Business Grid Components

for WebSphere Extended Deployment

Installation and Configuration Guide

Nov. 18, 2004

Table of Contents

Chapter 1 Introduction

This document describes the installation steps for Business Grid Components for WebSphere Extended Deployment for Linux running Red Hat Enterprise Linux (RHEL) 3.0.

Chapter 2 describes the pre-requisites needed to install and run the Business Grid Components.

Chapter 3 describes the Business Grid installation (bginstall) utility.

Chapter 4 outlines the steps needed to complete the configuration of the Business Grid Components for WebSphere Extended Deployment master node machine.

Chapter 5 outlines the steps needed to complete the configuration of the Business Grid Components for WebSphere Extended Deployment worker node machine.

Chapter 6 outlines the steps needed to manually install and configure the Business Grid Components for WebSphere Extended Deployment master node machine. Follow the steps outlined in this chapter if you cannot (for example the HA implementation described in chapter 8) or do not want to use the bginstall utility.

Chapter 7 outlines the steps needed to manually install and configure the Business Grid Components for WebSphere Extended Deployment worker node machine. Follow the steps outlined in this chapter if you cannot (for example the HA implementation described in chapter 8) or do not want to use the bginstall utility.

Chapter 8 describes the installation and configuration steps needed to create a Highly Available Business Grid setup. Start with this chapter if High Availability (HA) is a key requirement for your setup.

Chapter 9 contains links to some valuable and relevant resources.

Chapter 2 Pre-installation planning

The Business Grid Components for WebSphere Extended Deployment is an amalgamation of J2EE applications, shell scripts and other IBM products. For more information about the Business Grid architecture refer to the Business Grid Components White Paper (see resources for link). The remainder of this document assumes you are familiar with the basic architecture of the WebSphere Business Grid.

To simplify installation, the Business Grid Components for WebSphere Extended Deployment install process assumes your environment will consist of a single master machine and some number of worker machines.

Master Node: The master machine runs the WebSphere Application Server deployment manager, the WebSphere Application Server for the Business Grid Gateway, the WebSphere Business Grid Gateway, the LoadLeveler Central Manager and Scheduling Daemon and WebSphere MQ Queue Manager.

To support these functions, the following packages should be installed on the master machine (refer to the resources section for links to the relevant installation documents):

1. IBM LoadLeveler 3.2
2. WebSphere MQ 5.3.0.2
3. WebSphere MQ 5.3 fix pack 7
4. WebSphere Application Server Network Deployment (ND) 5.1
5. WebSphere Application Server Network Deployment (ND) 5.1 Fix Pack 1 (5.1.1)
6. WebSphere Application Server Network Deployment (ND) 5.1.1 Cumulative Fix 1 (5.1.1.1)
7. WebSphere Application Server 5.1
8. WebSphere Application Server 5.1 Fix Pack 1 (5.1.1)
9. WebSphere Application Server 5.1.1 Cumulative Fix 1 (5.1.1.1)
10. WebSphere Application Server 5.1.1 Cumulative Fix for SDKs
11. WebSphere Extended Deployment 5.1

Worker Node: The worker machines each run a WebSphere Application Server, WebSphere Business Grid executing machine logic, a LoadLeveler Startd and WebSphere MQ Client (except for the Highly Available configuration outlined in chapter 8, in which case WebSphere MQ server is needed as well).

To support these functions, the following packages should be installed on the worker machines (refer to the resources section for links to the relevant installation documents):

1. IBM LoadLeveler 3.2
2. WebSphere MQ 5.3.0.2
3. WebSphere MQ 5.3 fix pack 7
4. WebSphere Application Server 5.1
5. WebSphere Application Server 5.1 Fix Pack 1 (5.1.1)
6. WebSphere Application Server 5.1.1 Cumulative Fix 1 (5.1.1.1)

7. WebSphere Application Server 5.1.1 Cumulative Fix for SDKs
8. WebSphere Extended Deployment 5.1

Note:

1. Install RedHat Enterprise Linux Enterprise Server version 3.0 update 1 and bring all packages up current levels using the RedHat Network (http://www.redhat.com/rhn). If you have not already done so, you will need to obtain RHN entitlements to upgrade the machines. In addition to the default set of packages, make sure you install then openmotif-2.2.3-3.RHEL3 RPM package. It will be needed by IBM LoadLeveler.

2. Edit /etc/hosts and remove the machine's long and short host names from the entry for 127.0.0.1. "localhost.localdomain" and "localhost" should be the only hostnames on the line for IP address 127.0.0.1. The presence of the additional host names complicates and often breaks the process of adding the machine to the WebSphere cell.

3. Enable the NTP daemon on the machine. IBM LoadLeveler will not function correctly unless all clocks in the cluster are synchronized.

4. When installing WebSphere Application Server products, do **not** install the embedded messaging components. Business Grid Components for WebSphere Extended Deployment has only been tested with WebSphere MQ.

5. You need to install WebSphere Application Server Deployment Manager and WebSphere Application Server Base on the master node. The WebSphere Application Server Base on the master node will host the Business Grid Gateway.

6. Make sure the WebSphere Application Server Base is installed in the same location on all the worker nodes and the master node.

7. When upgrading to WebSphere XD on the master node, make sure you do it twice – once for the WebSphere Application Server ND deployment manager and once for the WebSphere Application Server Base.

8. Due to `wsadmin` limitations, we are not able to script the creation of the service classes needed by the sample applications. **Before** starting the `bginstall` script (described in the next chapter) you will need to manually define the `echoGridSC` and `mandelbrotGridSC` service classes via the following steps:

   a. Start the WebSphere Extended Deployment Deployment Manager using the `startManager` command.
   b. Log in to the WebSphere admin console.
   c. Expand "Operational Policies" and select "Service Policies".
   d. Press "New" and enter the following values: Name: `echoGridSC` and Press Next twice.
   e. Press "Finish".
   f. Repeat steps b-e to create a `mandelbrotGridSC` service class.

g. Save your changes.
h. Stop the Deployment manager using the `stopManager` command.

Chapter 3 Installation

The Business Grid Components for WebSphere Extended Deployment installation script assumes that you have installed all of the prerequisite products listed in the previous chapter and set up a WebSphere cell containing all of the master and worker machines. Before starting the Business Grid Components install, ensure that you have stopped the deployment manager and all node agents and application servers in the cell.

Untar the Business Grid Components tarball to a temporary directory:

```
# mkdir /tmp/bgrid
# tar xvzf BGrid.tar.gz –C /tmp/bgrid
```

Consult the README file for any corrections or additions to this install documentation.

The Business Grid Components installation and configuration script is located in the temporary directory you created and is named **bginstall**. It accepts the following parameters:

--install-type master|worker
  indicates the type of install to be performed (required for all installs)

--cell-name <cell_name>
  the name of your WebSphere cell; defaults to the first cell located in the WebSphere configuration directory

--mq-dir <mq_dir>
  location of the WebSphere MQ installation; defaults to /opt/mqm

--wasnd-dir <was_nd_dir>
  location of the WebSphere Application Server Network Deployment installation; defaults to /opt/WebSphere/DeploymentManager

--was-dir <was_dir>
  location of the WebSphere Application Server installation; defaults to /opt/WebSphere/AppServer

Since much of the configuration of Business Grid Components must be done in conjunction with the installation, the bginstall script does both. See the following chapters for more details on the configuration steps that will be performed by the bginstall script and details on additional configuration steps that you will need to perform manually. The bginstall script is written to check for existing configuration data and to avoid overwriting any configuration data it finds.

For both master and worker machines, the bginstall script must be run while logged on as root. The recommended invocation to install Business Grid Components on the master machine is:

```
# bginstall --install-type master 2>&1 | tee bginstall.log
```

For a worker machine, the recommended invocation is:

```
# bginstall --install-type worker 2>&1 | tee bginstall.log
```

Chapter 4 Configuring a Master Node

In addition to installing Business Grid Components, the bginstall script performs the following configuration steps on the master machine:

- adds entries to /etc/sudoers to allow the LoadLeveler id to execute selected Business Grid Components commands as the root and mqm users
- adds the root user to the mqm user group
- defines and starts a bgrid.queue.manager queue manager in WebSphere MQ
- defines BGRID.QUEUE, BGRIDRSP.QUEUE and ENDPT.REPLYQ queues for bgrid.queue.manager
- defines a BGRID.CHANNEL channel for bgrid.queue.manager and starts a TCP listener for the channel
- starts the LoadLeveler daemons
- sets the MQ_INSTALL_ROOT environment variable for the non-deployment manager WebSphere node
- creates WebSphere definitions corresponding to the WebSphere MQ queue manager and queues listed above
- creates a new bgridGateway application server on the non-deployment manager WebSphere node
- defines WebSphere listener ports for the BGRID.QUEUE and ENDPT.REPLYQ queues
- creates a new WebSphere Extended Deployment node group named bgridNG and adds the non-deployment manager WebSphere node to it
- defines a new URL provider to support the Business Grid Components JMS transport on the non-deployment manager WebSphere node
- adds host '*', ports 9081 and 9082 to the default_host WebSphere virtual host
- starts the WebSphere deployment manager, node agent and bgridGateway application server

Some aspects of Business Grid Components configuration are not amenable to automation and must be performed manually. These steps are described below.

1. Configure the Business Grid Components gateway

The Business Grid Components gateway has a number of configuration options that are not directly settable via bginstall. The default settings will work for simple environments, but will need to be updated for more complex scenarios (for example, if you wish to use JMS between the gateway and the executing machines). If you do have a need to change the settings, the correct procedure is described below.

a. Stop the bgridGateway server.

b. From a shell prompt on the node that runs the bgridGateway server, issue the following commands:

```
  # cd <WAS_base_dir>
  # cd <WAS_base_dir>/installedApps/<cell_name>/BGrid.ear
  # unzip BGridEJB.jar bgrid.properties

    <edit bgrid.properties as needed>
    < If you plan to use JMS, change the replyTo property to the following: >

replyTo=jms:/queue?channel=BGRID.CHANNEL/TCP&host=master_node_host
_name&\
     queueName=ENDPT.REPLYQ&queueManager=bgrid.queue.manager
     < Make sure you replace the master_node_host_name with the name of your
    master node>

  # zip BGridEJB.jar bgrid.properties
```

Comments in the bgrid.properties file describe the meaning, syntax and default value of each of the properties. Note that these changes will be overwritten if the BGrid.ear application is updated or redeployed to the server for any reason.

2. Configure the Business Grid Node Group to balance workload

The following steps will allow workload to be balanced between XD interactive applications and applications scheduled by the job scheduler:

a. Log into the WebSphere Deployment Manager Console.

b. Open the "System Administration" tree element (click on "+")

c. Select "Business Grid Node Groups"

d. Select the "BGridNG" node group to edit the group’s General Properties.

e. (Optionally) Edit the “Description” property to provide information about the node group.

f. Edit the property "BGUsage Mode" with a value of "true" to activate balancing,

g. Edit the property "XDMinNodes". Set the value to the number of gateways plus the minimum number of XD application server nodes to keep active. This value should always be at least 1 greater than the number of gateway nodes.

h. Edit the property "BGMinNodes" with a value greater than or equal to 1. The value of BGMinNodes plus XDMinNodes should usually equal the number of nodes in the node group.

i. Edit the property “BGOvercommittedMaxNodes” with a value greater than BGMinNodes, but less than or equal to the number of nodes in the node group minus XDMinNodes. BGOvercommittedMaxNodes is used by the balancer in situations where all nodes are overcommitted. The balancer will move with a value greater than BGMinNodes, but less than or equal to the number of nodes in the node group minus XDMinNodes. BGOvercommittedMaxNodes is used by the balancer in situations where all nodes are overcommitted. The balancer will move toward this number of Business Grid nodes over the node group to achieve a prescribed balance between nodes allocated to XD interactive work and Business Grid work in this situation.

j. Edit the property "BalancerInterval" with the value set to a number representing the number of minutes between balancer executions. As a starting point, use a value of 2 to 10 and adjust the value later as necessary.

k. Press “apply” and “ok” (Contine to the next step for initial configuring. Click “save” if no more editings are required during maintenance cycle).

l. Select “Node Group Member Information” in the “Additonal Properties” section to edit node group member scheduling capabilities.

m. Select each node capable of scheduling work in the node group, and click “set node BGUsage Candidacy”. Be sure this property is not active for the Gateway and DeploymentManager nodes.

n. Click “back” on the internet browser.

o. Select “Business Grid Balancer Matrix Properties” in the “Additonal Properties” section to edit the Balancer properties.

i. Select “new” to enter the matrix column information.
   a. Edit the name field by entering “com.ibm.websphere.xd.bgrid.matrixColHdrs”
   b. Edit the value field with a set of numbers separated by a dash (e.g. 0.0-25.0-50.0-75-100). The size of this number set determines the size of number of columns of the “Balancer Matrix”.
   c. Click “apply” and “ok”

ii. Select “new” to enter the matrix row information
   a. Edit the name field by entering “com.ibm.websphere.xd.bgrid.matrixRowHdrs”
   b. Edit the value field with a set of numbers separated by a dash (e.g. 0.0-25.0-50.0-75-100). The size of this number set determines the size of number of rows of the “Balancer Matrix”.
   c. Click “apply” and “ok”

iii. Select “new” to enter information for a row in the balancer matrix.
   a. Edit the name field by entering “com.ibm.websphere.xd.bgrid.matrixRowNN” where NN is a numeric value starting from 1.

b. Edit the value field with a set of action codes separated by a dash (e.g. z-z-x-x-x). The size of the set is the same as the number of columns as defined by “matrixColHdrs”.
c. Possible value for action codes are:
   Z = do nothing
   X = give a node to XD
   B = give a node to BG
   O = move to overcommitted max where the OC property value is used to determine how many BG nodes should be allocated.
d. Click “apply” and “ok”

iv. Repeat the adding of the row information for as many rows as defined by “matrixRowHdrs”.

p. Return to the “Business Grid Node Groups” Configuration panel and click “save” to apply changes to the master configuration.

Chapter 5 Configuring a Worker Node

In addition to installing Business Grid Components, the bginstall script performs the following configuration steps on worker machines:

- adds entries to /etc/sudoers to allow the LoadLeveler id to execute selected Business Grid Components commands as the root and mqm users

Some aspects of Business Grid Components configuration are not amenable to automation and must be performed manually. These steps are described below.

1. Add worker machine nodes to the bgridNG node group

a. Log on to the WebSphere admin console.
b. In the navigation tree, expand “System Administration” and select “Node Groups”.
c. Select “bgridNG”.
d. On the “Membership” row, select the worker nodes to be added to the bgridNG node group and press “Add >>”.
e. Save your changes.

Chapter 6 Manual Install and Configuration of a Master Node

Perform the steps, outlined in this chapter, to manually do the installation and configuration of the Business Grid Components for WebSphere Extended Deployment master node. Use this option only if you wish not to use the automatic install utility or if using it is not an option (e.g. HA implementation described in chapter 8):

Untar the Business Grid Components tarball to a temporary directory on the master node:

```
# mkdir /tmp/bgrid
# tar xvzf BGrid.tar.gz –C /tmp/bgrid
```

1. **Define WebSphere MQ as a JMS provider to the WebSphere cluster.**
   a. Log in to the WebSphere admin console.

   b. In the navigation tree, expand "Environment" and select "Manage WebSphere Variables".

   c. Select the non-Manager node on the master machine and press "Apply".

   d. Select "MQ_INSTALL_ROOT".

   e. Enter the following values:
   Value: /opt/mqm
   and press "OK".

   f. Save your changes.

2. **Add root to the mqm user group.**
   a. While logged on as root, execute the following command:

   ```
   # usermod -G $(id -G root | tr ' ' ','),mqm root
   ```

   b. Log off and back on and issue the following command:

   ```
   # id
   ```

   c. Verify that mqm shows up in the list of groups.

3. **Create and start the WebSphere MQ queue manager and queues.**
   While logged on as root, execute the following commands:

   ```
   # su – mqm
   ```

```
mqm$ export LD_ASSUME_KERNEL=2.4.19

mqm$ crtmqm bgrid.queue.manager

mqm$ strmqm bgrid.queue.manager

mqm$ runmqsc bgrid.queue.manager
```

Note: Verbiage from runmqsc shows up, enter the following commands to runmqsc. Be sure to type the queue name in capital letters.

```
define qlocal (BGRID.QUEUE)

define qlocal (BGRIDRSP.QUEUE)

define qlocal (ENDPT.REPLYQ)
```

Note: The following “define channel” command is too long for runmqsc and must be entered on 2 lines using '+' as continuation character.

```
define channel (BGRID.CHANNEL) chltype (SVRCONN) +
trptype (TCP) mcauser ('mqm')

end

mqm$ runmqlsr -t tcp -m bgrid.queue.manager &

mqm$ exit
```

4. **Define queues and queue manager to WebSphere.**

   a. Log in to the WebSphere admin console.

   b. In the navigation tree, expand "Resources" and select "WebSphere MQ JMS Provider".

   c. Select the non-Manager node on the master machine and press "Apply".

   d. Select "WebSphere MQ Queue Connection Factories".

   e. Press "New".

   e. Enter the following values:
   Name: BGGWQCF
   JNDI Name: jms/BGGWQCF
   Queue Manager: bgrid.queue.manager
   Transport Type: Bindings
   and press "OK".

f. Select "WebSphere MQ JMS Provider".

g. Select "WebSphere MQ Queue Destinations".

h. Press "New".

i. Enter the following values:
Name: BGGWInboundQ
JNDI Name: jms/BGGWQ
Base Queue Name: BGRID.QUEUE
Base Queue Manager Name: bgrid.queue.manager
and press "OK".

j. Press "New" again.

k. Enter the following values:
Name: BGGWOutboundQ
JNDI Name: jms/BGGWOUTQ
Base Queue Name: BGRIDRSP.QUEUE
Base Queue Manager Name: bgrid.queue.manager
and press "OK".

l. Press "New" once again.

m. Enter the following values:
Name: BGEndpointQ
JNDI Name: jms/BGEPTQ
Base Queue Name: ENDPT.REPLYQ
Base Queue Manager Name: bgrid.queue.manager
and press "OK".

n. Save your changes.

5. Define a new WebSphere server to run the WebSphere Business Grid gateway.

a. Log in to the WebSphere admin console.

b. In the navigation tree, expand "Servers" and select "Application Servers".

c. Press "New".

d. Enter the following values:
Select node: <master_node>
Server name: bgridGateway
and press "Next".

e. Press "Finish".

f. Select "bgridGateway".

g. Select "Message Listener Service".

h. Select "Listener Ports".
i. Press "New".

j. Enter the following values:
Name: BGGWInboundQ
Initial State: Started
Connection factory JNDI name: jms/BGGWQCF
Destination JNDI name: jms/BGGWQ
and press "OK".

k. Press "New" again.

l. Enter the following values:
Name: BGEndpointQ
Initial State: Started
Connection factory JNDI name: jms/BGGWQCF
Destination JNDI name: jms/BGEPTQ
and press "OK".

m. Save your changes.

6. Install and deploy the WebSphere Business Grid gateway application.

a. Issue the following command as root:

```
# tar xzf /tmp/bgrid/BGrid-gateway.tar.gz –C /opt/WebSphere/AppServer
```

b. Log in to the WebSphere admin console.

c. In the navigation tree, expand "Applications" and select "Install New Application".

d. Select "Server path", enter "/tmp/bgrid/BGrid.ear" and press "Next".

e. Continue pressing "Next" until you get to Step 6, "Map modules to application servers".

f. In step 6, be sure that all application modules are set to be deployed on the bgridGateway server.

g. Continue pressing "Next" until you get to Step 8, "Summary".

h. Press "Finish".

i. Save your changes.

7. Configure the WebSphere Business Grid gateway

This is done using a properties file contained in the BGridEJB.jar module. If you are setting up your environment as described here and you are using HTTP as the protocol throughout (e.g. from client to gateway and gateway to worker (endpoint) nodes), you will not need to change any of these settings. If using JMS either from client, to endpoint or both you will need to change these settings. If you do have a need to change the settings, the correct procedure is described below.

a. Stop the bgridGateway server.

b. From a shell prompt on the node that runs the bgridGateway server, issue the following commands:

```
  # cd <WAS_base_dir>
  # cd <WAS_base_dir>/installedApps/<cell_name>/BGrid.ear
  # unzip BGridEJB.jar bgrid.properties

  <edit bgrid.properties as needed>
 < If you plan to use JMS, change the replyTo property to the following: >

replyTo=jms:/queue?channel=BGRID.CHANNEL/TCP&host=master_node_host
_name&\
  queueName=ENDPT.REPLYQ&queueManager=bgrid.queue.manager
  < Make sure you replace the master_node_host_name with the name of your
 master node>

  # zip BGridEJB.jar bgrid.properties
```

Comments in the bgrid.properties file describe the meaning, syntax and default value of each of the properties. Note that these changes will be overwritten if the BGrid.ear application is updated or redeployed to the server for any reason.

8. Install and deploy the WebSphere Business Grid administration application.

a. Log in to the WebSphere admin console.

b. In the navigation tree, expand "Applications" and select "Install New Application".

c. Select "Server path", enter "/tmp/bgrid/BGridAdminApp.ear" and press "Next".

d. Press "Next".

e. Check the box next to "Deploy WebServices".

f. Continue pressing "Next" until you get to Step 3, "Map modules to application servers".

g. In step 3, be sure that all application modules are set to be deployed on the bgridGateway server.

h. Press "Next".

i. Press "Finish".

j. Save your changes.

9. Configure the WebSphere Business Grid administration application.

a. On the master node, edit /opt/WebSphere/DeploymentManager/config/cells/<cell_name>/applications/BGridAdminApp.ear/deployments/BGridAdminApp/BGridAdmin.war/WEB-INF/web.xml and make the following changes:

In the businessGridGatewayHost env-entry, replace the string "your.gateway.host:9080" with the fully qualified hostname and port number of the bgridGateway server.

b. Log in to the WebSphere admin console.

c. In the navigation tree, expand "System Administration" and select "Nodes".

d. Select the checkbox next to the node on which the bgridGateway server resides.

e. Press "Full Resynchronize".

10. Define a Node Group to run the WebSphere Business Grid applications.

a. Log in to the WebSphere admin console.

b. In the navigation tree, expand "System Administration" and select "Node Groups".

c. Press "New".

d. Enter the following values:
Name: bgridNG
and press "Next".

e. Add the non-Manager node on the master machine and the nodes on the worker machines as members of the node group and press "Next".

f. Press "Finish".

g. Save your changes.

11. Install and deploy the grid half of the sample applications.

1. Define a dynamic cluster for the Echo grid sample application.

a. Log in to the WebSphere admin console.

b. In the navigation tree, expand "Servers" and select "Dynamic Clusters".

c. Press "New".

d. Enter the following values:
Dynamic Cluster name: echoGridDC
Map to Node Group: bgridNG
and press "Next".

e. Press "Finish".

2. Install the Echo grid sample application EAR file.

a. In the navigation tree, expand "Applications" and select "Install New Application".

b. Select "Server path", enter "/tmp/bgrid/EchoGrid.ear" and press "Next".

c. Continue pressing "Next" until you get to Step 3, "Map modules to application servers".

d. In step 3, be sure that all application modules are set to be deployed to the echoGridDC dynamic cluster.

e. Press "Next".

f. Press "Finish".

3. Define a service and transaction class for the Echo grid sample application.

a. In the navigation tree, expand "Operational Policies" and select "Service Policies".

b. Press "New", enter the following values:
Name: echoGridSC
and press "Next".

c. On Step 3, "Define Service Policy Memberships", press "New", enter the following values:
Name: echoGridTC
and press "Next".

d. In Step 2, "Define Transaction Class Memberships", select the EchoGrid application and the EchoGridWeb.war module.

e. Add the "/Echo/*" URI as a member of echoGridTC and press "Next".

f. Press "Finish".

g. Press "Next".

h. Press "Finish".

4. Repeat for the Mandelbrot grid sample application.

a. Repeat steps 1-3 (above) for the Mandelbrot grid application. Use the following values for the Mandelbrot grid application:

- Dynamic cluster name (steps 1.d and 2.d): mandelbrotGridDC
- EAR file location (step 2.b): /tmp/bgrid/MandelbrotGrid.ear
- Service class name (step 3.b): mandelbrotGridSC
- Transaction class name (steps 3.c and 3.e): mandelbrotGridTC
- URI (step 3.e): "/Mandelbrot/*"

5. Save your changes.

12. Define additional port numbers for default_host.

When the dynamic clusters were defined for the grid applications above, WebSphere XD automatically generated server definitions for each dynamic cluster/node combination. As part of this generation, XD selected port numbers for the HTTP transports of these new servers so that the new servers' port numbers would not conflict with the port numbers defined for other servers on the same node. These new port numbers need to be associated with the default_host virtual host.

1. Find the port numbers assigned to the dynamic cluster servers

a. Log in to the WebSphere admin console.

b. In the navigation tree, expand "Servers" and select "Application Servers".

c. For each of the dynamic cluster servers defined by WebSphere XD, do the following:

i. Select the server.
ii. Select "Web Container".
iii. Select "HTTP transports".
iv. Note the port numbers used by the server

2. Add these port numbers to default_host

a. In the admin console navigation tree, expand "Environment" and select "Virtual Hosts".

b. Select "default_host".

c. Select "Host Aliases".

d. For any of the port numbers noted above that are NOT already in the list of port numbers defined for this virtual host, do the following:

i. Press "New".
ii. Enter the following values:
Host Name: *
Port: <port_number>
and press "OK"

3. Save your changes.

13. Install and deploy the sample client applications.

a. Log in to the WebSphere admin console.

b. In the navigation tree, expand "Applications" and select "Install New Application".

c. Select "Server path", enter "/tmp/bgrid/EchoSampleApp.ear" and press "Next".

d. Press "Next".

e. Check the box next to "Deploy WebServices".

f. Continue pressing "Next" until you get to Step 3, "Map modules to application servers".

g. In step 3, be sure that all application modules are set to be deployed on the bgridGateway server.

h. Press "Next".

i. Press "Finish".

j. Repeat steps b through i for
"/tmp/bgrid/MandelbrotSampleApp.ear",
but do NOT check the "Deploy WebServices" box in step e.

k. Save your changes.

14. Configure the sample client applications.

a. On the master node, edit
/opt/WebSphere/DeploymentManager/config/cells/<cell_name>/applications/EchoSampleApp.ear/deployments/EchoSampleApp/EchoSample.war/WEB-INF/web.xml
and make the following changes:

i. In the businessGridGatewayHost env-entry, replace the string "your.gateway.host:9080" with the fully qualified hostname and port number of the bgridGateway server.

ii. In the businessGridServiceHost env-entry, replace the string "node.in.echoGridDC:9081" with the fully qualified hostname and port number of one of the servers in the echoGridDC dynamic cluster.

b. On the master node, edit
/opt/WebSphere/DeploymentManager/config/cells/<cell_name>/applications/MandelbrotSampleApp.ear/deployments/MandelbrotSampleApp/MandelbrotSample.war/WEB-INF/web.xml
and make the following changes:

i. In the businessGridGatewayHost env-entry, replace the string "your.gateway.host:9080" with the fully qualified hostname and port number of the bgridGateway server.

ii. In the businessGridServiceHost env-entry, replace the string "node.in.mandelbrotGridDC:9081" with the fully qualified hostname and port number of one of the servers in the mandelbrotGridDC dynamic cluster.

iii. In the businessGridGatewayQueue and replyToQueue env-entry's, replace the string "your.gateway.host:9811" on the jndiProviderURL portion of each parameter with the fully qualified hostname and port number of the JNDI provider for the bgridGateway server.

c. Log in to the WebSphere admin console.

d. In the navigation tree, expand "System Administration" and select "Nodes".

e. Select the checkbox next to the node on which the bgridGateway server resides.

f. Press "Full Resynchronize".

15. Install and configure the BGrid (Async) JMS Transport

a. Copy /tmp/bgrid/BGridJmsTransport.jar to opt/WebSphere/AppServer/lib/ext

b. Log in to the WebSphere admin console.

c. In the navigation tree, expand "Resources" and select "URL Providers".

d. Select the non-Manager node on the master machine and press "Apply".

e. Click on "New".

f. Enter the following values:
Name: BGridJmsTransport
Stream Handler Classname: com.ibm.ws.bgrid.protocol.bgridjms.Handler
Protocol: bgridjms
and press "OK".

g. Save your changes.

16. Start the WebSphere Business Grid gateway.

a. Start the bgridGateway server using the WebSphere admin console.

17. Configure and start LoadLeveler on the master machine.

a. On the master machine, edit /home/loadl/LoadL_admin and add a line of the form:

```
<fully.qualified.host.name>:   type = machine
```

for each worker machine.

b. Issue the following command:

```
# su - loadl -c "llctl start"
```

18. Prepare for client application development.

Untar /tmp/bgrid/BGrid-develop.tar.gz in a directory that will be accessible in your client application development environment. This tar file contains XSL transforms you can apply to standard WSDL documents to generate new WSDL documents that exploit WebSphere Business Grid functionality.

19. Add Business Grid enhancement to WebSphere admin console.

The following steps will update the admin console:

a. Log on as root and issue the following command:

```
# cd /opt/WebSphere/DeploymentManager/bin
# ./PluginProcessor.sh -install –moduleExtension /tmp/bgrid/bgnodegroups.war
```

b. Restart the WebSphere Deployment manager.

```
# ./stopManager.sh
# ./startManager.sh
```

20. Install the Business Grid Balancer Code

The following steps will install the balancer code:

a. Log on as root and issue the following command:

```
# cp /tmp/bgrid/bgbalancer.jar /opt/WebSphere/DeploymentManager/lib
# tar xvzf /tmp/bgrid/bgbalancer.tgz -C /opt/WebSphere/DeploymentManager
```

21. Configure the Business Grid Node Group to balance workload

The following steps will allow workload to be balanced between XD interactive applications and applications scheduled by the job scheduler:

a. Log into the WebSphere Deployment Manager Console.

b. Open the "System Administration" tree element (click on "+")

c. Select "Business Grid Node Groups"

d. Select the "BGridNG" node group to edit the group’s General Properties.

e. (Optionally) Edit the “Description” property to provide information about the node group.

f. Edit the property "BGUsage Mode" with a value of "true" to activate balancing,

g. Edit the property "XDMinNodes". Set the value to the number of gateways plus the minimum number of XD application server nodes to keep active. This value should always be at least 1 greater than the number of gateway nodes.

h. Edit the property "BGMinNodes" with a value greater than or equal to 1. The value of BGMinNodes plus XDMinNodes should usually equal the number of nodes in the node group.

i. Edit the property “BGOvercommittedMaxNodes” with a value greater than BGMinNodes, but less than or equal to the number of nodes in the node group minus XDMinNodes. BGOvercommittedMaxNodes is used by the balancer in situations where all nodes are overcommitted. The balancer will move with a value greater than BGMinNodes, but less than or equal to the number of nodes in the node group minus XDMinNodes. BGOvercommittedMaxNodes is used by the balancer in situations where all nodes are overcommitted. The balancer will move toward this number of Business Grid nodes over the node group to achieve a prescribed balance between nodes allocated to XD interactive work and Business Grid work in this situation.

j. Edit the property "BalancerInterval" with the value set to a number representing the number of minutes between balancer executions. As a starting point, use a value of 2 to 10 and adjust the value later as necessary.

k. Press “apply” and “ok” (Contine to the next step for initial configuring. Click “save” if no more editings are required during maintenance cycle).

l. Select “Node Group Member Information” in the “Additonal Properties” section to edit node group member scheduling capabilities.

m. Select each node capable of scheduling work in the node group, and click “set node BGUsage Candidacy”. Be sure this property is not active for the Gateway and DeploymentManager nodes.

n. Click “back” on the internet browser.

o. Select “Business Grid Balancer Matrix Properties” in the “Additonal Properties” section to edit the Balancer properties.

i. Select “new” to enter the matrix column information.

d. Edit the name field by entering “com.ibm.websphere.xd.bgrid.matrixColHdrs”
e. Edit the value field with a set of numbers separated by a dash (e.g. 0.0-25.0-50.0-75-100). The size of this number set determines the size of number of columns of the “Balancer Matrix”.
f. Click “apply” and “ok”

ii. Select “new” to enter the matrix row information

d. Edit the name field by entering "com.ibm.websphere.xd.bgrid.matrixRowHdrs"

e. Edit the value field with a set of numbers separated by a dash (e.g. 0.0-25.0-50.0-75-100). The size of this number set determines the size of number of rows of the "Balancer Matrix".

f. Click "apply" and "ok"

iii. Select "new" to enter information for a row in the balancer matrix.

e. Edit the name field by entering "com.ibm.websphere.xd.bgrid.matrixRowNN" where NN is a numeric value starting from 1.

f. Edit the value field with a set of action codes separated by a dash (e.g. z-z-x-x-x). The size of the set is the same as the number of columns as defined by "matrixColHdrs".

g. Possible value for action codes are:

Z = do nothing
X = give a node to XD
B = give a node to BG
O = move to overcommitted max where the OC property value is used to determine how many BG nodes should be allocated.

h. Click "apply" and "ok"

iv. Repeat the adding of the row information for as many rows as defined by "matrixRowHdrs".

p. Return to the "Business Grid Node Groups" Configuration panel and click "save" to apply changes to the master configuration.

Chapter 7 Manual Install and Configuration of a Worker Node

Perform the steps, outlined in this chapter, to manually do the installation and configuration of the Business Grid Components for WebSphere Extended Deployment worker machine. Use this option only if you do not wish to use the automatic install utility or if using it is not an option (e.g. HA implementation described in chapter 8). Note that the master node can also act as a worker node provided that the steps outlined in this section are followed on it as well.

Untar the Business Grid Components tarball to a temporary directory on the master node:

```
# mkdir /tmp/bgrid
# tar xvzf BGrid.tar.gz –C /tmp/bgrid
```

1. Install the worker node piece of the Business Grid.

a. Log on as root and issue the following command:

```
# tar xvfz /tmp/bgrid/BGrid-execmachine.tar.gz -C /opt/WebSphere/AppServer
```

2. Prepare worker machines for grid application deployment code.

On each worker machine, perform the following steps:

a. Log on as root and issue the following command:

```
# visudo
```

b. Add the following lines to the end of the file

```
loadl   ALL = NOPASSWD: /opt/wasbgrid/bgjobwrapper.sh --setup *
loadl   ALL = (mqm) NOPASSWD: /opt/wasbgrid/bgridput
```

c. Save and exit vi.

Chapter 8 Building a Highly Available Business Grid Configuration

Using any software product in a business-critical or mission-critical environment requires that you consider availability, a measure of the ability of a system to do what it is supposed to do, even in the presence of crashes, equipment failures, and environmental mishaps. As more and more critical commercial applications move onto the Internet, providing highly available services becomes increasingly important.

In this chapter outlines the steps needed to install a Highly Available configuration of Business Grid Components for WebSphere Extended Deployment. High availability implementation of the Business Grid involves HA for WebSphere MQ, IBM LoadLeveler, WebSphere ND and XD Deployment managers and the Business Grid gateway.

8.1 Contents and requirements for the sample code

To get started the test scenarios, described in this chapter, require the following hardware:

- 4 systems that support Linux with Ethernet network adapters
- 1 shared external SCSI hard drive (twin tail disk)
- 1 IBM serial null modem cable

In our setup, we used IBM eServer xSeries 335 machines with 1GB RAM. For our shared disk, we used one of these as a NFS server.

The software requirement for our setup is remains the same as the pre-requisites for Business Grid Components for WebSphere Extended Deployment with the following additions:

- Heartbeat 1.2.2

The file **hahbcode.tar.gz** contains the code samples that accompany this chapter. You can download it from the Resources section below.

8.2 High availability concepts

High availability is the system management strategy of quickly restoring essential services in the event of system, component, or application failure. The goal is minimal service interruption rather than fault tolerance. The most common solution for a failure of a system performing critical business operations is to have another system waiting to assume the failed system's workload and continue business operations.

The term "cluster" has different meanings within the computing industry. Throughout this chapter, unless noted otherwise, *cluster* describes a *heartbeat cluster*, which is a collection of nodes and resources (such as disks and networks) that cooperate to provide high availability of services running within the cluster. If one of those machines should fail, the resources required to maintain business operations are transferred to another available machine in the cluster.

The two main cluster configurations are:

- **Standby configuration:** The most basic cluster configuration, in which one node performs work while the other node acts only as standby. The standby node does not perform work and is referred to as *idle*; this configuration is sometimes called *cold standby*. Such a configuration requires a high degree of hardware redundancy. **This chapter focuses on cold standby configuration.**

- **Takeover configuration:** A more advanced configuration in which all nodes perform some kind of work, and critical work can be taken over in the event of a node failure. In a *one-sided takeover* configuration, a standby node performs some additional, non-critical, non-movable work. In a *mutual takeover* configuration, all nodes are performing highly available (movable) work. This chapter does NOT address takeover configuration.

You must plan for several key items when setting up an HA cluster:

- The disks used to store the data must be connected by a private interconnect (serial cable) or LAN to the servers that make up the cluster.
- There must be a method for automatic detection of a failed resource. This is done by a software component referred to as a *heartbeat monitor*.
- There must be automatic transfer of resource ownership to one or more surviving cluster members upon failure.

8.3 Available HA software

Much currently available software performs heartbeat monitoring and resource takeover functionality. Here is a list of available software for building high-availability clusters on various operating systems (see Resources for links):

- heartbeat (Linux)
- High Availability Cluster Multiprocessing - HACMP (AIX)
- IBM Tivoli System Automation for Multiplatforms (AIX, Linux)
- Legato AAM 5.1 (AIX, HP-UX, Solaris, Linux, Windows)
- SteelEye LifeKeeper (Linux, Windows)
- Veritas Cluster Server (AIX, HP-UX, Solaris, Linux, Windows)

This chapter describes the open source HA software heartbeat. However, you can apply the concepts you learn here to any of the above software systems.

8.4 High-Availability Linux Project and Heartbeat

The basic goal of the High-Availability Linux project is to provide a high-availability (clustering) solution for Linux which promotes reliability, availability, and serviceability (RAS) through a community development effort. The Linux-HA project is widely used, and is an important component in many interesting High-Availability solutions.

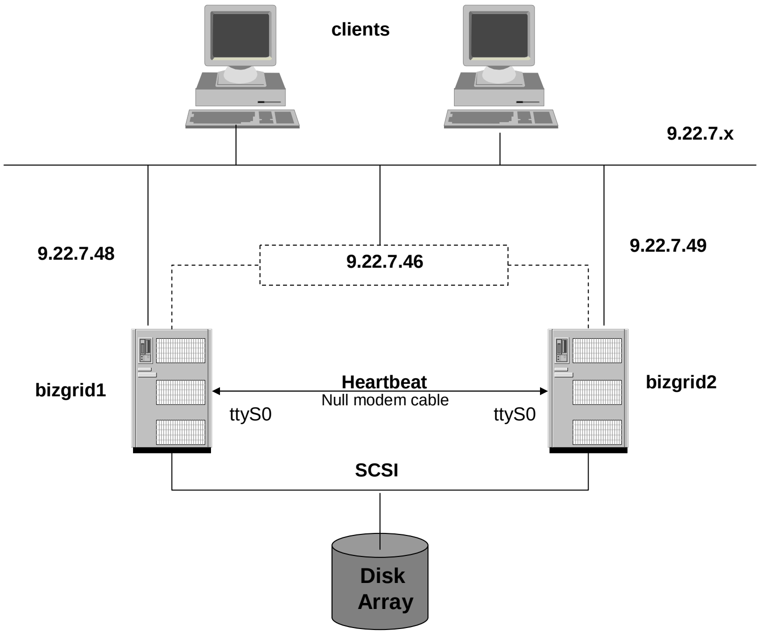
Heartbeat is one of the publicly available packages at the Linux-HA project web site. It provides the basic functions required by any HA system such as starting and stopping resources, monitoring the availability of the systems in the cluster, and transferring ownership of a shared IP address between nodes in the cluster. It monitors the health of a particular service (or services) through either a serial line or ethernet interface or both. The current version supports a 2-node configuration whereby special **heartbeat** "pings" are used to check the status and availability of a service.

For more information on **heartbeat** and projects where it's being used you can refer to the Linux-HA website (see Resources for a link).

8.5 Cluster configuration

The figure 8.1 shows a pair of clustered servers that both have access to a shared disk enclosure, containing multiple physical disks and in a cold standby mode. The application data needs to be on a shared device that both nodes can access. It can be a shared disk or a network file system. The device itself should be mirrored or have data protection to avoid data corruption. Such a configuration is frequently referred to as a "shared disk cluster", but it is actually a shared-nothing architecture as no disk is accessed by more than one node at a time.

Figure 8.1. Heartbeat cluster configuration in a production environment



For our test setup we have used NFS as the shared disk mechanism as shown in figure 8.2 although we recommend the option shown in figure 8.1 in a production environment. A null modem cable connected between the serial ports of the two systems is used to transmit heartbeats between the two nodes.

Figure 8.2. Heartbeat cluster configuration using NFS for shared file system

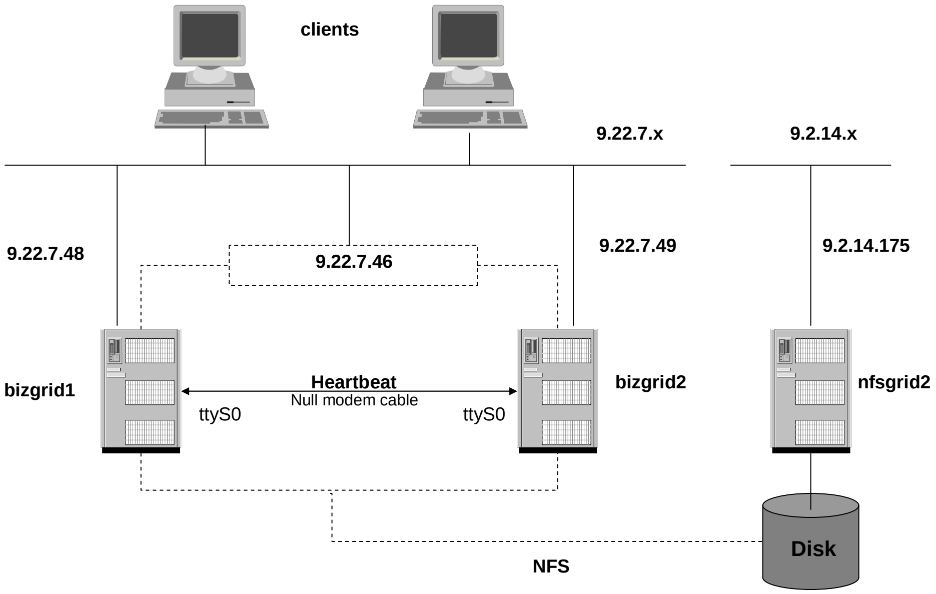


Table 8.1 shows the configuration for both nodes. The host names and IP addresses should be known to either the DNS or the /etc/hosts files on both nodes.

Table 8.1. Cluster configuration.

Role	Hostname	IP
Node1 (master)	bizgrid1.haw2.ibm.com	9.22.7.48
Node2 (master backup)	bizgrid2.haw2.ibm.com	9.22.7.49
Shared IP	bizgrid.haw2.ibm.com	9.22.7.46
NFS Server	nfsgrid.haw2.ibm.com	9.2.14.175
Node 3 (worker)	bizgrid3.haw2.ibm.com	9.23.7.50

In this chapter we will use the machine names and IP addresses, mentioned in the above table, for ease of explanation. You will have to replace those with machine names and IP addresses in your setup. Also we discuss using only one worker node (bizgrid3). You can easily repeat the setup steps for bizgrid3 for setting up multiple worker nodes.

8.6 Setting up the Serial Connection

Use a null modem cable to connect the two nodes through their serial ports. Now let’s test our serial connection as follows:

On bizgrid1 (receiver), type:

```
cat < /dev/ttyS0
```

On bizgrid2 (sender) type:

```
echo "Serial Connection test" > /dev/ttyS0
```

You should see the text on the receiver node (bizgrid1). If it works, change their roles and try again.

8.7 Setting up NFS for shared file system

As mentioned before, we will be using NFS for shared data between nodes for our test setup.

- The node **nfsgrid.haw2.ibm.com** is being used as a NFS sever.
- The file system **/ha** is being shared.

Follow the steps outlined below to get NFS up and running.

1. Create a directory **/ha** on **nfsgrid** node.

2. Edit the /etc/exports file. This file contains a list of entries; each entry indicates a volume that is shared and how it is shared. Listing 8.1 shows the relevant portion of the exports file for our setup.

Listing 8.1. exports file

```
...
/ha 9.22.7.48(rw,no_root_squash)
/ha 9.22.7.46(rw,no_root_squash)
/ha 9.22.7.49(rw,no_root_squash)
/ha 9.22.7.50(rw,no_root_squash)
...
```

3. Start the NFS services. If NFS is already running, you should run the command /usr/sbin/exportfs -ra to force nfsd to re-read the /etc/exports   file.

4. Add the file system /ha to your /etc/fstab file, on both the HA nodes - bizgrid1 and bizgrid2, the same way as local file systems. Listing 8.2 shows the relevant portion of the fstab file for our setup:

Listing 8.2. fstab file

```
...
nfsgrid.haw2.ibm.com:/ha    /ha    nfs    noauto,rw,hard 0 0
...
```

Later on, we will configure heartbeat to mount this file system.

5. Extract the code sample, hahbcode.tar.gz, on this file system using the commands shown in Listing 8.3. (First download the code sample from the Resources section)

Listing 8.3. Extract sample code

```
cd /ha
tar  xvfz  hahbcode.tar.gz
```

8.8 Download and Install Heartbeat

After downloading heartbeat (see resources for link), install it on both bizgrid1 and bizgrid2 machines, by entering the commands in listing 8.4 (in that order).

Listing 8.4. Commands for installing heartbeat

```
rpm -ivh <download_path>/heartbeat-pils-1.2.2-8.rh.el.3.0.i386.rpm
rpm -ivh <download_path>/heartbeat-stonith-1.2.2-8.rh.el.3.0.i386.rpm
rpm -ivh <download_path>/ heartbeat-1.2.2-8.rh.el.3.0.i386.rpm
```

8.9 Configure Heartbeat

You have to configure three files to get heartbeat to work: authkeys, ha.cf, and haresources. We show you the specific configuration we used for our implementation; if you require more information please refer to the heartbeat documentation (see resources for link).

1. Configure /etc/ha.d/authkeys

This file determines your authentication keys for the cluster that must be the same on both nodes. You can choose from three authentication schemes: crc, md5, or sha1. If your heartbeat runs over a secure network, such as the crossover cable in our example, you'll want to use crc. This is the cheapest method from a resources perspective. If the network is insecure, but you're either not very paranoid or concerned about minimizing CPU resources, use md5. Finally, if you want the best authentication without regard for CPU resources, use sha1. It's the hardest to crack.

The format of the file is as follows:
auth <number>
<number> <authmethod> [<authkey>]

For our test setup we chose the crc scheme. Listing 8.5 shows our /etc/ha.d/authkeys file. Make sure its permissions are safe, like 600.

Listing 8.5. authkeys file

```
auth 2
2 crc
```

2. Configure /etc/ha.d/ha.cf

This will be placed in the /etc/ha.d directory that is created after installation. It tells heartbeat what types of media paths to use and how to configure them. This file defines the nodes in the cluster and the interfaces that heartbeat uses to verify whether or not a system is up. Listing 8.6 shows the relevant portion of our /etc/ha.d/ha.cf file.

Listing 8.6. ha.cf file

```
...
```

```
#           File to write debug messages to
debugfile /var/log/ha-debug
#
#
#           File to write other messages to
#
logfile     /var/log/ha-log
#
#
#           Facility to use for syslog()/logger
#
logfacility         local0
#
#
#           keepalive: how long between heartbeats?
#
keepalive 2
#
#           deadtime: how long-to-declare-host-dead?
#
deadtime 60
#
#           warntime: how long before issuing "late heartbeat" warning?
#
warntime 10
#
#
#           Very first dead time (initdead)
#
initdead 120
#
...
#           Baud rate for serial ports...
#
baud        19200
#
#           serial      serialportname ...
serial      /dev/ttyS0
#           auto_failback:  determines whether a resource will
#           automatically fail back to its "primary" node, or remain
#           on whatever node is serving it until that node fails, or
#           an administrator intervenes.
#
auto_failback on
#
...
```

```
#
#       Tell what machines are in the cluster
#       node    nodename ...     -- must match uname -n
node    bizgrid1.haw2.ibm.com
node    bizgrid2.haw2.ibm.com
#
#       Less common options...
#
#       Treats 10.10.10.254 as a psuedo-cluster-member
#       Used together with ipfail below...
#
ping 9.22.7.1
#       Processes started and stopped with heartbeat.  Restarted unless
#               they exit with rc=100
#
respawn hacluster /usr/lib/heartbeat/ipfail
...
```

3. Configure /etc/ha.d/haresources
This file describes the resources that are managed by heartbeat. The resources are basically just start/stop scripts much like the ones used for starting and stopping resources in /etc/rc.d/init.d. Note that heartbeat will look in /etc/rc.d/init.d and /etc/ha.d/resource.d for scripts. Listing 8.7 shows our /etc/ha.d/haresources file:

Listing 8.7. haresources file

```
bizgrid1.haw2.ibm.com 9.22.7.46
Filesystem::nfsgrid.haw2.ibm.com:/ha::/ha::nfs::rw,hard httpd
```

This file must be the same on both the nodes.

This line dictates that on startup:

a. Have bizgrid1 serve the IP 9.22.7.46
b. Mount the NFS shared file system /ha
c. Start apache web server

We will be adding more resources to this file in the later sections.
On shutdown, heartbeat will:

a. Stop the apache server
b. Un-mount the shared file system
c. Give up the IP.

This assumes that the command "uname -n" spits out "bizgrid1.haw2.ibm.com" - yours may well produce "bizgrid1" and if it does, use that instead.

8.10 WebSphere MQ and HA

WebSphere MQ provides asynchronous message and queuing capabilities with assured, once-only delivery of messages. By using WebSphere MQ and heartbeat together, it is possible to further enhance the availability of WebSphere MQ queue managers.

In this section we will discuss the HA implementation for WebSphere MQ in a **cold standby** configuration using **heartbeat**. In this implementation, **heartbeat** detects that there is a problem with the primary. This could be hardware or software problem. The standby machine will:

- Take over the IP address.
- Take over the shared disks that store the queue and log files of the queue manager,
- Start the queue manager and associated processes

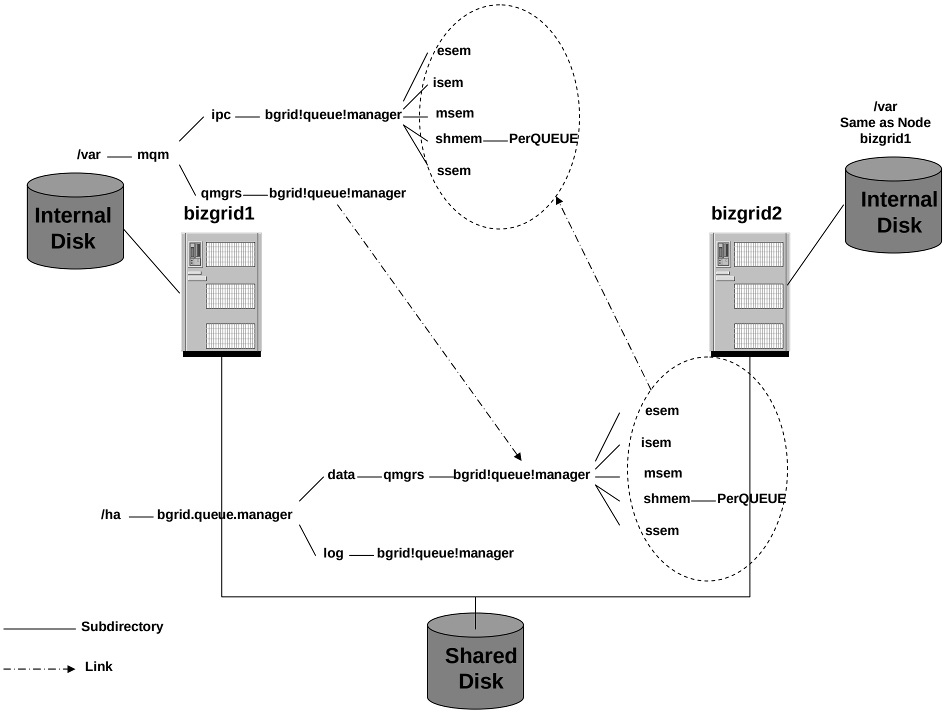
8.10.1 Implementing HA for WebSphere MQ

A queue manager that is to be used in a heartbeat cluster needs to have its logs and data on shared disks, so that they can be accessed by a surviving node in the event of a node failure. A node running a queue manager must also maintain a number of files on internal disks. These files include files that relate to all queue managers on the node, such as /var/mqm/mqs.ini, and queue manager specific files that are used to generate internal control information. Files related to a queue manager are therefore divided between internal and shared disks.

Regarding the queue manager files that are stored on shared disk it is possible to use a single shared disk for all the recovery data (logs and data) related to a queue manager. However, for optimal performance in a production environment, it is a recommended practice to place logs and data in separate filesystems such that they can be separately tuned for disk I/O.

Figure 8.3 shows the organization of the filesystem for our setup. The links shown were created automatically using shell scripts, explained in a later section.

Figure 8.3. File system organization for the queue manager - bgrid.queue.manager



In the sections to follow, we will take you through the steps of installing WebSphere MQ and creating and testing a highly available queue manager configuration.

8.10.1 Install WebSphere MQ

Follow the steps outlined in this section to install WebSphere MQ 5.3.0.2 and Fixpack 7 on both the primary master (bizgrid1) and the backup master node (bizgrid2) and the worker nodes (bizgrid3). For more information you can refer to the *WebSphere MQ for Linux for Intel and Linux for*.

1. Extract 5.3.0.2 RPMs

```
   rm -rf /tmp/mq5.3.0.2-install
   mkdir /tmp/mq5.3.0.2-install
   tar xzf C48UBML.tar.gz -C /tmp/mq5.3.0.2-install
   tar xf /tmp/mq5.3.0.2-install/MQ53Server_LinuxIntel.tar -C \
/tmp/mq5.3.0.2-install
```

Here C48UBML.tar.gz is the installation image file for WebSphere MQ. Your image filename might be different based on how you obtained the WebSphere MQ installables.

2. Set kernel level

```
   export LD_ASSUME_KERNEL=2.4.19
```

3. Replace the JRE that comes with MQ with IBM 1.4.2 JDK JRE.

```
  mv /tmp/mq5.3.0.2-install/lap/jre /tmp/mq5.3.0.2-install/lap/jre.mq

 ln –s /opt/IBMJava2-142/jre /tmp/mq5.3.0.2-install/lap/jre
```

4. Accept license

```
  /tmp/mq5.3.0.2-install/mqlicense.sh -accept -text_only
```

5. Install MQ RPMs

```
  rpm -Uvh /tmp/mq5.3.0.2-install/MQSeriesRuntime-5.3.0-2.i386.rpm
  rpm –Uvh /tmp/mq5.3.0.2-install/MQSeriesSDK-5.3.0-2.i386.rpm
  rpm -Uvh /tmp/mq5.3.0.2-install/MQSeriesServer-5.3.0-2.i386.rpm
  rpm –Uvh /tmp/mq5.3.0.2-install/MQSeriesClient-5.3.0-2.i386.rpm
  rpm -Uvh /tmp/mq5.3.0.2-install/MQSeriesSamples-5.3.0-2.i386.rpm
  rpm -Uvh /tmp/mq5.3.0.2-install/MQSeriesJava-5.3.0-2.i386.rpm
  rpm -Uvh /tmp/mq5.3.0.2-install/MQSeriesMan-5.3.0-2.i386.rpm
```

6. Cleanup

```
    rm –rf /tmp/mq5.3.0.2-install/
```

7. Extract fixpack 7 RPMs

```
     rm –rf /tmp/mq5.3.0.7-install/
  mkdir /tmp/mq5.3.0.7-install/
  tar xzf U496732.nogskit.tar.gz -C /tmp/mq5.3.0.7-install/
```

Here U496732.nogskit.tar.gz is the installation image file for WebSphere MQ Fix Pack 7. Your image filename might be different based on how you obtained the WebSphere MQ Fix Pack installables.

8. Install fixpack 7 RPMs

```
rpm –Uvh /tmp/mq5.3.0.7-install/MQSeriesRuntime-U496732-5.3.0-7.i386.rpm
rpm -Uvh /tmp/mq5.3.0.7-install/MQSeriesSDK-U496732-5.3.0-7.i386.rpm
rpm –Uvh /tmp/mq5.3.0.7-install/MQSeriesServer-U496732-5.3.0-7.i386.rpm
rpm -Uvh /tmp/mq5.3.0.7-install/MQSeriesClient-U496732-5.3.0-7.i386.rpm
rpm -Uvh /tmp/mq5.3.0.7-install/MQSeriesSamples-U496732-5.3.0-7.i386.rpm
rpm -Uvh /tmp/mq5.3.0.7-install/MQSeriesJava-U496732-5.3.0-7.i386.rpm
rpm -Uvh /tmp/mq5.3.0.7-install/MQSeriesMan-U496732-5.3.0-7.i386.rpm
```

9. Cleanup again

```
   rm –rf /tmp/mq5.3.0.7-install/
```

8.10.2 Create a Highly Available MQ Manager and Queues

On some platforms creation of a highly available queue manager is automated by scripts in WebSphere MQ HA Support Packs such as MC63 and IC61. However these support packs are not available for Linux.

The scripts used in this section are modified versions of the scripts in the MC63 support pack and have the following limitations:

- One file system only for log and data (/ha).
- One queue manager at a time.

Follow the steps outlined below to create a highly available queue manager, **bgrid.queue.manager**:

1. Create the following directories on the shared disk (/ha):
    a. /ha/bgrid.queue.manager
    b. /ha/bgrid.queue.manager/data
    c. /ha/bgrid.queue.manager/log

2. On the primary node (bizgrid1) create a highly available queue manager using the command shown below (as root) :

```
/ha/hacode/mq/hascripts/hacrtmqm bgrid.queue.manager
```

The **hacrtmqm** command will create the queue manager and will ensure that its directories are arranged to allow for HA operation. The source code for the **hacrtmqm** script is included with the sample code (see resources for a link)-.

3. Add the following line to your **.bashrc** file on both the nodes for the **mqm** user.

```
   LD_ASSUME_KERNEL=2.4.19
   export LD_ASSUME_KERNEL
```

4. You need to run the setmqcap command, inputting the number of processors that you have paid for. The command to run on **bizgrid1** for our setup is shown below:

```
/opt/mqm/bin/setmqcap 4
```

5. Login as user mqm.

```
su - mqm
```

6. Start the queue manager **bgrid.queue.manager**, using the **strmqm** command as user mqm.

```
/opt/mqm/bin/strmqm bgrid.queue.manager
```

7. Enable MQSC commands by typing:

```
/opt/mqm/bin/runmqsc bgrid.queue.manager
```

A message tells you that an MQSC session has started. MQSC has no command prompt.

8. Create a local queue **BGRID.QUEUE** by entering the following command:

```
define qlocal (BGRID.QUEUE)
```

9. Create a local queue **BGRIDRSP.QUEUE** by entering the following command:

```
define qlocal (BGRIDRSP.QUEUE)
```

10. Create a local queue **ENDPT.REPLYQ** by entering the following command:

```
define qlocal (ENDPT.REPLYQ)
```

11. Create a channel **BGRID.CHANNEL** by entering the following command:

```
define channel(BGRID.CHANNEL) chltype(svrconn) trptype(tcp) mcauser('mqm')
```

12. Create a channel **ENDPOINT.TO.BGRID** by entering the following command:

```
define channel(ENDPOINT.TO.BGRID) chltype(rcvr) trptype(tcp) mcauser('mqm')
```

13. Stop MQSC by typing: **end**. Some messages are displayed, and the command prompt is displayed again.

14. Stop the queue manager **bgrid.queue.manager** manually, using the **endmqm** command.

```
/opt/mqm/bin/endmqm bgrid.queue.manager
```

15. On the backup node (bizgrid2) create the queue manager using the command show below on one line as user **mqm** (you may have to mount /ha as root):

```
cd /ha/hacode/mq/hascripts/

./halinkmqm bgrid.queue.manager bgrid\!queue\!manager /ha/bgrid.queue.manager/data
standby
```

Internally, hacrtmqm uses a script called halinkmqm to relink the subdirectories used for IPC keys and create a symlink from /var/mqm/qmgrs/$qmgr to the /ha/$qmgr/data/qmgrs/$qmgr directory. Do not run halinkmqm on the node on which you created the queue manager with hacrtmqm - it has already been run there. The source code for the halinkmqm script is included with the sample code (see resources for a link).

16. You need to run the setmqcap command, inputting the number of processors that you have paid for. The command to run on **bizgrid2** for our setup is shown below:

```
/opt/mqm/bin/setmqcap 4
```

17. Start the queue manager **bgrid.queue.manager**, using the **strmqm** command, on the backup node. Make sure it starts.

18. Stop the queue manager **bgrid.queue.manager** on the backup node.

8.10.3 Configure heartbeat to manage MQ Server

The steps needed to configure heartbeat to manage the MQ server are outlined below:

1. As mentioned before, resources that are managed by heartbeat are basically just start/stop scripts. So let's create a script to start and stop the MQ queue manager and any associated processes. A very basic script is shown in listing 8.8. You can further customize it to suit your setup. This script has to be placed in the /etc/rc.d/init.d directory.

Listing 8.8. mqseries script

```
#!/bin/bash
#
```

```
#            /etc/rc.d/init.d/mqseries
#
# Starts the MQ Server
#
# description: Runs  MQ

. /etc/init.d/functions
# Source function library.

PATH=/usr/bin:/bin:/opt/mqm/bin
QMGRS=" bgrid.queue.manager"
PS=/bin/ps
GREP=/bin/grep
SED=/bin/sed
#==============================================================
========
SU="sh"
if [ "`whoami`" = "root" ]; then
   SU="su - mqm"
fi
#==============================================================
========
killproc() {            # kill the named process(es)
      pid=`$PS -e |
          $GREP -w $1 |
          $SED -e 's/^  *//' -e 's/ .*//'`
      [ "$pid" != "" ] && kill -9 $pid
}
#==============================================================
========
start() {
   for qmgr in $QMGRS ; do
      export qmgr
            echo "$0: starting $qmgr"
            $SU -c "strmqm $qmgr"
            $SU -c "nohup runmqlsr -m $qmgr -t tcp -p 1414 >> /dev/null 2>&1 <
/dev/null &"
   done
}
#==============================================================
========
stop() {
   for qmgr in $QMGRS ; do
      export qmgr
      echo ending $qmgr
            killproc runmqlsr
```

```
        $SU -c "endmqm -i $qmgr &"
    sleep 30
  done
}


case $1 in
'start')
  start
  ;;
'stop')
  stop
  ;;
'restart')
  stop
  start
  ;;
*)
  echo "usage: $0 {start|stop|restart}"
  ;;
esac
```

2. Now let’s configure the /etc/ha.d/haresources file (on both the nodes) to include the above mqseries script. The modified file is shown below:

```
bizgrid1.haw2.ibm.com 9.22.7.46
Filesystem::nfsgrid.haw2.ibm.com:/ha::/ha::nfs::rw,hard httpd
mqseries
```

So, this line dictates that on startup of **heartbeat**, have **bizgrid1** serves the cluster IP address, mount the shared file system /ha and start apache and mqseries as well. On shutdown, **heartbeat** will first stop mqseries, and then apache, then un-mount the file system, and finally gives up the IP.

8.10.4 Testing HA for MQ

This section outlines the steps needed to test the high availability of the queue manager, **bgrid.queue.manager**.

1. **Start the heartbeat service** on the primary and then on the backup node. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

If it fails, look in **/var/log/messages** to determine the reason and then correct it. After heartbeat starts successfully, you should see a new interface with the IP address that you configured in the **ha.cf** file. You can display if by running the ifconfig command. Shown below is the relevant portion of the output for our setup:

```
...
eth0:0    Link encap:Ethernet  HWaddr 00:D0:59:DA:01:50
          inet addr:9.22.7.46  Bcast:9.22.7.127  Mask:255.255.255.128
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:76541 errors:0 dropped:0 overruns:0 frame:0
          TX packets:61411 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8830515 (8.4 Mb)  TX bytes:6755709 (6.4 Mb)
          Interrupt:11 Base address:0x6400 Memory:c0200000-c0200038
...
```

Once you've started heartbeat, take a peek at your log file (default is /var/log/ha-log) before testing it. If all is peachy, the primary machine’s log (bizgrid1 in our example) should look something like shown below:

```
...
heartbeat: 2004/09/01_11:17:13 info: **************************
heartbeat: 2004/09/01_11:17:13 info: Configuration validated. Starting heartbeat 1.2.2
heartbeat: 2004/09/01_11:17:13 info: heartbeat: version 1.2.2
heartbeat: 2004/09/01_11:17:13 info: Heartbeat generation: 10
heartbeat: 2004/09/01_11:17:13 info: Starting serial heartbeat on tty /dev/ttyS0 (19200
baud)
heartbeat: 2004/09/01_11:17:13 info: ping heartbeat started.
heartbeat: 2004/09/01_11:17:13 info: pid 9226 locked in memory.
heartbeat: 2004/09/01_11:17:13 info: Local status now set to: 'up'
heartbeat: 2004/09/01_11:17:14 info: pid 9229 locked in memory.
heartbeat: 2004/09/01_11:17:14 info: pid 9230 locked in memory.
heartbeat: 2004/09/01_11:17:14 info: pid 9231 locked in memory.
heartbeat: 2004/09/01_11:17:14 info: pid 9232 locked in memory.
heartbeat: 2004/09/01_11:17:14 info: pid 9233 locked in memory.
heartbeat: 2004/09/01_11:17:14 info: Link 9.22.7.1:9.22.7.1 up.
heartbeat: 2004/09/01_11:17:14 info: Status update for node 9.22.7.1: status ping
...
heartbeat: 2004/09/01_11:19:18 info: Acquiring resource group: bizgrid1.haw2.ibm.com
9.22.7.46 httpd mqseries
heartbeat: 2004/09/01_11:19:18 info: Running /etc/ha.d/resource.d/IPaddr 9.22.7.46
start
heartbeat: 2004/09/01_11:19:18 info: /sbin/ifconfig eth0:0 9.22.7.46 netmask
255.255.255.128    broadcast 9.22.7.127
```

```
heartbeat: 2004/09/01_11:19:18 info: Sending Gratuitous Arp for 9.22.7.46 on eth0:0
[eth0]
...
heartbeat: 2004/09/01_11:19:49 info: No pkts missing from bizgrid2.haw2.ibm.com!
...
```

You can see that it is doing the IP take over and then starting WebSphere MQ processes. Use the **ps** command to make sure MQ is running on the primary node.

2. **Put a few persistent messages on the BGQUEUE**. This can be done by running the MQ Sender program, send.bat or send.sh (based on your OS). This program should be run from a machine that has the MQ Client installed. Shown below is the output of the run on the node **bizgrid1**:

```
[root@bizgrid1 mq]# ./send.sh
MSender is running
Hostname  = bizgrid.haw2.ibm.com
QManager = bgrid.queue.manager
Channel Name  = BGRID.CHANNEL
Channel Port  = 1414
Q  = BGRID.QUEUE

Enter a message:
Hello
This
is
a
test

Done Sending Message
[root@bizgrid1 mq]#
```

3. **Browse and get the messages.** Use the MQ Browse program receive.bat or receive.sh (based on your OS). We will fetch all the messages put before except the last one (test). We will get the last message after failover has happened. Shown below is the output of the run on the node **bizgrid1**:

```
[root@bizgrid1 mq]# ./receive.sh
MBrowse is running
Hostname  = bizgrid.haw2.ibm.com
QManager = bgrid.queue.manager
Channel Name  = BGRID.CHANNEL
Channel Port  = 1414
Q  = BGRID.QUEUE
```

```
Browsed message: Hello
Actually get message?y
Actually getting the message
Browsed message: This
Actually get message?y
Actually getting the message
Browsed message: is
Actually get message?y
Actually getting the message
Browsed message: a
Actually get message?y
Actually getting the message
Browsed message: test
Actually get message?n
MQJE001: Completion Code 2, Reason 2033
MQ exception: CC = 2 RC = 2033
```

Ignore the MQ Exception, with reason code 2033, at the end. It occurs because there are no more messages to get from the queue.

4. **Simulating failover.** This can be done simply stopping heartbeat on the primary system using the command shown below:

```
/etc/rc.d/init.d/heartbeat stop
```

You should see all the services come up on the second machine in under a minute. If you do not, look in **/var/log/messages** to determine the problem and correct it. You can fail back over to the primary by starting heartbeat again. **Heartbeat** will always give preference to the primary system and will start to run there if possible. Make sure WebSphere MQ is running by checking the **/var/log/ha-log** file and the **ps** command on the backup machine.

5. **Browse and get the last message.** Run the MQ Browse program receive.bat or receive.sh (based on your OS). We will get the last message this time. Shown below is the output of the run on the node **bizgrid2**:

```
[root@bizgrid2 mq]# ./receive.sh
MBrowse is running
Hostname  = bizgrid.haw2.ibm.com
QManager = bgrid.queue.manager
Channel Name  = BGRID.CHANNEL
Channel Port  = 1414
Q  = BGRID.QUEUE
Browsed message: test
Actually get message?y
Actually getting the message
MQJE001: Completion Code 2, Reason 2033
MQ exception: CC = 2 RC = 2033
```

6. **Start the heartbeat service back on the primary.** This should stop the apache and WebSphere MQ processes on the secondary and start them on the primary. The primary should also take over the cluster IP.

Thus we have shown how using shared disk the messages put on a queue before the failover can be recovered after it. In the next section we will discuss the HA implementation of IBM LoadLeveler scheduler.

8.11 Implementing LoadLeveler for High Availability

Each machine in the LoadLeveler cluster performs one or more roles in scheduling jobs. These roles and their failure implications are described below:

1. **Scheduling Machine:** A job submission results in it getting placed in a queue managed by a scheduling machine. The scheduling machine asks the central manager (this role is described below) to find a machine that can run the job, and also keeps persistent information about the job on the disk.

   The job information on one scheduling machine is not normally shared or accessed by other scheduling machines in the LoadLeveler cluster. The scheduling machines operate independently and in the event of a scheduling machine failure, the job information that resides on the failed scheduling machine will be temporarily unavailable (but not lost). Jobs waiting to be scheduled will not be considered for execution during this time. It is of critical importance to have the scheduling machine re-established as quickly as possible. In our example of high availability configuration for a scheduling machine, the necessary local files and directories are placed on external shared disk storage, which makes them available to a backup scheduling machine in the event of scheduler node failure.

2. **Central Manager Machine:** The role of the Central Manager is to find one or more machines in the LoadLeveler cluster, based on the job's requirements, which will run the job. Upon finding the machine(s), it notifies the scheduling machine.

   The central manager is the central point of control for LoadLeveler. LoadLeveler provides the capability to define an alternate central manager and have it take over the role of the primary central manager in case of failure. We will discuss this in detail in a later section.

   In the event of a central manager failure without an alternate central manager defined, jobs which have started run to completion without loss of information but their status are not reportable to the users. New jobs will be accepted by scheduling nodes and later forwarded to the central manager when it returns, or when an alternate takes over.

3. **Executing Machine:** The machine that runs the job is known as the executing machine.

   When an execution node fails the jobs running on the node fail and require restart when the node is restored. Jobs will start either from the beginning or from the last checkpoint. Job check-pointing can be selected as an option or coded in the application. The establishment of a backup execution node would provide immediately the capability to restart the job in a more timely fashion. With or without a backup node, jobs and their job information and checkpoints are not lost in the event of node failure, as long as their disks are still accessible using the appropriate cabling and disk techniques.

   Both the scheduling node and the execution nodes maintain persistent information about the state of their jobs. The scheduling node and the execution node use a protocol that ensures that the state information is kept on disk by at least one of them. This ensures the state can be recovered from either the scheduling node or the execution node in event of failure. Neither the scheduling node nor the execution node discard the job information until it is passed onto and accepted by the other node. In our configuration, the job information is stored on a shared disk.

4. **Submitting Machine:** These machines are used to submit, query and cancel jobs and have no persistent data about a job. Lack of any persistent data makes high availability non critical for these machines.

The role a client machine plays depends on which LoadLeveler daemons are configured on it. As a whole, the cluster provides the capability to build, submit, execute, and manage serial and parallel batch jobs.

In the sections to follow, we will discuss some of the built-in capabilities of LoadLeveler for high availability and how the overall system availability can be further enhanced by using **heartbeat**.

8.11.1 Install IBM LoadLeveler

This section outlines the steps we took to install IBM Loadleveler for Linux 3.2 on all our machines i.e. **bizgrid1**, **bizgrid2** and **bizgrid3**. These steps are based on the installation steps outlined in the chapter 4 of LoadLeveler Installation Memo (see Resource for link).

1. **Directories for LoadLeveler:** In our setup we used the following:
   Local directory: /var/loadl
   Home directory: /home/loadl
   Release directory: /opt/ibmll/LoadL/full
   Name of central manager machine: **bizgrid**

2. **Log in as root**

3. **Create the loadl group name:** We used the command shown below to create a group loadl with group id of 1000 on all the nodes:

```
groupadd –g 1000 loadl
```

4. **Create the loadl user ID:** We used the command shown below to create a userid loadl on all the nodes:

```
useradd –c loadleveler_user -d /home/loadl -s /bin/bash \
      -u 1000 –g 1000 -m loadl
```

5. **Install the LoadLeveler RPMs**
   a. Install license RPM (where LLIMAGEDIR contains the installation RPMS)

   ```
   cd $LLIMAGEDIR
   rpm -Uvh LoadL-full-license-3.2.0.0-0.i386.rpm
   ```

   b. Install the other RPM

   ```
   cd /opt/ibmll/LoadL/sbin
   ./install_ll -y -d "$LLIMAGEDIR"
   ```

6. **Run the initialization script llinit**

   a. Create the local directory:

   ```
   mkdir /var/loadl
   ```

   b. Set ownership:

   ```
   chown loadl.loadl /var/loadl
   ```

   c. Switch to the loadl ID

   ```
   su - loadl
   ```

d. Change the current directory to the bin subdirectory in the release directory by entering:

```
cd /opt/ibmll/LoadL/full/bin
```

e. Ensure that you have write privileges in the LoadLeveler home, local, and /tmp directories.

f. Enter the llinit command as shown below for our setup:

```
./llinit -local /var/loadl -release /opt/ibmll/LoadL/full -cm bizgrid
```

8.11.2 Create a Highly Available LL Scheduling Machine configuration

In our setup

- The machine **bizgrid1** will act as the primary scheduling machine
- The machine **bizgrid 2** will act as a standby scheduling machine
- The machine **bizgrid 3** will be used as a job execution machine.

In a high availability configuration, the necessary local files and directories for **bizgrid 1** are placed on external shared disk storage, which makes them available to **bizgrid 2** in the event of scheduler node failure. Shown below are the steps we followed to set this up:

1. **Log in as root**

2. Create the following directories on the shared disk (/ha):

   a. /ha/loadl/execute
   b. /ha/loadl/spool

3. Set ownership by running the following command on the node **bizgrid1**:

```
chown loadl.loadl /ha/loadl/
chown loadl.loadl /ha/loadl/execute
chown loadl.loadl /ha/loadl/spool
```

4. Switch to the loadl ID on the nodes **bizgrid1**and **bizgrid2**:

```
su - loadl
```

5. Set appropriate permissions on the shared directories using commands shown below on the node **bizgrid1**:

```
chmod 0777 /ha/loadl/execute
chmod 775 /ha/loadl/spool
```

6. On both the primary and backup scheduling machines delete the **execute** and **spool** directories under /var/loadl.

```
rm -rf /var/loadl/execute
rm –rf /var/loadl/spool
```

7. Create symbolic links to the shared directories using the following commands on the nodes **bizgrid1** and **bizgrid2**:

```
ln -s /ha/loadl/execute /var/loadl/execute
ln -s /ha/loadl/spool /var/loadl/spool
```

8. Edit the machines stanza. Shown below are the relevant portions of the **LoadL_admin** (under /home/loadl) file on the different machines:
   a. **bizgrid1** acts as a primary scheduling machine and the central manager. In a production environment our recommendation would be to put the central manager and scheduling daemon on separate machines.

```
...
bizgrid1:   type = machine
            central_manager = true
            schedd_host = true
bizgrid3:   type = machine
...
```

   b. **bizgrid2** acts as a backup scheduling machine and central manager

```
...
bizgrid2:   type = machine
            central_manager = true
            schedd_host = true
bizgrid3:   type = machine
...
```

   c. **bizgrid3** acts as an execution machine.

```
...
bizgrid:    type = machine
            central_manager = true
            schedd_host = true
bizgrid3:   type = machine
```

```
...
```

9. Change the RUNS_HERE flags in **LoadL_config** (under /home/loadl) file on the different machines:
   a. **bizgrid1 and bizgrid2**

   ```
   ...
   SCHEDD_RUNS_HERE      =     True
   STARTD_RUNS_HERE      =     False
   ...
   ```

   This will ensure that a job doesn't get scheduled to be executed on the scheduling machines. We want **bizgrid1** and **bizgrid2** to act as scheduling machines only.

   b. **bizgrid3**

   ```
   ...
   SCHEDD_RUNS_HERE      =     False
   STARTD_RUNS_HERE      =     True
   ...
   ```

10. Edit the /etc/hosts files on the three nodes as follows:
    a. **bizgrid1**

    ```
    ...
    9.22.7.46    bizgrid.haw2.ibm.com bizgrid
    9.22.7.46    bizgrid2.haw2.ibm.com bizgrid2
    ...
    ```

    b. **bizgrid2**

    ```
    ...
    9.22.7.46    bizgrid.haw2.ibm.com bizgrid
    9.22.7.46    bizgrid1.haw2.ibm.com bizgrid1
    ...
    ```

    c. **bizgrid3**

    ```
    ...
    9.22.7.46    bizgrid.haw2.ibm.com bizgrid
    9.22.7.46    bizgrid1.haw2.ibm.com bizgrid1
    9.22.7.46    bizgrid2.haw2.ibm.com bizgrid2
    ...
    ```

8.11.3 Configure heartbeat to manage LL Scheduling Machine

The steps needed to configure heartbeat to manage the loadleveler are outlined below:

1. Let's create a script to start and stop the loadleveler processes. A very basic script is shown in listing 8.9. You can further customize it to suit your setup. This script has to be placed in the /etc/rc.d/init.d directory.

Listing 8.9. loadl script

```
#!/bin/bash
#
#		/etc/rc.d/init.d/loadl
#
# Starts the loadleveler processes
#
# chkconfig: 345 89 56
# description: Runs  loadleveler

. /etc/init.d/functions
# Source function library.

PATH=/usr/bin:/bin:/opt/ibmll/LoadL/full/bin
#==============================================================
========
SU="sh"
if [ "`whoami`" = "root" ]; then
    SU="su - loadl"
fi
#==============================================================
========
start() {
		echo "$0: starting loadleveler"
		$SU -c "llctl start"
}
#==============================================================
========
stop() {
		echo "$0: Stoping loadleveler"
		$SU -c "llctl stop"
}


case $1 in
'start')
    start
    ;;
'stop')
    stop
    ;;
'restart')
    stop
    start
    ;;
```

```
*)
   echo "usage: $0 {start|stop|restart}"
   ;;
esac
```

2. Now let’s configure the /etc/ha.d/haresources file (on both the scheduling machine nodes) to include the above loadl script. The relevant portion of the modified file is shown below:

```
bizgrid1.haw2.ibm.com 9.22.7.46
Filesystem::nfsgrid.haw2.ibm.com:/ha::/ha::nfs::rw,hard httpd
mqseries loadl
```

So, this line dictates that on startup of **heartbeat**, have **bizgrid1** serves the cluster IP address, mount the shared file system, and start apache, WebSphere MQ and IBM LoadLeveler as well. On shutdown, **heartbeat** will first stop LoadLeveler, and then WebSphere MQ, and then apache, and then un-mount the shared file system, and finally give up the IP.

8.11.4 Testing LL Scheduling Machine Failover

This section outlines the steps needed to test the high availability of the scheduling daemon.

1. **Start the heartbeat service** on the primary and then on the backup node. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

After heartbeat starts successfully, you should see a new interface with the IP address that you configured in the **ha.cf** file. Once you've started heartbeat, take a peek at your log file (default is /var/log/ha-log) on the primary and make sure that it is doing the IP take over and then starting apache, WebSphere MQ processes and LoadLeveler processes. Use the **ps** command to make sure LoadLeveler daemons are running on the primary node. Heartbeat will not start any of the above processes on the backup. This happens only after the primary fails.

2. **Start the loadleveler daemons on bizgrid3,** as user **loadl**, using the following command:

```
llctl start
```

3. **Make bizgrid3 unavailable** as an executing machine by draining the startd daemon. This is needed to give us enough time, after submitting a job, to test failover. Use the following command as user **loadl** on node **bizgrid3**:

```
llctl drain startd
```

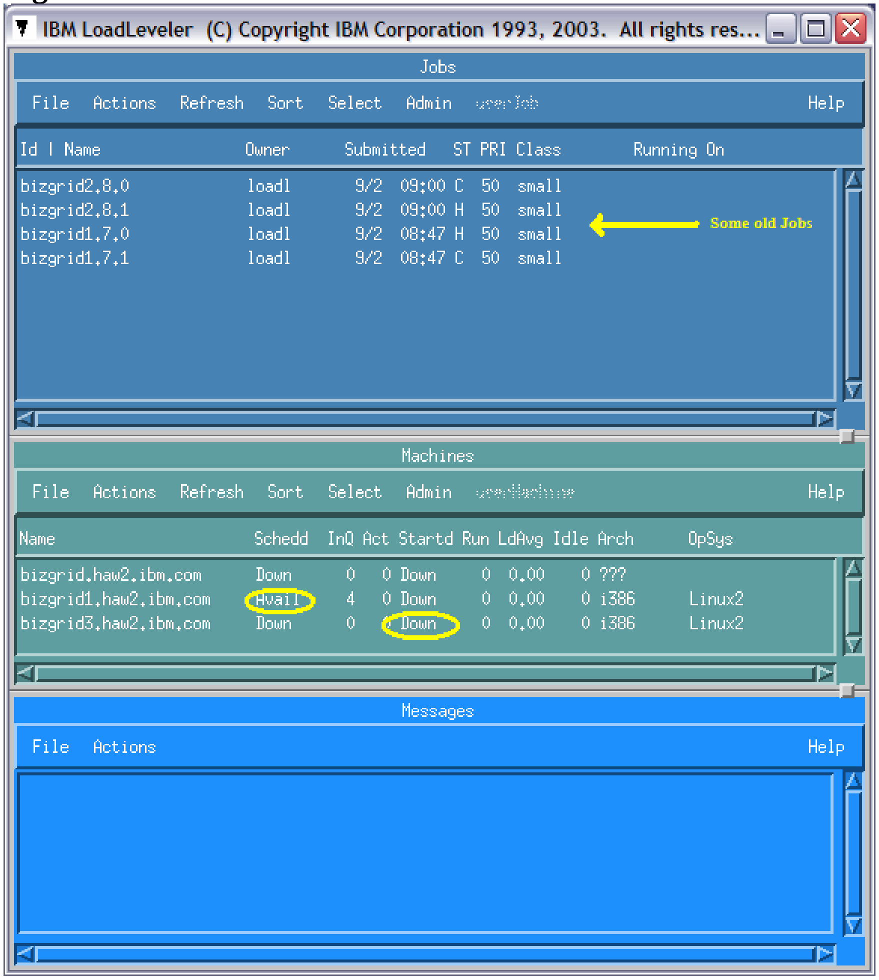
4. Start the GUI for managing the loadleveler cluster using the command below on the **bizgrid3** machine:

```
xloadl &
```

This should bring up the window shown in figure 8.4. In this figure you can see

a. Some old incomplete jobs (if any) in the Jobs pane
b. Schedd is available on **bizgrid1** machine
c. Startd is down on **bizgrid3** machine because of step 3 above.

Figure 8.4. Loadleveler GUI


IBM LoadLeveler (C) Copyright IBM Corporation 1993, 2003. All rights res...
Jobs
File Actions Refresh Sort Select Admin userJob Help
Some old Jobs
Machines
File Actions Refresh Sort Select Admin userMachine Help
Messages
File Actions Help

```
Id | Name                   Owner      Submitted   ST PRI Class        Running On
bizgrid2.8.0                loadl        9/2  09:00 C  50  small
bizgrid2.8.1                loadl        9/2  09:00 H  50  small
bizgrid1.7.0                loadl        9/2  08:47 H  50  small
bizgrid1.7.1                loadl        9/2  08:47 C  50  small
```

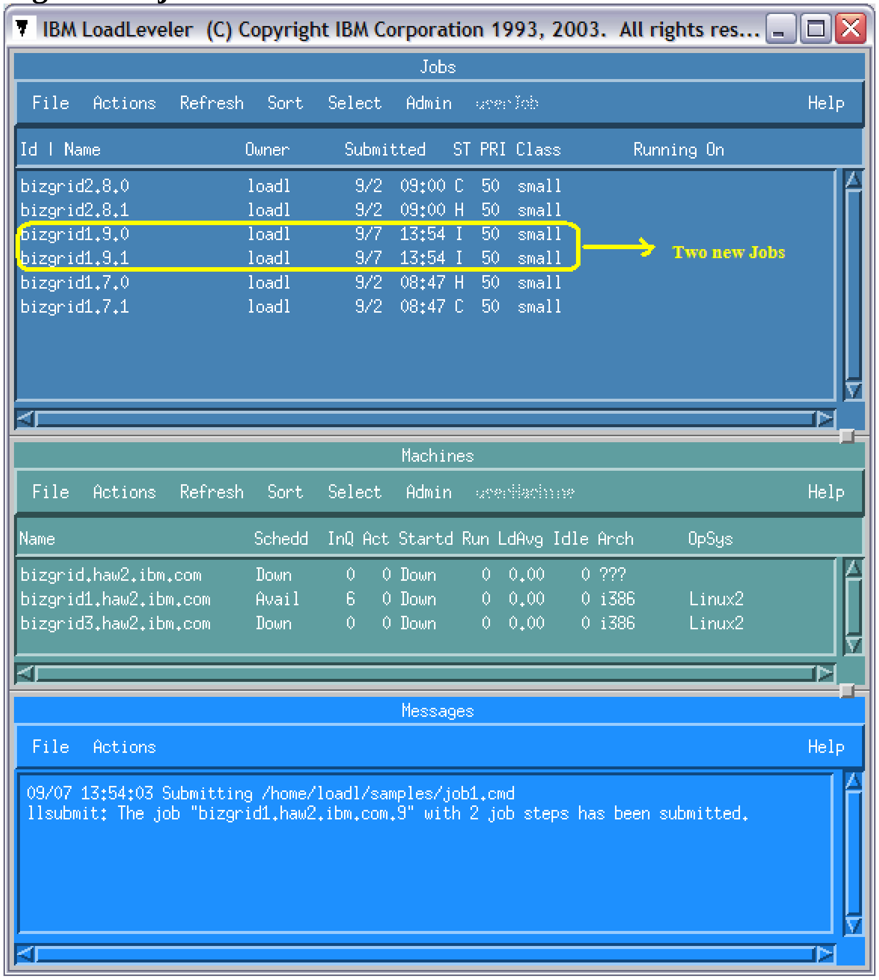
```
Name                      Schedd  InQ Act Startd Run LdAvg Idle Arch      OpSys
bizgrid.haw2.ibm.com       Down     0   0 Down     0  0.00    0 ???
bizgrid1.haw2.ibm.com      Avail    4   0 Down     0  0.00    0 i386      Linux2
bizgrid3.haw2.ibm.com      Down     0   0 Down     0  0.00    0 i386      Linux2
```

5. **Set ownership of the samples directory**. Use the command shown below on the node **bizgrid3** (where the job will be run):

```
chown loadl.loadl /opt/ibmll/LoadL/full/samples
```

6. **Submit a job.** This can be done by clicking File -> Submit a Job. This should bring up the Submit a Job dialog. Select job1.cmd from the /home/loadl/samples directory and click OK. The Messages windows should show that the job has been submitted. Click cancel on the Submit a Job dialog box. Two new jobs will get created (job1.cmd is a two step command). After a while these jobs will go to an idle state (I) for the lack of availability of an executing machine. This is shown in figure 8.5. Now let's fail the primary scheduling machine.

Figure 8.5. job1.cmd submitted



```
Id | Name                 Owner      Submitted    ST PRI Class        Running On
bizgrid2.8.0              loadl         9/2  09:00 C  50  small
bizgrid2.8.1              loadl         9/2  09:00 H  50  small
bizgrid1.9.0              loadl         9/7  13:54 I  50  small
bizgrid1.9.1              loadl         9/7  13:54 I  50  small
bizgrid1.7.0              loadl         9/2  08:47 H  50  small
bizgrid1.7.1              loadl         9/2  08:47 C  50  small
```

```
Name                       Schedd  InQ Act Startd Run LdAvg Idle Arch      OpSys
bizgrid.haw2.ibm.com       Down      0   0 Down     0  0.00    0 ???
bizgrid1.haw2.ibm.com      Avail     6   0 Down     0  0.00    0 i386      Linux2
bizgrid3.haw2.ibm.com      Down      0   0 Down     0  0.00    0 i386      Linux2
```

```
09/07 13:54:03 Submitting /home/loadl/samples/job1.cmd
llsubmit: The job "bizgrid1.haw2.ibm.com.9" with 2 job steps has been submitted.
```

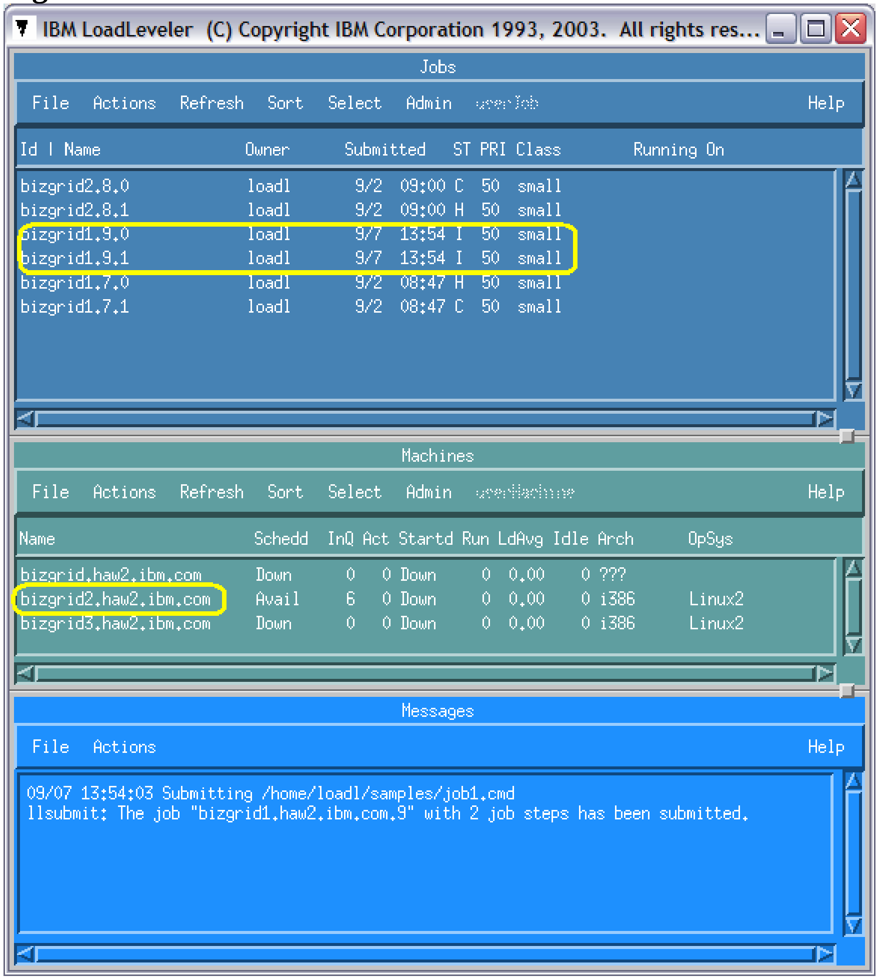
7. **Simulating failover.** This can be done simply stopping heartbeat on the primary system using the command shown below:

```
/etc/rc.d/init.d/heartbeat stop
```

You should see all the services come up on the backup machine in under a minute. You can verify that loadleveler is running on the backup by checking the **/var/log/ha-log** file and using the **ps** command on the backup machine. Once the backup has taken over control refresh the loadleveler GUI. Figure 8.6 shows a refreshed view. In this figure you can see

a. All the old incomplete jobs, including the two corresponding to the job submitted in step 6 above, in the Jobs pane. This means the jobs have survived a machine failover.
b. Schedd is now available on **bizgrid2** machine
c. Startd is still down on **bizgrid3** machine.

Figure 8.6. LL GUI after failover.


IBM LoadLeveler (C) Copyright IBM Corporation 1993, 2003. All rights res...
Jobs
File
Actions
Refresh
Sort
Select
Admin
Help

Id I Name	Owner	Submitted	ST	PRI	Class	Running On
bizgrid2.8.0	loadl	9/2 09:00	C	50	small	
bizgrid2.8.1	loadl	9/2 09:00	H	50	small	
bizgrid1.9.0	loadl	9/7 13:54	I	50	small	
bizgrid1.9.1	loadl	9/7 13:54	I	50	small	
bizgrid1.7.0	loadl	9/2 08:47	H	50	small	
bizgrid1.7.1	loadl	9/2 08:47	C	50	small	

Name	Schedd	InQ	Act	Startd	Run	LdAvg	Idle	Arch	OpSys
bizgrid.haw2.ibm.com	Down	0	0	Down	0	0.00	0	???	
bizgrid2.haw2.ibm.com	Avail	6	0	Down	0	0.00	0	i386	Linux2
bizgrid3.haw2.ibm.com	Down	0	0	Down	0	0.00	0	i386	Linux2

```
09/07 13:54:03 Submitting /home/loadl/samples/job1.cmd
llsubmit: The job "bizgrid1.haw2.ibm.com.9" with 2 job steps has been submitted.
```

8. **Resume the startd daemon on bizgrid3 machine.** This is done by executing the following command on **bizgrid3**:

```
llctl resume startd
```

Now the bizgrid3 machine is available for executing jobs. The two jobs should now go to a running state and finally should complete on the bizgrid3 machine. You can verify job completing by taking a peek at the .out files, created by the two steps of the job, in the /home/loadl/samples directory. For this run you should see two files, **job1.bizgrid3.9.0.out** and **job1.bizgrid3.9.1.out.** Thus we have shown how using shared disk the jobs submitted before the failure of the primary scheduling node can be recovered after the backup has taken over control.

9. **Start the heartbeat service back on the primary**. This should stop the loadleveler processes on the secondary and start them on the primary. The primary should also take over the cluster IP as well. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

8.11.5 HA for LL Central Manager

If you decide to keep the Central Manager on a separate node (**bizgrid4**) as the scheduling machine then you can leverage on the built-in high availability features of the Central Manager. To try out this configuration, define **bizgrid4** machine as the central manager in the configuration files of all machines.

Problems with network communication or software or hardware failures can cause the central manager to be unusable. In such cases, the other machines in the LoadLeveler cluster believe that the central manager machine is no longer operating. To remedy this situation, you can assign one or more alternate central managers in the machine stanza to take control.

The following machine stanza example defines the machine **bizgrid5** as an alternate central manager:

```
bizgrid5: type = machine
        central_manager = alt
```

If the primary central manager fails, the alternate central manager then becomes the central manager. When an alternate becomes the central manager, jobs will not be lost, but it may take a few minutes for all of the machines in the cluster to check in with the new central manager. As a result, job status queries may be incorrect for a short time.

When you define alternate central managers, you should set the following keywords in the configuration file:
CENTRAL_MANAGER_HEARTBEAT_INTERVAL = <amount of time in seconds>
CENTRAL_MANAGER_TIMEOUT = < the number of heartbeat intervals>

In our example (shown below), the alternate central manager will wait for 2 intervals, where each interval is 30 seconds:

```
CENTRAL_MANAGER_HEARTBEAT_INTERVAL = 30
CENTRAL_MANAGER_TIMEOUT = 2
```

8.11.6 Testing LL Central Manager Machine Failover

You can test the failover out by killing the central manager process called **Loadl_negotiator** on the **bizgrid4** machine. To prevent the central manager daemon from being restarted on the same node again, you should set the RESTARTS_PER_HOUR keyword to 0. This will bring up the central manager on the alternate, **bizgrid5**, in about a minute.

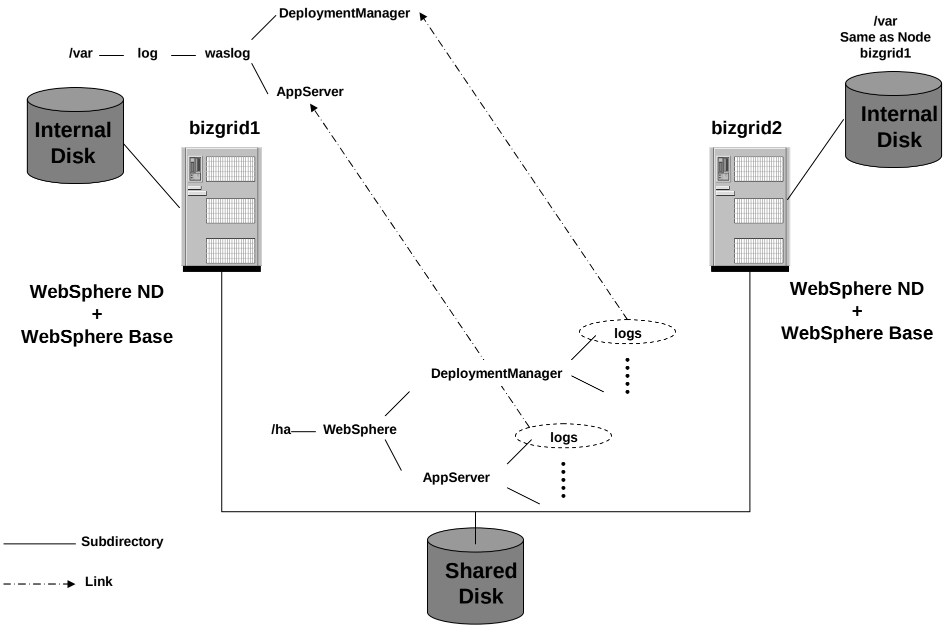
8.11.7 HA for LL Executing machines

This setup should be similar to HA for scheduling machines and hence has been skipped.

8.12 WebSphere Application Server and HA

Deployment managers, in WebSphere ND, are administrative agents that provide a centralized management view for all nodes in a cell. The management of clusters and the management of workload balancing of the application servers across one or several nodes are accomplished in the deployment manager. The deployment manager also hosts the administrative console and provides a single central point of administrative control for all the elements of the entire WAS distributed cell. The unavailability of the deployment manager impacts both the ability to make configuration changes and for the changes to be propagated to the application servers. This makes the deployment manager a single point of failure.

Figure 8.7. WebSphere ND and Base HA Setup



In the sections to follow, we will discuss making the deployment manager of the WAS-ND environment highly available. We will also show how a WebSphere base node can be failed over to a backup. As before we will place critical files on a shared file system (/ha for our example) so that it is available to a backup machine in the event of a WAS node failure.

Figure 8.7 shows the organization of the file system for our setup. In this setup:

1. The machine **bizgrid1** will serve as a primary WAS deployment manager machine and a WAS node.
2. The machine **bizgrid2** will serve as a backup for the WAS deployment manager and the WAS node.
3. The machine **bizgrid3** will serve as a WAS node.
4. The entire WebSphere Deployment manager (/ha/WebSphere/DeploymentManager) and WebSphere Node (/ha/WebSphere/AppServer) installation is kept on the shared disk. Only the log directories are kept on the local machines.

8.12.1 Installing WebSphere ND and Base in a HA Configuration

Follow the steps outlined below to install WAS-ND 5.1 with the necessary fixpacks on both the primary and the backup node:

1. Make sure heartbeat is running on both the nodes. This will ensure that **bizgrid1** is serving the cluster IP address **bizgrid** and the file system /ha is mounted on **bizgrid1** as well.

2. Create the installation directories for ND and Base using the command shown below on the **bizgrid1** node:

```
mkdir /ha/WebSphere/

mkdir /ha/WebSphere/DeploymentManager

mkdir /ha/WebSphere/AppServer
```

3. Extract the WAS ND 5.1 installation image using the commands shown below on the **bizgrid1** node:

```
rm -rf /tmp/was5.1nd-install

mkdir /tmp/was5.1nd-install

tar xf c53t6ml.tar -C /tmp/was5.1nd-install
```

   Here c53t6ml.tar is the installation tar file for WAS ND. Your image file name might be different based on how you obtained it.

4. Run the installation wizard on the node **bizgrid1** using the command shown below.

```
cd /tmp/was5.1nd-install/linuxi386
./launchpad.sh
```

   Use the following information in the wizard screens:
   a. **Installation directory**: /ha/WebSphere/DeploymentManager
   b. **Node**: bizgridManager
   c. **Host**: bizgrid.haw2.ibm.com
   d. **Cell**: bizgridNetwork
   e. In our setup, we already have a HTTP server and MQ installed so we chose not to install either of them. We also chose not to install the web services gateway.

5. Clean up the installation image directory using the following command:

```
rm –rf /tmp/was5.1nd-install
```

6. Extract the WAS ND 5.1 Fixpack 1 installation image using the commands shown below on the **bizgrid1** node:

```
rm -rf /tmp/was5.1.1nd-install
```

```
    mkdir /tmp/was5.1.1nd-install
    tar xzf was51_nd_fp1_linux.tar.gz -C /tmp/was5.1.1nd-install
```

Here was51_nd_fp1_linux.tar.gz is the installation tar file for WAS ND 5.1 Fix Pack 1. Your image file name might be different based on how you obtained it.

7. Run the silent update on **bizgrid1** using the commands shown below.

```
. /ha/WebSphere/DeploymentManager/bin/setupCmdLine.sh

cd /tmp/was5.1.1nd-install/

./updateSilent.sh –installDir /ha/WebSphere/DeploymentManager -fixpack -
install -fixpackDir /tmp/was5.1.1nd-install/fixpacks -fixpackID
was51_nd_fp1_linux -skipIHS -skipMQ
```

8. Clean up the fixpack installation image directory using the following command:

```
    rm –rf /tmp/was5.1.1nd-install
```

9. Extract the WAS ND 5.1.1 Cumulative Fix 1 installation image using the commands shown below on the **bizgrid1** node:

```
   rm -rf /tmp/was5.1.1.1nd-install

   mkdir /tmp/was5.1.1.1nd-install

   unzip -q was511_nd_cf1_linux.zip -d /tmp/was5.1.1.1nd-install
```

Here was511_nd_cf1_linux.zip is the installation zip file for WAS ND 5.1.1 Cumulative Fix 1. Your image file name might be different based on how you obtained it.

10. Run the silent update on **bizgrid1** using the command shown below.

```
. /ha/WebSphere/DeploymentManager/bin/setupCmdLine.sh

cd /tmp/was5.1.1.1nd-install

./updateSilent.sh –installDir /ha/WebSphere/DeploymentManager -fixpack -
install -fixpackDir /tmp/was5.1.1.1nd-install/fixpacks -fixpackID
was511_nd_cf1_linux -skipIHS -skipMQ
```

11. Clean up the fixpack installation image directory using the following command:

```
rm –rf /tmp/was5.1.1.1nd-install
```

12. Now let’s create the directory links shown in figure 1.
    a. Remove the Deployment Manager log directories from the installation by running the command shown below on **bizgrid1**:

    ```
    rm –rf /ha/WebSphere/DeploymentManager/logs
    ```

    b. Create directories for logs on a local filesystem by running the following commands on both the nodes **- bizgrid1** and **bizgrid2**:

    ```
    mkdir /var/log/waslog

    mkdir /var/log/waslog/DeploymentManager
    ```

    c. Set the correct permissions by running the following commands on both the nodes **- bizgrid1** and **bizgrid2**:

    ```
    chmod 755 /var/log/waslog

    chmod 755 /var/log/waslog/DeploymentManager
    ```

    d. Create the symbolic links by running the following commands on node **bizgrid1** only:

    ```
    ln -s /var/log/waslog/DeploymentManager
    /ha/WebSphere/DeploymentManager/logs
    ```

Follow the steps outlined below to install WAS 5.1 Base with the necessary fixpacks on both the primary and the backup node:

1. Extract the WAS Base 5.1 installation image using the commands shown below on the **bizgrid1** node:

```
rm -rf /tmp/was5.1base-install

mkdir /tmp/was5.1base-install

tar xf c53ipml.tar -C /tmp/was5.1base-install
```

Here c53ipml.tar is the installation tar file for WAS Base 5.1. Your image file name might be different based on how you obtained it.

2. Run the installation wizard on the node **bizgrid1** using the command shown below.

```
cd /tmp/was5.1base-install/linuxi386

./launchpad.sh
```

   Use the following information in the wizard screens:
   a. **Installation directory**: /ha/WebSphere/AppServer
   b. **Node**: bizgrid
   c. **Host**: bizgrid.haw2.ibm.com
   d. In our setup, we already have a HTTP server and MQ installed so we chose not to install either of them. Installation of these features can be disabled by selecting the **Custom Setup** option.

3. Clean up the installation image directory using the following command:

```
rm –rf /tmp/was5.1base-install
```

4. Extract the WAS Base 5.1 Fixpack 1 installation image using the commands shown below on the **bizgrid1** node:

```
rm -rf /tmp/was5.1.1base-install

mkdir /tmp/was5.1.1base-install

tar xzf was51_fp1_linux.tar.gz -C /tmp/was5.1.1base-install
```

   Here was51_fp1_linux.tar.gz is the installation tar file for WAS Base 5.1 Fix Pack 1. Your image file name might be different based on how you obtained it.

5. Run the silent update on **bizgrid1** using the command shown below.

```
. /ha/WebSphere/AppServer/bin/setupCmdLine.sh

cd /tmp/was5.1.1base-install

./updateSilent.sh –installDir /ha/WebSphere/AppServer -fixpack -install -
fixpackDir /tmp/was5.1.1base-install/fixpacks -fixpackID was51_fp1_linux -
skipIHS -skipMQ
```

6. Clean up the fixpack installation image directory using the following command on node **bizgrid1**:

```
rm –rf /tmp/was5.1.1base-install
```

7. Extract the WAS Base 5.1.1 Cumulative Fix 1 installation image using the commands shown below on the **bizgrid1** node:

```
rm -rf /tmp/was5.1.1.1base-install

mkdir /tmp/was5.1.1.1base-install

unzip -q was511_cf1_linux.zip -d /tmp/was5.1.1.1base-install
```

   Here was511_cf1_linux.zip is the installation zip file for WAS Base 5.1.1 Cumulative Fix 1.

8. Run the silent update on **bizgrid1** using the command shown below.

```
. /ha/WebSphere/AppServer/bin/setupCmdLine.sh

cd /tmp/was5.1.1.1base-install

./updateSilent.sh –installDir /ha/WebSphere/AppServer -fixpack -install -
fixpackDir /tmp/was5.1.1.1base-install/fixpacks -fixpackID was511_cf1_linux -
skipIHS -skipMQ
```

9. Clean up the fixpack installation image directory using the following command on node **bizgrid1**:

```
rm –rf /tmp/was5.1.1.1base-install
```

10. Now let’s create the directory links shown in figure 1.
    e. Remove the WebSphere log directories from the installation by running the command shown below on **bizgrid1**:

    ```
    rm –rf /ha/WebSphere/AppServer/logs
    ```

    f. Create directories for logs on a local filesystem by running the following command on both the nodes - **bizgrid1** and **bizgrid2**:

```
mkdir /var/log/waslog/AppServer
```

g. Set the correct permissions by running the following command on both the nodes - **bizgrid1** and **bizgrid2**:

```
chmod 755 /var/log/waslog/AppServer
```

h. Create the symbolic links by running the following command on node **bizgrid1** only:

```
ln -s /var/log/waslog/AppServer /ha/WebSphere/AppServer/logs
```

11. Install WAS base on the node **bizgrid3** (steps 1-9 above, only) with the following information:
    a. **Installation directory**: /opt/WebSphere/AppServer
    b. **Node**: bizgrid3
    c. **Host**: bizgrid3.haw2.ibm.com
    d. In our setup, we already have a HTTP server and MQ installed so we chose not to install either of them.

12. Start the deployment manager on **bizgrid1** by running the startManager.sh from the bin directory of the Deployment Manager Installation.

13. Add the WAS nodes **bizgrid** and **bizgrid3** (created during WAS Base installs above) to the cell **bizgridNetwork** (created in step 4 of WAS ND install) by executing the following command on each node (from the application server bin directory) :

```
addnode.sh bizgrid
```

14. Verify through the admin console that the cell appears correct. Open the console (http://bizgrid.haw2.ibm.com:9090/admin) and make sure that you see all of the nodes for the Application Servers.

15. Stop Everything. This should mean the deployment manager and the node agents on each of the WAS nodes. The commands are as follows:
    a. Node Agents: stopNode.sh (from the bin directory of the Application Server)
    b. Deployment Manager: stopManager.sh (from the bin directory of the deployment manager)

8.12.2 Configure heartbeat to manage Deployment Manager

The steps needed to configure heartbeat to manage the WAS ND deployment manager are outlined below:

1. Let’s create a script to start and stop the deployment manager process. A very basic script (wasdmgr) is shown in listing 8.10. You can further customize it to suit your setup. This script has to be placed in the /etc/rc.d/init.d directory.

Listing 8.10. wasdmgr script

```
#!/bin/bash
#
#            /etc/rc.d/init.d/wasdmgr
#
# Starts the WebSphere Deployment Manager
#
# chkconfig: 345 88 57
# description: Runs  WAS DMGR

. /etc/init.d/functions
# Source function library.

PATH=/usr/bin:/bin:/ha/WebSphere/DeploymentManager/bin
#===========================================================
================
SU="sh"
#===========================================================
========
start() {
            echo "$0: starting websphere deployment manager"
            $SU -c "startManager.sh"
}
#===========================================================
========
stop() {
            echo "$0: stopping websphere deployment manager"
            $SU -c "stopManager.sh"
      #sleep 30
}


case $1 in
'start')
   start
   ;;
'stop')
```

```
    stop
    ;;
'restart')
    stop
    start
    ;;
*)
    echo "usage: $0 {start|stop|restart}"
    ;;
esac
```

2. Now let’s configure the /etc/ha.d/haresources file (on both the scheduling machine nodes) to include the above wasdmgr script. The relevant portion of the modified file is shown below:

```
bizgrid1.haw2.ibm.com 9.22.7.46
Filesystem::nfsgrid.haw2.ibm.com:/ha::/ha::nfs::rw,hard httpd
mqseries loadl wasdmgr
```

So, this line dictates that on startup of **heartbeat**, have **bizgrid1** serves the cluster IP address, mount the shared file system, and start apache web server, WebSphere MQ, IBM LoadLeveler and the WebSphere deployment manager as well. On shutdown, **heartbeat** will first stop the deployment manager, and then LoadLeveler, and then WebSphere MQ, and then apache, and then un-mount the shared file system, and finally give up the IP.

8.12.3 Configure heartbeat to manage WAS Node

The steps needed to configure heartbeat to manage the WAS node agent and application server processes are outlined below:

1. Let’s create a couple of scripts to start and stop the node agent and the application server processes. A very basic script to start a node agent (wasnode) is shown in listing 8.11 and the script to start application servers (wasserver) on a node is shown in listing 8.12. You can further customize these to suit your setup. These scripts have to be placed in the /etc/rc.d/init.d directory.

Listing 8.11. wasnode script

```
#!/bin/bash
#
#            /etc/rc.d/init.d/wasnode
#
# Starts the WebSphere Node Agent
#
# chkconfig: 345 88 57
# description: Runs  WAS NODE
```

```
. /etc/init.d/functions
# Source function library.

PATH=/usr/bin:/bin:/ha/WebSphere/AppServer/bin
#===============================================================
========
SU="sh"
#===============================================================
========
start() {
        echo "$0: starting websphere node agent"
        $SU -c "startNode.sh"
}
#===============================================================
================
stop() {
        echo "$0: stopping websphere node agent"
        $SU -c "stopNode.sh"
    #sleep 30
}


case $1 in
'start')
  start
  ;;
'stop')
  stop
  ;;
'restart')
  stop
  start
  ;;
*)
  echo "usage: $0 {start|stop|restart}"
  ;;
esac
```

Listing 8.12. wasserver script

```
#!/bin/bash
#
#       /etc/rc.d/init.d/wasserver
#
# Starts the WebSphere Application Server
```

```
#
# chkconfig: 345 88 57
# description: Runs  WAS Server

. /etc/init.d/functions
# Source function library.

PATH=/usr/bin:/bin:/ha/WebSphere/AppServer/bin
WASSERVERS="server1"
#==============================================================
========
SU="sh"
#==============================================================
========
start() {
    for wasserver in $WASSERVERS ; do
        export wasserver
                echo "$0: starting websphere application server $wasserver"
                $SU -c "startServer.sh $wasserver"
    done
}
#==============================================================
================
stop() {
    for wasserver in $WASSERVERS ; do
        export wasserver
                echo "$0: stopping websphere application server $wasserver"
                $SU -c "stopServer.sh $wasserver"
        #sleep 30
    done
}


case $1 in
'start')
    start
    ;;
'stop')
    stop
    ;;
'restart')
    stop
    start
    ;;
*)
    echo "usage: $0 {start|stop|restart}"
    ;;
esac
```

2. Now let’s configure the /etc/ha.d/haresources file (on both the scheduling machine nodes) to include the above scripts. The relevant portion of the modified file is shown below:

```
bizgrid1.haw2.ibm.com 9.22.7.46
Filesystem::nfsgrid.haw2.ibm.com:/ha::/ha::nfs::rw,hard httpd
mqseries loadl wasdmgr wasnode wasserver
```

So, this line dictates that on startup of **heartbeat**, have **bizgrid1** serves the cluster IP address, mount the shared file system, and start apache web server, WebSphere MQ, LoadLeveler, the WebSphere deployment manager, node agent and application servers as well. On shutdown, **heartbeat** will first stop the application servers, and then the node agent, and then the deployment manager, and then LoadLeveler, and then WebSphere MQ, and then apache, and then un-mount the shared file system, and finally give up the IP.

8.12.4 Testing WAS Deployment Manager Failover

This section outlines the steps needed to test the high availability of the deployment manager.

1. **Start the heartbeat service** on the primary and then on the backup node. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

After heartbeat starts successfully, you should see a new interface with the IP address that you configured in the **ha.cf** file. Once you've started heartbeat, take a peek at your log file (default is /var/log/ha-log) on the primary and make sure that it is doing the IP take over and then starting dmgr, node agent, application servers and other resources. Heartbeat will not start dmgr or any other resource on the backup. This happens only after the primary fails.

2. **Start WebSphere node agent and application server** on the **bizgrid3** node.

3. From the admin console (http://bizgrid:9090/admin) make sure the application servers on both machines (**bizgrid1** and **bizgrid3**) are running. If not start them.

4. **Deploy the sample enterprise application**, TestWebSphereHA.ear (included with the sample code) under the was\sample_ver_1 directory, using the admin console. Make sure

you deploy it on both **bizgrid** and **bizgrid3** WAS node. Start the application using the console.

5. **Verify that the application runs** on both the nodes by pointing the browser at the following URLs:

    1. http://bizgrid.haw2.ibm.com:9080/TestWebSphereHAWeb/Test
    2. http://bizgrid3.haw2.ibm.com:9080/TestWebSphereHAWeb/Test

For both the URLs, the browser should display the following text:

```
Test:doGet() Invoked the HA Test Servlet.
```

We have successfully deployed an application on two WAS nodes that is being managed by a deployment manager running on **bizgrid1**. Let's see if this configuration information survives a failover to the back.

6. **Simulating failover.** This can be done simply stopping heartbeat on the primary system using the command shown below:

```
/etc/rc.d/init.d/heartbeat stop
```

You should see all the services come up on the backup machine. You can verify that deployment manager is running on the backup by checking the **/var/log/ha-log** file. Once the backup has taken over control, start the admin console again. You should see the two application server and the application TestWebSphereHA. This shows that the configuration information survived a failure. Also repeat step 5 above to verify that the application works on both the nodes after failover.

7. **Update the enterprise application** to a newer version TestWebSphereHA.ear (included with the sample code) under the was\sample_ver_2 directory, using the admin console. Make sure you select both **bizgrid** and **bizgrid3** WAS nodes while updating. After update make sure you save the master configuration. Also make sure that the **Synchronize changes with Nodes** option is selected. You should be able to successfully update the application. Run step 5 again.

a. For the URL http://bizgrid.haw2.ibm.com:9080/TestWebSphereHAWeb/Test the browser displays:

```
Test:doGet() Invoked the HA Test Servlet on remote host :
bizgrid2.haw2.ibm.com
```

The hostname of the machine gets printed as well. This verifies that the cluster IP bizgrid.haw2.ibm.com is being served by bizgrid2.haw2.ibm.com machine.

b. For the URL http://bizgrid3.haw2.ibm.com:9080/TestWebSphereHAWeb/Test the browser displays:

```
Test:doGet() Invoked the HA Test Servlet on remote host :
```

```
bizgrid3.haw2.ibm.com
```

8. Start the heartbeat service back on the primary. This should stop the WAS processes on the secondary and start them on the primary. The primary should also take over the cluster IP as well. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

Start the admin console again. You should see the two application server and the application TestWebSphereHA. This shows that the updated configuration information survived a failover to the primary. Also repeat step 5 above to verify that the application works on both the nodes.

We have successfully shown how configuration information of the deployment manager survives a failover from a primary machine to a standby machine.

8.12.5 Testing WAS Node Failover

For testing the failover for a node, we have modified the sample test application so that it keeps track of how many times it has been invoked by maintaining a persistent counter (count). Here we have chosen the file system as mechanism to keep the counter persistent. For the failover of the application to work, this data must be kept on the shared disk.

This section outlines the steps needed to test the high availability of a websphere node and application.

1. Start the heartbeat service on the primary and then on the backup node. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

2. **Start the WebSphere node agent and application server** on the **bizgrid3** node.

3. From the admin console (http://bizgrid:9090/admin) make sure the application servers on both machines (**bizgrid1** and **bizgrid3**) are running. If not start them. Also you should see the application TestWebSphereHA.

4. **Update the enterprise application** to a newer version TestWebSphereHA.ear (included in the sample code) under the was\sample_ver_3 directory, using the admin console. Make sure you select both **bizgrid** and **bizgrid3** WAS nodes while updating. After update make sure you save the master configuration. Also make sure that the **Synchronize changes with Nodes** option is selected. You should be able to successfully update the application.

5. **Verify that the application runs** on the **bizgrid** websphere node by pointing a browser to the following URL
http://bizgrid.haw2.ibm.com:9080/TestWebSphereHAWeb/Test

The output in the browser is as shown below:

```
Test:doGet() Invoked the HA Test Servlet on remote host : bizgrid1.haw2.ibm.com
Test:doGet() This servlet has been invoked 1 times
Test:doGet() Count data file :
/ha/WebSphere/AppServer/installedApps/bizgridNetwork/TestWebSphereHA.ear/
TestWebSphereHAWeb.war/WEB-INF/count.dat
```

This shows that the application is successfully running on the WAS node **bizgrid** which is being run on the primary node **bizgrid1**. Also note that the count.dat file is on the shared file system /ha.

Repeat this step one more time so that the count is 2.

6. **Simulating failover.** This can be done simply stopping heartbeat on the primary system using the command shown below:

```
/etc/rc.d/init.d/heartbeat stop
```

Once the backup has taken over control, start the admin console again. You should see the two application server and the application TestWebSphereHA.
Point a browser to the following URL
http://bizgrid.haw2.ibm.com:9080/TestWebSphereHAWeb/Test

The output in the browser, for this run, is shown below:

```
Test:doGet() Invoked the HA Test Servlet on remote host : bizgrid2.haw2.ibm.com
Test:doGet() This servlet has been invoked 3 times
Test:doGet() Count data file :
/ha/WebSphere/AppServer/installedApps/bizgridNetwork/TestWebSphereHA.ear/
TestWebSphereHAWeb.war/WEB-INF/count.dat
```

This shows that the application is successfully running on the WAS node **bizgrid** which is being run on the backup node **bizgrid2**. Also the application data (value of count) has survived a failover as it is on the shared disk. This is a trivial example of application data failover. A more practical example would be one that uses a database. In that case HA for the database should be implemented as well.

7. Start the heartbeat service back on the primary. This should stop the WAS processes on the secondary and start them on the primary. The primary should also take over the cluster IP as well. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

8.13 Implementing WebSphere XD for High Availability

As with WebSphere ND, the deployment manager of XD is a single point of failure. In the sections to follow, we will discuss making the deployment manager of the WAS-XD environment highly available. As before we will place critical files on a shared file system (/ha for our example) so that it is available for a backup machine in the event of a WAS node failure.

8.13.1 Installing WebSphere XD in a HA Configuration

Follow the steps outlined in this section to install WAS-XD 5. 1 on both the primary and the backup node. For more information refer to *WebSphere Extended Deployment V5.1 Planning and Installing Guide* (see Resources for link).

1. Make sure heartbeat is running on both the nodes. This will ensure that **bizgrid1** is serving the cluster IP address **bizgrid**.

2. **Install WAS ND 5.1 with the necessary fixpacks on bizgrid1**. Follow the steps outlined in one of the earlier sections.

3. **Install WAS Base 5.1 with the necessary fixpacks on bizgrid1 and bizgrid3**. Follow the steps outlined in one of the earlier sections. Don’t add the nodes to the deployment manager, yet.

4. Extract the WAS XD 5.1 installation image using the commands shown below on the **bizgrid1** node:

```
rm -rf /tmp/was5.1xd-install

mkdir /tmp/was5.1xd-install

    unzip -q xd.51.linux.ia32.zip -d /tmp/was5.1xd-install
```

5. Run the installation wizard on the node **bizgrid1** to install XD updates to the Deployment manager using the command shown below.

```
. /ha/WebSphere/DeploymentManager/bin/setupCmdLine.sh

cd /tmp/was5.1xd-install/
```

```
./LinuxInstaller.bin
```

Use the following information in the wizard screens:

a. Select **WebSphere Application Server Network Deployment Only** as the product to apply XD extensions to.
b. Verify that the feature **Deployment Manager Extensions** has been automatically selected to install.

6. Apply the WebSphere Application Server 5.1.1 Cumulative Fix for SDKs. Refer to WAS XD documentation for more details.

7. Run the installation wizard on the node **bizgrid1** to install XD updates to the Application server using the command shown below.

```
. /ha/WebSphere/AppServer/bin/setupCmdLine.sh

cd /tmp/was5.1xd-install/

./LinuxInstaller.bin
```

Use the following information in the wizard screens:

a. Select **WebSphere Application Server Only** as the product to apply XD extensions to.
b. Verify that the feature **Application Server Extensions** has been automatically selected to install.

8. Apply the WebSphere Application Server 5.1.1 Cumulative Fix for SDKs. Refer to WAS XD documentation for more details.

9. Clean up the installation image directory using the following command:

```
rm –rf /tmp/was5.1xd-install
```

10. Install the XD Application Server Extensions on the node **bizgrid3**. Follow steps 4, 7, 8 and 9 above with the following information:

- **Installation directory**: /opt/WebSphere/AppServer

11. Start the deployment manager on **bizgrid1** by running the startManager.sh from the bin directory of the Deployment Manager Installation.

12. Add the WAS nodes **bizgrid** and **bizgrid3** (created during WAS Base installs above) to the cell **bizgridNetwork** (created in step 3) by executing the following command on each node (if you did this as part of WebSphere ND HA test, you should remove the nodes from the bizgrid network before adding them):

```
cd /opt/WebSphere/AppServer/bin

./addnode.sh bizgrid
```

13. Verify through the admin console that the cell appears correct. Open the console (http://bizgrid.haw2.ibm.com:9090/admin) and make sure that you see all of the nodes for the Application Servers.

14. Stop Everything. This should mean the DMgr and the Node Agents on each of the Application Server Nodes. The commands are as follows:

   a. Node Agents: stopNode.sh (from the bin directory of the Application Server/ODR)
   b. Deployment Manager: stopManager.sh (from the bin directory of the DMgr)

8.13.2 Configure heartbeat to manage WAS-XD Deployment Manager

This was already done, if you followed the steps needed for WAS-ND deployment manager.

8.14 Implementing Business Grid Gateway for High Availability

The Business Grid gateway stores its persistent information in the directory defined by the **dataDir** property in the bgrid.properties file. In this section we will install the gateway on the WAS nodes on the two machines **bizgrid1** and **bizgrid2**. So for our setup, the default value for the **dataDir** property is /ha/WebSphere/AppServer/bgrid/data. As we can see that this is already on the shared disk. So we don’t have to do anything further than to test that the HA for the gateway works.

Follow the steps outlined in chapter 6 to install the business grid gateway on the nodes **bizgrid1** and **bizgrid2** so that they can act as a master node and the steps outlined in chapter 7 to install business grid on node **bizgrid3** so that it can act as a worker node, with the following in mind:

1. Ensure that heartbeat is started on both **bizgrid1** and **bizgrid2**. This would bring up WebSphere MQ, Loadleveler and WebSphere DMGR, node agent and server on the primary node.

2. Since you have already done the MQ configuration (in one of the previous sections) step 3 of the chapter 6 should be skipped.

3. In step 5 of the chapter 6, you are creating a server called **bgridGateway**. This has to be included in the scripts **wasserver** to allow **heartbeat** to manage this server.

4. In step 7 of the chapter 6, use the following **replyTo** property:

```
replyTo=jms:/queue?channel=SYSTEM.DEF.SVRCONN/TCP&host=localhost(1415)&\
queueName=ENDPT.REPLYQ&queueManager=bgridept.queue.manager
```

5. In step 9 and 14 of the chapter 6, make sure you use the cluster name as the hostname for the gateway.

6. Step 16 of the chapter 6 should be skipped as you have already done this in the LoadLeveler sections of this document.

7. On the worker node (**bizgrid3**) create a link as follows:

```
mkdir /ha

ln –s /ha/WebSphere /opt/WebSphere
```

This is needed to ensure that the directory structure is the same on both the submitting node and the executing node for LoadLeveler.

8. Create and start the WebSphere MQ queue manager and queues on the worker nodes (**bizgrid3**):

While logged on as root, execute the following commands on the worker node:

```
#su – mqm

mqm$export LD_ASSUME_KERNEL=2.4.19

mqm$crtmqm bgridept.queue.manager

mqm$strmqm bgridept.queue.manager

mqm$runmqsc bgridept.queue.manager

define qremote(ENDPT.REPLYQ) rname(ENDPT.REPLYQ) xmitq(TOBGRID) +
rqmname(‘bgrid.queue.manager’)

define qlocal(TOBGRID) usage(XMITQ) trigger trigtype(FIRST) +
trigdata(ENDPOINT.TO.BGRID) initq(SYSTEM.CHANNEL.INITQ)

define channel(ENDPOINT.TO.BGRID) chltype(SDR) trptype(TCP) +
xmitq(TOBGRID) conname(‘bgrid_gateway_host_name’)

start channel(ENDPOINT.TO.BGRID)

end
```

```
mqm$runmqlsr –t tcp –p 1415 –m bgridept.queue.manager  &

mqm$exit
```

8.14.1 Configure heartbeat to manage Business Grid Gateway

The business grid gateway runs as a J2EE application on the WAS node. We have already configured heartbeat to manage the WAS node processes (node agent and application servers). So nothing further is needed.

8.14.2 Testing Business Grid Gateway Failover

In this section we will do a few tests using the Echo and the Mandelbrot sample (included with the gateway) to ensure high availability of the gateway master node:

Scenario 1:
In this scenario, to begin with, we will make the LoadLeveler startd unavailable on the executing machine by draining. We will then submit a job to the gateway and then bring the primary master down and failover to the secondary. To try out this test scenario, follow the steps outlined below:

1. **Start the heartbeat service** on the primary and then on the backup node. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

2. **Start the loadleveler daemons on bizgrid3,** as user **loadl**, using the following command:

```
llctl start
```

3. **Make bizgrid3 unavailable** as an executing machine by draining the startd daemon. This is needed to give us enough time, after submitting a job, to test failover. Use the following command as user **loadl**:

```
llctl drain startd
```

4. Start the GUI for managing the loadleveler cluster using the command below on the **bizgrid3** machine:

```
xloadl &
```

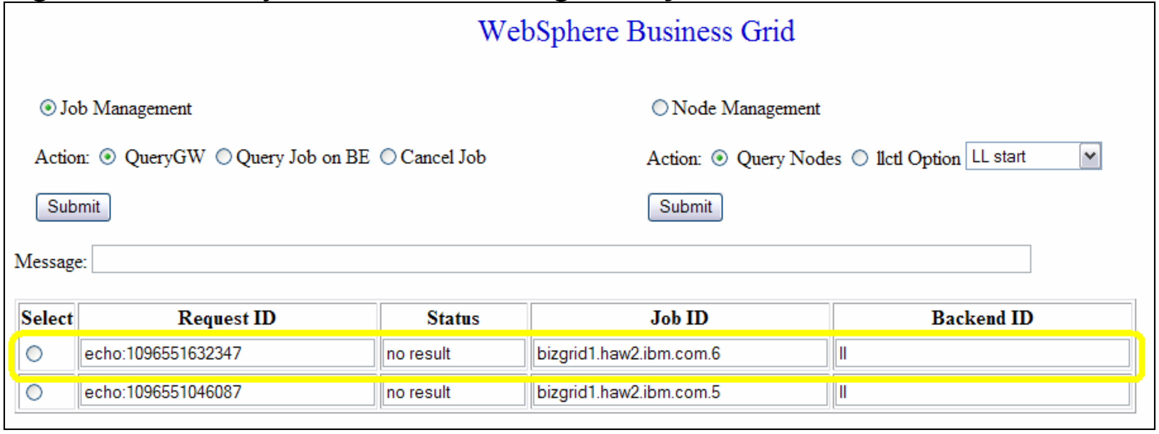
5. Open a browser to the following URL:

```
http://bizgrid.haw2.ibm.com:9080/EchoSample/gui
```

6. Enter a string **Hello** in the **String to echo** textbox and click **Submit**.

7. You should see a new job in the LoadLeveler console. The job should be in an idle state (I) for the lack of availability of an executing machine. You should also be able to see the job using the gateway admin GUI (http://bizgrid.haw2.ibm.com:9080/BGridAdmin/gui). The sample output is shown in figure 8.8.

Figure 8.8. Gateway Admin GUI showing active jobs



● Job Management

Action: ● QueryGW ○ Query Job on BE ○ Cancel Job

○ Node Management

Action: ● Query Nodes ○ llctl Option LL start

Message:

Select	Request ID	Status	Job ID	Backend ID
○	echo:1096551632347	no result	bizgrid1.haw2.ibm.com.6	II
○	echo:1096551046087	no result	bizgrid1.haw2.ibm.com.5	II

8. **Simulating failover.** This can be done simply stopping heartbeat on the primary system using the command shown below:

```
/etc/rc.d/init.d/heartbeat stop
```

You should see all the services come up on the backup machine. This takes some time. From the WAS admin console make sure that the application server for the gateway (**bgridGateway**) is running before proceeding to the next step.

9. **Resume the startd daemon on bizgrid3 machine.** This is done by executing the following command on **bizgrid3**:

```
llctl resume startd
```

Now the bizgrid3 machine is available for executing jobs. The job should now go to a running state and finally should complete on the **bizgrid3** machine. You can verify job completing by taking a peek at the bgjobwrapper.job.stdout file, in the /tmp directory. Click on the Refresh button on the Echo Sample GUI in the browser. The **Last string echoed** textbox should now show **Hello**. Also the Gateway Admin GUI should not display this job. Thus we have shown how using shared disk the jobs submitted before the failure of the primary scheduling node can be recovered after the backup has taken over control.

10. **Start the heartbeat service back on the primary**. This should stop the business gateway processes on the secondary and start them on the primary. The primary should also take over the cluster IP as well. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

Scenario 2:
In this scenario, we will submit a job to the gateway and wait till it gets scheduled to an endpoint. Once the job is running on the endpoint we will bring the primary master down and failover to the secondary. To try out this scenario we need to make the echo.sh sample script more long running to give us enough time to bring down the primary master node. We will modify echo.sh such that it will display a dialog box and wait for a user input before completing the execution. Follow the steps outlined below:

1. Download and install Xdialog (see Resources for link) on the endpoint (slave) machines.

2. Add the following lines to echo.sh after the line where the **TMPINPUT** variable is being set:

```
export DISPLAY=your.display.host:0.0
sudo /usr/local/bin/Xdialog --title "Echo Job" \
        --msgbox "Started Echo Job. Press the ENTER key to continue." 10 60
```

Change your.display.host to the host where you want the dialog box to be displayed.

3. Using **visudo** command, add the following line at the end:

```
loadl   ALL = NOPASSWD: /usr/local/bin/Xdialog
```

Now we are ready to test this scenario. Follow the steps outlined below:
1. **Start the heartbeat service** on the primary and then on the backup node. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

2. **Start the loadleveler daemons on bizgrid3,** as user **loadl**, using the following command:

```
llctl start
```

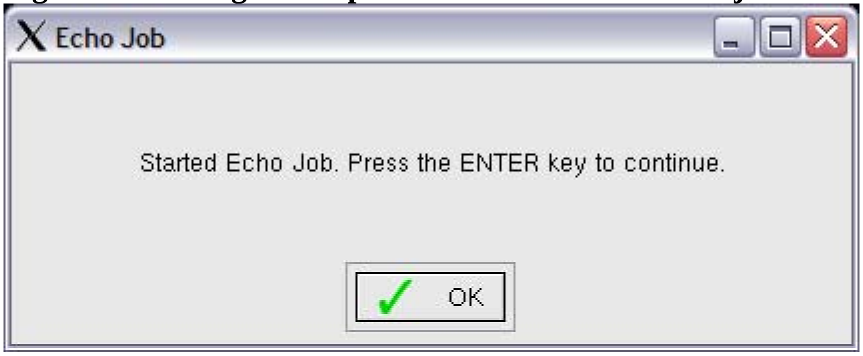
3. Open a browser to the following URL:

```
http://bizgrid.haw2.ibm.com:9080/EchoSample/gui
```

4. Enter a string **Hello** in the **String to echo** textbox and click **Submit**.

5. This should popup a dialog as shown below:

Figure 8.9. Dialog box to pause the execution of Echo job



Started Echo Job. Press the ENTER key to continue.

Do NOT click the OK button yet.

6. **Simulating failover.** This can be done simply stopping heartbeat on the primary system using the command shown below:

```
/etc/rc.d/init.d/heartbeat stop
```

You should see all the services come up on the backup machine. This takes some time. From the WAS admin console make sure that the application server for the gateway (**bgridGateway**) is running before proceeding to the next step.

7. Complete the execution of the echo job by clicking OK button on the dialog shown in figure 9.

8. Verify job completion by taking a peek at the bgjobwrapper.job.stdout file, in the /tmp directory on the node **bizgrid3**. Click on the Refresh button on the Echo Sample GUI in the browser. The **Last string echoed** textbox should now show **Hello**.

9. **Start the heartbeat service back on the primary**. This should stop the business grid gateway processes on the secondary and start them on the primary. The primary should also take over the cluster IP as well. The command is shown below:

```
/etc/rc.d/init.d/heartbeat start
```

Chapter 9 Resources

- Download the sample code package for chapter 8, hahbcode.tar.gz. More information about the contents can be obtained from the article High-availability middleware on Linux, Part 1: Heartbeat and Apache Web server.
- More information on IBM eServer 335 series can be obtained here.
- Check out the High-Availability Linux Project web site for heartbeat.
- Andre Bonhote's article on HA-NFS entitled High-Availability NFS Server with Linux Heartbeat, August, 2003 issue of the European publication Linux Magazine
- Read heartbeat success stories
- Find more information on the other high-availability solutions mentioned in this article:
    a. High Availability Cluster Multiprocessing (HACMP)
    b. IBM Tivoli System Automation for Multiplatforms
    c. Legato AAM
    d. SteelEye LifeKeeper
    e. Veritas Cluster Server
- Checkout the IBM Redbook, Implementing High Availability on RISC/6000 SP for implementing loadleveler high availability using HACMP.
- WebSphere MQ family books are available online here.
- WebSphere MQ for Linux for Intel and Linux for zSeries Quick Beginnings (GC34-6078-00)
- MC63: WebSphere MQ for AIX - Implementing with HACMP SupportPac.
- WebSphere Business Integration Message Broker and high availability environments (developerWorks, March 2004).
- IBM Loadleveler documentation:
    - LoadLeveler for AIX 5L and Linux: Installation Memo (GI11-2819-02)
    - LoadLeveler for AIX 5L and Linux: Diagnosis and Messages Guide (GA22-7882-02)
    - LoadLeveler for AIX 5L and Linux: Using and Administering (SA22-7881-02)
- IBM WebSphere Application Server support page contains links to Fix Packs and product documentation.
- Download Xdialog
- Business Grid Components for WebSphere Extended Deployment White paper.
- Business Grid Components for WebSphere Extended Deployment Administration Guide.
- WebSphere Extended Deployment Library contains links to the Infocenter and the Planning and Installing guide.

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
11-04

All statements regarding IBM future direction or intent are subject to change or withdrawal without notice and represent goals and objectives only.