HIGH-AVAILABILITY

Sun Solaris Volume Manager (SVM) for Clustering

High-Availability

Best Practice

Introduction

This guide explains the setup of Solaris Volume Manager (SVM) for use with clustering on Solaris and is intended to be used by customers and partners when setting up Solaris Volume Manager for a clustered environment. Solaris Volume Manager is not a new product but a new name for an old product; DiskSuite.

Solaris Volume Manager is an excellent product but does the work in software so there are performance considerations with high end solutions and often customers install hardware RAID devices but still manage the file systems with SVM. Many customers install two hardware RAID devices and use SVM to mirror between the two RAID devices.

Using SVM in a clustered environment is not very complicated but there are a few extra steps to follow when setting up what will ultimately be shared disks. The basic commands are mainly unchanged but each command must include an additional parameter to specify the 'set' be manipulated.

This guide is not intended to replace training and documentation provided by Sun and others but to be supplementary information to accelerate the successful deployment of clustering.

Planning Disk Layouts

The disks will need to be laid out in a logical fashion and if a hardware RAID device is used then this may need to be done outside of SVM control and largely outside the scope of this document. However, care should be taken to ensure that the layout is done sensibly. For instance Oracle roll-back logs should not be on the same physical disk 'spindles' as the main database.

The reliability of a cluster depends to a large extent on the reliability of the communication paths between the nodes. While network and serial are excellent paths, one of the most reliable communication paths available in a clustered environment is disk. RSF-1 heartbeats can use a very small (~10meg) partition to exchange status information. However, SVM uses 'SCSI reservation' which precludes the use of disk heartbeats on the same disk as a disk under the control of SVM. This restriction is not applicable to x86 solutions as SCSI reservation in not used with SVM for x86[1].

SCSI reservation is used by SVM as a safety feature. SCSI reservation is like a lock; one machine marks the disk with a "I'm in control" type message and while another machine can forcibly remove this 'lock' it requires the use of a extra 'flag' and the *first* machine frequently checks that the lock still exists, if the lock is changed then the *first* machine will panic as a safety mechanism to reduce the chances of data corruption on the disk.

With a hardware RAID device, like Sun's 3310 or 3510, 'logical disks' can be presented to the operating system which support SCSI reservation etc but remove this restriction. This is very desirable as the disk heartbeat is an excellent communications path but if the heartbeat is to be used without a logical disk *(logical partition)* configuration then the heartbeat must be on a disk that is not managed by SVM.

Hardware Configuration

Clustering software relies on the fact that all disks can be used and managed by all attached systems. This means that any disks or array must be configured with a single internal bus. This may require both hardware and software configuration. The following shows the single internal bus of a 3310 with hosts (nodes/machines) connected via ports 1 and 3 but each RAID device will differ.

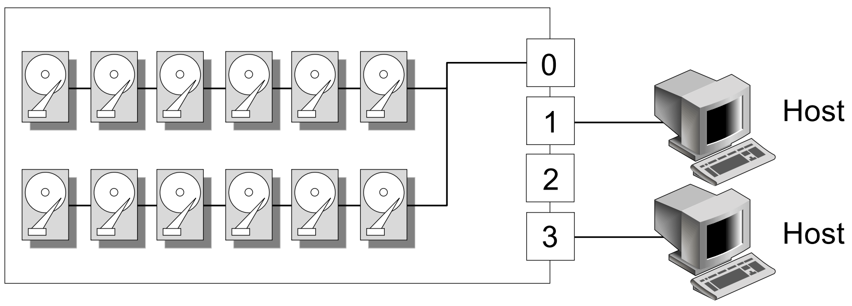

Figure 1.
Single Bus, Dual Attached

Disks which are to be shared must not be installed internal to one of the nodes, even if they are visible on a shared bus.

[1] true at the time this document was issued

BIOS / EEPROM / SCSI ID Configuration

Each node on a SCSI 'chain' must have an ID and no two machines can have the same ID, otherwise a conflict will occur. Most machines *ship* with the same default SCSI ID and as we intend to put two machines on the same SCSI chain one of the hosts will usually have to have the hostid changed. This isn't applicable for fibre channel. Sun have a global 'eeprom' parameter that changes all host adapters on a machine to the same hostid (with the exception of PCI adapters).

The setting of the global SCSI ID for all non-PCI SCSI interfaces can be done while the machine is running but does not take effect until the OS is rebooted.

To change the SCSI ID from Solaris;

```
        # eeprom scsi-initiator-id=5
```

To change the SCSI ID from the OBP

```
        ok scsi-initiator-id=5
```

The process of changing a PCI adapter SCSI ID is a bit more complex;

```
        ok probe-scsi-all                           - this creates a 'device tree' by probing the available hardware
        ok cd /sbus@1f,0/SUNW,fas@e,8800000         - this depends on the card you wish to change
        ok 5 " scsi-initiator-id" integer-property  - changes the SCSI ID
```

A fuller explanation of the process and commands can quickly be found on the web by searching with key words like "eeprom scsi id".

The choice of the 'other' SCSI ID is dependant on what other devices are in use on the same SCSI chain. The traditional (3 bit) Sun convention for numbering devices is;

SCSI ID	Use
0	1st Disk
1	3rd Disk
2	4th Disk
3	2nd Disk
4	1st Tape Drive
5	2nd Tape Drive
6	CDROM
7	Server

Figure 2.
Traditional Sun SCSI ID Assignments

High-Availability.Com usually recommend the use of SCSI ID 5 & 7 for servers on a shared SCSI chain, which means one server needs to be changed to use SCSI ID 5.

This document is written for users of Sun equipment but other brands of hardware has similar facilities. However, HP have intentionally prevented SCSI ID changes to the desktop and work station classes of machines, therefore these machines can only be used in a cluster with a shared disk where fibre channel is used or if when RAID device has an intelligent SCSI controller that prevents a conflict. Fibre channel devices use a form of serial number included in their 'address' and as such conflicts should never occur.

Meta Databases

SVM stores disk state, configuration, ownership and other information in special areas of a disk (partitions) and these are called meta databases. The information contained in the meta databases is critical & there must be multiple copies of the meta databases, warnings will be issued if less than three copies exist.

It is important to ensure that sufficient databases exist to ensure that a failure doesn't occur as the system will not boot unless more than half of the replicas are available and the system will panic if more than half of the replicas are corrupt. If one drive in a two drive mirror fails, and the system is rebooted for any reason it will not automatically (by default) boot as more that 50% of root disks are required to boot.

There are two types of meta databases; local and shared. The local must exist before shared meta databases can be established. In a two node cluster there will normally be at least nine partitions set aside for meta databases; three for each node's local databases and three for the shared databases.

Typically about 10MB of space is required for each copy of the meta database but often, with modern drives, a larger space will be allocated, as a complete *cylinder* should be used. System administrators will normally create a 'root disk' under the control of SVM, so that mirroring is performed not only on the data but also the base OS. This document will provide a worked example for reference purposes;

Setting up a booting SVM root file system

SVM is an optional component that is shipped with Solaris. If you need to install SVM the machines must be rebooted before proceeding.

Optionally, disable the safety feature that prevents booting a root disk mirrored system with only one working disk remaining issue the following command, which will take effect when the server is next rebooted;

```
# echo "set md:mirrored_root_flag=1" >> /etc/system
```

The output from **format c0t0d0,** ***partition table, print*** of a boot disk might look like the following;

Figure 3.
Example Disk Layout

Total disk cylinders available: 24620 + 2 (reserved cylinders)

Part	Tag	Flag	Cylinders	Size	Blocks
0	root	wm	0 - 283	400.62MB	(284/0/0) 820476
1	swap	wu	284 - 1701	1.95GB	(1418/0/0) 4096602
2	backup	wm	0 - 24619	33.92GB	(24620/0/0) 71127180
3	var	wm	1702 - 3136	1.98GB	(1435/0/0) 4145715
4	unassigned	wm	3137 - 3144	11.29MB	(8/0/0) 23112
5	opt	wm	3145 - 5271	2.93GB	(2127/0/0) 6144903
6	usr	wm	5272 - 11652	8.79GB	(6381/0/0) 18434709
7	home	wm	11653 - 24619	17.86GB	(12967/0/0) 37461663

Note: Partition 4 above (c0t0d0s4) has been set aside a spare partition, which we'll use for a meta database.

The partition table of the mirrored root drive should be identical to the boot drive. Copy the partition table of the boot drive to its mirror.

```
# prtvtoc /dev/rdsk/c0t0d0s2 | fmthard -s - /dev/rdsk/c0t1d0s2
fmthard: New volume table of contents now in place
```

Create the mirror for the root (/) filesystem, in this case c0t0d0 is the boot drive and c0t1d0 is the mirror. We will use c0t0d0s4 and c0t1d1s4 as meta databases. Ideally we would use different controllers for the boot and mirror disks but this depends on available resources.

```
# metadb -a -f -c2 /dev/dsk/c0t0d0s4 /dev/dsk/c0t1d0s4

# metainit -f d10 1 1 c0t0d0s0
d10: Concat/Stripe is setup
# metainit d20 1 1 c0t1d0s0
d20: Concat/Stripe is setup
# metainit d30 -m d10
d30: Mirror is setup
# metaroot d30
```

Create the mirror for all other filesystems
/var filesystem:

```
# metainit -f d11 1 1 c0t0d0s3
# metainit d21 1 1 c0t1d0s3
# metainit d31 -m d11
```

Swap filesystem:

```
# metainit -f d13 1 1 c0t0d0s1
# metainit d23 1 1 c0t1d0s1
# metainit d33 -m d13
```

/opt filesystem:

```
# metainit -f d15 1 1 c0t0d0s5
# metainit d25 1 1 c0t1d0s5
# metainit d35 -m d15
```

/usr filesystem:

```
# metainit -f d16 1 1 c0t0d0s6
# metainit d26 1 1 c0t1d0s6
# metainit d36 -m d16
```

/export/home filesystem:

```
# metainit -f d17 1 1 c0t0d0s7
# metainit d27 1 1 c0t1d0s7
# metainit d37 -m d17
```

Edit the /etc/vfstab to mount the new mirrors at boot.

Before :

```
#device to mount    device to fsck       mount point    FS type  fsck pass  mount at boot  mount options
...
/dev/dsk/c0t0d0s3   -                    -              swap     -          no             -
/dev/md/dsk/d30     /dev/md/rdsk/d30     /              ufs      1          no             -
/dev/dsk/c0t0d0s6   /dev/rdsk/c0t0d0s6   /usr           ufs      1          no             -
/dev/dsk/c0t0d0s3   /dev/rdsk/c0t0d0s3   /var           ufs      1          no             -
/dev/dsk/c0t0d0s7   /dev/rdsk/c0t0d0s7   /export/home   ufs      2          yes            -
/dev/dsk/c0t0d0s5   /dev/rdsk/c0t0d0s5   /opt           ufs      2          yes            -
...
```

Figure 4.
File Systems Prior to SVM Control

After :

```
#device to mount    device to fsck       mount point    FS type  fsck pass  mount at boot  mount options
...
/dev/md/dsk/d33     -                    -              swap     -          no             -
/dev/md/dsk/d30     /dev/md/rdsk/d30     /              ufs      1          no             -
/dev/md/dsk/d36     /dev/md/rdsk/d36     /usr           ufs      1          no             -
/dev/md/dsk/d31     /dev/md/rdsk/d31     /var           ufs      1          no             -
/dev/md/dsk/d37     /dev/md/rdsk/d37     /export/home   ufs      2          yes            -
/dev/md/dsk/d35     /dev/md/rdsk/d35     /opt           ufs      2          yes            -
...
```

Figure 5.
File Systems All Under SVM Control; Edited by Hand

Suppress warning messages

```
# metainit hsp001
```

Attach the second submirror to the mirror. This will cause the data from the boot disk to be synchronised with the mirrored drive.

```
# metattach d30 d20
# metattach d31 d21
# metattach d33 d23
# metattach d35 d25
# metattach d36 d26
# metattach d37 d27
```

You will hear a lot of disk thrashing at this point and your I/O will go through the roof.

Use **metastat** to track progress and ensure the sync is complete before rebooting the server.

Enable the mirror disk to be bootable:

```
# installboot /usr/platform/`uname -i`/lib/fs/ufs/bootblk /dev/rdsk/c0t1d0s0
```

This is the device path that you will use to define the alternate boot device at the hardware level.

```
ok nvalias mirror /pci@1f,4000/scsi@3/sd@1,0:a,raw
```

Issue a "show-disks" at the ok prompt to verify the correct path to the disk. Use "devalias" at the ok prompt to also give clues as to which device path to use. In case of primary boot disk failure, boot from the alternate disk

```
ok boot mirror
```

Creating a Shared Disk 'Set'

For metasets to work between machines (nodes) the rpc.metamedd process must be running on all nodes that will become members of the shared metaset. Further, the nodes must have /.rhosts setup to allow 'each other' access.

After setting up the 'local' meta databases then 'metaset'(s) can be created. Again we'll use an example to

SUNTONE™ CERTIFIED

illustrate this procedure. The procedure is often easier to complete if it is carried out on one node and then the other node is added to the metaset later, rather than adding machines then disks.

In the following example, we will create a metaset for use with Oracle, so we'll call the metaset oracle.

Add the first node to the metaset
# metaset -s oracle -a -h node1

Add two disks to the metaset

# metaset -s oracle -a c1t0d0 c2t1d0

Take ownership of the metaset

# metaset -s oracle -t

Create all the metadevices

```
# metainit -a -s oracle
# metainit -s oracle d11 1 1 c1t0d0s0
oracle/d11: Concat/Stripe is setup

# metainit -s oracle d12 1 1 c2t1d0s0
oracle/d12: Concat/Stripe is setup

# metainit -s oracle d10 -m d11
oracle/d10: Mirror is setup

# metattach -s oracle d10 d12
oracle/d10: submirror oracle/d12 is attached
```

Create the file system
# newfs /dev/md/oracle/rdsk/d10

Add the second node to the set
# metaset -s oracle -a -h node2

Possible Problems With Disk Suite (metasets)

Solaris assigns device names and numbers dynamically during a reconfiguration boot, by probing the attached hardware. Assignments are generally made according to the order in which hardware is detected (generally, lowest numbered slots first.)

When adding a disk to a multi-node diskset or when trying to add another node to the set you may get an error, indicating that the disk is not common between the nodes. The common fibre controller used is the Qlogic and a numbers of lines with 'qlc' would be key if this is the controller you have If you get this error then the OS links on the *new* node will need to be adjusted to match those of the existing node(s). **Always make a backup of the /etc/path_to_inst file before making changes**.

You can view the names and major & minor device numbers for the disks in a long listing of the device links.

```
crw-r-----   1 root   sys     32, 48 Nov 20 22:54 /dev/rdsk/c1t6d0s0
crw-r-----   1 root   sys     32, 49 Nov 20 22:54 /dev/rdsk/c1t6d0s1
crw-r-----   1 root   sys     32, 50 Nov 20 22:54 /dev/rdsk/c1t6d0s2
crw-r-----   1 root   sys     32, 51 Nov 20 22:54 /dev/rdsk/c1t6d0s3
crw-r-----   1 root   sys     32, 52 Nov 20 22:54 /dev/rdsk/c1t6d0s4
crw-r-----   1 root   sys     32, 53 Nov 20 22:54 /dev/rdsk/c1t6d0s5
crw-r-----   1 root   sys     32, 54 Nov 20 22:54 /dev/rdsk/c1t6d0s6
crw-r-----   1 root   sys     32, 55 Nov 20 22:54 /dev/rdsk/c1t6d0s7
```

The numbers, in the format 'major, minor', appear in the fifth column of the listing.

Enter this command (ls -lL /dev/rdsk/c1* for example) on both RSF-1 servers and compare the outputs.

For devices of the same type, the major device numbers will often be identical across different systems. However, the minor device numbers are assigned by Solaris during a reconfiguration boot and depend on what other devices of the same type have already been configured on the system. For two hosts with identical hardware configurations, these numbers should always cover identical ranges.

Before attempting to alter device names or numbers, ensure that all access to the devices is stopped and file systems on them are un-mounted.

Renaming devices

Warning: the following procedures should only be attempted by an experienced Solaris administrator!
Mistakes can render the system unbootable; if this occurs, you should contact your support provider for assistance.

The quickest and simplest way to rename devices is to rename the symbolic links in /dev/dsk and /dev/rdsk. Ensure that the new names do not conflict with existing devices and that you do not rename other devices, such as the boot disk. Use the move command to rename the controller, for example:

```
# mv /dev/dsk/c1t1d0s0 /dev/dsk/c2t1d0s0
```

Changing major device numbers

The major device numbers for each device driver are assigned in the /etc/name_to_major file. This contains a list of mappings in the form ‘drivername number’. Before changing the file, make a backup copy. Find the name of the driver you wish to change and modify the associated number. Perform a reconfiguration reboot to complete the change.

N.B. Changing the major device number for the device on which the root file system resides is not advisable and likely to render the system unbootable. If your root device has been incorporated by a disk manager such as SDS or Volume Manager and you wish to change the major device number for the disk manager, remove it from management control before proceeding.

Changing minor device numbers

The minor device numbers are calculated from the instance numbers for the devices in the /etc/path_to_inst file. Each line of the file has the following format:

“physical name” instance number “driver binding name”

The instance number is the second field on each line. The first field refers to the device file under /devices pointed to by the symbolic link in the /dev/dsk directory (you can list these links using ls -l /dev/rdsk). The third field is the name of the driver used for the device.

The minor device number is calculated from the instance number using the following formulas:

minor device number = (instance number * 8) + partition number

or

instance number = (minor device number - partition number)/8

Using this formula, you can edit the instance numbers in this file and hence alter the minor device numbers to match:

1. Make a backup copy of the /etc/path_to_inst file. You can boot with the backup copy of the file using the -a switch to the boot command.
2. Edit the original file. Locate the entries for the shared disk(s). Change the instance number as appropriate. If the instance number is already in use by a device on the same driver (third field), you must also change that entry, perhaps by swapping the numbers.
3. For all the affected devices, delete the device nodes (listed in the first field of the file above) from the /devices directory. Also delete the relevant symbolic links for each device from the /dev/dsk and /dev/rdsk directories. Be careful not to delete the device files referring to the system disk(s).
4. Perform a reconfiguration boot of the system.
5. Check that the relevant devices have been correctly renumbered by listing the /dev /dsk links.

   Note: Do not attempt to remake the device entries by running ***drvconfig***. This uses the instance numbering already in memory and will overwrite your changes to the /etc/path_to_inst file.

Here is an example extract from /etc/path_to_inst that needs to be changed from;

```
...
“/pci@1d,700000/SUNW,qlc@1” 0 “qlc”
“/pci@1d,700000/SUNW,qlc@1/fp@0,0” 0 “fp”
“/pci@1d,700000/SUNW,qlc@1/fp@0,0/sdd@w216000c0ff87e8d8,0” 0 “ssd”
“/pci@1d,700000/SUNW,qlc@1/fp@0,0/sdd@w256000c0ffc7e8d8,0” 1 “ssd”
“/pci@1d,700000/SUNW,qlc@1/fp@0,0/sdd@w256000c0ffc7e8d8,1” 2 “ssd”
“/pci@1d,700000/SUNW,qlc@1/fp@0,0/sdd@w256000c0ff87e8d8,0” 32 “ses”
“/pci@1d,700000/SUNW,qlc@1/fp@0,0/sdd@w256000c0ffc7e8d8,0” 33 “ses”
“/pci@1d,700000/SUNW,qlc@1/fp@0,0/sdd@w256000c0ff87e8d8,0” 5 “ssd”
...
“/scsi_vhci” 0 “scsi_vhci”
“/scsi_vhci/ssd@g600c0ff00000000007e8d85fe6ae6201” 3 “ssd”
“/scsi_vhci/ssd@g600c0ff00000000007e8d85fe6ae6200” 4 “ssd”
“/scsi_vhci/ssd@g600c0ff00000000007e8d80000000000” 34 “ses”
“/scsi_vhci/ssd@g600c0ff00000000007e8d85fe6ae6202” 6 “ssd”
...
```

Changing to match the other machine;

```
...
“/pci@1d,700000/SUNW,qlc@1” 0 “qlc”
“/pci@1d,700000/SUNW,qlc@1/fp@0,0” 0 “fp”
“/pci@1d,700000/SUNW,qlc@1/fp@0,0/sdd@w256000c0ffc7e8d8,0” 3 “ssd”
“/pci@1d,700000/SUNW,qlc@1/fp@0,0/sdd@w216000c0ff87e8d8,0” 4 “ssd”
...
“/scsi_vhci” 0 “scsi_vhci”
“/scsi_vhci/ssd@g600c0ff00000000007e8d80000000000” 32 “ses”
“/scsi_vhci/ssd@g600c0ff00000000007e8d878ba51fc01” 0 “ssd”
“/scsi_vhci/ssd@g600c0ff00000000007e8d878ba51fc00” 1 “ssd”
“/scsi_vhci/ssd@g600c0ff00000000007e8d85fe6ae6201” 2 “ssd”
“/scsi_vhci/ssd@g600c0ff00000000007e8d85fe6ae6200” 5 “ssd”
...
```

Edit /etc/path_to_inst on the *new* node and adjust the appropriate lines to match lines on the other nodes. The Solaris documentation should be consulted if you are unsure about this. Then delete the files referring to the disk, for example, if the controller was ‘c4’ and was a ‘Low Voltage Differential (LVD) SCSI channels’ - which is also the path used for Fibre Channels;

```
rm /dev/dsk/c4* /dev/rdsk/c4 /devices/scsi_vhci/*
```

Then reboot to cause the path_to_inst changes to take effect. Rebooting with the following syntax will cause the file references to be rebuilt correctly;

```
reboot -- -r
```

Please use caution when performing these tasks. Imprudent modifications to the path_to_inst file or deleting the wrong file references may make prevent the system from booting. **Always make a backup of the path_to_inst file before making changes.**

Performance Tuning

To improve synchronisation times it is possible to increase the I/O size. The optimal size is something outside the scope of this document but generally 1Mbyte is better than the 32Kbyts default. This will require two changes that will require a reboot before they take effect;

/etc/system - add the following line
    set maxphys=0x100000
/etc/rc2.d/S95svm.resync
    *modify the ‘$METASYNC -r’ line to be ‘***$METASYNC -r 2048***’*

UFS Logging

Always enable the **logging** option for all file systems as this improves the time taken to check and/or clean a filesystem on reboot.

UFS Filesystem Mirrors - Non-Oracle Installations

Write-On-Write is a phenonomon that occurs when data that is being sent to the disk is modified as it’s being written. This can result in different data being stored on different mirror copies. Oracle does not allow write-on-write but with other installations this can be prevented from occuring by;

/etc/system - add the following line
    md_mirror:md_mirror_wow_flg=0x20

Database & Filesystem I/O Cache

Databases normally cache and provide advanced optimisation for file system operations and by default so does the file system. This means that there is a potential risk to data that the database believes is physically written to disk but also additional memory is used to cache file operations. However, some installations do benefit from using the file system caching system and readers are advised to measure performance of the database before and after disabling file system caching.

To remove caching by the filesystem;

add the **forcedirectio** option when mounting the database file system partitions