Violin MEMORY

Best Practices

Best Practices Guide for Oracle® Database

Deployment and Tuning Recommendations for Oracle Database and Grid Infrastructure on Violin Flash Storage Platform (FSP)

Document Number: 535-0270-00 Rev 01
July 2016

LEGAL NOTICE

Violin Memory, Inc.
4555 Great America Parkway
Santa Clara, CA 95054
USA

Table of Contents

1 Introduction

1.1 Disclaimer

This document describes the process for building a generic system and does not take into account individual customer's requirements for security, performance, resilience and other operational aspects that may be relevant. Customers with existing operational guidelines should treat those guidelines with higher priority – and where any advice in this document conflicts with existing policies, those policies should be prioritized. Violin Memory does not accept liability for any issues experienced as a result of following this document.

1.2 About This Document

This document contains generic recommendations regarding the configuration of Oracle® as well as specific recommendations based on the Microsoft Windows and Linux operating systems. For customers using other operating systems, care should be taken to ensure that the concepts described for Microsoft® Windows® and/or Linux are followed where applicable. In addition, for simplicity, this document assumes the use of fibre-channel for connectivity with Violin Memory storage. If another connectivity option is to be used, consult the Violin documentation for connection specifics; all other recommendations still apply.

2 High Level Recommendations

Violin Memory makes the following recommendations for the use of Oracle Database software with Violin FSPs:

- Oracle Database and Grid Infrastructure software of version 11g Release 2 or later is recommended to support 4K sector storage devices, specifically allowing creation of 4K blocksize redo logs
- Use Oracle Automatic Storage Management (ASM) to achieve near-raw performance from block devices and avoid the overhead and manual file distribution of using (multiple) filesystems
- For deployments utilizing Oracle ASM, the use of ASMLib (Linux only) is also recommended for its IO optimizations, device labeling and ownership persistence, and its support of 4K native devices, such as the Violin 6000 All Flash Array
- Avoid the unnecessary use of partitions for LUNs. If partitions are used, care should be taken to ensure the partition begins on the boundary of a 4K block (see Appendix A: Partitioning LUNs).
- Databases placed on Violin Memory should have a database block size of 4K or greater (e.g. the default value of 8K is acceptable) for optimal performance
- Avoid use of striped LVM volume groups and other software striping on top of Violin LUNs, as this can impose fixed stripe sizes/widths that decrease efficiency for Oracle access. Note: ASM is software RAID specifically designed for use with Oracle Database and does not suffer from these inefficiencies.

2.1 LUN Count and Sizing Recommendations

The design of Violin Memory FSP allows for a single LUN to deliver the full performance capability of the array. However, since this performance capability is so high, many operating systems exhibit bottlenecks at the OS queue level if a single LUN is used. For this reason, Violin recommends using LUNs in groups of four (4) for each data storage point (e.g. ASM diskgroup), with extremely high performance configurations (greater than 200,000 IOPS requirement) having eight (8) LUNs per group.

- ASM diskgroups containing database data or fast recovery areas should comprise of no less than 4 LUNs, with very high performance groups having 8 LUNs; LUN counts should always be powers of 2
- If multiple arrays are used, the above recommendation should be adapted to allow a minimum of 2 LUNs per diskgroup from each array. For example, a +DATA diskgroup spread over four arrays would have 2 LUNs per array, making 8 LUNs in total
- For locations containing files with low I/O requirements (e.g. database parameter files, +SYSTEMDG diskgroups, etc.), the recommendation to use 4 LUNs can be ignored
- To avoid the unnecessary overhead of ASM rebalances, Violin recommends that customers do not add LUNs to diskgroups to increase capacity, but instead increase the size of existing LUNs (see Appendix C: Adding Storage to ASM Diskgroups).

3 Creating and Presenting LUNs from Violin FSP

The process of creating and exporting LUNs to be presented to the database server(s) is described in the Violin Memory Array User's Guide. The relevant version of this documentation can be found by accessing the Help section within the Violin Symphony Web management interface:

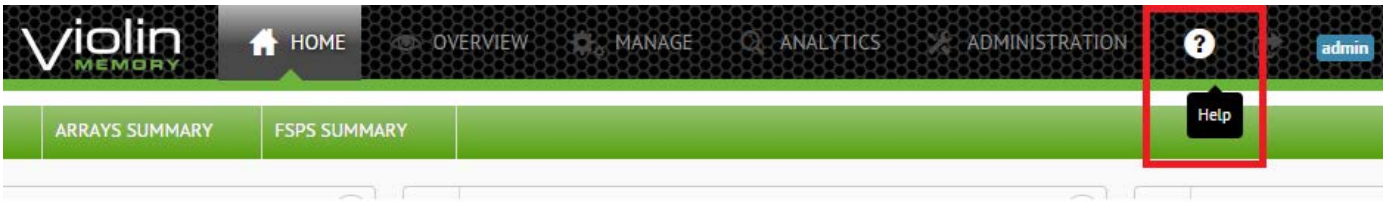

Figure 3-1. Symphony context-sensitive help

For reference, the Symphony 3.3 LUN creation screen is shown in Figure 3-2. Batch creation mode may be used to quickly create multiple LUNs of the same type and size.

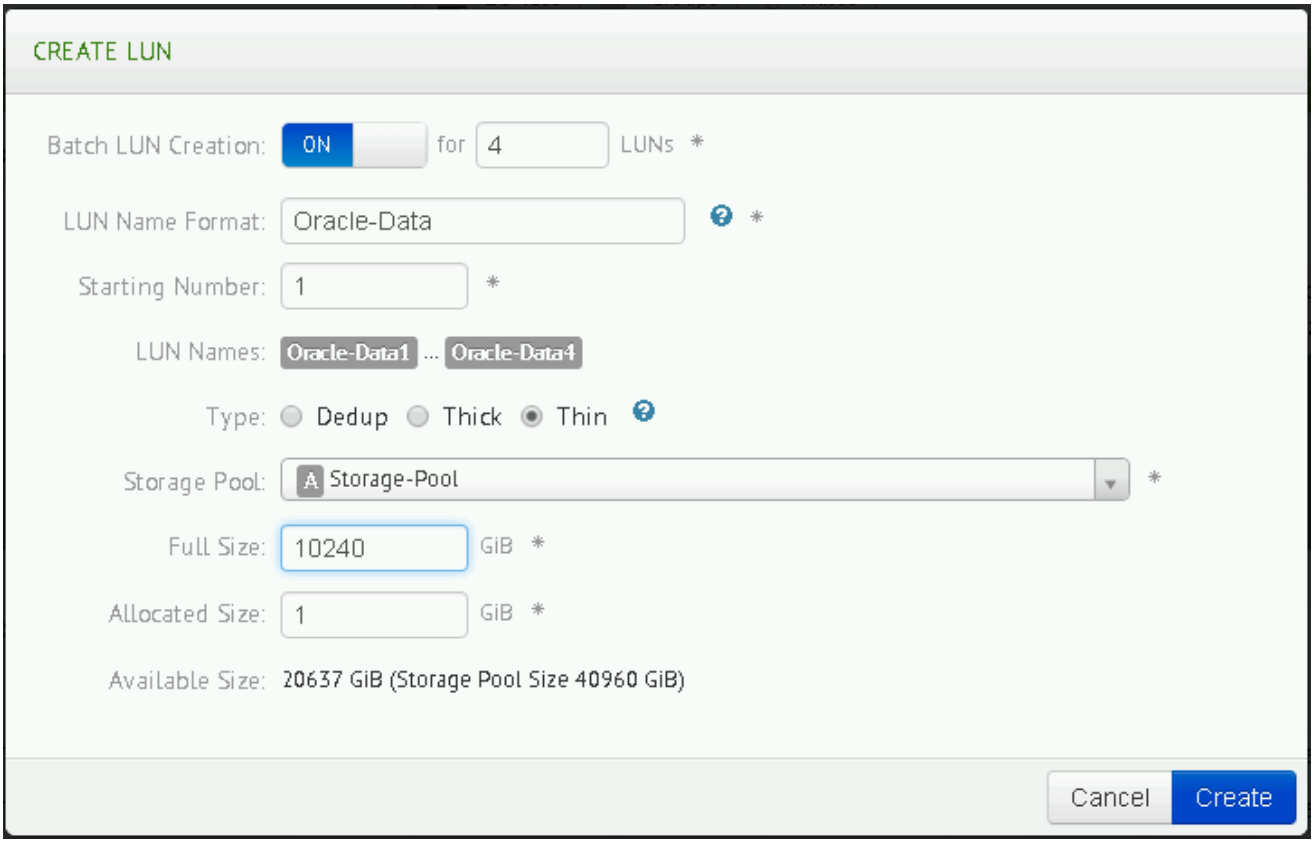

Figure 3-2. Symphony LUN creation dialog

Create LUNs of the type desired (thick-provisioned, thin-provisioned, or data-reduced) for each application based on desired performance and capacity characteristics. Thick-provisioned LUNs provide the lowest possible latency, while immediately consuming all space allocated to them. Thin-provisioned LUNs provide comparable latency to thick LUNs, with allocated space

being consumed as the data set grows. Data-reduced ("Dedup") LUNs may exhibit higher latency than the other LUN types, but they benefit from block deduplication followed by compression of all blocks stored on the physical medium, often providing significant capacity savings.

When considering deploying snapshots and thin clones of the LUNs created here, create all LUNs for a single logical storage purpose (e.g. an Oracle database) in the same storage pool.

4 Operating System Setup

This guide covers the use of Linux and Microsoft Windows operating systems (OSs). If other OSs are used, care should be taken to ensure that any similar concepts are followed, such as the configuration of multipathing software and the tuning of I/O schedulers and queues as allowed by the particular OS.

4.1 Multipathing Software

Note: If running in a virtual environment such as VMware, multipathing is likely to be handled by the hypervisor. Please review the companion document Concerto Best Practices for VMWare vSphere Deployment for such implementations. If the Violin Memory array is virtualized behind storage virtualization software such as USPV or IBM SVC, then please follow the advice from HDS or IBM on the correct configuration for multipathing.

Multipathing software allows for resilience when multiple paths exist between storage devices and servers. The multipathing software amalgamates multiple paths to a single physical device into a single virtual device, which can then be referenced by Oracle Database and Oracle ASM. The primary benefit of this virtual device is that any underlying path failure can be tolerated, provided there is at least one remaining path available. This capability is essential for Oracle products because they have no built-in functionality to perform the same task.

An additional benefit of multipathing software is the performance increase which can be gained by spreading I/O requests over numerous underlying paths. This is of particular importance when using high-performance storage such as Violin Memory arrays.

4.2 Configuration of Multipathing on Linux

4.2.1 Linux Multipath Daemon

Most versions of Linux have the multipathing daemon installed by default, but they do not always have it enabled. Check using the *chkconfig* command and enable if necessary:

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

This multipathing software uses a configuration file located in */etc/multipath.conf* to which entries for Violin Memory arrays must be added. The Violin Memory Interoperability Best Practices Guide contains the tested and validated entries required for using Violin Memory with the following versions of Linux:

- Red Hat Enterprise Linux 5 / Oracle Linux 5 / CentOS 5

- Red Hat Enterprise Linux 6 / Oracle Linux 6 / CentOS 6
- Red Hat Enterprise Linux 7 / Oracle Linux 7 / CentOS 7
- SUSE Linux Enterprise Server 11 SP3

For convenience, these settings are replicated in Appendix B: Linux Multipath Settings. Please confirm the settings against the Interoperability Best Practices Guide prior to implementation.

4.2.2 Starting Multipath Software (Linux only)

Once the relevant sections have been added to the Linux multipathing configuration file, the multipathing software can be started or restarted, e.g.:

```
[root ~]# service multipathd start
[root ~]# multipath
[root ~]# multipath –ll
mpathbr (V4JTEBRZQTTS) dm-1 VIOLIN ,CONCERTO ARRAY
size=100G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=50 status=active
  |- 5:0:4:0 sdb 8:16  active ready running
  |- 4:0:6:0 sdg 8:96  active ready running
  |- 5:0:7:0 sdc 8:32  active ready running
  |- 4:0:5:0 sdh 8:112 active ready running
  |- 5:0:8:0 sde 8:64  active ready running
  |- 4:0:7:0 sdf 8:80  active ready running
  |- 5:0:6:0 sdd 8:48  active ready running
  `- 4:0:8:0 sdi 8:128 active ready running
...
```

The listing shows the details of devices discovered by the multipath software. Initially these have default names which may not be sensible (e.g. /dev/dm-1, /dev/dm-0, etc.), but the “multipaths” section of the multipath.conf file can be updated to add meaningful names for each device if required.

4.2.3 Adding Multipath Aliases (Linux only)

The Linux multipath virtual devices corresponding to LUNs presented from Violin will have default names which may not be useful to administrators. For ease of identification and management, Violin Memory recommends applying aliases to these devices that are more obviously associated with their corresponding targets. Possible naming conventions include the use of the array name or the intended ASM disk (e.g. “DATA1”). Note that this is not required if using Oracle’s ASMLib, as that utility includes its own persistent device naming function.

Each LUN presented from Violin has a unique identifier or “serial”. These identifiers are used to create the user-friendly aliases in the multipath configuration file, so a list of the existing LUNs needs to be used – the command *multipath –ll* will also show all existing devices known to the multipathing software:

```
[root ~]# multipath -ll | grep VIOLIN | sort
DATA1 (GL52UNUKEUDB) dm-6 VIOLIN  ,CONCERTO ARRAY
DATA2 (VHDMF4WVEIZH) dm-7 VIOLIN  , CONCERTO ARRAY
<…output truncated…>
RECO1 (1Z3C16IYI9YF) dm-11 VIOLIN  , CONCERTO ARRAY
RECO2 (V4JTEBRZQTTS) dm-12 VIOLIN  , CONCERTO ARRAY
<…output truncated…>
SYSTEMDG (NWXCNQP2HBH8) dm-16 VIOLIN  , CONCERTO ARRAY
```

Based on these values, entries could be added to the /etc/multipath.conf file as such:

```
multipaths {
    multipath {
        wwid GL52UNUKEUDB
        alias DATA1
    }
    multipath {
        wwid VHDMF4WVEIZH
        alias DATA2
    }
<…output truncated…>
    multipath {
        wwid 1Z3C16IYI9YF
        alias RECO1
    }
    multipath {
        wwid V4JTEBRZQTTS
        alias RECO2
    }
<…output truncated…>
multipath {
        wwid NWXCNQP2HBH8
        alias SYSTEMDG
    }
}
```

The final step in the process is now to flush the device mapper and order multipath to pick up the new configuration:

```
[root ~]# multipath -F
[root ~]# multipath –v1
DATA1
DATA2
DATA3
DATA4
<…output truncated…>
```

The devices now exist in the /dev/mapper directory as expected:

```
[root ~]# ls -l /dev/mapper/DATA*
lrwxrwxrwx 1 root root 7 Oct  7 13:44 /dev/mapper/DATA1 -> ../dm-6
lrwxrwxrwx 1 root root 7 Oct  7 13:44 /dev/mapper/DATA2 -> ../dm-7
lrwxrwxrwx 1 root root 7 Oct  7 13:44 /dev/mapper/DATA3 -> ../dm-8
lrwxrwxrwx 1 root root 7 Oct  7 13:44 /dev/mapper/DATA4 -> ../dm-9
<…output truncated…>
```

4.3 Adjusting I/O Scheduler Properties Using UDEV Rules (Linux only)

The I/O scheduler determines the way in which block I/O operations are submitted to storage. There are a number of different I/O schedulers available in the Linux kernel by default, but a common theme in their behavior is the aim to reduce the impact of hard drive “seek time”. Flash memory has no concept of seek times and exhibits latencies which are frequently less than a millisecond, so there is minimal gain—and some small processing overhead—from using this scheduler. Tests have consistently shown a performance increase when switching to the simple noop scheduler.

In order to set all Violin Memory devices to use these values consistently, a UDEV rule must be created. UDEV is the Linux device manager which dynamically creates and maintains the device files found in the /dev directory. UDEV uses a number of rules files

located in the */etc/udev/rules.d* directory, so the new file should be created there. The name of the file – and its contents – will be dependent on the version of Linux in use.

Note: If the Violin array is virtualized behind storage virtualization software such as USPV or IBM SVC, or if the operating system is running within a hypervisor such as VMware, then customization may be required to the rules files shown below. Please consult the storage virtualization provider for guidance in those cases.

4.3.1 Red Hat Enterprise Linux 5 / Oracle Linux 5

Create a file with the name 60-vshare.rules:

```
[root ~]# vi /etc/udev/rules.d/60-violin.rules
```

This file will contain the following UDEV rules (take care not to introduce any additional carriage returns):

```
# UDEV rules for RH5 and OL5 with Violin Memory FSP SCSI devices
KERNEL=="sd*[!0-9]|sg*", BUS=="scsi", SYSFS{vendor}=="VIOLIN", SYSFS{model}=="CONCERTO
ARRAY*", RUN+="/bin/sh -c 'echo noop > /sys/$devpath/queue/scheduler && echo 256 >
/sys/$devpath/queue/nr_requests'"
```

Finally, UDEV must be told to reread and apply the new rules:

```
[root ~]# udevcontrol reload_rules
[root ~]# udevtrigger
```

4.3.2 Red Hat Enterprise Linux 6 / Oracle Linux 6

Create a file with the name 12-violin.rules:

```
[root ~]# vi /etc/udev/rules.d/12-violin.rules
```

This file will contain the following UDEV rules (take care not to introduce any additional carriage returns):

```
# UDEV rules for RH6 and OL6 with Violin Memory FSP SCSI devices
KERNEL=="sd*[!0-9]|sg*", ATTRS{vendor}=="VIOLIN", ATTRS{model}=="CONCERTO ARRAY",
ATTR{queue/nr_requests}="256", ATTR{queue/scheduler}="noop", ATTR{queue/rotational}="0",
ATTR{queue/rq_affinity}="1", ATTR{queue/nomerges}="1", ATTR{queue/add_random}="0"
#
# UDEV rules for RH6 and OL6 with Violin Memory FSP multipath devices
KERNEL=="dm-[0-9]*", PROGRAM=="/sbin/scsi_id --whitelisted --replace-whitespace --
page=0x80 --device=%N", RESULT=="SVIOLIN_CONCERTO_ARRAY*", OWNER:="oracle", GROUP:="dba",
MODE:="660", ATTR{queue/rotational}="0", ATTR{queue/nr_requests}="256",
ATTR{queue/scheduler}="noop", ATTR{queue/rotational}="0", ATTR{queue/rq_affinity}="1",
ATTR{queue/nomerges}="1", ATTR{queue/add_random}="0"
```

Finally, UDEV must be told to reread and apply the new rules:

```
[root ~]# udevadm control --reload-rules
[root ~]# udevadm trigger
```

4.3.3 Red Hat Enterprise Linux 7 / Oracle Linux 7

Create a file with the name 20-violin.rules:

```
[root ~]# vi /etc/udev/rules.d/20-violin.rules
```

This file will contain the following UDEV rules (take care not to introduce any additional carriage returns):

```
# UDEV rules for RH7 and OL7 with Violin Memory FSP SCSI devices
KERNEL=="sd*[!0-9]|sg*", ATTRS{vendor}=="VIOLIN", ATTRS{model}=="CONCERTO ARRAY",
ATTR{queue/nr_requests}="256", ATTR{queue/scheduler}="noop", ATTR{queue/rotational}="0",
ATTR{queue/rq_affinity}="1", ATTR{queue/nomerges}="1", ATTR{queue/add_random}="0"
#
# UDEV rules for RH7 and OL7 with Violin Memory FSP multipath devices
KERNEL=="dm-[0-9]*", PROGRAM=="/usr/lib/udev/scsi_id --whitelisted --replace-whitespace --
page=0x80 --device=%N", RESULT=="SVIOLIN_CONCERTO_ARRAY*", OWNER:="oracle", GROUP:="dba",
MODE:="660", ATTR{queue/rotational}="0", ATTR{queue/nr_requests}="256",
ATTR{queue/scheduler}="noop", ATTR{queue/rotational}="0", ATTR{queue/rq_affinity}="1",
ATTR{queue/nomerges}="1", ATTR{queue/add_random}="0"
```

Finally, UDEV must be told to reread and apply the new rules:

```
[root ~]# udevadm control --reload-rules
[root ~]# udevadm trigger
```

4.3.4 SUSE Linux Enterprise Server 11 SP3

Create a file with the name 12-violin.rules:

```
[root ~]# vi /etc/udev/rules.d/12-violin.rules
```

This file will contain the following UDEV rules (take care not to introduce any additional carriage returns):

```
KERNEL=="sd*[!0-9]|sg*", BUS=="scsi", SYSFS{vendor}=="VIOLIN", RUN+="/bin/sh -c 'echo noop
> /sys/$devpath/queue/scheduler'"
KERNEL=="sd*[!0-9]|sg*", BUS=="scsi", SYSFS{vendor}=="VIOLIN", RUN+="/bin/sh -c 'echo 0 >
/sys/$devpath/queue/rotational'"
KERNEL=="sd*[!0-9]|sg*", BUS=="scsi", SYSFS{vendor}=="VIOLIN", RUN+="/bin/sh -c 'echo 256
> /sys/$devpath/queue/nr_requests'"
KERNEL=="sd*[!0-9]|sg*", BUS=="scsi", SYSFS{vendor}=="VIOLIN", RUN+="/bin/sh -c 'echo 1 >
/sys/$devpath/queue/rq_affinity'"
KERNEL=="sd*[!0-9]|sg*", BUS=="scsi", SYSFS{vendor}=="VIOLIN", RUN+="/bin/sh -c 'echo 1 >
/sys/$devpath/queue/nomerges'"
KERNEL=="sd*[!0-9]|sg*", BUS=="scsi", SYSFS{vendor}=="VIOLIN", RUN+="/bin/sh -c 'echo 0 >
/sys/$devpath/queue/add_random'"
ENV{DM_UUID}=="mpath-36001b97?*", RUN+="/bin/sh -c 'echo noop >
/sys/$devpath/queue/scheduler'"
ENV{DM_UUID}=="mpath-36001b97?*", RUN+="/bin/sh -c 'echo 0 >
/sys/$devpath/queue/rotational'"
ENV{DM_UUID}=="mpath-36001b97?*", RUN+="/bin/sh -c 'echo 64 >
/sys/$devpath/queue/nr_requests'"
ENV{DM_UUID}=="mpath-36001b97?*", RUN+="/bin/sh -c 'echo 1 >
/sys/$devpath/queue/rq_affinity'"
ENV{DM_UUID}=="mpath-36001b97?*", RUN+="/bin/sh -c 'echo 1 >
/sys/$devpath/queue/nomerges'"
ENV{DM_UUID}=="mpath-36001b97?*", RUN+="/bin/sh -c 'echo 0 >
/sys/$devpath/queue/add_random'"
```

Caution: The value for NR_REQUESTS is different between the SCSI (sd*) and multipath (dm-*) devices.

Finally, UDEV must be told to reread and apply the new rules:

```
[root ~]# udevadm control --reload-rules
[root ~]# udevadm trigger
```

4.4 Verifying I/O Attribute Settings (Linux only)

The Linux kernel offers a means of querying the running configuration through the use of the "SYSFS" filesystem – a collection of files placed in a directory structure under the /sys top-level directory.

SCSI devices using the naming convention of /dev/sd* can be queried by examining files in the directory /sys/block/<device-name>/queue, for example (using device /dev/sdc):

```
[root ~]# cd /sys/block/sdc/queue
```

Multipath devices using the naming convention of /dev/dm-* can also be queried in the directory using this same naming convention, for version 6 and later (e.g. RHEL6, Oracle Linux 6, RHEL7, etc). Note that the dm-* device name must be used, not the corresponding /dev/mapper/* device name (which is actually a symbolic link). For example (using device /dev/mapper/DATA1):

```
[root ~]# ls -l /dev/mapper/DATA1
lrwxrwxrwx  1 root root    8 May  9 00:32 /dev/mapper/DATA1 -> ../dm-0
[root ~]# cd /sys/block/dm-0/queue
```

Careful examination of the appropriate UDEV rule will result in a list of files to check. For example, the I/O scheduler can be examined to ensure that "noop" is in use:

```
[root queue]# cat scheduler
[noop] deadline cfq
```

Care should be taken to ensure that all devices presented from Violin Memory have the correct settings as configured in the UDEV rules file, both for SCSI devices and multipath devices (where applicable).

4.5 Microsoft Windows Multipathing

Microsoft Server does not have the MPIO feature enabled by default. Verify this in the Tools list under Server Manager -> Local Server, as illustrated in Figure 4-1.

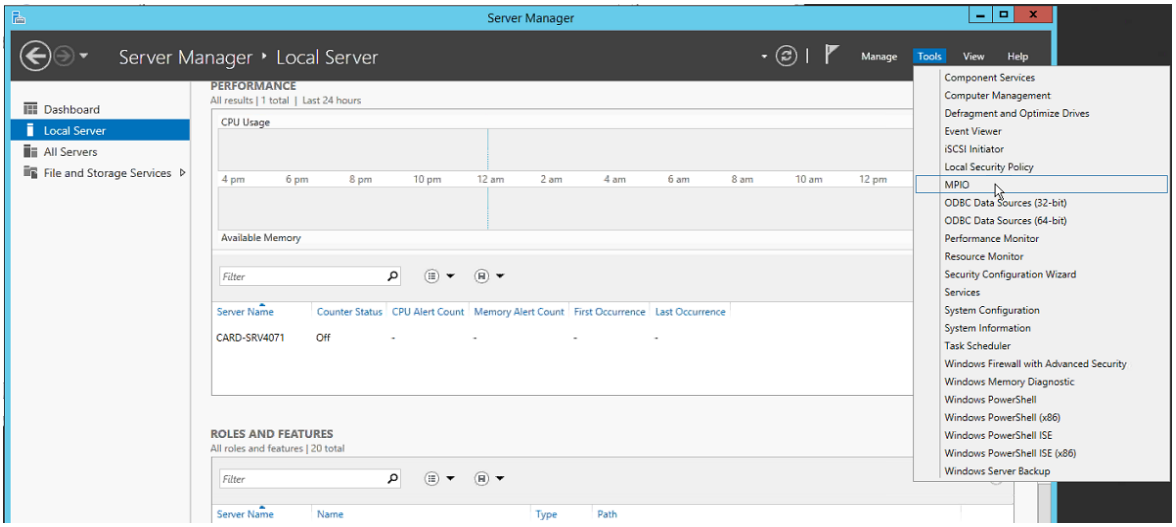

Figure 4-1. Verifying MPIO is enabled on MS Windows

If necessary, enable MPIO by checking the selection box at Server Manager -> Local Server -> Add Roles and Features -> Features -> Multipath I/O, as in Figure 4-2.

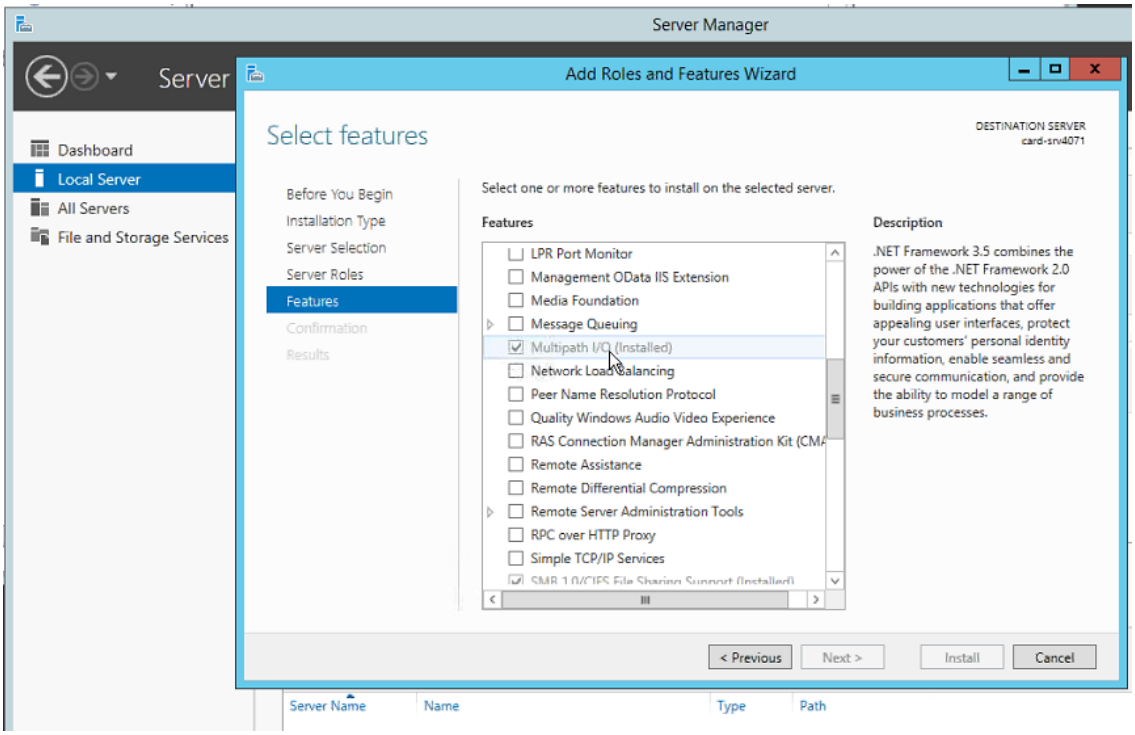

Figure 4-2. Enabling MPIO on MS Windows

Once MPIO is enabled, confirm the Violin Concerto Array is listed under MPIO Devices, as in Figure 4-3.

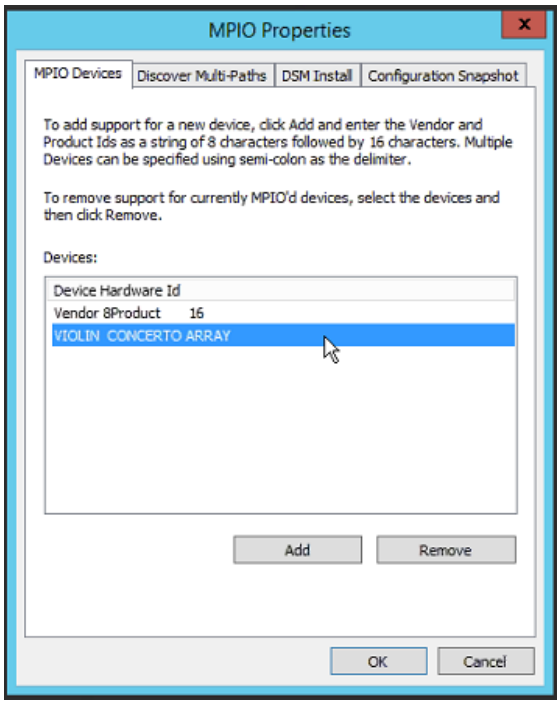

Figure 4-3. Violin Concerto Array is listed under MPIO Devices

If the array is not listed here, select it under the Discover Multi-Paths tab and click Add.

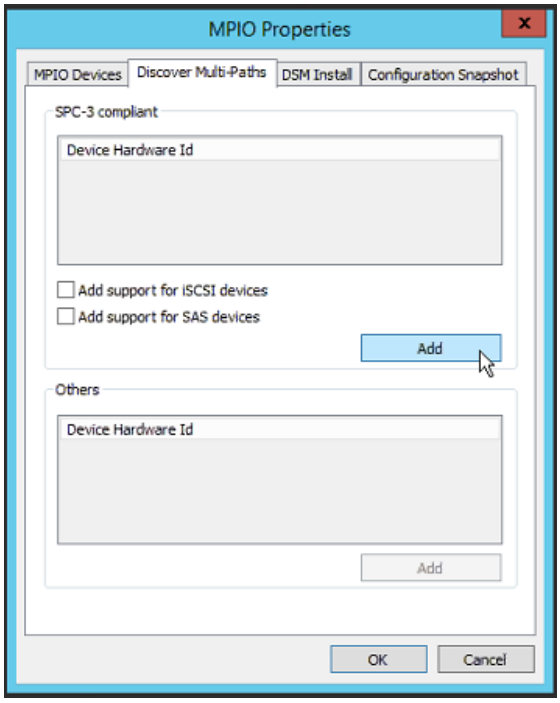

Figure 4-4. Add Violin Concerto Array under MPIO Devices if not there already

Microsoft Windows defaults to the Round Robin policy, and there is no need to change this from its default setting. Verify the MPIO policy in use by selecting a particular Disk in Storage Manager (Figure 4-5), followed by reviewing the settings on the MPIO tab (Figure 4-6).

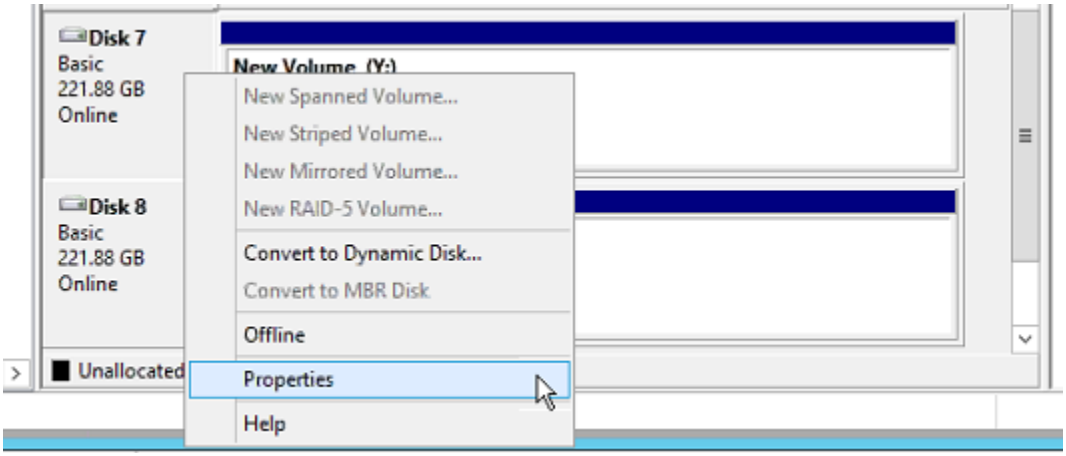

Figure 4-5. Examine the properties of a Violin Concerto Array based Windows Disk

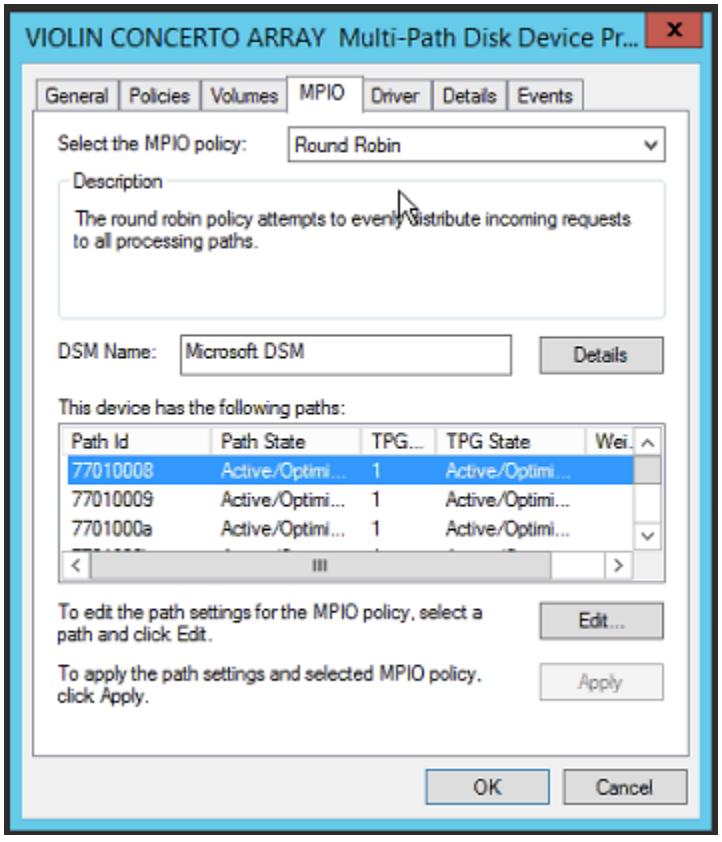

Figure 4-6. MPIO properties of a Violin Concerto Array Windows Disk

5 Virtualized Oracle Considerations

Oracle databases are increasingly run inside virtual machines, with Oracle VM and VMWare ESXi being two popular hypervisor platforms. Consider the following recommendations when deploying Oracle Database inside a VM using Violin Memory FSP storage:

- Follow all best practice recommendations for configuring the host and hypervisor per the manufacturer; further follow recommendations in the companion document *Best Practices Guide for VMWare vSphere*
- Follow LUN count and sizing guidelines in this document just as for a physical host-based deployment
- Multipathing should be handled by the hypervisor, so setting up DM-MPIO in the guest OS is unnecessary
- When defining storage for Oracle ASM, create one virtual disk on each data store, and only one data store per Violin LUN; this will allow snapshots of the database to be easily cloned and mapped to the same or other hosts
- Create all virtual disks as thick-provisioned, eager-zeroed
- Assign virtual disks evenly across multiple Paravirtual SCSI controllers (4 controllers recommended) (VMWare ESXi™)
- All storage devices presented to a VM are seen as "VMware" devices by the guest OS and cannot be identified using vendor strings as shown in the UDEV rules section of this document. As a result, other means of in-guest device identification are recommended. Violin Memory recommends use of ASMLib in such deployments to preserve device names and ownership. As an alternative, if the system is a dedicated database server, a blanket UDEV rule may assign ownership and permissions of all SCSI storage devices as follows:

```
# UDEV rules for RH6/OL6 and RH7/OL7 with SCSI devices in VMWare
KERNEL=="sd*[!0-9]|sg*", OWNER:="oracle", GROUP:="dba", MODE:="660",
ATTR{queue/nr_requests}="256", ATTR{queue/scheduler}="noop", ATTR{queue/rotational}="0",
ATTR{queue/rq_affinity}="1", ATTR{queue/nomerges}="1", ATTR{queue/add_random}="0"
```

6 Oracle ASM Configuration

The procedure for installation of Oracle Automatic Storage Management and the creation of diskgroups follows the standard process described in the Oracle documentation. No special procedures are required. Violin Memory recommends specifying EXTERNAL redundancy for most deployments, as our highly optimized and proprietary vRAID runs natively inside every Concerto Array and provides full data protection for that array. For configurations using multiple Violin Memory arrays with Oracle Automatic Storage Management, additional design considerations exist. Further assistance with this design process can be sought from the Violin Memory technical organization.

6.1 Using Multiple Arrays with Oracle ASM

6.1.1 ASM Redundancy Options

As the Violin FSP comes in all cases with built-in highly redundant and efficient vRAID protection, the standard recommendation for ASM redundancy is to use the EXTERNAL option, i.e. where data will not be mirrored by ASM. However, for multi-array configurations there are situations where NORMAL or HIGH redundancy may be preferable:

- Oracle Real Application Clusters extended cluster configurations where data must be mirrored across sites
- Configurations which require the highest level of availability where data will be mirrored across multiple arrays (thereby adding ASM redundancy on top of vRAID)
- Diskgroups containing the critical Oracle files listed in the following section Best Practice for Multi-Array Configurations

6.1.2 Best Practice for Multi-Array Configurations

For any multi-array configuration, Violin Memory recommends that the following critical Oracle files should be mirrored across more than one array:

- Oracle Database control files
- Oracle Database online redo logs
- Oracle Clusterware voting disks
- Oracle Clusterware cluster registry (OCR)

This can be achieved either through the creation of multiplexed files (for example by placing online redo log members on separate arrays) or through the use of ASM mirroring (the creation of a single ASM diskgroup across multiple arrays using NORMAL or HIGH redundancy).

6.1.3 ASM Failure Groups and ASM Preferred Read

Oracle ASM allows the configuration of *failure groups* to define units of self-contained storage (see the Oracle Automatic Storage Management documentation for details). For multi-array configurations, Violin recommends that each array be defined as a single failure group.

The exception to this recommendation applies to Oracle extended RAC configurations, where each group of arrays per site should be defined as a failure group. Consideration should also be given to the use of ASM Preferred Read to ensure that RAC nodes read from local extents rather than extents based at a remote site. See the Oracle Automatic Storage Management documentation for further details.

7 Oracle on File Systems Configuration

While Violin Memory does recommend the use of ASM for Oracle Database object storage, modern file systems can provide similar performance and ease of use if configured properly.

7.1 Set filesystemio_options = SETALL

My Oracle Support has a thorough note discussing this initialization parameter, its benefits and caveats at: *Things To Consider For Setting filesystemio_options And disk_asynch_io (Doc ID 1987437.1)*.

In Summary, while ASM always performs direct, asynchronous IO requests, most file systems by default perform buffered IO requests that may also be synchronous. Since Oracle may only access data blocks from its own buffer cache (for cached reads only; does not apply to direct path reads), having the OS perform a second layer of caching serves to add a step to the process when accessing data on a file system, forcing the buffer to be copied from storage to file system buffer to Oracle buffer space. In this mode, a read may be serviced from the host memory yet appear to Oracle to have been serviced by underlying storage, which throws off Oracle's accounting of IO service times. Worse, a write could be acknowledged by the file system, only to be flushed to physical media at some later point, creating the possibility for data loss or corruption in the event of an outage.

With these considerations in mind, Violin Memory strongly recommends setting this parameter and increasing the Oracle buffer cache allocation if necessary to make up for the loss of caching that was provided by the OS previously. Further assistance with understanding this setting can be sought from the Violin Memory technical organization.

7.2 Avoid LUN partitions

Most modern file systems do not require partitions on block storage devices. As partitions present an opportunity to cause file system blocks to be unaligned with the underlying physical media, it is safest to avoid their use unless necessary. If a partition is required for the chosen file system, please refer to Appendix A of this document for example syntax to create a properly aligned partition.

8 Creation of Oracle Databases

To achieve optimal performance, there are two elements to consider when configuring the Oracle database to run on 4K physical sector storage, such as the Violin Memory FSP:

- Database block size (set by parameter **db_block_size**): Allowable values for this parameter in Oracle are 2K, 4K, 8K (the default), 16K, and 32K. In order to ensure 4K alignment and optimal performance, **values of 4K or greater** should always be used with Violin Memory.
- Online redo log block size: by default this is 512 bytes, but as of Oracle 11g Release 2 it can be changed using the BLOCKSIZE clause. To achieve optimal performance on Violin Memory **this should be set to 4K**.

8.1 Creating Databases

There are no special procedures required during the creation of databases. However, once a database has been successfully created, it is **strongly recommended** to change the block size of all online redo logs to 4096 bytes (4K). This is only possible in Oracle Database 11g Release 2 and later versions – and it has been demonstrated to have a significant performance benefit.

To create online redo logs with a block size of 4096 bytes, it is first necessary to set the Oracle parameter *_disk_sector_size_override* to *true* (its default value is *false*). This parameter allows Oracle to ignore the fact that the underlying diskgroup uses a sector size of 512 bytes. The parameter should be set in all database instances as follows:

```
SQL> alter system set "_disk_sector_size_override"="true" scope=both;
```

Note that since this is an "underscore" parameter, it is usually necessary to seek approval from Oracle Support if any support contract is in place with Oracle. Oracle's support note on the topic of using flash storage with ASM describes setting the parameter, and experience has shown that approval is always granted. See this My Oracle Support document for further details:

USING 4K REDO LOGS ON FLASH AND SSD-BASED STORAGE (DOC ID 1681266.1)

Once the parameter has been set on all instances, existing online redo log groups should be dropped and replaced with corresponding redo log groups that use the 4096 byte block size. For example:

```
SQL> alter database orcl add logfile group 4 ('+RECO') size 500M blocksize 4k;
```

Note that the block size of any redo log group can be seen by querying the dynamic view v$log.

9 Appendix A: Partitioning LUNs

Violin Memory recommends against the use of partitions on LUNs, whether they are presented to Oracle ASM or used for file systems. Neither ASM nor most modern file system types require partitioned LUNs. However, if partitions must be used, it is essential to ensure that each partition begins on the boundary of a 4K physical block, otherwise performance can be affected.

In Linux versions 5 and 6, through the use of either fdisk or parted with default options, partitions will not be aligned on these 4K boundaries, so the following adjustments are recommended to override this behavior.

9.1 Linux fdisk and GPT partitions

Fdisk by default assumes 63 sectors per track. This can be overridden by specifying a value, as in the first example below. This does not change the geometry of the device but merely causes the partition to start at sector 1024, which lies on a 4K boundary. Another method is to specify the use of sectors as the units for fdisk, which then causes the default starting sector to be 1024.

The GPT partition type created by the parted utility requires some space for partition information at the beginning of a device. Reserve 1024 sectors (512KB) for the GPT partition as demonstrated in the second example below.

Examples:

fdisk:

```
fdisk –S 1024 /dev/mapper/vmem
```

or

```
fdisk –u /dev/mapper/vmem
```

parted:

```
parted /dev/mapper/vmem mklabel gpt
parted /dev/mapper/vmem -s -- mkpart primary 1024s -0
parted /dev/mapper/vmem unit s print
```

The above parted commands will create a single 4K-aligned partition on the device starting at sector 1024 and ending at the end of the LUN.

Linux version 7 has an updated fdisk utility that defaults to use of sectors for units and defaults to sector 2048 as the start for any partition. This results in a 1MB offset from the start of the LUN, and the partition is thereby aligned on a 4K boundary.

Further assistance with these commands may be sought from the Violin Memory technical organization.

9.2 Using ASMLib with Non-Partitioned LUNs

Violin Memory recommends against the use of partitions on LUNs presented to Oracle ASM, but the Oracle ASMLib kernel driver responds with an error when an attempt is made to label a single-path device (via the “createdisk” command) that has not been partitioned. Note that this error does not occur when using the preferred option of presenting LUNs via the Linux multipath

software. There are occasions, however, when multipath software will not be required (for example, when installing Oracle within a virtual machine whereby multipathing functionality is handled at the hypervisor layer).

The following command can be used to override the ASMLib partition check. See the My Oracle Support website for further details.

```
# /usr/sbin/asmtool -C -l /dev/oracleasm -n DATA1 -s /dev/sdc -a force=yes
asmtool: Device "/dev/sdc" is not a partition
asmtool: Continuing anyway
```

10 Appendix B: Linux Multipath Settings

The Violin Memory Interoperability Best Practices Guide contains the tested and validated entries required for using Violin Memory with the following versions of Linux:

- Red Hat Enterprise Linux 5 / Oracle Linux 5
- Red Hat Enterprise Linux 6 / Oracle Linux 6
- Red Hat Enterprise Linux 7 / Oracle Linux 7
- SUSE Linux Enterprise Server 11 SP3

For convenience, these entries are replicated here – however, please obtain the latest version of the Interoperability Guide and confirm the entries printed here. In addition, for configurations where Boot From SAN is required, follow the instructions in the Interoperability Guide.

10.1 Red Hat Enterprise Linux 5 / Oracle Linux 5

If no multipath.conf file currently exists, a blank file should be created with the following content:

```
blacklist {
        devnode "*"
}
blacklist_exceptions {
        devnode "sd*"
}
defaults {
        user_friendly_names yes
}
```

For new or existing files, the following Violin configuration should be added to the *devices* section:

```
#Concerto LUNs
    device {
        vendor                  "VIOLIN"
        product                 "CONCERTO ARRAY"
        path_grouping_policy    multibus
        path_selector           "round-robin 0"
        getuid_callout          "/sbin/scsi_id -p 0x80 -g -u –s /block/%n"
        path_checker            tur
        failback                immediate
        rr_weight               priorities
        no_path_retry           300
        rr_min_io               100
        prio                    alua
        features                "1 queue_if_no_path"
    }
}
```

10.2 Red Hat Enterprise Linux 6 / Oracle Linux 6

If no multipath.conf file currently exists, a blank file should be created with the following content:

```
blacklist {
```

```
        devnode "*"
}
blacklist_exceptions {
        devnode "sd*"
}
defaults {
        user_friendly_names yes
}
```

For new or existing files, the following Violin configuration should be added to the *devices* section:

```
#Concerto LUNs
    device {
        vendor                  "VIOLIN"
        product                 "CONCERTO ARRAY"
        path_grouping_policy    multibus
        path_selector           "round-robin 0"
        getuid_callout "/lib/udev/scsi_id --whitelisted --replace-whitespace --page=0x80 -
-device=/dev/%n"
        path_checker            tur
        failback                immediate
        rr_weight               priorities
        no_path_retry           300
        rr_min_io               100
        prio                    alua
        features                "1 queue_if_no_path"
    }
}
```

10.3 Red Hat Enterprise Linux 7 / Oracle Linux 7

If no multipath.conf file currently exists, a blank file should be created with the following content:

```
blacklist {
        devnode "*"
}
blacklist_exceptions {
        devnode "sd*"
}
defaults {
        user_friendly_names yes
}
```

For new or existing files, the following Violin configuration should be added to the *devices* section:

```
#Concerto LUNs
    device {
        vendor                  "VIOLIN"
        product                 "CONCERTO ARRAY"
        uid_attribute           "ID_SCSI_SERIAL"
        path_grouping_policy    multibus
        path_selector           "round-robin 0"
        path_checker            tur
        failback                immediate
        rr_weight               priorities
        no_path_retry           300
        rr_min_io               100
        prio                    alua
```

```
        features                "1 queue_if_no_path"
    }
}
```

10.4 SUSE Linux Enterprise Server 11 SP3

If no multipath.conf file currently exists, a blank file should be created with the following content:

```
defaults {
    path_grouping_policy "failover"
}
blacklist {
    devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
    devnode "^hd[a-z]"
    devnode "^dcssblk[0-9]*"
}
blacklist_exceptions {
    property "(ID_SCSI_VPD|ID_WWN)"
}
```

For new or existing files, the following Violin configuration should be added to the *devices* section:

```
#Concerto LUNs
    device {
        vendor                          "VIOLIN"
        product                         "CONCERTO ARRAY"
        path_grouping_policy            multibus
        path_checker                    tur
        path_selector                   "round-robin 0"
        failback                        immediate
        rr_weight                       priorities
        no_path_retry                   300
        rr_min_io                       100
        prio                            alua
        features                        "1 queue_if_no_path"
    }
```

It is noted that the default value for *path_selector* changed in SLES 11 SP3 from “round-robin 0” to “service-time 0”. However, extensive testing performed by Violin Memory has consistently shown that the round-robin algorithm allows for better results.

11 Appendix C: Adding Storage to ASM Diskgroups

Storage and database administrators frequently add capacity to ASM disk groups by creating new LUNs on the storage and then adding these LUNs into ASM. When a new LUN is added, ASM's default behavior is to perform a *dynamic rebalance* operation in order to evenly spread data across all disks within a diskgroup.

However, when adding LUNs from Violin Memory there is no benefit in performing this operation, so the unnecessary I/O may be avoided. Violin Memory therefore recommends the following procedure to be used when adding storage to ASM disk groups from existing arrays:

- Instead of creating new LUNs at the storage layer, extend each existing LUN in the ASM Disk Group by the same amount (see the Violin Memory Array User's Guide for details on how to perform this operation). As per Oracle's recommendation, all LUNs for a given diskgroup should be equally sized.
- Re-scan the SCSI bus on the Operating System. On most Linux versions the following commands should be used to ensure existing devices are re-scanned. These commands will dynamically update the size of the extended LUNs at the OS level and have been shown to be non-disruptive to the Oracle database and ASM instances – although customers should fully test this procedure with any specific configuration before adopting in a production environment.

```
# rescan-scsi-bus.sh --forcerescan
# multipath -F
# multipath -r
# multipath -ll
```

  At this point, the new LUN sizes should be visible from the output of the last command.

- Resize all the disks in the ASM Disk Group using the following ASM command:

```
$ sqlplus / as sysasm
SQL> alter diskgroup DG_NAME resize all;
SQL> select name, total_mb from v$asm_disk;
```

This will cause all disks in the disk group to be resized in a single operation and expand the size of the disk group without generating unnecessary I/O.