ORACLE

LINUX

An Oracle White Paper
February 2014

OCFS2 Best Practices Guide

ORACLE

Introduction

OCFS2 is a high performance, high availability, POSIX compliant general-purpose file system for Linux. It is a versatile clustered file system that can be used with applications that are non-cluster aware and cluster aware. OCFS2 is fully integrated into the mainline Linux kernel as of 2006 and is available for most Linux distributions. In addition, OCFS2 is embedded in Oracle VM and can be used with Oracle products such as the Oracle Database and Oracle RAC solutions.

OCFS2 is a useful clustered file system that has many general purpose uses beyond Oracle workloads. Utilizing shared storage, it can be used for many general computing tasks where shared clustered storage is required. Its high performance and clustering capabilities set it apart from many other network based storage technologies. Cluster aware applications can take advantage of cache-coherent parallel I/O from more than one node at a time to provide better performance and scalability. Uses for OCFS2 are virtually unlimited, but some examples are a shared file system for web applications, database data files, and storing virtual machine images for different types of open source hypervisors.

OCFS2 is completely architecture and endian neutral and supports file system cluster sizes from 4KB to 1MB and block sizes from 512 bytes to 4KB. It also supports a number of different features such as POSIX ACL’s, Indexed Directories, REFLINK, Metadata Checksums, Extended Attributes, Allocation Reservation and User and Group Quotas

OCFS2 Overview

Clustering is the concept of connecting multiple servers together to act as a single system, providing additional resources for workloads and failover capabilities for high availability. Clustered systems frequently use a heartbeat to maintain services within the cluster. A heartbeat provides information–such as membership and resource information–to the nodes within the cluster and can be used to alert nodes of potential failures. Clustered systems typically are designed to share network and disk resources and communicate with one another using a node heartbeat to maintain services within the cluster. Clustered systems often contain code that will detect a non-responsive node and remove it from the cluster to preserve the ability to continue its services and avoid failure or data corruption.

OCFS2 utilizes both a network and disk based heartbeat to determine if nodes inside the cluster are available. If a node fails to respond, the other nodes in the cluster are able to continue operation by removing the node from the cluster. The following diagram shows the functional overview of a three node OCFS2 cluster. The private network interconnect is shown in red and the shared storage interconnect is shown in gray. The shared storage in this example could be Fibre Channel, iSCSI or another type of storage that allows multiple nodes to attach to a storage device. Each node has access to the shared storage and is able to communicate with other nodes using the private network interconnect. In the event that a node is unable to communicate via the network or the shared storage, the node will be fenced, which means it will be evicted from the cluster and the kernel panics in order to return to an operational state.

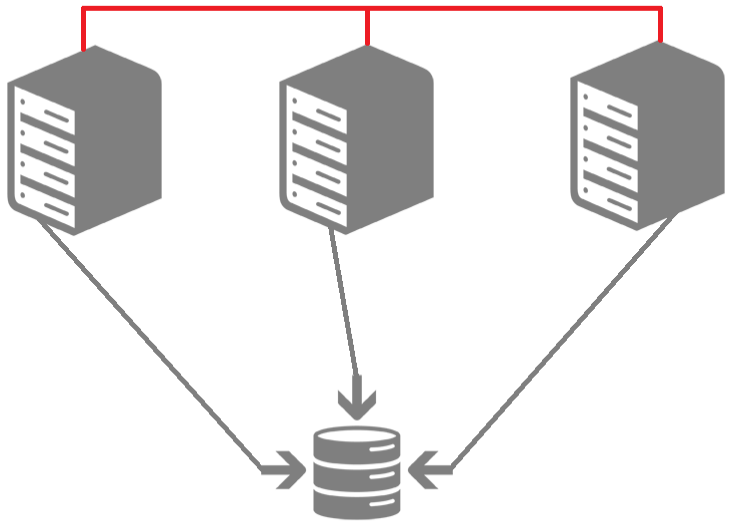

Figure 1. Functional Overview of a Three Node OCFS2 Cluster

OCFS2's disk-based heartbeat works in the following way. Each node writes to its block in the system heartbeat file every two seconds. Nodes will communicate to each node in the cluster with a TCP keepalive packet on port 7777 every two seconds to determine if the node is alive. The nodes in the cluster will read the blocks written by other nodes on disk every 2 seconds and if the timestamps on

the blocks of the other nodes are changing, those nodes are considered alive. If there is no response, the time-out timer starts and will mark the node as not responding when the timeout value is reached.

OCFS2 uses the network in addition to a separate disk based heartbeat to determine if nodes are alive and for communication of file locks and file system access. Once a node stops responding on the network and all the timers have expired, the quorum call is performed. When nodes lose connection via the network they are in split brain mode as they communicate via the disk heartbeat but not on the network heartbeat. This situation results in a temporary hang. To recover, some of the disconnected nodes have to be evicted. This decision is made by the quorum code. The quorum code acts with the following logic: if there are 2 nodes in a cluster and they lose connection with each other, the node with the lowest node number survives. In case of more than 2 nodes, a node survives if it can communicate with more than half the nodes in the cluster. If the node communicates with half the nodes, a node will survive if it can communicate to the lowest node number. During a network timeout situation the nodes can still determine node functionality and perform quorum calls using the disk heartbeat.

Best practices for setup and installation vary depending on the intended purpose. Specific installation guides can be found in the additional resources section of this document. For the purpose of this guide, examples will cover some of the basic configuration and best practices for installing and configuring OCFS2 for general file system usage and differences when using OCFS2 with other products, such as Oracle RAC and Oracle VM. It is recommended that readers refer to the OCFS2 Users Guide for additional details not covered in this document.

OCFS2 as a General File System

Although the amount of traffic generated by OCFS2's network heartbeat process is low, the heartbeats are critical to operation thus it is recommended that a dedicated network interface be used for the network interconnect. Optimally this dedicated network interface should be on a private network or VLAN that only the nodes in the cluster have access to. This will allow the cluster's network traffic to have better access and less latency without competing with other network traffic. For example, a node that was accessing an OCFS2 file system and was used as a file server would have the file transfer competing with the cluster's network traffic. This latency could cause the node to be removed from the cluster if it did not respond to the keep alive packets in the allotted time out values. Dedicated network interfaces also make the process of troubleshooting easier as you can look at the traffic on the dedicated interface without other traffic being seen in your packet capture.

It is important that all nodes that will access the file system have read and write access to the storage being used to store the file system. For SAN and iSCSI based storage access, multi-pathed disk volumes are highly recommended for redundancy. This will protect against the individual nodes losing access to the storage in the event of a path failure. Optimally this should be done on redundant switches in the event of a switch failure causing the path failure. Complete configuration of `/etc/multipath.conf` should be completed before the configuration of the file system. Further resources for configuring `multipathd` can be found in the additional resources section of this document.

When designing your OCFS2 file system it is important to keep in mind the number of nodes that you select for your cluster. Two node clusters are not optimal because of the possibility of both nodes self fencing and causing a completely offline file system. When there is a connectivity failure, there's a 50/50 chance that the node that is still alive has a higher node number. This can cause both systems to become unavailable. It is also important to keep in mind the possibility for expansion in the future for your cluster if you decide to add additional nodes. When formatting the file system, it is important to consider additional node slots for expansion as when node slots are added after the fact they can cause performance related issues with the file system at a later date. Additional node slots do consume some additional disk space but have no performance impact on the file system.

It is important to note with later releases of OCFS2 the graphical `ocfs2console` utility has been removed. Configuration of the `/etc/ocfs2/cluster.conf` file can be done manually but the recommended option is using the `o2cb` cluster registration utility for the `o2cb` cluster stack. The use of the `o2cb` cluster registration utility is preferred to avoid mistakes in editing the `/etc/ocfs2/cluster.conf` file as formatting errors can cause problems. The following is an example of a working `/etc/ocfs2/cluster.conf` file for a 3 node OCFS2 cluster:

```
node:
      name = ocfs2-1
      cluster = ocfs2demo
      number = 1
      ip_address = 10.0.0.1
      ip_port = 7777

node:
      name = ocfs2-2
      cluster = ocfs2demo
      number = 2
      ip_address = 10.0.0.2
      ip_port = 7777

node:
      name = ocfs2-3
      cluster = ocfs2demo
      number = 3
      ip_address = 10.0.0.3
      ip_port = 7777

cluster:
      name = ocfs2demo
      heartbeat_mode = local
      node_count = 3
```

Here are the configuration options for the `/etc/ocfs2/cluster.conf` file:

`node`

This section defines the configuration for a node in the cluster. This is one of the systems in the cluster.

`Name`

This defines the hostname of the system.

`Cluster`

This defines the cluster that the system is a member of.

`Number`

This defines the node number of the system. Each system needs a unique node number. This is used for qurorum.

`ip_address`

This defines the IP address of the system in the cluster.

`ip_port`

This defines the port number that OCFS2 uses for the network heartbeat. Default is 7777.

`cluster`

This section defines the configuration of the cluster.

`Name`

This is the name of the OCFS2 cluster.

`heartbeat_mode`

This defines the type of heartbeat being used in the cluster configuration.

`node_count`

This defines the total number of nodes in the cluster.

The following are the commands you would use with the `o2cb` cluster registration utility to build your configuration. These commands will populate the contents of the `/etc/ocfs2/cluster.conf` file on the local system:

```
o2cb add-cluster ocfs2demo
o2cb add-node --ip 10.0.0.1 --port 7777 --number 1 ocfs2demo ocfs2-1
o2cb add-node --ip 10.0.0.2 --port 7777 --number 2 ocfs2demo ocfs2-2
o2cb add-node --ip 10.0.0.3 --port 7777 --number 3 ocfs2demo ocfs2-3
o2cb register-cluster ocfs2demo
o2cb start-heartbeat ocfs2demo
```

Once the configuration is completed, it needs to be populated onto each node in the cluster. The `/etc/ocfs2/cluster.conf` file must be copied to each node in the cluster and registered with the `o2cb` cluster service or the same `o2cb` cluster commands can be run on each node on the cluster. Copying the configuration manually ensures that each node has the same version of the file. Running the commands on each node has the advantage that it could be scripted and performed remotely via a

shell script running over `ssh`. Both methods are valid and have their advantages depending on your environment and preference, but running the commands on each node is the best of the two options.

For environments with multiple OCFS2 clusters administrators should consider the global heartbeat feature. With multiple clusters the disk and network heartbeats can consume significant network and disk I/O resources. Global heartbeat reduces the overall network and disk I/O consumed with maintaining the individual clusters.

When formatting an OCFS2 volume, the utility `mkfs.ocfs2` has a number of options depending on the use case and requirements you have for the file system. For a general purpose file system a 4KB block size and cluster size is recommended. It is also recommended that OCFS2 volumes are partitioned and labeled for ease of use and protection against unintended use. The `mkfs.ocfs2` utility will protect users from accidently overwriting an OCFS2 volume but it will not protect users from overwriting other types of file systems that may be contained on a disk. It is important to take care and verify disks before formatting to avoid overwriting essential data. The `mkfs.ocfs2` utility will set a number of defaults if left unspecified during the formatting process. All of these options can be modified later using the `tunefs.ocfs2`. When you create the file system, it is important to consider the number of node slots you will need now and also in the future. This controls the number of nodes that can concurrently mount a volume. While you can add node slots in the future, there is a performance impact that may occur if they are added later due to the possibility of the slots being added at the far edge of the disk platter. The `-T` option in `mkfs.ocfs2` also allows you to specify the file system type and has the options mail, datafiles and vmstore to configure your file system for these types of files.

Below is an example of an OCFS2 file system being created for a previously created cluster. All the file system features have been selected manually and there are 16 node slots to allow future expansion of the cluster:

```
mkfs.ocfs2 -L ocfs2demo --cluster-name=ocfs2demo --fs-feature-
level=max-features -N 16
```

It is important to note that creating OCFS2 volumes on logical volumes (LVM) is not supported. This is due to the fact that logical volumes are not cluster aware and corruption of the OCFS2 file system may occur.

It is recommended that OCFS2 file systems be mounted via `/etc/fstab` to ensure file system mounting through reboots. In order to accomplish this you must first ensure the `o2cb` driver loads on boot and is configured to start your cluster. The command `service o2cb configure` will bring you into a menu driven configuration utility for the `o2cb` driver. Current configuration values will be shown during configuration and the user will have the option of keeping the current configuration or changing the configuration during this process. Here is an example of the command being run on the example cluster and not changing the configuration:

```
[root@ocfs2-1 ~]# service o2cb configure
Configuring the O2CB driver.

This will configure the on-boot properties of the O2CB driver.
```

```
The following questions will determine whether the driver is loaded
on boot.  The current values will be shown in brackets ('[]').
Hitting <ENTER> without typing an answer will keep that current
value.  Ctrl-C will abort.

Load O2CB driver on boot (y/n) [y]:
Cluster stack backing O2CB [o2cb]:
Cluster to start on boot (Enter "none" to clear) [ocfs2demo]:
Specify heartbeat dead threshold (>=7) [31]:
Specify network idle timeout in ms (>=5000) [30000]:
Specify network keepalive delay in ms (>=1000) [2000]:
Specify network reconnect delay in ms (>=2000) [2000]:
Writing O2CB configuration: OK
Loading filesystem "configfs": OK
Mounting configfs filesystem at /sys/kernel/config: OK Loading stack
plugin "o2cb": OK Loading filesystem "ocfs2_dlmfs": OK Mounting
ocfs2_dlmfs filesystem at /dlm: OK Setting cluster stack "o2cb": OK
Registering O2CB cluster "ocfs2demo": OK Setting O2CB cluster
timeouts : OK
```

NOTE: Attention should be paid to the heartbeat dead threshold when using multipath devices. The storage vendor should be consulted to determine how long a path failover takes and the timeout should be adjusted accordingly.

Once you have configured the `o2cb` driver to load on boot you can add an entry to `/etc/fstab` for your file system. The following is an example of the `/etc/fstab` on one the nodes in your demo cluster. In this example, the OCFS2 file system is `ocfs2demo` and the device is `/dev/sdb1`:

```
# /etc/fstab
# Created by anaconda on Fri May  3 18:50:48 2013 # # Accessible filesystems,
by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info #
/dev/mapper/vg_ocfs22-lv_root /                       ext4    defaults
       1 1
UUID=6ee45d9d-e41e-479c-89d0-cee5b1620ebb /boot                   ext4
   defaults        1 2
/dev/mapper/vg_ocfs22-lv_swap swap                    swap    defaults
       0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
/dev/sdb1               /ocfs2demo              ocfs2   defaults        0 0
```

You can mount your file system by issuing the command `mount -a` which mounts all the file systems in `/etc/fstab`. Once you have mounted your file systems you should see your file system mounted in the output of `df -h`, in this example the file system is `ocfs2demo`.

```
[root@ocfs2-2 ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_ocfs22-lv_root
                       16G  2.4G   13G  16% /
tmpfs                 1004M     0 1004M   0% /dev/shm
/dev/sda1              485M   98M  362M  22% /boot
/dev/sdb1               12G  1.3G   11G  11% /ocfs2demo
```

An empty OCFS2 volume will show significant used space due to the embedded journals. In the previous example you have 1.3G of space showing used on an empty volume. It is also important to note that due to this, the outputs of commands like `df` and `du` often will not match.

A best practice for mounting OCFS2 volumes is to mount the device via UUID rather than the device name. This is because device names are not always consistent on reboots. Here is an example of how to determine the UUID and how this information would appear in the `/etc/fstab` when using UUID to mount a device. In the following output from the command `blkid` you can see the physical device name and the UUID. For example `/dev/sdb1` has the volume label of `ocfs2demo` and the UUID of `d193a367-a1b6-4e95-82b6-2efdd98dd2fd`:

```
[root@ocfs2-1 ~]# blkid
/dev/sda1: UUID="3fbd6ccd-6bc0-4d41-a21d-b5274f5b2238" TYPE="ext4"
/dev/sda2: UUID="zgeNML-VQDc-HMe3-g14R-Fqks-me5P-LCu03R"
TYPE="LVM2_member"
/dev/sdb1: LABEL="ocfs2demo" UUID="d193a367-a1b6-4e95-82b6-
2efdd98dd2fd" TYPE="ocfs2"
/dev/mapper/vg_ocfs21-lv_root: UUID="79aa15e4-50c4-47be-8d3a-
ba5d118c155a" TYPE="ext4"
/dev/mapper/vg_ocfs21-lv_swap: UUID="99bfb486-3c37-47ca-8dc7-
4d46ee377249" TYPE="swap"
```

Here is an example of a `/etc/fstab` file configured to mount an OCFS2 filesystem via UUID rather than device name:

```
# /etc/fstab
# Created by anaconda on Fri May  3 18:49:58 2013
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/vg_ocfs21-lv_root
/                       ext4    defaults        1 1
UUID=3fbd6ccd-6bc0-4d41-a21d-b5274f5b2238
/boot                   ext4    defaults        1 2
/dev/mapper/vg_ocfs21-lv_swap
swap                    swap    defaults        0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
UUID=d193a367-a1b6-4e95-82b6-2efdd98dd2fd     /ocfs2demo    ocfs2  defaults
       0 0
```

To provide access of the file system over the network OCFS2 file systems can be exported via NFS, but it is important to note that only export via NFS version 3 and above is supported. Older clients can connect if the server exports the volumes with no_subtree_check enabled but due to security issues it is not recommended.

OCFS2 and Oracle Real Application Clusters (RAC)

For Oracle RAC, most of the best practices for general purpose OCFS2 files systems apply with some differences. During file system creation Oracle RAC file systems should be created with a cluster size that's no smaller than the database block size. A larger cluster size is possible and can have some advantages depending on the specific needs of the file system. It is also recommended to set the file system type to datafiles to take advantage of the optimization of the journal for large database files. OCFS2 file systems containing the voting diskfile (CRS), data files, redo logs, archive logs, control files and cluster registry (OCR) must be mounted with `datavolume` and `nointr` mount options. The `datavolume` option allows the Oracle process to open these files with the `o_direct` flag and `nointr` insures that file system access is not interrupted. The following is an example of the mount command using `datavolume` and `nointr`:

```
mount –o datavolume,nointr –t ocfs2  /dev/sdb1 /u01/db
```

OCFS2 with Oracle VM

Once again, many of the best practices for general purpose OCFS2 file systems will apply with Oracle VM with a few minor differences. OCFS2 is incorporated into Oracle VM Server 3 as the underlying cluster file system for server pools and storage repositories. It is also used for configuring high availability for virtualized guests in an Oracle VM environment. During file system creation Oracle VM file systems can take advantage of specification of a file system type of `datafiles`. This optimizes

the journal for the large VM images that the file system contains. This can be specified with the `mkfs.ocfs2` command. Here's an example of the command syntax specifying the file system type of `datafiles` with a file system label of `testvolume` on the disk `/dev/sda1`:

```
mkfs.ocfs2 –T datafiles –L "testvolume" /dev/sda1
```

It is important to note that for production systems it is not a best practice to have an OCFS2 volume on a virtual disk inside an Oracle VM repository due to the nature of clustered file systems. For OCFS2 volumes it is best to have physical disk. For additional information on OCFS2 and Oracle VM, please refer to the technical white papers found in the additional resource section of this document.

OCFS2 Troubleshooting

Issues arise in production, no matter how diligent an administrator is in their testing and validation phase. When creating solutions involving clustering technologies and file systems problems may occur. It is important for administrators to have knowledge for troubleshooting these issues in order to determine where the root cause of a problem exists so it can be corrected and avoided in the future.

When node(s) fail there are several things administrators can look at to determine the chain of events in an attempt to find the root cause. One of the first places administrators can look is in `/var/log/messages` on each node in the cluster to gather information regarding an issue or point of failure.

NOTE: In some situations when a node has to self-fence, the default action is often a machine restart and logs may not be written to disk prior to the restart.

Administrators can also look at `/etc/sysconfig/o2cb` and `/etc/ocfs2/cluster.conf` for information about their configuration. The document titled *"Diagnostic Information To Collect for OCFS2 Issues"* found in the additional resources section of this document provides more comprehensive information about this.

The following example shows a 3 node cluster where node 3 experienced a network failure and has been evicted from the cluster. This example shows the syslog entries from Node 1 in the cluster. When there is a network failure, OCFS2 will attempt to re-establish connection, as described earlier in this document. Once the connection is re-established, the node will attempt to rejoin the cluster. The log shows Node 3 was allowed to reconnect to the cluster once it regained network connectivity:

```
May 10 15:41:43 ocfs2-1 kernel: o2net: Connection to node ocfs2-3 (num 3) at
10.0.0.3:7777 has been idle for 30.60 secs, shutting it down.
May 10 15:41:43 ocfs2-1 kernel: o2net: No longer connected to node ocfs2-3
(num 3) at 10.0.0.3:7777
May 10 15:42:13 ocfs2-1 kernel: o2net: No connection established with node 3
after 30.0 seconds, giving up.
May 10 15:42:29 ocfs2-1 o2hbmonitor: mount debugfs at /sys/kernel/debug
May 10 15:42:43 ocfs2-1 kernel: o2net: No connection established with node 3
after 30.0 seconds, giving up.
May 10 15:43:13 ocfs2-1 kernel: o2net: No connection established with node 3
after 30.0 seconds, giving up.
May 10 15:43:43 ocfs2-1 kernel: o2net: No connection established with node 3
after 30.0 seconds, giving up.
May 10 15:43:51 ocfs2-1 kernel: o2cb: o2dlm has evicted node 3 from domain
D193A367A1B64E9582B62EFDD98DD2FD
May 10 15:43:52 ocfs2-1 kernel: o2dlm: Waiting on the recovery of node 3 in
domain D193A367A1B64E9582B62EFDD98DD2FD
May 10 15:43:55 ocfs2-1 kernel: o2dlm: Begin recovery on domain
D193A367A1B64E9582B62EFDD98DD2FD for node 3
May 10 15:43:55 ocfs2-1 kernel: o2dlm: Node 1 (me) is the Recovery Master for
the dead node 3 in domain D193A367A1B64E9582B62EFDD98DD2FD
May 10 15:44:00 ocfs2-1 kernel: o2dlm: End recovery on domain
D193A367A1B64E9582B62EFDD98DD2FD
May 10 15:44:03 ocfs2-1 kernel: ocfs2: Begin replay journal (node 3, slot 2)
on device (8,17)
May 10 15:44:05 ocfs2-1 kernel: ocfs2: End replay journal (node 3, slot 2) on
device (8,17)
May 10 15:44:05 ocfs2-1 kernel: ocfs2: Beginning quota recovery on device
(8,17) for slot 2
May 10 15:44:05 ocfs2-1 kernel: ocfs2: Finishing quota recovery on device
(8,17) for slot 2
May 10 15:46:39 ocfs2-1 kernel: o2net: Accepted connection from node ocfs2-3
(num 3) at 10.0.0.3:7777
May 10 15:46:43 ocfs2-1 kernel: o2dlm: Node 3 joins domain
D193A367A1B64E9582B62EFDD98DD2FD ( 1 2 3 ) 3 nodes
```

Here is the same eviction shown in the syslog from Node 2 in the cluster:

```
May 10 15:41:44 ocfs2-2 kernel: o2net: Connection to node ocfs2-3 (num 3) at
10.0.0.3:7777 has been idle for 30.88 secs, shutting it down.
May 10 15:41:44 ocfs2-2 kernel: o2net: No longer connected to node ocfs2-3
(num 3) at 10.0.0.3:7777
May 10 15:42:14 ocfs2-2 kernel: o2net: No connection established with node 3
after 30.0 seconds, giving up.
May 10 15:42:35 ocfs2-2 o2hbmonitor: mount debugfs at /sys/kernel/debug
May 10 15:42:45 ocfs2-2 kernel: o2net: No connection established with node 3
after 30.0 seconds, giving up.
May 10 15:43:15 ocfs2-2 kernel: o2net: No connection established with node 3
after 30.0 seconds, giving up.
May 10 15:43:45 ocfs2-2 kernel: o2net: No connection established with node 3
after 30.0 seconds, giving up.
May 10 15:43:53 ocfs2-2 kernel: o2cb: o2dlm has evicted node 3 from domain
D193A367A1B64E9582B62EFDD98DD2FD
May 10 15:43:56 ocfs2-2 kernel: o2cb: o2dlm has evicted node 3 from domain
D193A367A1B64E9582B62EFDD98DD2FD
May 10 15:43:56 ocfs2-2 kernel: o2dlm: Begin recovery on domain
D193A367A1B64E9582B62EFDD98DD2FD for node 3
```

```
May 10 15:43:56 ocfs2-2 kernel: o2dlm: Node 1 (he) is the Recovery Master for
the dead node 3 in domain D193A367A1B64E9582B62EFDD98DD2FD
May 10 15:43:56 ocfs2-2 kernel: o2dlm: End recovery on domain
D193A367A1B64E9582B62EFDD98DD2FD
May 10 15:46:39 ocfs2-2 kernel: o2net: Accepted connection from node ocfs2-3
(num 3) at 10.0.0.3:7777
May 10 15:46:43 ocfs2-2 kernel: o2dlm: Node 3 joins domain
D193A367A1B64E9582B62EFDD98DD2FD ( 1 2 3 ) 3 nodes
```

When evaluating the same issue on the node without connectivity, you notice the node rebooted after being evicted from the cluster.

```
May 10 15:41:43 ocfs2-3 kernel: o2net: Connection to node ocfs2-1 (num 1) at
10.0.0.1:7777 has been idle for 30.27 secs, shutting it down.
May 10 15:41:43 ocfs2-3 kernel: o2net: No longer connected to node ocfs2-1
(num 1) at 10.0.0.1:7777
May 10 15:41:43 ocfs2-3 kernel:
(kworker/u:2,26496,0):dlm_send_remote_convert_request:395 ERROR: Error -112
when sending message 504 (key 0x6d3d88d5) to node 1
May 10 15:41:43 ocfs2-3 kernel: o2dlm: Waiting on the death of node 1 in
domain D193A367A1B64E9582B62EFDD98DD2FD
May 10 15:41:44 ocfs2-3 kernel: o2net: Connection to node ocfs2-2 (num 2) at
10.0.0.2:7777 has been idle for 30.70 secs, shutting it down.
May 10 15:41:44 ocfs2-3 kernel: o2net: No longer connected to node ocfs2-2
(num 2) at 10.0.0.2:7777
May 10 15:41:48 ocfs2-3 o2hbmonitor: mount debugfs at /sys/kernel/debug
May 10 15:42:13 ocfs2-3 kernel: o2net: No connection established with node 1
after 30.0 seconds, giving up.
May 10 15:42:14 ocfs2-3 kernel: o2net: No connection established with node 2
after 30.0 seconds, giving up.
May 10 15:42:43 ocfs2-3 kernel: o2net: No connection established with node 1
after 30.0 seconds, giving up.
May 10 15:42:44 ocfs2-3 kernel: o2net: No connection established with node 2
after 30.0 seconds, giving up.
May 10 15:43:17 ocfs2-3 kernel: imklog 5.8.10, log source = /proc/kmsg
started.
May 10 15:43:17 ocfs2-3 rsyslogd: [origin software="rsyslogd"
swVersion="5.8.10" x-pid="1111" x-info="http://www.rsyslog.com"] start
May 10 15:43:17 ocfs2-3 kernel: Initializing cgroup subsys cpuset
May 10 15:43:17 ocfs2-3 kernel: Initializing cgroup subsys cpu
May 10 15:43:17 ocfs2-3 kernel: Linux version 2.6.39-400.21.2.el6uek.x86_64
(mockbuild@ca-build44.us.oracle.com) (gcc version 4.4.7 20120313 (Red Hat
4.4.7-3) (GCC) ) #1 SMP Tue Apr 23 21:52:38 PDT 2013
May 10 15:43:17 ocfs2-3 kernel: Command line: ro root=/dev/mapper/vg_ocfs23-
lv_root rd_NO_LUKS LANG=en_US.UTF-8 rd_NO_MD SYSFONT=latarcyrheb-
sun16  rd_LVM_LV=vg_ocfs23/lv_swap  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM
rd_LVM_LV=vg_ocfs23/lv_root rhgb quiet
May 10 15:43:17 ocfs2-3 kernel: BIOS-provided physical RAM map:
May 10 15:43:17 ocfs2-3 kernel: BIOS-e820: 0000000000000000 -
000000000009fc00 (usable)
May 10 15:43:17 ocfs2-3 kernel: BIOS-e820: 000000000009fc00 -
00000000000a0000 (reserved)
May 10 15:43:17 ocfs2-3 kernel: BIOS-e820: 00000000000f0000 -
0000000000100000 (reserved)
May 10 15:43:17 ocfs2-3 kernel: BIOS-e820: 0000000000100000 -
000000007fff0000 (usable)
```

```
May 10 15:43:17 ocfs2-3 kernel: BIOS-e820: 000000007fff0000 -
0000000080000000 (ACPI data)
May 10 15:43:17 ocfs2-3 kernel: BIOS-e820: 00000000fffc0000 -
0000000100000000 (reserved)
```

(Truncated)

Due to the verbose output of the system rebooting the output was truncated. Here is is the output of Node 3 rejoining the cluster.

```
May 10 15:46:01 ocfs2-3 kernel: OCFS2 Node Manager 1.8.0
May 10 15:46:01 ocfs2-3 kernel: OCFS2 DLM 1.8.0
May 10 15:46:01 ocfs2-3 kernel: ocfs2: Registered cluster interface o2cb
May 10 15:46:01 ocfs2-3 kernel: OCFS2 DLMFS 1.8.0
May 10 15:46:01 ocfs2-3 kernel: OCFS2 User DLM kernel interface loaded
May 10 15:46:01 ocfs2-3 o2cb.init: online ocfs2demo
May 10 15:46:01 ocfs2-3 kernel: o2hb: Heartbeat mode set to local
May 10 15:46:01 ocfs2-3 o2hbmonitor: Starting
May 10 15:46:01 ocfs2-3 o2hbmonitor: mount debugfs at /sys/kernel/debug
May 10 15:46:38 ocfs2-3 kernel: o2net: Connected to node ocfs2-1 (num 1) at
10.0.0.1:7777
May 10 15:46:38 ocfs2-3 kernel: o2net: Connected to node ocfs2-2 (num 2) at
10.0.0.2:7777
May 10 15:46:43 ocfs2-3 kernel: OCFS2 1.8.0
May 10 15:46:43 ocfs2-3 kernel: o2dlm: Joining domain
D193A367A1B64E9582B62EFDD98DD2FD ( 1 2 3 ) 3 nodes
May 10 15:46:43 ocfs2-3 kernel: ocfs2: Mounting device (8,17) on (node 3,
slot 2) with ordered data mode.
```

Troubleshooting Issues in the Cluster

If there are issues with nodes in the cluster, the first step is to identify if there are any factors influencing the behavior. Common issues with clusters can be traced to either misconfiguration or environment issues such as network latency. To isolate issues involving the network, administrators can utilize `tcpdump` to look into the traffic going on the wire for the private interface and verify nodes can communicate with one another. Below is an example of using `tcpdump` to look at the traffic on node 3 of the OCFS2 cluster. The `tcpdump` command uses –i to specify the interface `eth1` and using the –n option prevents DNS lookups in the example. This is optional and was done to make the example more readable by showing the IP address contained in the `/etc/ocfs2/cluster.conf` file.

```
[root@ocfs2-3 ~]# tcpdump -i eth1 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
18:24:59.161673 IP 10.0.0.2.cbt > 10.0.0.3.38167: Flags [P.], seq
2073493988:2073494012, ack 19101096, win 758, options [nop,nop,TS val
6916984 ecr 6908736], length 24
18:24:59.161694 IP 10.0.0.3.38167 > 10.0.0.2.cbt: Flags [.], ack 24, win
1093, options [nop,nop,TS val 6910697 ecr 6916984], length 0
18:24:59.161761 IP 10.0.0.3.38167 > 10.0.0.2.cbt: Flags [P.], seq 1:25, ack
24, win 1093, options [nop,nop,TS val 6910697 ecr 6916984], length 24
18:24:59.168157 IP 10.0.0.3.58796 > 10.0.0.1.cbt: Flags [P.], seq
2277480151:2277480175, ack 3914928161, win 457, options [nop,nop,TS val
6910704 ecr 6917185], length 24
18:24:59.168351 IP 10.0.0.1.cbt > 10.0.0.3.58796: Flags [.], ack 24, win 758,
options [nop,nop,TS val 6919146 ecr 6910704], length 0
18:24:59.168438 IP 10.0.0.1.cbt > 10.0.0.3.58796: Flags [P.], seq 1:25, ack
24, win 758, options [nop,nop,TS val 6919146 ecr 6910704], length 24
18:24:59.190282 IP 10.0.0.1.cbt > 10.0.0.2.56383: Flags [P.], seq
16480484:16480508, ack 2193337195, win 1089, options [nop,nop,TS val
6919168 ecr 6915012], length 24
18:24:59.190496 IP 10.0.0.2.56383 > 10.0.0.1.cbt: Flags [.], ack 24, win 762,
options [nop,nop,TS val 6917013 ecr 6919168], length 0
18:24:59.190497 IP 10.0.0.2.56383 > 10.0.0.1.cbt: Flags [P.], seq 1:25, ack
24, win 762, options [nop,nop,TS val 6917013 ecr 6919168], length 24
18:24:59.201254 IP 10.0.0.2.cbt > 10.0.0.3.38167: Flags [.], ack 25, win 758,
options [nop,nop,TS val 6917024 ecr 6910697], length 0
18:24:59.208141 IP 10.0.0.3.58796 > 10.0.0.1.cbt: Flags [.], ack 25, win 457,
options [nop,nop,TS val 6910744 ecr 6919146], length 0
18:24:59.230796 IP 10.0.0.1.cbt > 10.0.0.2.56383: Flags [.], ack 25, win
1089, options [nop,nop,TS val 6919208 ecr 6917013], length 0 ^C
12 packets captured
12 packets received by filter
0 packets dropped by kernel
```

In this short example there is traffic from 10.0.0.1, 10.0.0.2 and 10.0.0.3 which represents all three nodes in the cluster. If a node were missing in the `tcpdump` output it would be an indicator of a potential problem with the node.

Monitoring network traffic between nodes and logging it in a self-maintaining logged type output using tcpdump is also an option. The following example shows traffic on the node being logged to `/tmp/hostname` with `$DEVICE` being the one in use by OCFS2.

```
tcpdump -Z root -i $DEVICE -C 50 -W 10 -s 2500 -Sw /tmp/`hostname -
s`_tcpdump.log -ttt 'port 7777' &
```

Once you determine there is a missing node based on evaluation of the traffic, you can do a bit more troubleshooting to determine if there is connectivity between the nodes. This can be accomplished with common tools attempting to connect to the host on port 7777. One example is to use `telnet` to connect to the other host. The following shows an attempt to connect to a node using telnet over port 7777 where the initial connection is established then closed since `telnet` is not providing the expected input:

```
[root@ocfs2-1 ~]# telnet 10.0.0.3 7777
Trying 10.0.0.3...
Connected to 10.0.0.3.
Escape character is '^]'.
Connection closed by foreign host.
```

Here is an example of a failed connection using `telnet`. Notice the connection is refused when attempting to connect to port 7777:

```
[root@ocfs2-1 ~]# telnet 10.0.0.3 7777
Trying 10.0.0.3...
telnet: connect to address 10.0.0.3: Connection refused
```

A much better technique would be to install the utility `netcat` to attempt to make a connection to the other system. The utility `netcat` provides clear output and has more options than `telnet` for this type of connection troubleshooting. `Netcat` is not installed by default so you will need to install it first. You can install it using the following command:

```
yum install nc.x86_64
```

Here is an example of a successful connection using `netcat` to the system ocfs2-3 with the IP of 10.0.0.3 from the system ocfs2-1 with the IP address 10.0.0.1:

```
[root@ocfs2-1 ~]# nc -zv 10.0.0.3 7777
Connection to 10.0.0.3 7777 port [tcp/cbt] succeeded!
```

Here is an example of a failed connection using `netcat`:

```
[root@ocfs2-1 ~]# nc -zv 10.0.0.3 7777
nc: connect to 10.0.0.3 port 7777 (tcp) failed: Connection refused
```

Once you isolate a connection failure, you need to identify the culprit. Start with the basics, such as the machine's local `iptables`, to determine if there are any firewall rules that are preventing access. You can use the following command to determine if there are any firewall rules in place that may be blocking traffic. In this example, the machine has no `iptables` rules in place at all:

```
# iptables --list
Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

If there are a number of `iptables` rules you can add the following to `/etc/sysconfig/iptables` before the last line on all nodes and restart the `iptables` service to allow OCFS2 traffic.

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport
7777 -j ACCEPT
```

You can also utilize the `livenodes.sh` and `livegather.sh` diagnostic scripts to gather diagnostic information about your network traffic. For more information, please see the document titled *"Identifying Problematic Nodes in an OCFS2 Cluster*" found in the additional resources section of this document for information about these diagnostic scripts.

As mentioned earlier, failure to complete critical configuration steps is another common reason for cluster failures. Here are some common things to look for when issues come up with an OCFS2 file system. Many of these are very simple and often are overlooked during initial configuration.

Ensure o2cb and ocfs2 services are configured to start at boot:

```
[root@ocfs2-1 ~]# chkconfig --list o2cb
o2cb              0:off 1:off 2:on  3:on  4:on  5:on  6:off
[root@ocfs2-1 ~]# chkconfig --list ocfs2
ocfs2             0:off 1:off 2:on  3:on  4:on  5:on  6:off
```

If they are not turned on by default you can perform the following operation:

```
chkconfig o2cb on
chkconfig ocfs2 on
```

Check to see if SELinux is disabled on a node. Presently OCFS2 will not operate while SELinux is enabled. You can check this with the following command. It should return disabled as in the following example:

```
[root@ocfs2-1 ~]# getenforce
Disabled
```

If SELinux is enabled on the system it can be disabled by editing `/etc/selinux/config`. The following shows how to disable SELinux in the configuration file:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=disabled
```

Also, look at the configuration files for each node and verify they match exactly. Any differences in the configuration files between nodes can cause issues with the cluster's ability to function. Here is an example of the configuration file of the three node cluster configuration file used for many of the demonstrations in this article. This configuration file should contain the same information on all nodes in the cluster.

In addition, look at the UUID's and device labels on each node to determine if there are variations. Each node in the cluster should have the same UUID and label for the file system. You can use `blkid` a command line utility to locate and print block device attributes to obtain this information. Here's an example of the output of the `blkid` command. The UUID and label for only the OCFS2 file system will match. The boot drives and any other file systems that the nodes may have will likely have different labels and UUID's.

```
[root@ocfs2-1 ~]# blkid
/dev/sda1: UUID="3fbd6ccd-6bc0-4d41-a21d-b5274f5b2238" TYPE="ext4"
/dev/sda2: UUID="zgeNML-VQDc-HMe3-g14R-Fqks-me5P-LCu03R"
TYPE="LVM2_member"
/dev/sdb1: LABEL="ocfs2demo"
UUID="d193a367-a1b6-4e95-82b6-2efdd98dd2fd" TYPE="ocfs2"
/dev/mapper/vg_ocfs21-lv_root:
UUID="79aa15e4-50c4-47be-8d3a-ba5d118c155a" TYPE="ext4"
/dev/mapper/vg_ocfs21-lv_swap:
UUID="99bfb486-3c37-47ca-8dc7-4d46ee377249" TYPE="swap"
```

Some of this information is also available from `mounted.ocfs2`. Here's two examples of using `mounted.ocfs2` to gather information about your OCFS2 file system:

```
[root@ocfs2-1 ~]# mounted.ocfs2 -d
Device     Stack  Cluster    F  UUID                              Lab
el
/dev/sdb1  o2cb   ocfs2demo     D193A367A1B64E9582B62EFDD98DD2FD  ocf
s2demo

[root@ocfs2-1 ~]# mounted.ocfs2 -f
Device     Stack  Cluster    F  Nodes
/dev/sdb1  o2cb   ocfs2demo     ocfs2-2, ocfs2-1
```

It is also possible to do checks on the integrity of the OCFS2 file system and determine if there is any corruption or other issues with the file system using the `fsck.ocfs2` utility.

NOTE: The file system must be unmounted when this check is performed.

The following is an example of the output from a healthy OCFS2 file system using `fsck.ocfs2`:

```
[root@ocfs2-1 ~]# fsck.ocfs2 /dev/sdb1
fsck.ocfs2 1.8.0
Checking OCFS2 filesystem in /dev/sdb1:
  Label:              ocfs2demo
  UUID:               D193A367A1B64E9582B62EFDD98DD2FD
  Number of blocks:   3144715
  Block size:         4096
  Number of clusters: 3144715
  Cluster size:       4096
  Number of slots:    16

/dev/sdb1 is clean.  It will be checked after 20 additional mounts.
```

The following is an example of an unhealthy OCFS2 file system. Notice how the fsck.ocfs2 utility will attempt to correct the problems with the file system:

```
Checking OCFS2 filesystem in /dev/sdb1:
  Label:              ocfs2demo
  UUID:               D193A367A1B64E9582B62EFDD98DD2FD
  Number of blocks:   3144715
  Block size:         4096
  Number of clusters: 3144715
  Cluster size:       4096
  Number of slots:    16

** Skipping slot recovery because -n was given. **
/dev/sdb1 was run with -f, check forced.
Pass 0a: Checking cluster allocation chains
Pass 0b: Checking inode allocation chains
Pass 0c: Checking extent block allocation chains
Pass 1: Checking inodes and blocks.
Pass 2: Checking directory entries.
Pass 3: Checking directory connectivity.
Pass 4a: checking for orphaned inodes
Pass 4b: Checking inodes link counts.
All passes succeeded.
```

The `debgugfs.ocfs2` command opens up many possibilities of troubleshooting and diagnostic functions. In order to utilize `debugfs.ocfs2` first you must mount the debugfs file system for OCFS2. Adding the following to /etc/fstab will allow you to utilize debugfs.ocfs2. Once you have the entry in /etc/fstab you can use the `mount -a` command to mount the filesystem.

```
debugfs     /sys/kernel/debug debugfs  defaults 0 0
```

There are a number of different things you can accomplish with `debugfs.ocfs2` once it is mounted. Below is an example listing all the trace bits and their status. This can be used to determine what tracing is active on your file system for further troubleshooting and information gathering.

```
[root@ocfs2-1 ~]# debugfs.ocfs2 -l
TCP off
MSG off
SOCKET off
HEARTBEAT off
HB_BIO off
DLMFS off
DLM off
DLM_DOMAIN off
DLM_THREAD off
DLM_MASTER off
DLM_RECOVERY off
DLM_GLUE off
VOTE off
CONN off
QUORUM off
BASTS off
```

```
CLUSTER off
ERROR allow
NOTICE allow
KTHREAD off
```

You can also examine file system locks with `debugfs.ocfs2`. In the following example you will list the file system locks on your system:

```
[root@ocfs2-1 tmp]# echo "fs_locks" | debugfs.ocfs2 /dev/sdb1
debugfs.ocfs2 1.8.0
debugfs: fs_locks
Lockres: W000000000000000000001d5a08570b  Mode: Invalid
Flags: Initialized
RO Holders: 0  EX Holders: 0
Pending Action: None  Pending Unlock Action: None Requested Mode:
Invalid  Blocking Mode: Invalid
PR > Gets: 0  Fails: 0    Waits Total: 0us  Max: 0us  Avg: 0ns
EX > Gets: 0  Fails: 0    Waits Total: 0us  Max: 0us  Avg: 0ns
Disk Refreshes: 0

Lockres: O000000000000000000001d00000000  Mode: Invalid
Flags: Initialized
RO Holders: 0  EX Holders: 0
Pending Action: None  Pending Unlock Action: None Requested Mode:
Invalid  Blocking Mode: Invalid
PR > Gets: 0  Fails: 0    Waits Total: 0us  Max: 0us  Avg: 0ns
EX > Gets: 0  Fails: 0    Waits Total: 0us  Max: 0us  Avg: 0ns
Disk Refreshes: 0

Lockres: M000000000000000000001d5a08570b  Mode: Protected Read
Flags: Initialized Attached
RO Holders: 0  EX Holders: 0
Pending Action: None  Pending Unlock Action: None Requested Mode:
Protected Read  Blocking Mode: Invalid
PR > Gets: 1  Fails: 0    Waits Total: 993us  Max: 993us  Avg:
993802ns
EX > Gets: 0  Fails: 0    Waits Total: 0us  Max: 0us  Avg: 0ns
Disk Refreshes: 1
```

(Truncated)

You can also run `debugfs.ocfs2` in interactive mode just by executing the command. A question mark will provide us all of the available options for the command.

```
[root@ocfs2-1 tmp]# debugfs.ocfs2
debugfs.ocfs2 1.8.0
debugfs: ?
bmap <filespec> <logical_blk>             Show the corresponding
physical block# for the inode
cat <filespec>                            Show file on stdout
cd <filespec>                             Change directory
chroot <filespec>                         Change root
close                                     Close a device
controld dump                             Obtain information from
ocfs2_controld
curdev                                    Show current device
decode <lockname#> ...                    Decode block#(s) from the
lockname(s)
dirblocks <filespec>                      Dump directory blocks
dlm_locks [-f <file>] [-l] lockname       Show live dlm locking state
dump [-p] <filespec> <outfile>            Dumps file to outfile on a mounted
fs
dx_dump <blkno>                           Show directory index information
dx_leaf <blkno>                           Show directory index leaf block
only
dx_root <blkno>                           Show directory index root block
only
dx_space <filespec>                       Dump directory free space list
encode <filespec>                         Show lock name
extent <block#>                           Show extent block
findpath <block#>                         List one pathname of the
inode/lockname
frag <filespec>                           Show inode extents / clusters ratio
fs_locks [-f <file>] [-l] [-B]            Show live fs locking state
group <block#>                            Show chain group
grpextents <block#>                       Show free extents in a chain group
hb                                        Show the heartbeat blocks
help, ?                                   This information
icheck block# ...                         List inode# that is using the
block#
lcd <directory>                           Change directory on a
mounted flesystem
locate <block#> ...                       List all pathnames of the
inode(s)/lockname(s)
logdump [-T] <slot#>                      Show journal file for the node slot
ls [-l] <filespec>                        List directory
net_stats [interval [count]]              Show net statistics
ncheck <block#> ...                       List all pathnames of the
inode(s)/lockname(s)
open <device> [-i] [-s backup#]           Open a device
quit, q                                   Exit the program
rdump [-v] <filespec> <outdir>            Recursively dumps from src
to a dir on a mounted filesystem
refcount [-e] <filespec>                  Dump the refcount tree for
the inode or refcount block
slotmap                                   Show slot map
stat [-t|-T] <filespec>                   Show inode
stat_sysdir                               Show all objects in the
system directory
```

```
stats [-h]                                  Show superblock
xattr [-v] <filespec>                       Show extended attributes
```

OCFS2: Tuning and Performance

During the regular usage of an OCFS2 file system you may detect issues related to performance or functionality of the cluster. The file system has many options that can be selected to customize it for a specific environment or use cases. Issues such as network latency or file system performance can be easily tuned to maximize availability and performance for the file system.

One of the more common problems is network latency causing timeouts in the network heartbeat. The livenodes.sh and livegather.sh scripts can be used to collect diagnostic information for use in tuning the heartbeat timeouts. If timeouts are detected making gradual changes in the timeouts may resolve the issue. The timeouts can be adjusted by configuring the o2cb driver. The following is an example of how to tune the o2cb driver:

```
[root@ocfs2-1 etc]# /etc/init.d/o2cb configure
Configuring the O2CB driver.

This will configure the on-boot properties of the O2CB driver.
The following questions will determine whether the driver is loaded
on
boot.  The current values will be shown in brackets ('[]').  Hitting
<ENTER> without typing an answer will keep that current value.  Ctrl-
C
will abort.

Load O2CB driver on boot (y/n) [y]:
Cluster stack backing O2CB [o2cb]:
Cluster to start on boot (Enter "none" to clear) [ocfs2demo]:
Specify heartbeat dead threshold (>=7) [31]:
Specify network idle timeout in ms (>=5000) [30000]:
Specify network keepalive delay in ms (>=1000) [2000]:
Specify network reconnect delay in ms (>=2000) [2000]:
Writing O2CB configuration: OK
Setting cluster stack "o2cb": OK
```

When tuning these values, it is important to understand how each value affects the operation of the cluster. It is important to understand the root cause of the issue in order to be able to tune the file system effectively using as much real data as possible. OCFS2 1.8 logs countdown timing into /var/log/messages and this can be utilized as a guide to begin the tuning process.

The heartbeat dead threshold is the number of two second cycles before a node is considered dead. To set this value we must convert our time by taking the desired time out in seconds and dividing that value by two and adding 1 to it. For example, for a 120 second time out we would set this value to 61.

The network idle timeout specifies the time in milliseconds before a network connection is considered dead. This value can be set directly in milliseconds. The default setting is 30000 milliseconds.

The network keep alive delay the maximum delay before a keep alive packet is sent. This can also be set directly and is set in milliseconds. The default setting is 5000 milliseconds.

The network reconnect delay is the minimum delay between network connection attempts. This value can be set directly in milliseconds and its default setting is 2000 milliseconds.

Performance related issues are best solved by measurement of actual performance and a review of the actual file system usage. For example, a file system used for general purpose storage may have different needs than a file system used to store VM images. There are several areas that can be tuned to maximize performance for an OCFS2 file system. Block size and cluster size can be tuned to provide better performance by matching the needs of the file system use to its actual configuration. The document *"OCFS2 Performance: Measurement, Diagnosis and Tuning"* in the additional resources section of this document contains information on how to measure performance and tune a file system for a number of workloads.

Significant code changes were made in later releases of OCFS2 to prevent file system fragmentation. On earlier releases file system fragmentation sometimes does occur. Currently there is no defragmentation tool available for OCFS2, but there are some workarounds when a file system becomes fragmented. On file systems with a extra pre-defined node slots these node slots can be removed to provide extra space for the file system if there is no continuous space available for writing to the file system. In cases that this is not possible adjustments can be made on the local pre-allocated chunk size during file system mounting. This needs to be done on each node accessing the file system and the file system needs to be re-mounted. The following is an example of adjusting the local pre-allocated chunk size to 32MB using the `localalloc` option. Smaller and larger local pre-allocated chunk sizes are possible depending on the specific needs of the situation.

```
# Created by anaconda on Fri May  3 18:50:48 2013 # # Accessible filesystems,
by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info #
/dev/mapper/vg_ocfs22-lv_root /                       ext4    defaults
       1 1
UUID=6ee45d9d-e41e-479c-89d0-cee5b1620ebb /boot                   ext4
   defaults        1 2
/dev/mapper/vg_ocfs22-lv_swap swap                    swap    defaults
       0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
/dev/sdb1               /ocfs2demo              ocfs2   localalloc=32   0 0
```

Summary

The information in this document provides the standard configuration and most common issues seen with OCFS2. The examples include details on the tools used to configure and tune an OCFS2 environment but this does not represent a comprehensive list of options available. It is recommended administrators also take time to review the additional information found in the "Additional Resources" section at end of this document.

Additional Resources

OCFS2 1.6 Users Guide
https://oss.oracle.com/projects/ocfs2/dist/documentation/v1.6/ocfs2-1_6-usersguide.pdf

Oracle VM High Availability: Hands-on Guide to Implementing Guest VM HA
http://www.oracle.com/us/technologies/026966.pdf

Oracle VM 3: Backup and Recovery Best Practices Guide
http://www.oracle.com/technetwork/server-storage/vm/ovm3-backup-recovery-1997244.pdf

Configuration and Use of Device Mapper Multipathing on Oracle Linux
https://support.oracle.com/epmos/faces/DocumentDisplay?id=555603.1

OCFS2 man page
http://linux.die.net/man/5/exports

Troubleshooting a multi node OCFS2 installation
https://support.oracle.com/epmos/faces/DocumentDisplay?id=806645.1

Diagnostic Information to Collect for OCFS2 Issues
https://support.oracle.com/epmos/faces/DocumentDisplay?id=1538837.1

Identifying Problematic Nodes in an OCFS2 Cluster
https://support.oracle.com/epmos/faces/DocumentDisplay?id=1551417.1

OCFS2 Performance: Measurement, Diagnosis and Tuning
https://support.oracle.com/epmos/faces/DocumentDisplay?id=727866.1

OCFS2 Best Practices Guide
February 2014
Author: Robert Chase

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together