Configuring udev and device mapper for Oracle RAC 10g Release 2 on SLES9

An Oracle White Paper
June 2006

Table of Contents

Introduction

With the advent of Oracle Database 10g and Linux 2.6, customers have a true low cost computing solution. However, some customers that want high availability solutions have deployed products that has added to the overall price-point of implementations, thus increasing overall total cost of ownership (TCO). One such example is the deployment of 3rd party I/O multi-pathing utilities by storage vendors. Customers now want a low or no cost replacement for these utilities, that are out of the box and stable solutions.

This paper will focus on the configuration of two utilities that provide out-of-the-box I/O multipathing solutions and device naming persistency in the Linux 2.6 environment. Specifically we will illustrate how Automatic Storage Management (ASM) and Real Application Clusters (RAC) deployments can leverage these utilities.

The following utilities will be reviewed:

- udev – the role of udev is to provide device persistency and naming consistency. This is especially important for the OCR and Voting disks required by Oracle Clusterware.
- device-mapper – Device-mapper will provide I/O path management for all Oracle devices, including ASM disk devices and OCR/Voting disks.
- ASM - ASM will provide volume management for all Oracle database files.
- ASMLib – ASMLib will provide device management specifically for ASM disk devices.

udev

In Linux 2.6, a new feature was introduced to simplify device management and hotplug capabilities. This feature is called *udev* and is a standard package in SLES9. Udev is a userspace utility for dynamic device node creation. A device node is an entry in the /dev directory; e.g., sda or sdh1. One main benefit of udev for Oracle environments, is that it provides persistent disk naming; i.e., prevents device renames upon SCSI reconfigurations.

By default, udev is used only to create the device nodes for hotplug devices. Many devices on the system are considered "cold-plug" and will not be handled by default. In order to have udev handle the device node creation for all devices, the boot.udev script should be enabled to run on boot using the following command

```
#insserv /etc/init.d/boot.udev
```

This will cause all devices on the system to be named by udev. It can also be started manually by calling /etc/init.d/boot.udev on SLES9. Udev uses hotplug events sent by the kernel whenever a device is added or removed from the system. The details about the newly added devices are exported by the kernel to the sysfs filesystem. The sysfs filesystem, which is a new filesystem in the 2.6 kernel, is managed by the kernel and

exports basic information about the devices currently plugged into your system. Udev manages the /dev directory by monitoring the /sys directory.

By leveraging the information in sysfs, udev can determine which devices need to have a /dev entry created and the corresponding device names to be used. This infrastructure of udev and sysfs allows predictable behavior when devices are added or removed from the system.

Once udev is notified that a device is added, it uses the scsi_id call to retrieve and generate a unique SCSI identifier. The scsi_id call queries a SCSI device via the SCSI INQUIRY command and leverages the vital product data (VPD) page 0x80 or 0x83. The output from this query is used to to generate a value that is unique across all SCSI devices that properly support pages 0x80 or 0x83[1].

udev uses three basic configuration files that control its functionality.

- /etc/udev/udev.conf - the main udev configuration file
- /etc/udev/udev.permissions - udev permission files
- udev rules files. – naming rule definitions.

See the udev man page for details on modifying these files.

udev.conf file

The main udev configuration file, /etc/udev/udev.conf, controls the directory locations for the udev permission and rules files, the udev database, and the default location where udev device nodes are created. The udev man page shows a sample udev.conf file.

udev.permissions

Once udev is started and the device nodes are created, udev applies the ownership and permissions of the device node to what is defined in the udev.permissions file. This is done every time udev is reloaded. This file is essentially used to keep file attributes persistent across reboots. Thus, this file should contain an entry for the disks that will be used for ASM, OCR, and Voting disks. Either specify individual disks or use wildcards. Below is a sample *udev.permissions*:

```
#name:user:group:mode
sda*:oracle:oinstall:660
 sdd1:oracle:oinstall:660
 sde1:oracle:oinstall:660
 sdf1:oracle:oinstall:660
 sdg1:oracle:oinstall:660
```

[1] The devices must support either page x80/x83 to enable this.

udev.rules

The udev rules file is used to drive the naming scheme for device nodes. udev passes in a device node and applies the appropriate rule. The output is a meaningful and consistent device name. Every line in the rules files defines how a specific device attribute is mapped to a device file. If all keys that are specified in a rule match the device that was found, the specified device file is created.

SLES9 comes with several packaged wrapper scripts that can be used directly in the udev rules file. In this example, the wrapper script udev.get_persistent_device_name.sh, is used. This wrapper script actually calls /sbin/udev.get_unique_hardware_path.sh and /sbin/udev.get_unique_drive_id.sh. Below is an example of udev rule mapped to a SCSI device

```
BUS="scsi", PROGRAM="/sbin/udev.get_persistent_device_name.sh",
NAME="%k" SYMLINK="%c{1+}"
```

This rule indicates that for each SCSI device, we pass the device name to the udev.get_persistent_device_name.sh program. The "%k" represents the kernel assigned device name, such as sdc. The "%c" stands for the output string to be generated from the specified program. The program will output the persistent name for that device and udev will create a symbolic link (of the persistent name) that points to the original device.

For example, for device sdc, running the script manually produces the following device name:

```
fstlcg01:~ # /sbin/udev.get_persistent_device_name.sh \
/sys/block/sdc

disk/by-path/pci-0000:01:06.0-scsi-0:0:0:0
disk/by-id/36000393000011a3c01000000d52ac983
```

Two symbolic links will be created, one referenced in /dev/disk/by-id/ and another in /dev/disk/by-path

```
lrwxrwxrwx  1 root root 9 Aug 26 17:01 /dev/disk/by-path/pci-
0000:01:06.0-scsi-0:0:0:0 -> ../../sdc
lrwxrwxrwx  1 root root  9 Aug 26 17:01 /dev/disk/by-
id/36000393000011a3c01000000d52ac983 -> ../../sdc
```

If the disk contains multiple partitions, each partition will have a symbolic link with the partition number added to the name, such as:

```
lrwxrwxrwx  1 root root 10 Aug 26 17:01 /dev/disk/by-
id/36000393000011a3c01000000d52ac983p1 -> ../../sdc1
```

In this way we can achieve persistent naming by referring to the device by the persistent named symbolic link.

The /usr/sbin/*udevinfo -d* command can be used to display all devices defined in the udev database. In addition to displaying the contents of the udev database, the *udevinfo* command can be used to display information about the devices udev has configured. For example the following *udevinfo* command will display all of the symbolic links (symlinks) that udev has created that point to /dev/sdc1.
udevinfo -q symlink -n /dev/sdc1.

To show all the udevinfo stored in the udev database about a specific device, use the following command:
udevinfo -q all -n /dev/sdc1

See the udevinfo man page for more details.

Adding new storage device to a host

There will be cases when new storage disks/LUNs will be added to a host. Therefore, when adding new storage, the HBA driver must be instructed to rescan the SCSI bus to discover the new storage area (LUN)[2]. This is required since the Linux 2.6 kernel is unable to dynamically scan for new devices. However, the exact commands or method may depend on the driver. If the driver supports it, the following command can be used to request a SCSI scan for each HBA port. .
For example, for a QLogic 2300 HBA, the command is

echo scsi-qlascan >/proc/scsi/qla2xxx/<*host number*>

and for an Emulex HBA, it is

echo 1 >/sys/class/scsi_host/host<*host number*>/issue_lip

For details on this procedure see the following URL:
https://secure-support.novell.com/KanisaPlatform/Publishing/390/817_f.SAL_Public.html

Device mapper- DM

An I/O path generally consists of an initiator port, fabric port, target port, and LUN. Each permutation of this I/O path is considered an independent path. Dynamic Multi-pathing/failover tools aggregate these independent paths into a single logical path. This path virtualization provides I/O load-balancing across the host bus adapters (HBAs), as well as non-disruptive failover on HBA failures.

Multipathing is an important aspect of high availability configurations. Linux 2.6 offers a Path Manger utility called “device mapper” (DM). DM uses pseudo devices with the “dm-” prefix in the “/dev” directory. It automatically maps paths to drives, using the drive’s unique world wide id (WWID).

[2] This is required if you are not rebooting to pickup the new storage.

Setup

DM is dependent on the "device-mapper", "udev", and "multipath-tools" packages. The following are some best practices for deploying DM:

- If the host server has multiple HBAs, the HBA driver may be configured with its own failover mode enabled, thus preventing the paths except the first HBA from being discovered. For QLogic HBAs, you can prevent this by disabling the QLogic HBA failover by setting "ql2xfailover=0" in /etc/modules.conf file.

- Set "HOTPLUG_USE_SUBFS=no" in the "/etc/sysconfig/hotplug" file.

- Configure INITRD_MODULES parameter in the "/etc/sysconfig/kernel" file with the disk controller modules. See comments for in the file for further information. For example:

INITRD_MODULES=" qla2xxx_conf qla2xxx qla2300 ext3" DM multipathing can be initialized, using the following command:

```
# /etc/init.d/boot.multipath start
```

It is recommended that DM be integrated into the boot sequence using the following command:

```
# insserv /etc/init.d/boot.multipath
```

The /sbin/*multipath* command invokes DM using the settings defined in the "/etc/multipath.conf" file[3]. The multipath.conf can be edited to using the following stanza format[4]:

```
multipaths {

      multipath {
            wwid                    36006016064f51000d2e7d1feb052d911
            alias                   ocr
            path_grouping_policy    failover
            path_checker            readsector0
            path_selector           "round-robin 0"
            failback                immediate
      }
```

The "wwid" attribute can be determined using the multipath –v2 –d. It is recommended that the alias attribute be added for the device. This is especially important for OCR and Voting disk devices, so that these devices can be easily identified. The multipath.conf file should contain the entries for all required multipathed devices. Once the multipath.conf file is edited, run command *multipath –v2* to allow the multipathed

[3] If the file does not exist, DM uses the default settings.

[4] Note, that the stanza applies to a device level; i.e., all partitions of a device share the same attributes.

process to pickup the new configuration file, and enable the new multipath device nodes to be created in the /dev/dm-x and /dev/mapper/wwn directories.

DM's *multipath* command can be used to dynamically change policies or policy parameters as well as display multipathed devices. For example, the current multipath device status, including the state, serial number, priority, and node name can be obtained with the */sbin/multipath –v2 –ll* command.

Priority Groups

Priority groups are the central component of DM. Paths are grouped into an ordered list of Priority Groups. Each Priority Group has a Path Selector which chooses which of the Priority Group's paths is to be used for each IO request. If an IO request returns an error, then the path that it uses, gets disabled and an alternative path is tried. If all the paths in a Priority Group fail, then another Priority Group is selected. There are also management commands such as *fail_path* and *reinstate_path*, that manually disable or re-enable paths; respectively. A path tester module, which currently runs in userspace, is responsible for monitoring paths that have failed and reinstating them should they come back.
A drive can have one or more Priority Groups. At any given time; however, only one of the Priority Groups can be active. DM issues all I/O requests to the drive paths in the active priority group.
If a failed path in the active priority group is restored, DM will begin re-issuing I/O requests to it as well as. If a failed path in an inactive priority group is restored, DM will failback to the inactive priority group if and only if the group has a higher priority than the currently active group. Other management commands can be use to switch immediately to a specified Priority Group or to disable a particular Priority Group so it will only be tried after there are no more left.

The grouping policy controls how device mapper groups paths. It can be set as a multipath command option or from the "/etc/multipath.conf" file. DM offers five grouping policies:

- **failover** The failover policy places each path in a separate priority group. All priority groups have equal priority. This policy is good for low-bandwidth databases where multiple paths are for providing availability, not throughput. It is the default policy.
- **multibus** The multibus policy places all paths in one priority group. This policy provides both availability and throughput.
- **group_by_serial** In general, storage array controllers have unique serial numbers. The group_by_serial policy groups paths by serial number, placing paths through the same controller in the same priority group. It can be used to failover between active/passive controllers and to load-balance between paths through the active controller.
- **group_by_prio** The group_by_prio policy places paths with the same priority value into the same priority group. A program is specified in the

"/etc/multipath.conf" file that takes a path name as its input and returns the priority for the given path. If no such program is specified, all paths have the same priority.

- **group_by_node_name** The group_by_node_name policy places all paths with the same target node name in the same priority group.

The grouping policy is generally storage dependent. For most high-end storage arrays, the *multibus* policy is the recommended setting.

Device-mapper devices

As discussed above, device mapper devices are created as /dev/dm-x devices. DM devices use a linear name space, i.e., they are named "/dev/dm-0", "/dev/dm-1", "/dev/dm-2", etc. DM only handles whole drives, if a drive has multiple partitions, the device mapping of each partition is handled by "kpartx". kpartx is a user-space tool in the "multipath-tool" package. It is automatically invoked whenever device mapper device reconfigurations occur.
For example, suppose a system has one drive with two paths: "sdc" and "sdj".
The drive has two partitions, "sdc1"/"sdj1" and "sdc2"/"sdj2". device mapper and kpartx will create the following dm devices:

/dev/dm-0: sdc, sdj
/dev/dm-1: sdc1, sdj1
/dev/dm-2: sdc2, sdj2

DM can be integrated with udev. A udev rule can be written so that persistent names are obtained from udev. For example, the following udev rule produces persistently named DM devices:

```
KERNEL="dm-[0-9]*", PROGRAM="/sbin/devmap_name %M %m", NAME="%k"
SYMLINK="disk/by-name/%c"
```

If the multipath.conf alias attribute for a device is specified as the input parameter, then the /sbin/devmap_name script will return that value. If the alias attribute is not specified, then the script will return the WWID of the device. This has the affect of creating a persistent device name for a particular device. If the device is partitioned, the partitions will appear as /dev/disk/by-name/<WWID>pX or /dev/disk/by-name/<alias>pX where X is the partition number.

Therefore Oracle Clusterware can leverage udev and device-mapper for its OCR and Voting disks by using the devmap_name generated paths. It is recommended that the OCR and Voting disks use /dev/mapper/*alias* path format.

Automatic Storage Management

Automatic Storage Management (ASM) is a storage management tool specifically built to simplify the job of the DBA. It provides a simple storage management interface across all servers and storage platforms. ASM provides the DBA flexibility to manage a dynamic database environment with increased efficiency. This feature is a key component of Enterprise Grid Computing and Database Storage Consolidation. A detailed overview of ASM is available on the Oracle Technology Network (OTN) ASM web site:

http://www.oracle.com/technology/products/database/asm/index.html

When ASM scans for disks, it will use the asm_diskstring parameter from the ASM instance init.ora. ASM will discover any devices that it has permissions to open. Upon successful discovery, the V$ASM_DISK view on the ASM instance, will reflect which disks were discovered. When using udev and DM based paths with ASM, the asm_diskstring must be specify the following:

asm_diskstring = '/dev/dm-*'

If ASMLib is to be used in conjunction with ASM, then see the next section for disk device paths for ASM.

If new disks are to be added to ASM diskgroups, see the section, Adding new storage device to a host, first.

ASMLib

ASMLib, which is an add-on to ASM, is a device management utility that is used to exploit the capabilities and strengths of vendor storage arrays. The purpose of ASMLib, is to provide an alternative interface for ASM to identify and access block devices. This is particularly important in an Oracle RAC environment. For a more detailed overview of ASMLib please refer the following URL:
http://www.oracle.com/technology/tech/linux/asmlib/index.html

Disks to be managed by ASMLib must have a partition table on them. Thus in the case of DM devices, the partition table must be created on one of the subpaths. For example, if /dev/sdg is to be used as an ASMLib disk, then fdisk must be used to create a partition table, producing a /dev/sdg1 device. Determine the DM device associated with /dev/sdg1. Now create the ASMLib disk using the DM device. For example,
/etc/init.d/oracleasm createdisk VOL1 /dev/dm-1

ASMLib upon startup scans the devices available on the system and discovers ASMLib labeled disks. ASMLib needs to be limited to scan only the dm devices instead of any

underlying subpaths. This can be done by specifying the following in the ASMLib configuration file `/etc/sysconfig/oracleasm`:

```
ORACLEASM_SCANORDER = "dm"
ORACLEASM_SCANEXCLUDE = "sd"
```

This must be done one each node of a RAC cluster. As stated in the ASM section, when ASM scans for disks, it will use the asm_diskstring to find any devices that it has permissions to open. Upon successful discovery, the V$ASM_DISK view on the ASM instance, will reflect which disks were discovered. When ASMLib and device-mapper based paths are used, the asm_diskstring must be specified as the following:

asm_diskstring = 'ORCL:*'

Steps for Oracle Clusterware installation on SLES 9

1. Install SUSE Linux Enterprise Server 9 with Service Pack 3 or later and include the multipath-tools package during the installation.

2. The SLES multipathing tools will not automatically detect newly created partitions when restarting the multipath tools. It is recommended that you create a partition table on all the disks that will be used for Oracle Clusterware and ASM. This can be done using `/sbin/fdisk` on the standard devices before starting the multipath tools. If you create a partition after starting the multipath tools, you will need to restart the multipath tools for the partition to appear with the other multipath devices.

3. Start the multipathing tools by running the following commands as the root user:

```
# /etc/init.d/boot.multipath start
# /etc/init.d/multipathd start
```

4. Configure the multipathing tools to run on boot by running the following commands as the root user:

```
# insserv boot.multipath multipathd
```

5. The shared storage devices should now be available as */dev/disk/by-name/<WWID>* and the partitions as */dev/disk/by-name/<WWID>pX* where X is the partition number.

6. Identify all the disk paths that will be used by Oracle Clusterware (OCR/Voting) and ASM; e.g., /dev/sdd.

7. Edit the /etc/multipath.conf file to include all multipathed devices. Use the format shown on page 10 and the "multipath –v2 –d " command to assist in generating this file. Add an alias attribute for each device. For example, use ocr*x*, for OCR; vote*x* for Voting disks, and asmx for ASM disks.

8. Run command /sbin/multipath –v2. This will invoke multipathd daemon to re-read the multipath.conf file and generate the */dev/mapper/<WWN>* and */dev/mapper/alias* device nodes.

9. Edit the /etc/raw file so that the proper raw devices are mapped to the proper multipath devices[5]. Note, the raw bindings for the ASM disks are not required. ASM can use the block devices.

```
# /etc/raw
#
raw1:/dev/mapper/ocr1
raw2:/dev/mapper/ocr2
raw3:/dev/mapper/vote1
raw4:/dev/mapper/vote2
raw5:/dev/mapper/vote3
raw6:/dev/mapper/vote4
raw7:/dev/mapper/vote5
```

Restart the raw service to pickup the new bindings.

10. Start the Oracle Clusterware installation. When prompted for the OCR and Voting disk names, enter the /dev/mapper/ocrx and /dev/mapper/votex; respectively.

[5] Associate the raw mapping to /dev/mapper devices and not the /dev/dm-x devices. The /dev/dm-x device names are not consistent across nodes.

Configuring udev and device mapper for Oracle RAC 10g Release 2 on SLES9

Feburary 2006
Author: Nitin Vengurlekar
Contributing Authors: Joel Becker , Kevin Lyon (Novell), Xiaoming Liu

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