Oracle Linux 10

Managing the Oracle Cluster File System Version 2

G25063-01
June 2025

Oracle Linux 10 Managing the Oracle Cluster File System Version 2,

G25063-01

Contents

Preface

This chapter includes information about managing the Oracle Cluster File System Version 2 (OCFS2) in Oracle Linux 10, including tasks for configuring, administering, troubleshooting, and using OCFS2.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1 About OCFS2

OCFS2 (Oracle Cluster File System Version 2) is a general-purpose shared-disk file system intended for use with clusters. OCFS2 offers high performance and high availability. You can also mount an OCFS2 volume on a standalone system that's not clustered.

Oracle Cluster File System Version 2 (OCFS2) is available on Unbreakable Enterprise Kernel (UEK) releases only, starting with Unbreakable Enterprise Kernel Release 6 (UEK R6).

Using OCFS2 offers the following benefits:

- You can use the `reflink` command with OCFS2 to create copy-on-write clones of individual files. You can also use the `cp --reflink` command similarly to how you would on a Btrfs file system. Typically, these clones let you save disk space when storing copies of files that are similar, such as virtual machine (VM) images or Linux Containers.

  Note:

  When you use the `reflink` command, the resulting file system becomes a clone of the original file system, which means that their UUIDs are identical. When creating a clone with the `reflink` command, you must change the UUID by using the `tunefs.ocfs2` command. See Querying and Changing Volume Parameters

- Mounting a local OCFS2 file system lets you later migrate the file system to a cluster file system without requiring any conversion.
- OCFS2 provides local file system semantics. Therefore, most applications can use OCFS2. Applications that are cluster-aware can use cache-coherent parallel I/O from many cluster nodes to balance activity across the cluster, or they can use the available file-system functionality to fail over and run on another node if a node fails.

2

OCFS2 Use Cases

This section describes some typical use cases for OCFS2.

Load Balancing Use Case

You can use OCFS2 nodes to share resources between client systems. For example, the nodes could export a shared file system by using Samba or NFS. To distribute service requests between the nodes, you can use round-robin DNS, a network load balancer; or, you can specify which node each client uses.

Oracle Real Application Cluster Use Case

Note:

Check the Oracle Database installation documentation for Linux to verify that the platform is certified for use with the current Oracle Linux release: https://docs.oracle.com/en/database/oracle/oracle-database/index.html.

Oracle Real Application Cluster (RAC) uses its own cluster stack, Cluster Synchronization Services (CSS). You can use O2CB with CSS, but note that each stack is configured independently for timeouts, nodes, and other cluster settings. You can use OCFS2 to host the voting disk files and the Oracle cluster registry (OCR), but not the grid infrastructure user's home, which must exist on a local file system on each node.

Because both CSS and O2CB use the lowest node number as a tie breaker in quorum calculations, ensure that the node numbers are the same in both clusters. If required, edit the O2CB configuration file, `/etc/ocfs2/cluster.conf`, to make the node numbering consistent. Then, update this file on all the nodes. The change takes effect when the cluster is restarted.

Oracle Database Use Case

Note:

Check the Oracle Database installation documentation for Linux to verify that the platform is certified for use with the current Oracle Linux release: https://docs.oracle.com/en/database/oracle/oracle-database/index.html.

Specify the `noatime` option when mounting volumes that host Oracle datafiles, control files, redo logs, voting disk, and OCR. The `noatime` option disables unnecessary updates to the access time on the inodes.

Specify the `nointr` mount option to prevent signals interrupting I/O transactions that are in progress.

By default, the `init.ora` parameter `filesystemio_options` directs the database to perform direct I/O to the Oracle datafiles, control files, and redo logs. Specify the `datavolume` mount option for volumes that contain the voting disk and OCR. Do *not* specify this option for volumes that host the Oracle user's home directory or Oracle E-Business Suite.

To prevent database blocks from becoming fragmented across a disk, ensure that the file system cluster size is at least as large as the database block size, which is typically 8KB. If you specify the file system usage type as `datafiles` when using the `mkfs.ocfs2` command, the file system cluster size is set to 128KB.

To enable many nodes to maximize throughput by concurrently streaming data to an Oracle datafile, OCFS2 deviates from the POSIX standard by not updating the modification time (`mtime`) on the disk when performing non-extending direct I/O writes. The value of `mtime` is updated in memory. However, OCFS2 doesn't write the value to disk unless an application extends or truncates the file or performs a operation to change the file metadata, such as using the `touch` command. This behavior leads to results with different nodes reporting different time stamps for the same file. Use the following command to view the on-disk timestamp of a file:

```
sudo debugfs.ocfs2 -R "stat /file_path" device | grep "mtime:"
```

3

Setting Up an OCFS2 Cluster

A cluster consists of members called nodes. For best performance, give each node in the cluster at least two network interfaces. The first interface is connected to a public network to provide general access to the systems, while the second interface is used for private communications between the nodes and the *cluster heartbeat*. The second interface lets the cluster nodes coordinate their access to shared resources and monitor each other's state.

Important:

Both network interfaces must be connected through a network switch. Also, you must ensure that all the network interfaces are configured and working before configuring the cluster.

Planning for an OCFS2 Cluster

In addition to the hardware requirements for the cluster, you must also decide on the hostnames and IP addresses of the cluster members and which heartbeat mode the cluster uses.

In a cluster configuration, small packets travel throughout the entire setup over a specific UDP port, including all the networks configured for the cluster. These packets establish routes between the cluster nodes and show the health of the cluster network. For this reason, the packets are also called *heartbeats*.

You can configure a cluster to run in either of the following heartbeat modes:

- Local heartbeat thread for each shared device (default heartbeat mode).

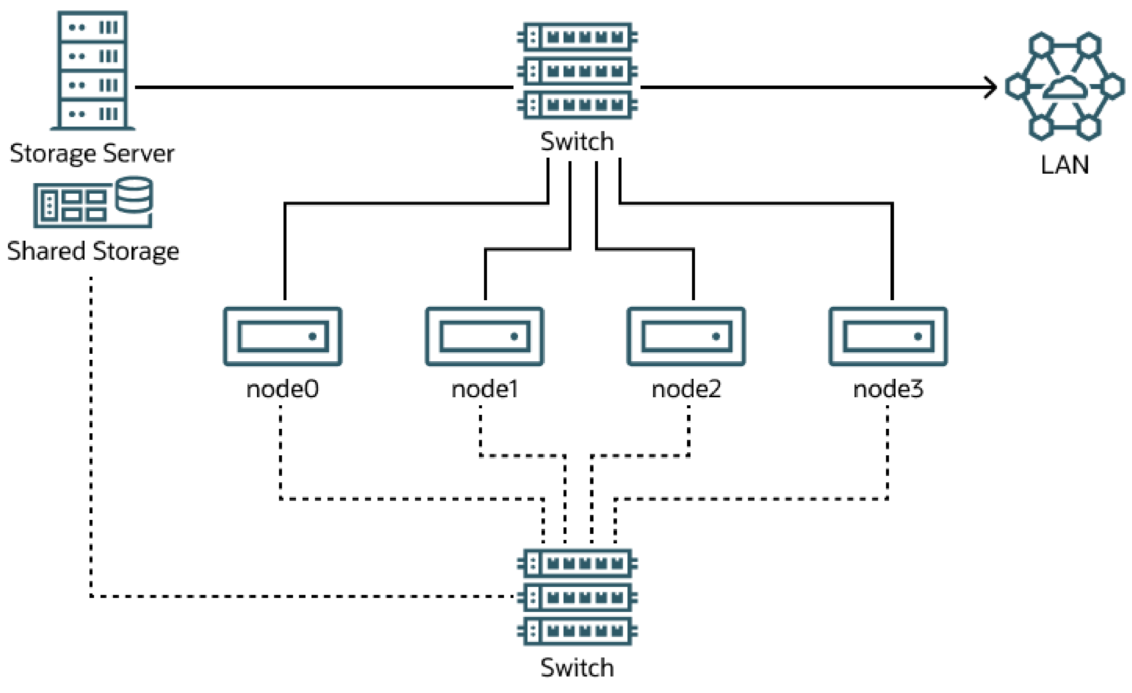
  In this configuration, a node starts a heartbeat thread when it mounts an OCFS2 volume and stops the thread when it unmounts the volume. CPU overhead is large on nodes that mount many OCFS2 volumes because each mount requires a separate heartbeat thread. Likewise, many mounts increases the risk of a node becoming isolated from the cluster, because of a heartbeat I/O timeout on a single mount. This is known as *fencing*.

- Global heartbeat on specific shared devices.

  This mode lets you configure any OCFS2 volume as a global heartbeat device, if the volume occupies a whole disk device and not a partition. In this mode, the heartbeat to the device starts when the cluster becomes online and stops when the cluster goes offline. This mode is recommended for clusters that mount many OCFS2 volumes. A node fences itself out of the cluster if a heartbeat I/O timeout occurs on more than half the global heartbeat devices. To provide redundancy against failure of one of the devices, configure at least three global heartbeat devices.

The following figure shows a cluster with four nodes that are connected using a network switch to a LAN and a network storage server. The nodes and storage server are also connected using a switch to a private network that's used for the local cluster heartbeat.

Figure 3-1 Cluster Configuration by Using a Private Network



Although you can configure and use OCFS2 without using a private network, this increases the probability of a node fencing itself out of the cluster because of an I/O heartbeat timeout.

The following table provides recommendations for minimum cluster size settings for different file system size ranges:

File System Size	Suggested Minimum Cluster Size
1 GB - 10 GB	8K
10GB - 100 GB	16K
100 GB - 1 TB	32K
1 TB - 10 TB	64K
10 TB - 16 TB	128K

Installing the Cluster Software

This task shows you how to install OCFS2. For greatest efficiency, use the same version of OCFS2 and a compatible UEK release on all cluster nodes. In a rolling update of a cluster, the nodes can have different versions of OCFS2 and UEK. When this is the case, the cluster node running the earliest version sets the usable features of the software for the entire cluster.

For a tutorial on how to configure OCFS2, see Use Oracle Cluster File System Tools on Oracle Linux.

Important:

Perform the following procedure on each cluster node.

1. Check which version of the kernel the node is running.

   ```
   uname -r
   ```

   If required, update each node to ensure that all nodes are running the same kernel version.
2. Install the OCFS2 packages.

   ```
   sudo dnf install -y ocfs2-tools
   ```

3. Configure the firewall.

   Configure the firewall to provide access on the interface that the cluster uses for private cluster communication.

   By default, the cluster uses both TCP and UDP over port 7777.

   ```
   sudo firewall-cmd --permanent --add-port=7777/tcp --add-port=7777/udp
   ```

4. Disable SELinux.

   Edit the `/etc/selinux/config` file to disable SELinux, by entering the setting shown in bold:

   ```
   # This file controls the state of SELinux on the system.
   # SELINUX= can take one of these values:
   #     enforcing - SELinux security policy is enforced.
   #     permissive - SELinux prints warnings instead of enforcing.
   #     disabled - No SELinux policy is loaded.
   SELINUX=disabled
   ```

Configuring the Cluster Layout

The following task describes how to create a cluster, add nodes, and configure the chosen heartbeat mode. Run the commands on a single host that you want to be a node in the cluster.

The examples shown create a cluster called `mycluster`, with the following hostnames and IP addresses as nodes in the cluster:

- `ol-sys0`: 10.1.0.100
- `ol-sys1`: 10.1.0.110
- `ol-sys2`: 10.1.0.120

1. Define the cluster.

   Create a cluster definition by running the following command on any node:

   ```
   sudo o2cb add-cluster cluster-name
   ```

   For example, to create the cluster `mycluster` from the `ol-sys1` host, run the following command on `ol-sys1`:

   ```
   sudo o2cb add-cluster mycluster
   ```

2. Add nodes to the cluster.

Add the current host and all other required hosts as nodes to the cluster.

```
sudo o2cb add-node cluster-name hostname --ip ip_address
```

The IP address is the IP address that's used by the node for private communication in the cluster.

For example, to add nodes to `mycluster` from `ol-sys1`, you would type the following commands on `ol-sys1`:

```
sudo o2cb add-node mycluster ol-sys0 --ip 10.1.0.100
sudo o2cb add-node mycluster ol-sys1 --ip 10.1.0.110
sudo o2cb add-node mycluster ol-sys2 --ip 10.1.0.120
```

> **Note:**
>
> OCFS2 only works with IPv4 addresses.

3. (Optional) Configure the global heartbeat mode.

   The default heartbeat mode for the cluster is local. If you want the cluster to run in global heartbeat mode, do the following:

   a. Run the following command on every cluster device:

      ```
      sudo o2cb add-heartbeat cluster-name device-name
      ```

      For example, for `mycluster`, you set the cluster heartbeat on `/dev/sdd`, `/dev/sdg`, and `/dev/sdj` as follows:

      ```
      sudo o2cb add-heartbeat mycluster /dev/sdd
      sudo o2cb add-heartbeat mycluster /dev/sdg
      sudo o2cb add-heartbeat mycluster /dev/sdj
      ```

   b. Set the cluster's heartbeat mode to global.

      ```
      sudo o2cb heartbeat-mode cluster-name global
      ```

   > **Important:**
   >
   > You must configure the global heartbeat feature to use whole disk devices. You can't use disk partitions for global heartbeat mode.

4. Copy the cluster configuration file to each cluster node.

   Copy the cluster `/etc/ocfs2/cluster.conf` file to each node in the cluster.

5. (Optional) Show information about the cluster.

   Run the following command to display information about the cluster:

   ```
   sudo o2cb list-cluster cluster-name
   ```

A three-node cluster with a global heartbeat would display information similar to the following:

```
node:
        name = ol-sys0
        cluster = mycluster
        number = 0
        ip_address = 10.1.0.100
        ip_port = 7777

node:
        name = ol-sys1
        cluster = mycluster
        number = 1
        ip_address = 10.1.0.110
        ip_port = 7777

node:
        name = ol-sys2
        cluster = mycluster
        number = 2
        ip_address = 10.1.0.120
        ip_port = 7777

cluster:
        name = mycluster
        heartbeat_mode = global
        node_count = 3

heartbeat:
        cluster = mycluster
        region = 7DA5015346C245E6A41AA85E2E7EA3CF

heartbeat:
        cluster = mycluster
        region = 4F9FBB0D9B6341729F21A8891B9A05BD

heartbeat:
        cluster = mycluster
        region = B423C7EEE9FC426790FC411972C91CC3
```

The heartbeat regions are represented by the UUIDs of their block devices.

The same cluster `mycluster` with a local heartbeat would display the following information:

```
node:
        name = ol-sys0
        cluster = mycluster
        number = 0
        ip_address = 10.1.0.100
        ip_port = 7777

node:
        name = ol-sys1
        cluster = mycluster
        number = 1
        ip_address = 10.1.0.110
        ip_port = 7777

node:
        name = ol-sys2
        cluster = mycluster
        number = 2
        ip_address = 10.1.0.120
        ip_port = 7777

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

Configuring and Starting the O2CB Cluster Stack

The following steps configure and start the O2CB cluster stack and must be run on every node in the cluster.

1. Configure the node.

   Run the following command to configure the node:

   ```
   sudo /sbin/o2cb.init configure
   ```

   The configuration process prompts you for extra information. You need to specify the following:

   - **Load the O2CB driver on boot**: Specify `y` or `n`, which is the default setting.
   - **Cluster to start at boot**: Specify the name of the cluster. The name must match that in the `/etc/ocfs2/cluster.conf` file.
   - For the remaining parameters, the default values are typically adequate.

2. Check the status of the cluster stack.

   Verify the settings for the cluster stack:

   ```
   sudo /sbin/o2cb.init status
   ```

   A cluster that uses local heartbeat mode displays information similar to the following:

   ```
   Driver for "configfs": Loaded
   Filesystem "configfs": Mounted
   Stack glue driver: Loaded
   Stack plugin "o2cb": Loaded
   Driver for "ocfs2_dlmfs": Loaded
   Filesystem "ocfs2_dlmfs": Mounted
   Checking O2CB cluster "mycluster": Online
     Heartbeat dead threshold: 61
     Network idle timeout: 30000
     Network keepalive delay: 2000
     Network reconnect delay: 2000
   ```

```
  Heartbeat mode: Local
Checking O2CB heartbeat: Active
```

A cluster that uses global heartbeat mode displays information similar to the following:

```
Driver for "configfs": Loaded
Filesystem "configfs": Mounted
Stack glue driver: Loaded
Stack plugin "o2cb": Loaded
Driver for "ocfs2_dlmfs": Loaded
Filesystem "ocfs2_dlmfs": Mounted
Checking O2CB cluster "mycluster": Online
  Heartbeat dead threshold: 61
  Network idle timeout: 30000
  Network keepalive delay: 2000
  Network reconnect delay: 2000
  Heartbeat mode: Global
Checking O2CB heartbeat: Active
  7DA5015346C245E6A41AA85E2E7EA3CF /dev/sdd
  4F9FBB0D9B6341729F21A8891B9A05BD /dev/sdg
  B423C7EEE9FC426790FC411972C91CC3 /dev/sdj
```

3. Enable the `o2cb` and `ocfs2` services.

   Enable the `o2cb` and `ocfs2` services so that they start on boot after networking is enabled.

   ```
   sudo systemctl enable o2cb
   ```

   ```
   sudo systemctl enable ocfs2
   ```

4. Configure the `sysctl`kernel parameters.

   Edit the `/etc/sysctl.d/99-sysctl.conf` file to configure the following kernel settings for cluster operations:

   - `kernel.panic` = 30

     This setting specifies the number of seconds after a panic before a system reboots. The default value is zero. If you want a memory image to be created before the system reboots, assign a larger value.

   - `kernel.panic_on_oops` = 1

     This setting causes the system to panic if a kernel `oops` occurs. So, if a kernel thread required for cluster operation fails, the system reboots. Otherwise, a node might not know whether another node is slow to respond or unavailable, which eventually causes all cluster operations to suspend.

5. Save the changes to the `sysctl` configuration.

   Apply the configuration by running the following command:

   ```
   sudo sysctl -p
   ```

The `o2cb.init` command accepts other subcommands which let you administer the cluster, such as the following:

- `/sbin/o2cb.init status`: Check the status of the cluster stack.

- `/sbin/o2cb.init online`: Bring the cluster stack online.
- `/sbin/o2cb.init offline`: Take the cluster stack offline.
- `/sbin/o2cb.init unload`: Unload the cluster stack.

To view other available subcommands, run the command `o2cb.init` by itself.

4

Working With OCFS2 Volumes

Use the `mkfs.ocfs2` command to configure OCFS2 volumes. The command accepts different options and arguments which let you control how how volumes are created, including the following:

`-b` ***`blocksize`***, **`--block-size`** ***`blocksize`***
Specifies the unit size for I/O transactions to and from the file system, and the size of inode and extent blocks. Available block sizes are 512 (512 bytes), 1K, 2K, and 4K. The default and recommended block size is 4K (4 kilobytes).

`-C` ***`clustersize`***, **`--cluster-size`** ***`clustersize`***
Specifies the unit size for space used to allocate file data. The available cluster sizes are 4K, 8K, 16K, 32K, 64K, 128K, 256K, 512K, and 1M (1 megabyte). The default cluster size is 4K (4 kilobytes).

`--fs-feature-level=` ***`feature-level`***
Lets you select a set of file system features from the following choices:

- `default`: Lets you use sparse files, unwritten extents, and inline data features.
- `max-compat`: Only makes available features that are understood by older versions of OCFS2.
- `max-features`: Makes all features of OCFS2 available.

`--fs_features=` ***`feature`***
Lets you enable or disable individual features such as sparse files, unwritten extents, and backup superblocks. For more information, see the `mkfs.ocfs2(8)` manual page.

`-J` ***`journalsize`***, **`--journal-size`** ***`journalsize`***
Specifies the size of the write-ahead journal. If not specified, the size is calculated from the file system usage type that you specify using the `-T` option, and, otherwise, from the volume size. The default size of the journal is 64M (64 MB) for `datafiles`, 256M (256 MB) for `mail`, and 128M (128 MB) for `vmstore`.

`-L` ***`label`***, **`--label`** ***`label`***
Specifies a descriptive name for the volume that lets you identify it easily on different cluster nodes.

`-N` ***`number-of-slots`***, **`--node-slots`** ***`number-of-slots`***
Specifies the maximum number of nodes that can concurrently access a volume, which is limited by the number of node slots for system files such as the file system journal. For best performance, set the number of node slots to at least twice the number of nodes. If you later increase the number of node slots, performance can suffer because the journal is no longer contiguously laid out on the outer edge of the disk platter.

`-T` ***`file-system-usage-type`***
Specifies the usage type of the file system, which is one of the following three options:

- `datafiles`: Database files are typically few in number, fully allocated, and relatively large. Such files require few metadata changes, and don't benefit from having a large journal.
- `mail`: Mail server files are typically many in number, and relatively small. Such files require many metadata changes, and benefit from having a large journal.
- `vmstore`: Virtual machine image files are typically few in number, sparsely allocated, and relatively large. Such files require a moderate number of metadata changes and a medium sized journal.

Creating and Mounting OCFS2 Volumes

When creating OCFS2 volumes, consider the following:

- Don't create an OCFS2 volume on an LVM logical volume, as LVM isn't cluster-aware.
- After you have created an OCFS2 volume, you can't change the block and cluster size of that volume. You can use the `tunefs.ocfs2` command to change other file system settings, with certain restrictions. For more information, see the `tunefs.ocfs2(8)` manual page.
- If you intend the volume to store database files, don't specify a cluster size that's smaller than the block size of the database.
- The default cluster size of 4 KB isn't suitable if the file system is larger than a few gigabytes.

1. Create an OCFS2 volume.

   The following command creates the volume with a label on the specified device:

   ```
   sudo mkfs.ocfs2 -L "myvol" /dev/sdc1
   ```

   Without extra options or arguments, the volume uses default values for some of its properties, such as 4 KB block and cluster size, eight node slots, 256 MB journal, and makes default file system features available. These default settings are only suitable to create volumes for file systems that are no larger than a few gigabytes.

   Tip:

   Ensure that the device corresponds to a partition so that you can use the label when mounting the volume.

   Specify options to the `mkfs.ocfs2` command to create volumes with different characteristics. Consider the following examples:

   - Create a labeled volume for use as a database.

     ```
     sudo mkfs.ocfs2 -L "dbvol" -T datafiles /dev/sdd2
     ```

     In this case, the cluster size is set to 128 KB and the journal size to 32 MB.
   - Create a volume with specific property settings.

     ```
     sudo mkfs.ocfs2 -C 16K -J size=128M -N 16 --fs-feature-level=max-
     features --fs-features=norefcount /dev/sde1
     ```

This command specifies cluster and journal sizes, and the number of node slots. All file system features are enabled, except `norefcount` trees.

2. On each cluster member, mount the created volume.
   a. Create a mount point.

      ```
      sudo mkdir /u01
      ```

   b. Mount the volume.

      ```
      sudo mount -L myvol /u01
      ```

   c. Check the status of the heartbeat mode.

      ```
      sudo o2cb.init status
      ```

      The heartbeat becomes active after the volume is mounted.

3. (Optional) Permanently mount the OCFS2 volume.

   You can automate the mount operation across system restarts by adding an entry to the `/etc/fstab` file. For example:

   ```
   myvol  /u01   ocfs2     _netdev,defaults  0 0
   ```

   In this entry, `_netdev` instructs the system to mount an OCFS2 volume at boot time only after networking is started and unmounts the file system before networking is stopped.

Querying and Changing Volume Parameters

Use the `tunefs.ocfs2` command to query or change volume parameters.

For example, to find out the label, UUID, and number of node slots for a volume, use the following command:

```
sudo tunefs.ocfs2 -Q "Label = %V\nUUID = %U\nNumSlots =%N\n" /dev/sdb
```

The output of the command is similar to the following:

```
Label = myvol
UUID = CBB8D5E0C169497C8B52A0FD555C7A3E
NumSlots = 4
```

Generate a new UUID for a volume by using the following commands:

```
sudo tunefs.ocfs2 -U /dev/sda
sudo tunefs.ocfs2 -Q "Label = %V\nUUID = %U\nNumSlots =%N\n" /dev/sdb
```

The output is similar to the following:

```
Label = myvol
UUID = 48E56A2BBAB34A9EB1BE832B3C36AB5C
NumSlots = 4
```

5 Creating a Local OCFS2 File System

You can create an OCFS2 file system that you can mount locally, which isn't associated with a cluster. Use the following command syntax:

```
sudo mkfs.ocfs2 -M local --fs-features=local -N 1 [options] device
```

The following example creates a locally mountable OCFS2 volume on `/dev/sdc1`, with one node slot and the label `localvol`:

```
sudo mkfs.ocfs2 -M local --fs-features=local -N 1 -L "localvol" /dev/sdc1
```

To convert a local OCFS2 file system for use in a cluster, use the `tunefs.ocfs2` utility as follows:

```
sudo umount /dev/sdc1
sudo tunefs.ocfs2 -M cluster --fs-features=clusterinfo -N 8 /dev/sdc1
```

This example also increases the number of node slots from 1 to 8, letting up to eight nodes mount the file system.

6

Troubleshooting OCFS2 Issues

The following information can help you resolve issues that you might come across when administering OCFS2.

Recommended Debugging Tools and Practices

Use the following tools to troubleshoot OCFS2 issues:

- Install `netconsole` on the nodes to capture an `oops` trace.
- Use the `tcpdump` command to capture the DLM's network traffic between nodes. For example, to capture TCP traffic on port 7777 for the private network interface `em2`, you could use the following command:

  ```
  sudo tcpdump -i em2 -C 10 -W 15 -s 10000 -Sw /tmp/`hostname -
  s`_tcpdump.log \
  -ttt 'port 7777' &
  ```

- Use the `debugfs.ocfs2` command to trace events in the OCFS2 driver, view the status of locks, walk directory structures, examine inodes, and so on. This command is similar in behavior to the `debugfs` command that's used for the `ext3` file system.

  For more information, see the `debugfs.ocfs2(8)` manual page.
- Use the `o2image` command to save an OCFS2 file system's metadata, including information about inodes, file names, and directory names, to an image file on another file system. Because the image file contains only metadata, it's much smaller than the original file system. You can use the `debugfs.ocfs2` command to open the image file and analyze the file system layout to discover the cause of a file system corruption or performance problem.

  For example, to create the image `/tmp/sda2.img` from the OCFS2 file system on the device `/dev/sda2`, you would use the following command:

  ```
  sudo o2image /dev/sda2 /tmp/sda2.img
  ```

  For more information, see the `o2image(8)` manual page.

Mounting the debugfs File System

OCFS2 uses the `debugfs` file system to enable userspace access to information about its in-kernel state. You must mount the `debugfs` file system to use the `debugfs.ocfs2` command.

1. Edit the `/etc/fstab` file

   Add the following line to the `/etc/fstab` file:

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

2. Mount the `debugfs` file system

   Mount the `debugfs` file system by running the following command:

   ```
   mount -a
   ```

Configuring OCFS2 Tracing

Use the following commands and methods to trace issues in OCFS2.

Commands for Tracing OCFS2 Issues

The following commands are useful for tracing OCFS2 issues.

`debugfs.ocfs2 -l`
Lists all the trace bits and their statuses.

`debugfs.ocfs2 -l SUPER allow|off|deny`
Allows, disables, or disallows tracing for the superblock. If you specify `deny`, then even if another tracing mode setting implicitly allows it, tracing is still disallowed.

`debugfs.ocfs2 -l HEARTBEAT ENTRY EXIT allow`
Enable heartbeat tracing.

`debugfs.ocfs2 -l HEARTBEAT off ENTRY EXIT deny`
Disable heartbeat tracing. `ENTRY` and `EXIT` parameters are set to `deny`, as these parameters exist in all trace paths.

`debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow`
Enable tracing for the file system.

`debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE allow`
Disable tracing for the file system.

`debugfs.ocfs2 -l ENTRY EXIT DLM DLM_THREAD allow`
Enable tracing for the DLM.

`debugfs.ocfs2 -l ENTRY EXIT deny DLM DLM_THREAD allow`
Disable tracing for the DLM.

OCFS2 Tracing Methods and Examples

To obtain a trace, first enable the trace, sleep for a short while, and then disable the trace. To avoid unnecessary output, reset the trace bits to their default settings after you have finished tracing, as shown in the following example:

```
sudo debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow && sleep 10 &&
sudo debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE off
```

To limit the amount of information that's displayed, enable only the trace bits that are relevant to diagnosing the problem.

If a specific file system command, such as `mv`, is causing an error, you might use a sequence of commands that are shown in the following example to trace the error:

```
sudo debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow
mv source destination & CMD_PID=$(jobs -p %-)
echo $CMD_PID
sudo debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE off
```

Because the trace is enabled for all mounted OCFS2 volumes, knowing the correct process ID can help you to interpret the trace.

For more information, see the `debugfs.ocfs2(8)` manual page.

Debugging File System Locks

If an OCFS2 volume hangs, you can use the following procedure to find out which locks are busy and which processes are likely to be holding the locks.

In the following procedure, the `Lockres` value refers to the lock name that's used by DLM, which is a combination of a lock-type identifier, inode number, and a generation number. The following table lists the various lock types and their associated identifier.

Table 6-1 DLM Lock Types

Identifier	Lock Type
D	File data
M	Metadata
R	Rename
S	Superblock
W	Read-write

1. Mount the debug file system.

   Mount the debug file system using the following command:

   ```
   sudo mount -t debugfs debugfs /sys/kernel/debug
   ```

2. View the lock statuses.

   Dump the lock statuses for the file system device, which is `/dev/sdx1` in the following example:

   ```
   echo "fs_locks" | sudo debugfs.ocfs2 /dev/sdx1 | sudo tee /tmp/fslocks


   Lockres: M000000000000006672078b84822 Mode: Protected Read
   ...
   ```

3. Retrieve the inode and generation number.

Use the `Lockres` value from the previous output to obtain the inode number and generation number for the lock.

```
sudo echo "stat lockres-value" | sudo debugfs.ocfs2 -n /dev/sdx1
```

For example, for the `Locres` value `M000000000000006672078b84822` from the previous step, the command output might resemble the following:

```
Inode: 419616   Mode: 0666   Generation: 2025343010 (0x78b84822)
...
```

4. Look up the file system object.

   Relate the file system object to the inode number from the previous output:

   ```
   sudo echo "locate inode" | sudo debugfs.ocfs2 -n /dev/sdx1
   ```

   For example, for the `Inode` value `419616` from the previous step, the command output might resemble the following:

   ```
   419616 /linux-2.6.15/arch/i386/kernel/semaphore.c
   ```

5. Obtain the lock names for the file system object.

   Obtain the names of the locks that are associated with the file system object, which in the previous step's output is `/linux-2.6.15/arch/i386/kernel/semaphore.c`. Thus, you would type:

   ```
   sudo echo "encode /linux-2.6.15/arch/i386/kernel/semaphore.c" | sudo
   debugfs.ocfs2 -n /dev/sdx1
   ```

   ```
   M000000000000006672078b84822 D000000000000006672078b84822
   W000000000000006672078b84822
   ```

   In the previous example, a metadata lock, a file data lock, and a read-write lock are associated with the file system object.

6. Retrieve the DLM domain.

   Establish the DLM domain of the file system by running the following command:

   ```
   sudo echo "stats" | sudo debugfs.ocfs2 -n /dev/sdX1 | grep UUID: | while
   read a b ; do echo $b ; done
   ```

   ```
   82DA8137A49A47E4B187F74E09FBBB4B
   ```

7. Enable debugging.

   Using the values of the DLM domain and the lock name, run the following command to let you debug that DLM:

   ```
   sudo echo R 82DA8137A49A47E4B187F74E09FBBB4B M000000000000006672078b84822 |
   sudo tee /proc/fs/ocfs2_dlm/debug
   ```

8. View the debug messages.

   Examine the debug messages by using the `dmesg | tail` command, for example:

   ```
   struct dlm_ctxt: 82DA8137A49A47E4B187F74E09FBBB4B, node=3, key=965960985
     lockres: M000000000000006672078b84822, owner=1, state=0 last used: 0,
     on purge list: no granted queue:
         type=3, conv=-1, node=3, cookie=11673330234144325711,
   ast=(empty=y,pend=n),
         bast=(empty=y,pend=n)
       converting queue:
       blocked queue:
   ```

   The DLM has three lock modes: no lock (`type=0`), protected read (`type=3`), and exclusive (`type=5`). In the previous example, the lock is owned by node 1 (`owner=1`) and node 3 has been granted a protected-read lock on the file-system resource.

9. Identify sleeping processes.

   Use the following command to search for processes that are in an uninterruptable sleep state, which are indicated by the `D` flag in the `STAT` column:

   ```
   ps -e -o pid,stat,comm,wchan=WIDE-WCHAN-COLUMN
   ```

   Note that at least one of the processes that are in the uninterruptable sleep state is responsible for the hang on the other node.

If a process is waiting for I/O to complete, the problem could be anywhere in the I/O subsystem, from the block device layer through the drivers, to the disk array. If the hang concerns a user lock (`flock()`), the problem could lie with the application. If possible, end the process holding the lock. If the hang is because of lack of memory or fragmented memory, you can free up memory by ending nonessential processes. The most immediate solution is to reset the node that's holding the lock. The DLM recovery process can then clear all the locks owned by the dead node, enabling the cluster to continue to operate.

Configuring the Behavior of Fenced Nodes With Kdump

If the hearbeat mechanism detects that a node with a mounted OCFS2 volume has lost contact with the other cluster nodes, that node is removed from the cluster in a process called *fencing*. Fencing prevents other nodes from hanging while trying to access resources that are held by the fenced node. By default, a fenced node automatically restarts so that it can rejoin the cluster as soon as possible.

However, under some circumstances, you might not want this default behavior. For example, if a node often restarts for no obvious reason, then causing the node to panic instead of restarting is preferable, so that you can troubleshoot the issue. By enabling Kdump on the node, you can obtain a `vmcore` crash dump from the fenced node and analyze it to diagnose the cause of frequent node restarts.

1. Configure the fence method.

   To configure a node to panic at the next fencing, set `fence_method` to `panic` by running the following command on the node after the cluster starts:

   ```
   echo "panic" | sudo tee /sys/kernel/config/cluster/cluster-name/
   fence_method
   ```

2. Persist the change across reboots.

   To set the value after each system reboot, add the same line to the `/etc/rc.local` file.

To restore the default behavior, change the value of `fence_method` back to `reset`.

```
echo "reset" | sudo tee /sys/kernel/config/cluster/cluster-name/fence_method
```

Then, remove the `panic` line from `/etc/rc.local` if the line exists in the file.