all from the same model.
- Requests stay on the local machine unless all local clones fail.
- Clones have a yellow background behind the icon. The model has a blue background.
## Model Properties--General

**Model**: Big3 Master  

<table>
<thead>
<tr>
<th>Model</th>
<th>General</th>
<th>Advanced</th>
<th>Debug</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Executable in use:**

**Command line arguments:**

**Environment:** Environment...

**Process ID:**

**Working directory:**

**Standard input:**

**Standard input in use:**

**Standard output:** Jcpt/WebSphere/App Server;

**Standard output in use:**

**Standard error:** Jcpt/WebSphere/App Server;

**Standard error in use:**
Clone Properties--General

Application Server: Big3Clone

Current State: Running
Desired State: Running
Start Time: Jul 26, 2000 7:19:10 PM
Executable in use: /opt/WebSphere/AppServer/java/nm/bin/java
Command line arguments: 
Environment: Environment...
Process ID: 18743
Working directory: 
Standard input: 
Standard input in use: 
Standard output: /opt/WebSphere/AppServer/logs/Big3Server.out
Standard output in use: /opt/WebSphere/AppServer/logs/Big3Server.out
Standard error: /opt/WebSphere/AppServer/logs/Big3Server.err
Standard error in use: /opt/WebSphere/AppServer/logs/Big3Server.err

IBM
Model/Clone--Advanced

- Note: For clones, the Workload management selection policy will be unavailable.

- The graphic above shows some of the options you can choose when defining the model/clone.
Server Groups

A server group is a collection of servers participating in workload management.

- Using a network link is slower than in-process calls.
- Performance gains are realized when the additional servers handle some of the workload that would cause a single application server to be overloaded.
- The key to using server groups to increase performance is finding the balance in the tradeoff between in-process calls to one server on one node, and the increased performance and processing power of additional servers using out-of-process calls.
- In this example, server group 1 consists of Server1 and Server2, both of which reside on one physical node. Server group 2 consists of four application servers, Server4, Server5, Server7, and Server8. Server group 2 is spread across two physical nodes. Finally, server group 3 comprises Server3, Server6, and Server9, each on a separate physical node.
Server Groups

- Server groups have the following characteristics:
  - A collection of identical application servers
  - Can have more than one clone on a single physical node
  - Can have multiple clones distributed across nodes

Hint: For now, you can only clone across nodes that have the same operating system and directory structure.
Servlet Clustering
Round Robin Routing of servlet requests occurs between the WebSphere plug-in and the cloned servlet engines.

Once the ProcessClaim servlet is invoked, it will make in-process calls to the processClaimEJB (assuming that the stubs for the EJB are WLM-enabled).

Example of queues.properties:

ose.srvgrp.ibmoselink1
ose.srvgrp.ibmoselink1.type=FASTLINK
ose.srvgrp.ibmoselink1.clonescount=2
ose.srvgrp.ibmoselink1.clone1.port=8111
ose.srvgrp.ibmoselink1.clone1.type=remote
ose.srvgrp.ibmoselink1.clone2.port=8112
ose.srvgrp.ibmoselink1.clone2.type=remote
Performance Implications

- Efficient use of system resources through use of clones
- Failover at the process level

- If one clone fails, it no longer receives requests.
- When a failed clone becomes available again, it is inserted back into the routing algorithm.
EJB Workload Management Basics

- Requests to EJBs are workload managed for
  - EJB homes
  - Stateless session EJBs
  - Entity EJBs

- Stateful EJBs are not workload managed
EJB request routing occurs in EJB remote interface stubs.

- The stubs are generated during EJB deployment.
- Stubs generated by VA-J are WLM-enabled by default.
Stubs generated by the WebSphere Administrative GUI are WLM-enabled if you choose that option during deployment.

- Supported in version 3.5
EJB WLM routing policy follows those policies outlined in the WLM overview with the following exceptions:

- **Process Affinity**
- **Transaction Affinity**

- Process affinity overrides the binding policy. Requests from EJB clients (servlets or another EJB) running in a process are routed to an EJB in the same process if available.

- Transaction Affinity:
  - If a server clone is picked from a server group in a transaction, it will continue to use the same server clone for all requests within that transaction.
  - Will select a clone based on selection policy for next transaction.
  - Enforced per JVM, not globally.
EJB WLM Request Processing

- The stub is aware of the clone configuration and follows the routing policy to send the request to a working clone.
- The stub is notified of configuration changes as part of the response processing.
- The stub uses the new configuration when handling new requests.
EJB WLM Failover Processing

In Flight Request Processing

- Failed requests that were never started by a failed server will be routed to a working server.
- Failed requests that were processed by a failed server but not completed result in an exception being thrown back to the client. Retries will go to a working server.

An In Flight Request that fails and was never started will have a CORBA completion status of COMPLETE=NO. The stub will automatically reroute failed requests that were not started by the failed server.

An In Flight Request that was started but not completed by a failed server will have a CORBA completion status of COMPLETE=MAYBE. The stub throws an exception for requests that were being handled by the failed server. How the client handles these exceptions are application dependent.

For example, if the request was doing a lookup, the request should be retried. If, on the other hand, the request that started but did not complete was a product order, more sophisticated recovery would be necessary.
WLM Client Initial Access

1. Client tries to access cloned EJB.
2. WLM runtime obtains Model info from Admin Server.
3. WLM runtime creates proxies to clones.
4. WLM runtime directs access through a proxy based on affinity and selection policy.

Admin Server

Admin DB

WLM Client

Proxy1

Server Clone1

Bean Clone1

Server Clone2

Bean Clone2

Bean Clone1

Bean Clone2
WLM Client Subsequent Access

1. Client tries to access cloned EJB again.

2. WLM runtime directs access through a proxy based on affinity and selection policy.

2.1 If clone is down, increment MaxCommFailure counter and try another clone. If counter is > MaxCommFailure MAX, mark server as bad for UnusableInterval seconds.

2.2 If attempt to all clones failed then try to get new model info from Admin Server.

Client tries to access cloned EJB again.

WLM runtime directs access through a proxy based on affinity and selection policy.

If clone is down, increment MaxCommFailure counter and try another clone. If counter is > MaxCommFailure MAX, mark server as bad for UnusableInterval seconds.

If attempt to all clones failed then try to get new model info from Admin Server.
Red boxes indicate separate processes.

Process Affinity overrides the binding policy.

If a servlet is running in the same process as an entity or stateless session EJB, the servlet's methods will be directed to that EJB and not some out-of-process EJB.

If the request is part of an active user transaction, transaction affinity overrides all other affinities.
In this configuration, incoming requests are routed to one of the clones in the Big3ServletServer.

The servlet clone routes the request to one of the EJB Container clones in the Big3EJB server. The servlets can route to either EJB Container clone, balancing the workload.

In this scenario, it is possible to place a firewall between your servlet engine and the EJB server in order to protect your business logic.

Note: not all relationships are shown in this diagram.
• EJBs are secure behind the firewall
• Each container holding EJBs may hold 1:n clones of the EJBs
• Not all relationships are shown in this diagram
Client Requirements for EJB WLM

Remote Java clients (running on a machine without an Admin Server) must have the following properties set on the Java command line:

- `-Dcom.ibm.CORBA.BootstrapHost=adminserver_machine_name`
- `-Dcom.ibm.CORBA.Bootstrap_port (defaults to 900)`

These properties are used by EJB WLM support to look up information from the name server.

If there is an Admin Server in the same Admin Domain running on the client's system, the properties may be omitted because the default is to pick the local name server. Thus, these properties do not need to be set for Servlets that are accessing EJBs.
Stateful Session Beans and WLM
A stateful session bean is associated with a specific client. That bean may be accessed across several client requests.

The standard web shopping cart example often uses a stateful session EJB to store its information.

The handle to the stateful session EJB is stored in the HTTP session for that client.
Disadvantage of Using Stateful Session

1. The first request from Client1 was routed to Clone1 where the stateful session bean was created.
2. The next request from Client1 may be routed to the second clone. To maintain the client's session, the information in the stateful session EJB on Clone 1 must be accessed.
3. Accessing the EJB on Clone 1 from Clone 2 requires communicating using out-of-process RMI/IIOP, which significantly slows performance.
4. The likelihood of an in-process call is only 1/N for N clones.
5. If the first clone fails, the state information for Client1 is lost.
Best Practice: Use entity EJBs to store state information.

Using entity EJBs to store state information improves performance and failover by taking full advantage of WebSphere's workload management capabilities.

It is important to manage the destruction of expired state information from the database. This can be accomplished using HTTP Session Events or a timed cleanup script.
IBM WebSphere Web Multi-Platform Configuration

TCP/IP Spraying
Network Dispatcher (ND), a part of WebSphere Edge Server, can be placed in front of one or many HTTP servers.

You must have ND or its equivalent to take advantage of TCP/IP spraying.

Client requests for HTML pages, servlets, or JSPs come into the ND machine, which forwards the requests to the servers using one of several routing algorithms.

To the client, ND looks like an HTTP server.

Note: The HTTP server can reside on a separate physical node. For simplicity, we will show the HTTP server on the same node as the WAS.
Network Dispatcher Environments

- AIX
- Red Hat Linux
- Solaris
- Windows NT

For more information, go to the SecureWay Network Dispatcher Web site: http://www.software.ibm.com/network/dispatcher/library
Network Dispatcher (ND) can be configured to route requests in the following ways:
- round robin
- load balancing
- custom advisor
- combining custom advisor with load balancing
- content-based
**ND-Round Robin**

- **How it works**: ND blindly routes requests, in order, to HTTP servers

- **Advantages**: simple, spreads workload

- **Disadvantages**:  
  - if one machine or process crashes, then all requests going to that machine or process fail  
  - slower or lower capacity servers get same workload as fast, high capacity servers

---

IBM
How it works

- Periodically, ND sends a request to the server and times the response.
- Routing decisions are made based on response times.

- ND receives the client requests.
- Load balanced servers:
  - send their response to the client without any involvement of ND
  - do not require any additional code to communicate with ND
- ND directs requests to the most optimal server according to dynamically set weights.
- ND forwards traffic around a failed server, but checks the server periodically to see if it is available. If the server becomes available, it will become part of the routing again.
**Advantages**
- Faster servers handle more requests.
- ND will route around a failed machine or HTTP server.

**Disadvantages**
- Processing time required for ND to check servers.
- ND checks the availability of the HTTP server only, not the availability of WAS or database.

---

- If ND is configured between the client and the HTTP server, ND has no way of knowing how the WebSphere application is performing.

- If the WebSphere server or the application has failed but the HTTP server is still running, ND never finds out. It will continue to send requests to machine.

Which leads us to advisors...
Java servlet running in WebSphere
- ND periodically checks the servlet
- Can be as simple or as complex as needed
- Can be written to verify if WebSphere and the back-tier data sources are still available
- May be written to keep track of response times to WebSphere and the database; then used for load balancing
- If the advisor gets no response from the servlet, ND knows that that path is not functioning. Whether the HTTP server or the WAS failed, the advisor doesn't care.
- ND will ignore that path except for periodic checks to see if the path is running correctly.

Note: ND is now shipped with a standard advisor to check WAS.
Advantage: You can write a custom advisor that can go all the way through to the WAS to check to see if the WAS is available.

Disadvantage: Writing custom advisors is somewhat difficult.
ND-Content Based Routing

- Works with WTE, a caching proxy server, to
  - proxy HTTP requests to specified servers
  - filter web page content using specified rule types
- Allows you to specify a set of servers to handle requests based on the request content
- Detects where one server in a set has failed and stops routing to that server

WTE is Web Traffic Express, a caching and filtering component of IBM WebSphere Performance Pack.
- Usually examines the value of a cookie.
- Advantage: session management via smart router.
- Disadvantages:
  - It's difficult and expensive to set up
  - Requires at least two layers of proxies (physical nodes)

For more information, see "IBM Network Dispatcher: User's Guide Version 3.0 for Multiplatforms", GC31-8496
Performance/Failover Implications

- Network Dispatcher allows you to distribute workload across multiple nodes
- Failover support at the HTTP server level
- Becomes a single point of failure

IBM
To alleviate the single point of failure problem, ND can be configured for high availability (automatic or manual):
- Requires two physical nodes on the same subnet
- Nodes must be configured alike

On the primary dispatcher node, use these commands:
- `ndcontrol high backup add <primary IP addr> <backup IP addr>`
- `ndcontrol high backup add primary <manual | automatic> <port>`
- `ndcontrol high stat (show status of HA config)`

On the secondary dispatcher node, use these commands:
- `ndcontrol high backup add <backup IP addr> <primary IP addr>`
- `ndcontrol high backup add primary <same as primary> <primary port>`
- `ndcontrol high stat (show status of HA config)`

Note: Not supported when using content-based routing.

ND can also be configured for mutual high availability. Both machines actively perform load balancing while providing backup for each other.

For more information, see "IBM Network Dispatcher: User's Guide Version 3.0 for Multiplatforms", GC31-8496
Session Management and WLM
Session Management and WLM

What is a session?
- The ability to maintain state information (shopping cart, etc.) across multiple HTTP requests

How does it work?
- Cookie holding session identification is stored in the browser
- Server holds session identification and associated session information
Session Management and WLM

Client

Network Dispatcher (ND)

HTTP

OSE plug-in

Clone_A

Clone_B

HTTP

OSE plug-in

Clone_C

Clone_D
In general, session information is maintained in the memory space of the process that first registered the session.

WLM introduces a problem because multiple requests may not return to the same process (clone, network-dispatched HTTP server) and each process has its own memory space.
Two solutions:

- Session affinity (Network Dispatcher)
- Session affinity and persistence (WebSphere)
Session affinity forces requests from the same IP address to go to the same server. This is a Network Dispatcher option that can be enabled.

Session affinity is often referred to as "sticky port."
The client request enters the network dispatcher, which remembers the IP address and routes the request to an HTTP server.

Subsequent requests from the same IP address are routed to the HTTP server that served the first request.

Two problems with this configuration:
- IP address routing ("AOL" Problem)
- WebSphere Clones

<table>
<thead>
<tr>
<th>Client IP</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client 1</td>
<td>WAS 1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The client request is being routed through a set of proxy servers.
When the client makes a request, Network Dispatcher stores the IP address of the proxy server in the Network Dispatcher routing table.
The next request from that client may come from Proxy 2.

Network Dispatcher does not know that the request is really from the same client and may route the request to WAS 2.

A better way would be to use the session key rather than the IP address to route requests. However, if SSL is enabled, Network Dispatcher cannot view the session key.
WebSphere Cloning Problem

Client —> Network Dispatcher (ND) —> HTTP

OSE plug-in

Big3Clone_A

Big3Clone_B

OSE plug-in

Big3Clone_C

Big3Clone_D
• The client request comes in and is routed to the upper HTTP server.

• The request is routed to Big3Clone_A. Session information is stored in the memory space for Big3Clone_A.
Another request from the same client is correctly routed from the network dispatcher to the HTTP server, but this time the request may be routed to a different clone. When this happens, the client session information is not available.
With session persistence, each clone can store session information in a session database.

You can also use session persistence in configurations that do not include the network dispatcher.

Note: The WAS repository, not shown here, may or may not reside on the same node. It is possible to configure the WAS repository to hold session information.
The client request is routed to Big3Clone_A. Session information is stored in the session database.
The next request from the same client may get routed to another clone or even another HTTP server.

The clone serving the request has access to the session information stored in the session database.
Client accesses the HTTP server, which passes the request to the plug-in.

Since the request has been through the HTTP server, the plug-in can view the session key, even if SSL is enabled.

In this case, there is no session key. This is the first request from the client. The plug-in uses normal round robin to route the request to a clone.

Each clone has a unique set of session keys that it can generate.

The clone creates the session and returns both the response and the session key to the client.

The session information is persisted to the session database.
The client accesses the HTTP server again. Based on the session key passed by the client, the plug-in routes the request to the same clone that generated the key. The clone checks the timestamp of the session in the database and compares it with the timestamp of the session it is holding in memory. If the timestamps match, the clone uses the in-memory session object to process the request. In this way, the clone does not have to read the entire session object from the database every time the session is accessed. The clone returns the response to the client and writes the session to the database with a new timestamp.
The client accesses the HTTP server again. Based on the session key, the plug-in knows the request belongs in Clone 1. However, the plug-in detects that Clone 1 has failed. The plug-in sends the request to Clone 2. Clone 2 does not have the session information stored in memory so it must read the session information from the session database. Clone 2 returns the response to the client and writes the session to the database with a new timestamp. Until Clone 1 is active, client requests will be routed to Clone 2, which now holds the session in memory.
Clone 1 becomes available again.
The client accesses the HTTP server. Based on the session key passed by the client, the plug-in routes the request to Clone 1.
Clone 1 must read the session information from the session database because the session information is not in memory.
The persistence type field has two choices: directodb, and EJB (not enabled)

Version 3.02 and above use connection pooling

Best practice is to use a database separate from the WAS repository for storing session information.
Summary

- In this presentation you've learned about workload management and the following concepts:
  - Servlet clustering
  - EJB workload management
  - TCP/IP spraying
  - Session management
Servlet Clustering and EJB Workload Management

What This Exercise is About

In this exercise you will be using Servlet Clustering by creating a model of the Application Server, choosing a workload management type, and adding an additional clone of the model. At the end of the exercise, you will run a workload management-enabled Java client program to access the Big3 application. You will use the resource analyzer to demonstrate that the work is being distributed evenly across the EJBs.

User Requirement

To understand Work Load Management, you need to understand how Servlet Clustering works. This lab will explain all the steps that are necessary to create a model and then clone the model.

What You Should Be Able to Do

You should become familiar with the steps that are necessary to perform Servlet Clustering. This involves creating a model of the Application Server, choosing the Workload Management Type, and then creating clones of the model. A model is required to create clones of your server. When you make changes to the model, they are automatically propagated into all clones.

Introduction

The lab consists of the following parts:

- Part One: Creating a Model Application Server
- Part Two: Creating Clones of Your Model Application Server
- Part Three: Starting the Model
- Part Four: Installing Resource Analyzer
- Part Five: Testing/Verifying
- Part Six: Using a WLM client and verifying EJB WLM using Resource Analyzer
Exercise Instructions

To complete this lab use the Solaris lab computer. Estimated time: 1 hour, 15 minutes

This exercise requires that the Big3 application be installed and correctly configured. Make sure the application is installed before proceeding with the lab. Simply follow the steps outlined below to complete the lab.

This lab is performed from the Administrative Console. To begin, launch the console and wait for it to finish its startup tasks. When it is ready, you will see the following at the bottom of the console GUI:

<table>
<thead>
<tr>
<th>11/12/99 12:57 PM</th>
<th>Loading ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/12/99 1:00 PM</td>
<td>Console Ready.</td>
</tr>
</tbody>
</table>

Part One: Creating a Model Application Server

You are now ready to create a model of the Big3Server Application Server.

1. The first task is to open the Topology view and expand the node that contains the Big3Server Application Server. Highlight Big3Server server.

Note: In this example, the node name is “wssun5”.

© Copyright IBM Corporation 2000. All rights reserved.
2. Right-click **Big3Server** and select Create --> Model.

3. Type the Model Name, **Big3Master** and check the **Make Big3Server a Clone** and **Recursively Model all Instances under Big3Server** check boxes.

   ![Model Properties](image)

   * Model Name: **Big3Master**
   * Make Big3Server a Clone
   * Recursively Model all Instances under Big3Server

   **Note:** Checking these boxes will turn the Application Server into a clone when the model is created. The first box ensures that the Big3Server becomes a clone and will participate in workload management. In addition, any changes to Big3Master will get propagated to both Big3Server and any clones created from the model. Making the server a clone is strongly recommended. The second box causes objects and containers under Big3Server to be cloned. If you do not select this box, your model will not have an EJB container, Servlet container, etc.

4. Select the **Advanced** tab and confirm the **Round Robin Perfer Local** Workload management selection policy is used. This is the default choice and usually is the best option to choose for most environments.

5. Click the **OK** button to create the model. The application server, Big3Server, will be converted into a clone at the same time. WebSphere will process your request. This may take a few minutes, depending on your configuration. When the model has been created, an Information dialog will display.

**Part Two: Creating Clones of Your Model Application Server**

At this point, you have a model created and one clone. The follow section of the lab will show you how to create a second clone.
1. Right click the **Big3Master** and select **Create --> Clone**.

2. Change the name to Big3clone.

3. Select your node and click the **Create** button.

Note: In WebSphere Application Server Advanced Edition Version 3.0, the **Big3EJBContainer** and the **Big3ServletEngine** are not listed under the new clone. They actually are there, just that they are currently not displayed. This is done for display speed. You can manually refresh the display to show the information. In Version 3.02 and higher, it is not necessary to refresh the view.
Part Three: Starting the Model

At this point, you have a model and two clones. The following section of the lab you will start both clones at the same time.

1. Right-click the **Big3Master** model and select **Start**.

Note: You could also start each clone individually, but starting the Model is easier because all clones are started at the same time.

Part Four: Installing Resource Analyzer

WebSphere Resource Analyzer is a technology preview that you can use to monitor objects in your configuration. You can download Resource Analyzer from the IBM WebSphere Web site. Resource Analyzer has already been downloaded for you and resides in the `/usr/big3/ra` directory.

1. Copy the Resource Analyzer compressed file from `/usr/big3/ra` to `<as_root>`. In a terminal window, change directories to `<as_root>` and type the following command:

   ```
   cp /usr/big3/ra/* .
   ```

2. Expand the compressed file with the following command. As the file uncompresses, the `/ra` directory and subdirectories will be created under `<as_root>`.

   ```
   unzip ra.zip
   ```

3. Change directories to `<as_root>/ra/bin`. Using a text editor, edit `ra.sh` to make sure the following entries are correct:

   - `$WAS_HOME`
   - `$JAVA_HOME` (hint: use the `find / -name java` command to locate the path)

4. Save and exit the editor.

Part Four: Testing/Verifying

You are now ready to make sure the clones are working. This section of the lab will show you how to use the Resource Analyzer to determine if the clones are working.

1. In a terminal window, go to the `/bin` directory where you installed Resource Analyzer. Start the analyzer by typing `./ra.sh` on the command line.
2. When the Resource Analyzer application starts, look at the content tree in the left pane. Click the icon to the left of Big3Server and Big3clone to open the tree. Locate Big3Server and, below that, servlet engine. Click the symbol to the left of servlet engine to open the object. Click the symbol to the left of default_host and Big3WebApp as well.

3. Locate Big3clone and open the tree as you did above.
4. Select **ProcessClaimServlet**. Hold down the **Ctrl** key and select **ProcessClaimServletClone**. Your display should look something like this:

5. On the menu bar at the top of the window, select **Actions-->Run**.

6. Clear any values by clicking the **Reset to Zero** button on the toolbar, then click the **Refresh** button.

7. Start a web browser and enter the URL: **http://<machine_name>/Big3/index.html**.

Note: Change `<machine_name>` to your computer’s name in the lab.
8. Click the **Submit Form** button several times (at least 10). Don’t change any other values on the web page. You will have to hit the “back” button on the browser and hit the submit button each time.

9. Look at the Resource Analyzer. You should see each instance of the ProcessClaim servlet serviced approximately the same number of requests.

### Part Six: Using a WLM Client /Verifying EJB WLM Using Resource Analyzer

1. On the Windows NT machine, edit the RunWLMClient.bat file found in the C:\big3 directory. Locate the Java command line and make sure the following text follows the cmdString=java entry: 

   ```
   -Dcom.ibm.CORBA.BootstrapHost=<admin_node_name>
   ```

   make sure you replace the `<admin_node_name>` with your Solaris host name.

2. On the Solaris, make sure that Big3Master is running and that the clones are running, as well.

   
   - Click the icon to the left of **enterprise bean --> Big3EJBContainer** on both Big3Server and Big3clone.
   - Hold down the **Ctrl** key and select the ProcessClaim, Claim and Entity beans from both EJB Containers. In the example that follows, the columns have been rearranged to show the number of instantiates for the session beans, and the method calls to the entity beans.
4. On the Windows NT machine, open an MS-DOS command window and go to the C:\big3 directory. Enter the following command, substituting the Solaris machine name for the `<machine_name>` parameter:

```
RunWLMClient <policy_number> <claim_number> <amount> <machine_name>
```

This command runs a batch file that submits 50 requests to the beans.

5. After the batch file has completed its work, view the Resource Analyzer. You should have a display that looks similar to this:
Notice that there are two rows of each bean; one for each clone. Each ProcessClaim session bean handled 25 of the 50 requests you submitted. ProcessClaim invokes the Claim bean once and the Policy bean twice. The workload is being balanced between the EJBs. The “extra” instantiation is created by the remove method in the client.

**What you did in this exercise**

In this exercise, you created a model application server out of a preexisting application server (Big3Server). The application server was then turned into a clone. From there you created an additional clone and then started both clones at the same time. You then tested the clones by submitting several claims and confirmed that both clones were working using the Resource Analyzer.

Once both clones were working and verified, you used a WLM-enabled Java client to access the EJBs directly to process claims. This action was verified using the Resource Analyzer.
Configuring IBM Network Dispatcher for TCP/IP Spraying

What This Exercise is About

In this exercise, you will configure IBM Network Dispatcher so that incoming requests are automatically routed to two separate HTTP Servers on two separate machines.

User Requirement

You will need one IP address for each machine, plus one additional IP address for the cluster address. These addresses must be recognized by the DNS server. It is important that all machines share the same LAN segment and that the data passing between the machines does not pass through routers or bridges.

You will also need to have installed IBM Network Dispatcher (ND), a part of IBM Edge Server. In this lab exercise, ND has been installed for you on the AIX machine in your configuration.

Finally, you will need WebSphere installed on the Windows NT machine and the Solaris machine, with the Big3 application deployed on both.

What You Should Be Able to Do

After completing this lab, you should be able to:

- Configure a loopback interface on the machines where the HTTP servers are installed.
- Configure ND to accept incoming requests from the Web client and route them to the HTTP servers.
- Verify that the requests are being routed to the HTTP servers.

Introduction

IBM Network Dispatcher allows you to route incoming requests to separate HTTP servers. You can configure the Network Dispatcher to perform simple round-robin routing, or use a more complex configuration to perform load balancing or content-based routing. There is a course designed specifically for Network Dispatcher. The intent of this exercise is to give you the basics on starting and running Network Dispatcher, and performing basic configuration tasks to allow incoming requests to be sprayed to separate HTTP servers.
**Exercise Instructions**

Network Dispatcher (ND) is installed on the AIX machine. You will configure a loopback interface on the Windows NT machine and on the Solaris machine, then perform the configuration tasks for ND on the AIX machine.

To run ND and successfully spray requests to the two HTTP servers, you must have the Web servers configured correctly. You must be logged into the ND machine as the root user to configure ND. In your configurations, WebSphere must be installed and Big3 must be deployed on the Windows NT machine and on the AIX machine.

Your ending configuration should match the diagram below:

![Diagram of network configuration](image)

In this lab you will:

- Make sure the hosts can talk to each other
- Configure the loopback interface on the HTTP server machines
- Configure ND
- Test the configuration

**Part One: Checking Communications**

1. From a terminal window on the AIX, use the `ping` command to make sure you can communicate with both the Solaris and the Windows NT machines:

   ```
   ping <machine_name>  (use Ctrl+C to break out of the ping if you need to)
   ```

2. From a terminal window on the Solaris machine use the `ping` command to make sure you can communicate with both the AIX and the Windows NT machine.

3. On the Windows NT machine, from either an MD-DOS command window or the Start-->Run prompt, ping the Solaris and the AIX machines.
4. Ensure that the HTTP servers are running on the Windows NT machine and the Solaris machine. Use a Web browser to request pages directly from http://<NT_machine_name> and http://<Solaris_machine_name>.

You should see a screen like the following:

![Welcome to the IBM HTTP Server](welcome_to:http:_ibm_com.png)

Welcome to the
IBM HTTP Server
powered by Apache

Everything you need to start
and use this server begins here...

- Configure server
- View documentation
- Visit our Web site

Part Two: Configuring the Loopback Interface on NT

Use the following steps to configure the loopback interface on the HTTP Server residing on the Windows NT machine. The loopback interface allows the HTTP Servers to accept traffic for big3cX.com (where X is your group number).
__1. Click **Start-->Settings-->Control Panel.** Then double-click **Network.**

__2. Add the MS Loopback Adapter driver.  
   — In the Network window, click **Adapters.**  
   — Click **Add.**  
   — Select **MS Loopback Adapter**, then click **OK.**  
   — Accept the default frame type.  
   — Accept the default location for copying files. Click **Continue.**

   ![Windows NT Setup](image)

   — In the Network window, click **Protocols.**  
   — Select **TCP/IP Protocol**, then click **Properties.**  
   — Select **MS Loopback Adapter**, then click **OK.**  
   — Set the loopback address to your cluster address (the IP address for big3cX.com, where x is your group number). Accept the default subnet mask (255.0.0.0) and do not enter a gateway address.  
   — Click **Apply**

   **Note:** You may have to exit and reenter Network Settings before the MS Loopback Driver shows up under TCP/IP Configuration.

__3. Sometimes, adding a loopback adapter causes extra routes to be added. You need to delete any extra routes. Open an MS-DOS command prompt and execute the following command: **route print**
You will see a table that may look like this:

<table>
<thead>
<tr>
<th>Network Destination</th>
<th>Netmask</th>
<th>Gateway</th>
<th>Interface</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>9.5.119.212</td>
<td>9.5.119.215</td>
<td>1</td>
</tr>
<tr>
<td>9.0.0.0</td>
<td>255.0.0.0</td>
<td>9.5.119.216</td>
<td>9.5.119.216</td>
<td>1</td>
</tr>
<tr>
<td>9.5.119.215</td>
<td>255.255.255.255</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>1</td>
</tr>
<tr>
<td>9.5.119.216</td>
<td>255.255.255.255</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>1</td>
</tr>
<tr>
<td>9.255.255.255</td>
<td>255.255.255.255</td>
<td>9.5.119.216</td>
<td>9.5.119.216</td>
<td>1</td>
</tr>
<tr>
<td>127.0.0.0</td>
<td>255.0.0.0</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>1</td>
</tr>
<tr>
<td>224.0.0.0</td>
<td>224.0.0.0</td>
<td>9.5.119.216</td>
<td>9.5.119.216</td>
<td>1</td>
</tr>
<tr>
<td>9.5.119.216</td>
<td>255.255.255.255</td>
<td>9.5.119.216</td>
<td>9.5.119.216</td>
<td>1</td>
</tr>
</tbody>
</table>

__4. Look in the Gateway column. In the diagram above, the cluster address is 9.5.119.216. Note that there is more than one entry matching the cluster address. Now look at your display for your cluster address. **What you are looking for is a number that starts with the same digit(s) as the beginning of your cluster address, followed by three zeros.** The extra address has been circled in the example above.

Note: Your cluster address may be different.

__5. Delete the extra route using the following command:

```
routedelete <network_destination> <cluster_address>
```

**Important note:** On Windows NT machines, you will have to delete the extra route every time you reboot the server. You can create a script to delete the extra route at startup.

---

**Part Three: Configuring the Loopback Interface on Solaris**

Use the following steps to configure the loopback interface on the HTTP Server residing on the Solaris machine. The loopback interface allows the HTTP Servers to accept traffic for big3cX.com.

__6. Configure the loopback interface using the following command from a terminal window on the Solaris (replace the X with your group number):

```
ifconfig lo0:1 big3cX.com 127.0.0.1 up
```
7. Check for extra routes using by typing `netstat -nr`. Like you did on the NT, you should be looking in the Gateway column for duplicates of your cluster address. If the address in the Destination column begins with the same digit(s) as the beginning of the IP address, followed by three zeros, you have an extra route. Your display should look something like this:

```
Sun Microsystems Inc. SunOS 5.7 Generic October 1998
# netstat -nr
Routing Table:
<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Flags</th>
<th>Ref</th>
<th>Use</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>9.5.119.216</td>
<td>WH</td>
<td>0</td>
<td>0</td>
<td>lo0:1</td>
</tr>
<tr>
<td>9.5.90.0</td>
<td>9.5.119.137</td>
<td>UG</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9.5.119.128</td>
<td>9.5.119.224</td>
<td>U</td>
<td>3</td>
<td>560</td>
<td>hme0</td>
</tr>
<tr>
<td>224.0.0.0</td>
<td>9.5.119.224</td>
<td>U</td>
<td>3</td>
<td>0</td>
<td>hme0</td>
</tr>
<tr>
<td>default</td>
<td>9.5.119.129</td>
<td>UG</td>
<td>0</td>
<td>2577</td>
<td></td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>WH</td>
<td>0</td>
<td>8916</td>
<td>lo0</td>
</tr>
</tbody>
</table>
#```

Make sure there are no extra routes. If you find an extra route, delete it using this command:
```
route delete -net <destination_address> <cluster_address>
```

For more information about configuring the loopback interface on different operating systems, see the *IBM Network Dispatcher: User’s Guide*, GC31-8496.

**Part Four: Configuring the Dispatcher component**

In this section, you will configure Network Dispatcher using the command line. All the steps in this part of the exercise will be performed on the AIX machine. During this part of the exercise, you will:

- Start the ndserver
- Start the executor function
- Add the cluster address to the Dispatcher configuration
- Add the HTTP protocol port to the Dispatcher configuration
- Add each of the HTTP servers to the Dispatcher configuration
— Configure the workstation to accept traffic for the cluster address

Note: If you were configuring Dispatcher on a Solaris machine using a 100Mbps Ethernet adapter, you would need to perform additional configuration steps. See the “Configuring the Dispatcher component” chapter in the *IBM Network Dispatcher: User’s Guide* for more details.

____1. Start the `ndserver` on Dispatcher by opening a terminal window and typing, `ndserver` on the command line.

   Note: a default configuration file (default.cfg) gets automatically loaded when starting ndserver. If you decide to save the Dispatcher configuration in default.cfg, then everything saved in this will be automatically loaded next time the ndserver gets started.

____2. To start the executor function, enter the `ndcontrol executor start` command.

   Note: You may also change various executor settings at this time. You'll find more information about these settings in Appendix B of the *IBM Network Dispatcher: User’s Guide*.

____3. Define the nonforwarding address (if different from the host name). The nonforwarding address is used to connect the Dispatcher machine for administrative purposes, such as using Telnet or SMTP to this machine. By default, this address is the hostname.

   To define the nonforwarding address, enter
   `ndcontrol executor set nfa <IP_address>`
   `<IP_address>` is either the machine name or the actual IP address.

____4. Dispatcher will balance the requests sent to the cluster address to the corresponding servers configured on the ports for that cluster. To define the cluster, issue the command `ndcontrol cluster add <cluster_name>`. Your cluster name is big3cX.com, where X is your group number.

____5. Once the cluster had been defined, you should configure the cluster address on one of the network interface cards of the Dispatcher machine. To do this, issue the command `ndcontrol cluster configure <cluster_address>`. This will look for an adapter with an existing address that belongs on the same subnet as the cluster address. It will then issue the system's adapter configuration command for the cluster address, using the adapter found and the netmask for the existing address found on that adapter.
Note: in rare cases, you may have a cluster address that does not match any subnet for existing addresses. In this case, use the second form of the cluster configure command and explicitly provide the interface name and netmask. Use `ndcontrol cluster configure <cluster_address> <interface_name> <netmask>`. For example, on a Solaris, the command could be `ndcontrol cluster configure 9.5.119.216 le0:1 255.255.0`. You will not need this command for this exercise!

6. Define the ports and set port options. To define a port, enter the `ndcontrol port add <cluster_address>:80` where `<cluster_address>` can be either big3cX.com or the IP address.

7. To add a load-balanced HTTP servers to the Dispatch configuration, enter `ndcontrol server add <cluster_address>:80:<Solaris_machine_name>` `ndcontrol server add <cluster_address>:80:<NT_machine_name>`

Note: you must define more than one server to a port to perform load balancing.

Your basic configuration is now complete. But you’re not done yet!

**Part Five: Configuring WebSphere to Recognize the Cluster**

Important! The following steps must be performed on BOTH the Windows NT machine and on the Solaris machine.

1. Start WebSphere and open the Administrative Console

2. Under the Topology tab, select default_host. Add entries for the cluster IP address and cluster name.

Note: the addresses in the picture that follows may not be the same as yours!
3. Open the node in the Administrative Console. Stop, then restart the Default Server (important if any of your web applications are use the default server).

4. Stop, then restart Big3Master (on Solaris) or Big3 (on Windows NT).

Part Six: Testing your Configuration

Now you’re ready to test your configuration. Using the web client, you will submit claims using the ProcessClaim servlet. You can use the HTTP servers access logs to see that your requests are being handled by both servers.

1. Make sure that the HTTP servers have been started. On the NT, look in Settings-->Control Panel-->Services to see if the IBMHTTP server is started. On Solaris, open a terminal window and type: `ps -ef | grep httpd`

2. Open a browser window on the AIX and access the following URL: `http://big3cX.com/Big3/index.html` (note: use your group number in place of the X).
If you open a browser on either of your HTTP server machines, all requests will go to that machine because of the loopback interface.

__3. Press and hold the **Shift** key while clicking the **Refresh** button several times.

__4. Open a terminal window on the AIX machine. On the menu bar of the terminal window, select **Options-->Terminal...** Disable End-of-line wrapping. Drag the edge of the window to make it wide. On the command line, type:

   `ndcontrol server report ::`

You should see a table that looks like the following:

```
+-----------------+-------+-------+-------+-------+-------+-------+-------+
| Address         | Total | TCP   | UDP   | Active | FINned | Complete| Scan  |
|-----------------+-------+-------+-------+-------+-------+-------+-------|
| 9.5.119.224     | 44035 | 44035 | 0     | 0     | 0     | 44035  | -1    |
| 9.5.119.215     | 43455 | 43455 | 0     | 0     | 0     | 43455  | -1    |
      +-----------------+-------+-------+-------+-------+-------+-------+
```

The numbers under the Total column should be close to equal.

Optional step: For a more interactive way to observe that both servers are handling requests, do the following:

- On the Solaris, change directories to `/opt/IBMHTTPD/logs`. View the access log by entering: `tail -f access_log`

- On the NT, right click the Start button. Select Explore. Find the `Program Files\IBM HTTP Server\Logs` directory on drive C.
• Open the file, **access.log**. You should see something like this:

![Image of access.log in Notepad](image)

The file shows that the HTTP server on that machine handled requests to get .html pages. The 200 that follows the HTML address indicates that the page was found successfully.

• Look at the Solaris window you opened earlier. Did some of the requests go to that server as well?

___5. On the Windows NT machine, stop the HTTP server.

___6. On the AIX machine, press **Shift+Refresh** several times. What happens?

___7. Restart the HTTP server on the Windows NT machine.

**Part Seven: Adding the Manager and Advisors**

___1. Start the manager function. The manager improves load balancing. To start the manager, enter the `ndcontrol manager start` command.

___2. Repeat steps 2 through 7 (omit the optional steps) from Part Six to test your configuration and compare the differences between the server reports. Is one server handling more requests? What happened when the HTTP server stopped responding to requests?

In order for Network Dispatcher to detect when an HTTP server is not responding so that the Dispatcher can route requests around it, you need an advisor running that will monitor the HTTP servers.
3. Start the advisor function. The advisors give the manager more information about the ability of the load balanced server machines to respond to requests. The advisors currently available are listed below, along with their default ports:

<table>
<thead>
<tr>
<th>Advisor Name</th>
<th>Protocol</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftp</td>
<td>FTP</td>
<td>21</td>
</tr>
<tr>
<td>telnet</td>
<td>Telnet</td>
<td>23</td>
</tr>
<tr>
<td>smtp</td>
<td>SMTP</td>
<td>25</td>
</tr>
<tr>
<td>http</td>
<td>HTTP</td>
<td>80</td>
</tr>
<tr>
<td>pop3</td>
<td>POP3</td>
<td>110</td>
</tr>
<tr>
<td>nntp</td>
<td>NNTP</td>
<td>119</td>
</tr>
<tr>
<td>ssl</td>
<td>SSL</td>
<td>443</td>
</tr>
<tr>
<td>Workload Manager</td>
<td>private</td>
<td>10,007</td>
</tr>
<tr>
<td>WTE</td>
<td>HTTP</td>
<td>80</td>
</tr>
<tr>
<td>PING</td>
<td>ping</td>
<td>0</td>
</tr>
</tbody>
</table>

To start the advisor, enter `ndcontrol advisor start http 80`

4. Set manager proportions as required. If you have started any advisors, you must change the manager proportions to allow the advisor information to included in the load balancing decisions. To set the manager proportions, type `ndcontrol manager proportions 50 50 0 0`

The four values above are related to the external factors the manager may use in its weighting decisions:

- The number of active connections on each load balanced server machine (as tracked by the executor)
- The number of new connections on each load balanced server machine (as tracked by the executor)
- Input from the advisors
- Input from system monitoring tools, such as ISS

You can change the relative proportion of importance to the manager four values. Think of the proportions as being percentages. The sum of the relative proportions must equal 100 percent. The default ratio is 50 50 0 0, which ignores the advisor and system information. However, the manager is still providing information to the Dispatcher, so some load balancing is taking place. You may wish to adjust these proportions to find the combination that gives the best performance. For load balancing, avoid setting the first two values below 20.

5. Repeat steps 2 through 7 (omit the optional steps) from Part Six to test your configuration and compare the differences between the server reports. Is one server handling more requests? What happened when the HTTP server stopped responding to requests?
6. Now, change the proportions the manager uses to give more relative weight to the advisor. Reset the manager proportions:
   `ndcontrol manager proportions 40 40 20 0`

7. Repeat steps 2 through 7 (omit the optional steps) from Part Six to test your configuration and compare the differences between the server reports. Is one server handling more requests? What happened when the HTTP server stopped responding to requests?

Dispatcher will now make sure that client requests are not sent to a failed HTTP server.

8. For more fun, have your neighboring group access your cluster address and watch the access logs for activity

**What you did in this exercise**

During this exercise, you configured your HTTP servers on two platforms to accept requests addressed to the cluster. You also configured WebSphere to recognize that address as well. You configured IBM Network Dispatcher on an AIX host to accept incoming requests and to distribute them to the HTTP servers. During the configuration, you set the parameters to allow Network Dispatcher to balance the work load between the receiving HTTP servers. Finally, you tested your configuration by accessing the cluster using a web browser and by looking at the access logs on each HTTP server.

This exercise gave an overview of configuring and testing IBM Network Dispatcher using the manager and advisor to perform load balancing and to detect and status of the HTTP servers. Many of the configuration steps you just completed can also be performed using the GUI (on AIX, type `ndadmin` to start the GUI), or by editing a configuration file. For more information on these tasks and more, see the *IBM Network Dispatcher: User’s Guide*, or the *IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher* Redbook, SG24-5858-00, or look for whitepapers on the WebSphere Web site.