Chapter 14.

February 2008

Welcome to the February 2008 edition of IBM Information Server Developer's Notebook. This month we answer the question;

> How do I create my own grid installation of IBM Information Server on the Red Hat Linux operating system?
>
> *Excellent question! Every software server ever created has a process architecture, be it a (single process) multi-threaded architecture, a two process architecture, and dozens more. Based on its process architecture, IBM Information Server works, scales, and operates fabulously in a grid configuration. Because grid hardware and operating system setup requires much more than the average amount of work, IBM offers a canned grid installation of IBM Information Server. But if you wish to roll your own IBM Information Server grid, you can do that too.*

In short, we are going discuss all of the relevant terms and technologies above, and provide examples that detail how to deliver this functionality using IBM Information Server.

Software versions

All of these solutions were *developed and tested* on IBM Information Server (IIS) version 8.01, FixPak 1A, using the Microsoft Windows XP/SP2 platform to support IIS client programs, and a RedHat Linux Advanced Server 4 (RHEL 4) FixPak U6 32 bit SMP server (Linux kernel version 2.6.9-67.EL-smp) to support the IIS server components.

IBM Information Server allows for a single, consistent, and accurate view of data across the full width of the corporate enterprise, be it relational or non-relational, staged or live data. As a reminder, the IBM Information Server product contains the following major components;

> WebSphere Business Glossary Anywhere™, WebSphere Information Analyzer™, WebSphere Information Services Director™, WebSphere DataStage™, WebSphere QualityStage™, WebSphere Metadata Server and Metabridges™, WebSphere Metadata Workbench™, WebSphere Federation Server™, Classic Federation™, Event Publisher™, Replication Server™, Rational Data Architect™, DataMirror Transformation Server™, and others.

Obviously, IBM Information Server is a large and capable product, addressing many strategic needs across the enterprise, and supporting different roles and responsibilities.

14.1 Terms and core concepts

Grid versus cluster, a 2000 year old struggle

There are persons in the Information Technology field who would argue the differences between a grid or cluster configuration of hardware and/or software servers for hours. Since this document is really about achieving benefits from time spent, we're not really going to head down that road. Before we work with the terms cluster and grid, there are other terms we are dependent on, which are in need of defining;

- Hardware server- In this context, a hardware server refers to a physical computer; for example, a Dell laptop, or an IBM pSeries box. Some hardware servers offer a choice in the operating system that may be installed and run.
- Software server- In this context, is a further refinement over the term hardware server, and refers specifically to the operating system that is present and running.

  Through virtualization, it is possible to have numerous software servers running on one hardware server. That topic is not expanded upon further here.
- High speed inter-connect, NIC- NIC is an acronym for network interface card.

  One software server may have zero or more network cards, and network cards can operate at different communication speeds. It is common to have a software server with multiple NICs; each configured to support a dedicated type of network traffic.
- Node- In this context, is a synonym for software server.
- Node type- A refinement over the term *node*, implies that a given node is configured to support a specialized purpose in a collection of nodes. Often this means a specific subset of application software is installed and running.
- Homogenous/heterogeneous- A homogenous collection of nodes would include minimally the same type of hardware, and same type and version of operating system. A heterogeneous collection of nodes has different hardware or operating systems and versions.
- High availability, fault tolerant- The ability to suffer a complete loss of a hardware server, and still be able to provide a given software service. Generally involves a stand by (second/redundant) hardware server, often with shared disk between both boxes.

  A software agent of some sort monitors availability of the primary server, and redirects service requests to the backup server as needed.

WikiPedia.com states that, "grid computing (generally differs) from typical cluster computing, (in) that grids tend to be more loosely coupled, heterogeneous, and geographically dispersed". Also, on **WikiPedia.com** and other sources, you see clusters mentioned in the discussion of high availability systems, where grid often is not. Here is what we are going to say regarding clusters and grids as it relates to IBM Information Server (IIS);

- IIS can be placed in a clustered configuration to support high availability.

  This topic is not expanded upon further here.

- IIS can be placed in a grid configuration. Generally this is done to achieve a high performance and scalability quotient, on low cost or commodity hardware.

  If a given 16 CPU Sun Microsystems SMP hardware server costs $500K, how much do 8 (count) two CPU commodity Intel servers cost- Even if the Sun box has a faster back plane and related, can the Intel boxes be made to be fast enough-

Note: Operating system preparation (specifically, RedHat Linux Advanced Server) towards operating an IBM Information Server (IIS) system in a grid configuration, is the topic of this document. Clustering and high availability of IIS is not.

- IIS has three type of nodes;
  - Metadata repository node.

    This node operates a SQL database server. While IIS ships with a limited use copy of DB2/UDB, Oracle and Microsoft SQL/Server are also supported SQL database server types.
  - Java/J2EE compliant Application Server node.

    This node operates a limited use WebSphere Application Server (WAS) installation. Currently, WAS is the only supported Java/J2EE compliant application server. Other J2EE servers may and should work, as is the promise of Java/J2EE; only WAS is tested and supported by IBM.
  - Compute node (basically, DSEngine).
- With IIS, the three node types can be on different types of hardware and/or software servers.

All of the Compute nodes must be on the same hardware and software server types. (This is due to a dependency towards C++ language compilation of certain end user created units of work within IIS.)

Note: The above block states what is possible. The earliest versions of the IIS supported platform documentation could be interpreted to imply that all 3 node types had to be on the same type and version of operating system.

If you are considering running any of the 3 node types on different hardware or software servers, it'd be best to check with IBM Technical Support first.

- While the Compute nodes should probably share a high speed inter-connect between homogenous system types, the Metadata repository node and Application Server node could be remote.
- You can have a high availability system with IIS by clustering all or part of the above node types; meaning, you can cluster the Metadata repository node, the Application Server node, and Compute nodes.

  This topic is not expanded upon further here; this document discusses grid computing, not high availability computing.

APT_CONFIG_FILE, and engaging a resource manager

APT_CONFIG_FILE is a core performance and configuration file inside IBM Information Server (IIS). This file specifies the location and amount of resource that a given IIS Job is permitted to consume. One of the resources this file specifies, is the name and location of any IIS Compute nodes. When operating IIS in a grid configuration, an optional *resource manager* may be used to generate a dynamic copy (version) of an APT_CONFIG_FILE at run time, for each individual Job.

Note: IBM Tivoli Load Leveler is one example of resource manager software. There are also open source resource management systems that are available.

Generally, resource manager type software reports the availability of given nodes (is that node up/working), and then the business of given nodes.

Figure 14-1 displays a mixture of sample Jobs operating inside an IIS system that is configured for grid use. For simplicity, this example assumes a 4 (count) Compute node configuration, with resource (memory, CPU, and disk) being given out by a working resource manager (RM) at the whole node level; meaning, a given Job can receive 25% of the available resource, 50%, 75% or 100%.

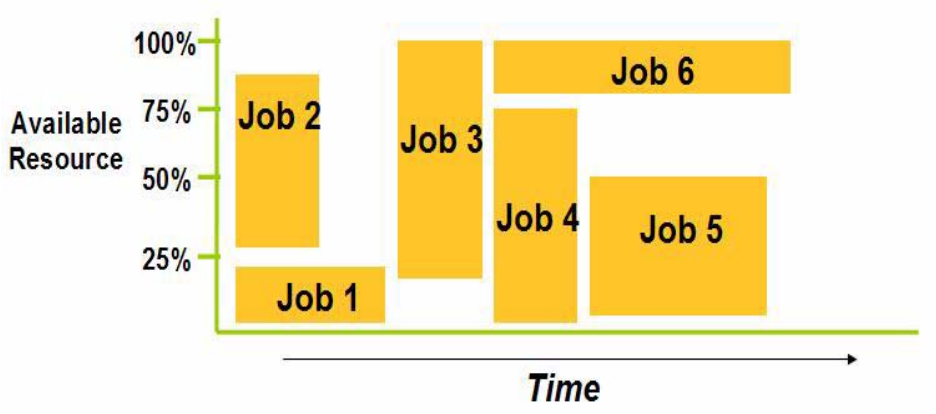

Figure 14-1 Resource requirements per IIS Job.

From Figure 14-1, the following is offered;

- From Figure 14-1, the Jobs are named/numbered, 1 through 6, and would run in that order if allowed.
- Job-1 requires 25% or less of the available resource, and so the working resource manager (RM) generates an APT_CONFIG_FILE for Job-1, that assigns Job-1 to work only on 1 node.
- Job-2 requires 75% or less of the available resource, and so the RM assigns Job-2 to work on the remaining 3 nodes. Job-2 is running at the same time as Job-1, with separate (dedicated) memory, process and disk resources. With proper configuration and tuning of network resources, Job-1 and Job-2 can not tell each other is running.
- Even though Job-2 finishes before Job-1 (based on its required workload), Job-3 is queued because it is configured to wait for all (more than the available 75%, 100% - 25%) available resource on the grid. This condition is configurable system wide, or Job by Job.

Note: The thought is Job-3 will run faster by waiting for all available resource, versus running with less resource, and then overflowing its resource need; having to do things like use disk drives for virtual (extended) memory.

- After Job-3 completes, Job-4 requests 75% of resource and its granted same.
- Although Job-5 is next, Job-5 requires 50% of the available resource when only 25% is available. The RM holds/gates Job-5 and runs job-6 instead, because Job-6 only requires 25% of the available resource.

Again, how to handle collisions and gating is configurable system wide (default), or on a Job by Job basis, and is the responsibility of the RM.

Note: The key point is, a resource manager type of software is configured to observe and manage the IIS Compute nodes, and automatically generate an APT_CONFIG_FILE for each Job. It is the APT_CONFIG_FILE which determines resource availability for any given IIS Job.

If you don't have a resource manager type of software (there are open source offerings of this type of software), you'd have two choices;

- Give every Job access to run on all or a subset of Compute nodes. (No management of nodes, just let things fall where they may.)
- Write your own resource manager type of software.

Generating the APT_CONFIG_FILE above is easy. There would be one master (full) version of this file that grants all resource (access to all nodes), and then subsets of that file; listing just subsets of nodes and resource.

Given our 4 (count) Compute nodes above, we'd need 4! (4 factorial) versions of the APT_CONFIG_FILE; one for all nodes, one for just Node-1, just Node-2, just Node-1 and Node-2, and so on.

Rather than edit those files, we'd write a smallish script to generate same.

Sample IIS grid configuration

Figure 14-2 displays a sample IBM Information Server (IIS) install using a grid configuration. 4 IIS Compute nodes are displayed, in addition to a Metadata repository node and a Java/J2EE compliant Application Server node.

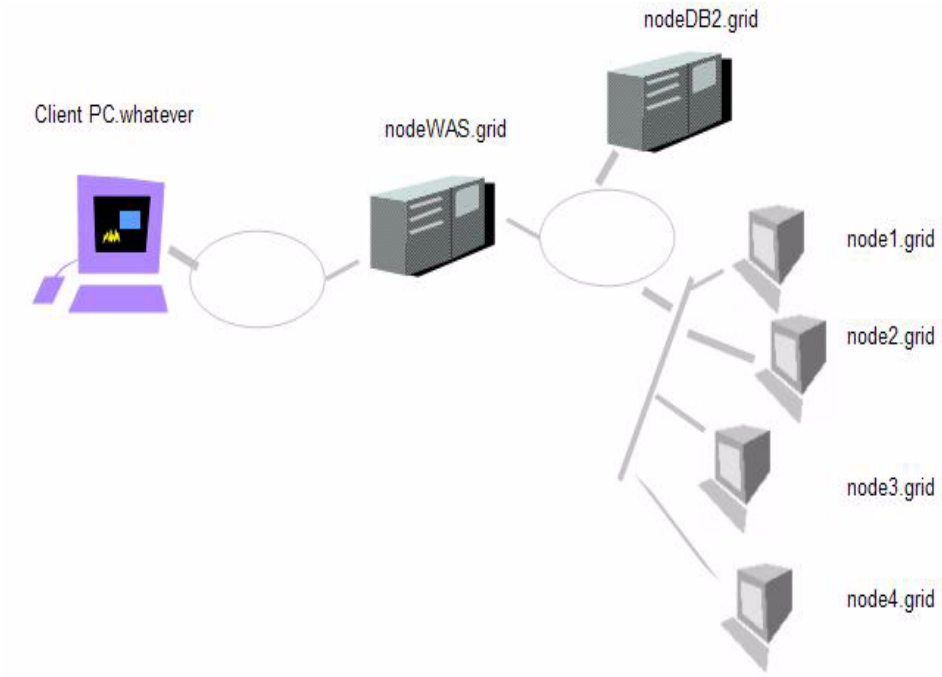

Figure 14-2 Sample IIS grid configuration.

From Figure 14-2, the following is offered;

- A client workstation runs the IBM Information Server (IIS) client programs; programs like the IIS Console, IIS DataStage Designer, and so on.

  This client workstation connects to the Java/J2EE compliant Application Server node (sample name from above, nodeWAS.grid) for user authentication, as well as other support requests.

- nodeWAS.grid hosts the Java/J2EE compliant Application Server. In addition to running core internal componentry of IIS, this is also the node where WISD (IIS WebSphere Information Services Director operations are hosted. E.g., IIS published and hosted Web services.)

  Generally, this is the only node that client programs need access to.

- nodeDB2.grid hosts the metadata repository, basically a SQL database server. While this database server could be Oracle or Microsoft SQL/Server, a free limited use copy of DB2/UDB is supplied.

- node[1-4].grid are the IIS Compute nodes. You could have one node, or hundreds, depending our your licence agreement, and performance needs.

To save time, and sacrifice only a very small amount of performance, you could install the IIS software on just node1.grid, and then cross mount (provide an NFS link) node1.grid's IIS binaries to all remaining Compute nodes.

Installing IIS in a grid configuration, overview

In general, the software installation of the above system (Figure 14-2) would be as follows;

(Version 8.01, and addition of FixPak 1a. Windows XP/SP1 client, RedHat Advanced Server V4 for all server hosting.)

– Install IIS client programs version 8.01, and then FixPak 1a.

  On an appropriately sized Windows PC, this work could take 20 to 40 minutes.

  This work may be done before or after, or concurrently, to any of the processes listed below.

– Install the Metadata repository node-

  • 32 bit DB2/UDB; 14 minutes to install, 3 minutes for the FixPak. (The DB2 FixPak is tiny.)

  • 64 bit DB2/UDB; have to install DB2 from 64 bit media, then install IIS afterwards, plus then the FixPak, 18 minutes.

  • *This step must be completed before installing the Java/J2EE compliant Application Server below.*

  • Inside the IIS graphical install program, select-

    Metadata repository -> Metadata Repository

    Metadata repository -> WebSphere Information Analyzer

    (The line above assume you've licensed IIS Information Analyzer.)

– Install Java/J2EE compliant Application Server node-

  • 32 bit WAS; 28-38 minutes, including more than one reboot of WAS.

  • 64 bit WAS; have to install WAS from 64 bit media, then install 3 WAS specific FixPaks (RedHat 64 bit specific comment), then install IIS proper; takes close to 55 minutes.

    Then install IIS FixPak 1a.

  • We always complete this step before installing the Compute nodes.

– Installing the Compute nodes (DSEngine)-

As mentioned above, this step only needs to be completed once, for the first Compute node. Then, all remaining Compute nodes may just cross mount (using an NFS link), the necessary binary programs.

Or, use IBM Information Server Blades

The procedures and technologies listed above (installing and configuring a 6 (count) node IBM Information Server system in a grid configuration), can take a full 7 hours to complete; 7 hours if you've done it before and have reasonable skill, and that's just the Information Server piece. Otherwise, that install can take much more than 7 hours. *There are numerous operating system dependencies which we cover in a remaining section, below.*

To avoid all of the above grief, and to allow for a pre-tuned, pre-proven grid install and configuration, IBM offers the IBM Information Server (IIS) Blade system. Both hardware and software, IIS comes pre-installed and pre-optimized on IBM HS21 dual-core x86 blade mount servers. These systems run RedHat Enterprise Linux 4, with 4 to 8 GB of RAM, and up to 12 (count) 76 GB high performance drives.

The Tivoli Load Leveler system (a resource manager) comes pre-installed and pre-configured with the above.

14.2 IIS grid, operating system dependencies, RedHat

As mentioned above, this document outlines how to install IBM Information Server (IIS) using a grid configuration; specifically, on the RedHat Linux Advanced Server version 4 operating system. In addition to operating system specific procedures and conventions that must be followed, this section of this document lists specific debugging or tracing commends we found to be of use.

Note: The examples that follow make use of the IIS grid configured system diagrammed in Figure 14-2.

Operating system settings and defaults

Following Figure 14-2, we are detailing how to install IBM Information Server (IIS) is a 6 (count) node grid configuration. The following is offered;

– All nodes in this install have the very same RedHat Linux Advanced Server operating system install; same install media, same packages installed.

  One node only (nodeWAS.grid) will activate its Domain Name Server subsystem.

– All server IP addresses and related are set to static. Using a class-C sub network, the following IPs are in effect.
  - The Gateway (Router) is 192.168.0.1
  - Subnet mask, 255.255.255.0
  - nodeDB2.grid, 192.168.0.10

    This is the node hosting DB2/UDB and the 2 IIS databases.
  - nodeWAS.grid, 192.168.0.11

    This is the node hosting WAS, and the DNS server.
  - node1.grid, 192.168.0.12

    This is the node we install DSEngine on. All remaining nodes simply cross mount a given directory from this node using NFS.
  - node2.grid, 192.168.0.13
  - node3.grid, 192.168.0.14
  - node4.grid, 192.168.0.15

Making Linux system users

To successfully install and then operate IBM Information Server (IIS), we run the following Linux administration commands to make groups, users, and set passwords. Example as shown in Example 14-1.

Example 14-1 Linux administration command to make groups, users, and set passwords.

```
groupadd -g 400 xmeta

groupadd -g 401 db2iadm1

groupadd -g 402 db2fadm1

groupadd -g 403 dsadm1

groupadd -g 404 wasadmin

groupadd -g 405 iauser

groupadd -g 406 dstage

groupadd -g 408 loadl


useradd -g  400 -u 400 xmeta
```



```
useradd -g  401 -u 401 db2inst1

useradd -g  402 -u 402 db2fenc1

useradd -g  403 -u 403 dasusr1

useradd -g  404 -u 404 wasadmin

useradd -g  405 -u 405 iauser

useradd -g  406 -u 406 dsadm

useradd -g  406 -u 407 iisadmin

useradd -g  408 -u 408 loadl   # The Tivoli Load Leveler Administrator


passwd xmeta

passwd db2inst1

passwd db2fenc1

passwd dasusr1

passwd wasadmin

passwd iauser

passwd dsadm

passwd iisadmin

passwd loadl
```

Notes related to Example 14-1;

- The users iisadmin and dsadm need to belong to the same operating system group.
- After creating all groups, users, and setting passwords, we test logging in as every user on every node. We do this to ensure the accuracy of each created user; are they prompted to change their password upon log in (bad: the IIS install script would not be able to handle that event), do they have a log in shell, and so on.

Configuring a RedHat Linux domain name server (DNS)

A domain name server allows one to locate servers on a network by their server name alone. Editing the Linux/Unix "/etc/hosts" file, or Windows "c:/windows/system32/drivers/etc/hosts" file is not enough. In order to operate an IBM Information Server (IIS) system in a grid configuration, some DNS style entity has to be made available to supply hostname resolution services.

Configuring a Linux DNS server is probably the hardest of the Linux administration topics we discuss. It is the hardest topic, because of a dependency on a hideously cryptic and whitespace sensitive (Linux) DNS configuration file, seen below.

The following Linux packages must be installed in order to successfully configure DNS;

- bind
- bind_utils
- bind_libs
- bind_chroot
- caching_nameserver

On each node, we run the following commands. Example as shown in Example 14-2;

Example 14-2 First set of commands to configure a Linux DNS server.

```
unalias cp 2> /dev/null


cp -f hosts       /etc

cp -f resolv.conf /etc


[ ‘hostname‘ ==  “nodewas.grid” ] && {


  cp -f named.conf          /var/named/chroot/etc

  cp -f grid.zone /var/named/chroot/var/named/

  cp -f 192.168.0.zone      /var/named/chroot/var/named/
```



```
service named start

} || {

sleep 1;

}
```

A code review of Example 14-2 follows;

- The 'unalias' line is largely superfluous; it changes the behavior of the 'cp' command.
- The first two 'cp' lines copy given configuration files to named locations. These files are displayed in Example 14-3 and Example 14-4.
- The line with 'hostname' in it starts a conditional; the remainder of this file will only execute on one node, the nodeWAS.grid node. This node is our DNS server node.
  - The next 3 lines copy more files, detailed below.
  - The 'service' line calls to start the DNS subsystem.

Example 14-3 Contents of '/etc/hosts' file.

```
# Do not remove the following line, or various programs
# that require network functionality will fail.

127.0.0.1       localhost localhost.localdomain

192.168.0.10    nodedb2   nodedb2.grid
192.168.0.11    nodewas   nodewas.grid
192.168.0.12    node1     node1.grid
192.168.0.13    node2     node2.grid
192.168.0.14    node3     node3.grid
```



```
192.168.0.15     node4     node4.grid
```

Example 14-4 Contents of '/etc/resolv.conf'.

```
search grid
nameserver 192.168.0.11
```

The DNS subsystem is dependent on 3 ASCII text configuration files. These files are;

- /var/named/chroot/etc/named.conf
- /var/named/chroot/var/named/grid.zone

  Where "grid" is the name of our domain; we named our IIS nodes, "nodeDB2.grid, nodeWAS.grid, node1.grid, etcetera.
- /var/named/chroot/var/named/192.168.0.zone

  Where 192.168.0 is the prefix to our subnet.

Examples of these 3 named files follow.

Example 14-5 Contents of '/var/named/chroot/etc/named.conf'.

```
//
// named.conf for Red Hat caching-nameserver
//

acl grid { 192.168.0.10/15;  127.0/8; };

options {
directory "/var/named";
dump-file "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";

        allow-query { grid; };
```



```
/*
 * If there is a firewall between you and nameservers you want
 * to talk to, you might need to uncomment the query-source
 * directive below.  Previous versions of BIND always asked
 * questions using port 53, but BIND 8.1 uses an unprivileged
 * port by default.
 */
 // query-source address * port 53;
};

//
// a caching only nameserver config
//
controls {
inet 127.0.0.1 allow { localhost; } keys { rndckey; };
};

zone "." IN {
type hint;
file "named.ca";
};

zone "localdomain" IN {
type master;
file "localdomain.zone";
allow-update { none; };
};

zone "localhost" IN {
type master;
file "localhost.zone";
allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
type master;
```



```
file "named.local";
allow-update { none; };
};

zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa" IN {
        type master;
file "named.ip6.local";
allow-update { none; };
};

zone "255.in-addr.arpa" IN {
type master;
file "named.broadcast";
allow-update { none; };
};

zone "0.in-addr.arpa" IN {
type master;
file "named.zero";
allow-update { none; };
};

zone "grid" IN {
        type master;
        file "grid.zone";
};

zone "0.168.192.in-addr.arpa" IN {
        type master;
        file "192.168.0.zone";
};

include "/etc/rndc.key";
```

A code review of Example 14-5 follows.

- Most of the file contents in Example 14-5 are default, and remain unchanged by us.
- The first line we have to change starts with the word, 'acl'.

  'ACL' is short for access control list, in our case, IP addresses in one of 2 ranges;

  192.168.0.10 through 15

  127.0.0.0 through 8

  Which in our smallish test system as displayed, basically equals anything connected to our router, anything on our smallish Class-C sub network.
- The 'allow-query' line is added as displayed.
- All but 2 new paragraphs in this file are default, as supplied by Linux, and left unchanged.

  The only 2 paragraphs we add are near the end of the file, and begin with the word 'zone'.

  zone "grid.zone" ..

  zone "0.168.192" ..

  These two lines/paragraphs specify further configuration files which we review next, in Example 14-6 and Example 14-7.

Example 14-6 Contents of '/var/named/chroot/var/named.grid.zone'.

```
$TTL 1D

grid.   IN SOA nodewas.grid. foo.bar.tld. (
                20080315   ; serial
                2H         ; refresh slaves
                5M         ; retry
                1W         ; expire
                1M         ; negative TTL
                )

@               IN NS  nodewas.grid.
```



```
nodewas          IN   A  192.168.0.11   ;
nodedb2          IN   A  192.168.0.10   ;
node1            IN   A  192.168.0.12   ;
node2            IN   A  192.168.0.13   ;
node3            IN   A  192.168.0.14   ;
node4            IN   A  192.168.0.15   ;
```

Example 14-7 Contents of '/var/named/chroot/var/named/192.168.0.zone'.

```
$TTL 1D

@           IN SOA nodewas.grid foo.bar.tld. (
                20080315   ; serial
                2H         ; refresh slaves
                5M         ; retry
                1W         ; expire
                1M         ; negative TTL
                )

              IN NS  nodewas.grid.

11            IN PTR nodewas.grid.
```



```
10      IN PTR nodedb2.grid.

12      IN PTR node1.grid.

13      IN PTR node2.grid.

14      IN PTR node3.grid.

15      IN PTR node4.grid.
```

Minus a small amount of surrounding syntax, the code in Example 14-6 and Example 14-7 are largely used to specify our known server host names.

A command named, 'named_checkzone', may be used to check any entries in the configuration files used by the DNS subsystem. If 'named_checkzone' reports zero errors, generally you are good to go.

"dig" is the Domain (name server) Information Groper tool. 'dig' is similar to ping, except that dig specifically routes its request through the active DNS server subsystem. Basically, if you can dig every hostname in our grid from every other hostname, then you are also certain you have configured DNS properly. 'dig' every hostname and every partial hostname. For example;

- dig nodewas.grid
- dig nodewas

The DNS subsystem is started, stopped, etcetera, via the following commands;

- service named start
- service named stop
- service named restart # equal to a stop and start
- service named status

Configuring a RedHat Linux NFS subsystem

NFS is an acronym for network mounted file system; in effect, one software server is making available all or part of a given hard disk to other servers or client workstations. We can optionally make use of NFS mounted file systems when using IBM Information Server (IIS) in a grid configuration. With NFS, we need only install the DSEngine component of IIS on one Compute node in our grid, and then make this software available to all remaining Compute nodes through an NFS mount.

Note: Comparing NFS to DNS-

In a given domain (an organized collection of computers), only one server or related entity may provide DNS type services; there is always one DNS primary server, and then optional backup DNS servers.

Unlike DNS, zero of more software servers inside a domain may provide NFS type services. In our example, only node1.grid needs to supply NFS mounts, and so only that server is configured to host NFS. Other nodes in our grid are configured to make use of the node1.grid supplied NFS mount, to be NFS clients.

NFS is a lot easier to configure than DNS.

On each node, we run the following commands. Example as shown in Example 14-8.

Example 14-8 First set of command to configure NFS.

```
unalias cp 2> /dev/null

mkdir /tmp/nfs
chmod 777 /tmp/nfs

mkdir /opt/IBM
chmod 777 /opt/IBM

mkdir /opt/IBM/InformationServer
chmod 777 /opt/IBM/InformationServer

[ 'hostname' ==  "nodewas.benchmark.grid" ] && {

  cp -f exports     /etc
  cp -f hosts.allow /etc

  rpc.mountd
  rpc.nfsd
```



```
} || {

mount nodewas.benchmark.grid:/tmp/nfs                /tmp/nfs

[ 'hostname' ==  "nodedb2.benchmark.grid" ] && {
  sleep 1
} || {
  mount nodewas.benchmark.grid:/opt/IBM/InformationServer \
    /opt/IBM/InformationServer
}



}
```

A code review of Example 14-8 follows;

- The 'unalias' line is largely superfluous; it changes the behavior of the 'cp' command.
- Three sets of commands follow-
  - "mkdir" is the Unix/Linux command to make a new directory.

    We make a directory named, "/tmp/nfs/" just as a place were all servers can share (read and write) files. nodeWAS.grid will host this directory, all other servers will cross mount (NFS mount) this directory.

    We make a directory named "/opt/IBM/InformationServer/" on all nodes. nodeWAS.grid will host this directory, only the Compute nodes will cross mount this directory.

Note: We've been saying we would use node1.grid as the server where we installed the first (and only) Compute node. That is standard and possibly best practice.

Since we needed/wanted the nodeWAS.grid server to host files anyway (the /tmp/nfs/ files), we opted to save time and also have nodeWAS.grid host the Compute node files (/opt/IBM/InformationServer/).

  - "chmod" is the Unix/Linux command to set file permissions.
- The line with 'hostname' in it starts a conditional; the next 4 lines only run on nodeWAS.grid.

- The next 2 lines do file copies; these files are detailed below.
- The 2 "rpc." command enable the NFS server system. "nfsd" is the actual NFS server sub system daemon.

– The next line with the keyword "mount", is run by all servers wishing to have access to the "/tmp/nfs" directory.

- The next "hostname" line forms a conditional; basically this line excludes the nodeDB2.grid server from mounting nodeWAS.grid's /opt/IBM/InformationServer/ directory.
- And then the last "mount" command is similar to the first, only this one mounts a different directory.

Two dependencies from above are ASCII text control files that are referenced by the NFS server subsystem, /etc/exports, and /etc/hosts.allow. /etc/exports Is displayed in Example 14-9.

Example 14-9 Sample contents of /etc/exports.

```
/tmp/nfs            nodedb2.grid(rw,no_root_squash,sync)
                    node1.grid(rw,no_root_squash,sync)
                    node2.grid(rw,no_root_squash,sync)
                    node3.grid(rw,no_root_squash,sync)
                    node4.grid(rw,no_root_squash,sync)

/opt/IBM/InformationServer  nodedb2.grid(rw,no_root_squash,sync)
                    node1.grid(rw,no_root_squash,sync)
                    node2.grid(rw,no_root_squash,sync)
                    node3.grid(rw,no_root_squash,sync)
                    node4.grid(rw,no_root_squash,sync)
```

A code review of Example 14-9 follows;

– From Example 14-9, there are only two physical lines; lines with a carriage return. Because these lines are so wide/long, they appear to be multiple lines.

  These two entries must be on one line each, or NFS will not do what we wish it to do.

– Each of the two lines is similar, so we will explain only the first;

- This file only has to appear on the NFS server.

- The first word (by example our word is, /tmp/nfs), lists the directory we are willing to share.
- Next are blocks indicating which remote host may access the given directory. This block begins with the remote hostname, followed by a variable number of properties surrounded in parenthesis "()".
- Our properties allow read and write access "rw", and allow someone to access super user files "root", and affect how file caching is managed.

the final ASCII text file referenced by the NFS server subsystem in /etc/host.allow, displayed in Example 14-10.

Example 14-10 Sample contents of /tmp/hosts.allow.

```
#
# hosts.allowThis file describes the names of the hosts which are
#allowed to use the local INET services, as decided
#by the '/usr/sbin/tcpd' server.
#


nodedb2.benchmark.grid
nodewas.benchmark.grid
node1.benchmark.grid
node2.benchmark.grid
node3.benchmark.grid
node4.benchmark.grid
```

A code review of Example 14-10 follows;

- This file only needs to be present on the NFS server.
- This file lists which remote servers are "trusted", and may access the NFS server without supplying credentials.

A summary of the NFS server subsystem follows;

- It is the two "rpc" commands in Example 14-8, executed on the NFS server, that make NFS work.
- The only command run on an NFS client is, "mount".
- If you get an error on the NFS client similar to,

  "error: program not registered"

  This indicates that the rpc.nfsd daemon is not running on the NFS server.

- The "rpcinfo -p" command will inform you that current state of the NFS server subsystem.
- The "exportfs -ra" command causes the /tmp/exports file to be re-read dynamically.

Configuring RedHat Linux SSH (secure shell)

SSH is an acronym for the Unix/Linux secure shell; the ability to run commands on remote servers, using an encrypted transport line. Previously, when we were configuring DNS and then NFS, these were Linux server subsystems that may be used by all users. SSH isn't a subsystem per se, its merely a program capability. With SSH, we need to configure this user by user, and server by server.

The only user we need to set SSH for is dsadm. dsadm is the IBM Information Server (IIS) user who administers the component of IIS we are grid enabling.

On each node, we run the following command. Examples as shown in Example 14-11.

Example 14-11 First set of command to configure SSH.

```
unalias rm 2> /dev/null

[ 'hostname' ==  "nodewas.grid" ] && {

  ssh-keygen -trsa

  cp $HOME/.ssh/id_rsa.pub       $HOME/.ssh/authorized_keys2
  cp $HOME/.ssh/authorized_keys2  /tmp/nfs

  } || {

  mkdir $HOME/.ssh
  chmod 700 $HOME/.ssh
  cp /tmp/nfs/authorized_keys2 /$HOME/.ssh



  }
```

A code review of Example 14-11 follows;

- *These commands should be run as user, dsadm. (Or by any user wanting this capability, but we must set them for dsadm.)*
- The 'unalias' line is largely superfluous; it changes the behavior of the 'rm' command.
- The line with 'hostname' in it begins a conditional; the next 3 lines execute only on nodeWAS.grid.
  - "ssh-keygen" generates a new encryption key that allows for this bidirectional encrypted command communication.
  - The next 2 lines do file copies.

    Here we make use of the previously configured /tmp/nfs/ directory; in effect, this allows us to copy these encryption keys to other servers.
- The line that begins with "mkdir", runs on all other servers.

  Here we set file permissions and do file copies.

The following is offered;

- To test the accuracy of the above work, test a remote log in attempt to every server in the grid from nodeWAS.grid. Also test a loop back, a remote log in to nodeWAS.grid from nodeWAS.grid. Example as follows;

  ```
  # From nodeWAS.gri, as user, dsadm

  ssh node1.grid
  ```

  After your initial attempt, every subsequent attempt should grant a log in shell on the remote box without any prompts.
- Upon failure of the above, re-run the steps from Example 14-11.

Grid enabling IBM Information Server (IIS)

At this point, we have prepared the operating system to allow IBM Information Server (IIS) to operate in a grid configuration; we have configured Linux DNS, NFS, and SSH. To prepare the IIS system proper, the following is offered;

- Assuming that IBM Information Server is installed in,

  /opt/IBM/InformationServer/

  the directory that must be made available to all Compute nodes is,

  /opt/IBM/InformationServer/Server/

  If you followed the NFS server subsystem configuration steps above, this step is already complete.

Note: The instructions in this document stated that standard practice would be to install the DSEngine component on node1.grid alone, and then allow for an NFS mount to all remaining Compute nodes. *This is the common practice.* Then, we went ahead and installed DSEngine on nodeWAS.grid, and mounted from all Compute nodes. *Either way works.*

About the only point we would like to make at this time is-

Install IBM Information Server (IIS) without the NFS mounts being in place, then restore the NFS mounts. If you install with the NFS mounts in place, you will likely saturate your network subsystem; the install may fail due to operating system limits, *or at the very least, the install will take much longer than you would prefer.*

– Again assuming default installation directories, edit the following files,

  /opt/IBM/InformationServer/Server/PXEngine/etc/remsh.example

  and, remcp.example

  - Save these files without the ".example" in the file name.
  - In each file, change the string "rsh" to "ssh", and "rcp" to "scp".
  - chmod these files to 755.

– Edit the

  /opt/IBM/InformationServer/Server/DSEngine/dsenv

  file. Add to the bottom of this file,

  export PATH=$APT_ORCHHOME/bin:$PATH

  export LD_LIBRARY_PATH=$APT_ORCHHOME/lib:$LD_LIBRARY_PATH

  Afterwards, "source" this file; that is, run a ". {filename}", where "filename" is "dsenv".

– As a test of your IIS grid installation, create an APT_CONFIG_FILE that uses all of your Compute nodes.

  - The default APT_CONFIG_FILE is located in,

    /opt/IBM/InformationServer/Server/Configurations/default.apt

    Copy this file to a new file, example; My4Node.apt

    Edit the above file to include multiple Compute nodes.
  - Set the environment variable APT_CONFIG_FILE, to equal the absolute pathname to the above file. Example,


```
export APT_CONFIG_FILE=/opt/IBM/InformationServer/Server/Configurations/My4Node.apt
```

Note: To test the APT_CONFIG_FILE (and pretty much everything else), run the following,

```
cd /opt/IBM/InformationServer/Server/PXEngine/bin
export PATH=$PATH:‘pwd‘
osh “hostname [par]”
```

The above command should run the Unix/Linux “hostname” command on every node listed in your APT_CONFIG_FILE. If this works without error, you are ready for IBM Information Server grid computing.

Further things to be aware of

Listed below are random comments that may be of value;

– As we have configured things in this document, the DNS and SSH changes we made are persistent across Linux server reboots.

  The NFS steps we completed are not persistent across Linux Server reboots. Upon every Linux server startup, you will need to;

  • Run the “rpc” commands from Example 14-8 on the NFS server.
  • Run the “mount” command from Example 14-8 on every NFS client.

– The Linux network configuration files are viewable as ASCII text files, and are located under,

  /etc/sysconfig/networking/devices/

– To debug the availability of IP Addresses and port numbers, use,

  telnet *IP-Address Port-number*

  nslookup *hostname*

  dig *hostname*

  The Unix/Linux “top” command is useful to see what programs are running; “top” is similar to the Windows Task Manager in this regards.

– To see what DB2/UDB databases are present, find the “db2” command on the nodeDB2.grid server, and run a command equal to,

  db2 list database directory

  After a successful install of IBM Information server on the nodeDB2.grid node, you should see the XMeta repository database (default name), and the optional (if licensed) Information Analyzer database.

- The default URL to the WebSphere Application Server (WAS) Administration Console is,

  https://nodeWAS.grid:9043/ibm/console/logon.jsp

  Where nodeWAS.grid is the server name hosting WAS.

  Follow the left panel menu go to, Applications -> Enterprise Applications

  to confirm or deny if the Java/J2EE component of IBM Information Server is installed and running.

- The default URL to the IBM Information Server Administration Web Console is,

  http://nodeWAS.grid:9080/loginForm.jsp

  Where nodeWAS.grid is the correct server name.

- The Windows equivalent to the Unix/Linux /etc/hosts file is found at,

  c:/windows/system32/drivers/etc/hosts

  There are other hosts files present on a Windows box, but the above file is the critical one.

14.3 Summation

In this document, we reviewed or created:

We overviewed installing IBM Information Server (IIS) in a grid configuration, with special attention given to preparation of the RedHat Linux Advanced Server operating system.

We covered the Linux server subsystems; DNS, NFS, and (SSH).

Persons who help this month.

Peter '24x7' Marshall.

Additional resources:

WikiPedia.com article on 'computer clusters'-

http://en.wikipedia.org/wiki/Computer_cluster

WikiPedia.com article on 'grid computing'-

http://en.wikipedia.org/wiki/Grid_computing

IBM.com article on 'IBM Information Server Blade'-

http://www-306.ibm.com/software/data/integration/info_server/blade/

IBM Tivoli Load Leveler documentation home-

http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.loadl.doc/loadl34/am2ug30509.html

Setting up a Linux NFS server, How to

http://www.linux.org/docs/ldp/howto/NFS-HOWTO/server.html

Setting up a Linux DNS server, How to

http://www.redhat.com/magazine/025nov06/features/dns/

http://www.redhat.com/magazine/026dec06/features/dns/

Setting up Linux SSH/RSH, How to

http://www.seriss.com/rush-current/misc/rsh-config.html

Legal statements:

Special attributions:

The listed trademarks of the following companies require marking and attribution:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Microsoft trademark guidelines

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intel trademark information

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

Other company, product, or service names may be trademarks or service marks of others.

