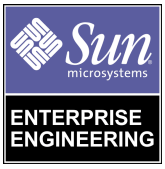
Toward a Reference Configuration for VxVM Managed Boot Disks

By Gene Trantham - Enterprise Engineering
John S. Howard - Enterprise Engineering

Sun BluePrints™ OnLine - August 2000



http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-6197-11
Revision 01, 1/29/03
Edition: August 2000

Copyright 2000 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Sun, Sun Microsystems, the Sun logo, Sun StorEdge, Ultra Enterprise, Sun Enterprise Volume Manager, Sun BluePrints and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X / Open Company, Ltd.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6 / 87) and FAR 52.227-19(6 / 87), or DFAR 252.227-7015(b)(6 / 95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, Californie 94303 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Sun, Sun Microsystems, le logo Sun, Sun StorEdge, Ultra Enterprise, Sun Enterprise Volume Manager, Sun BluePrints, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X / Open Company, Ltd.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Please
Recycle

Adobe PostScript

Toward a Reference Configuration for VxVM Managed Boot Disks

Introduction

The Volume Manager from Veritas Software (VxVM) is widely used in mission critical environments for a variety of storage management tasks. It is considered the volume manager of choice by many for virtually all storage management on the most critical systems. The one exception to the near universal use of VxVM is the management of the boot disk.

VxVM does not enjoy a reputation as an easy and effective storage manager for the boot device on hosts running the Solaris™ Operating Environment. This is only partly due to the internal architecture of the software, many of the problems which arise from an encapsulated and mirrored boot device under VxVM are the result of casual installation procedures. If VxVM is installed according to strict and consistent guidelines, it can be a most effective and unobtrusive mechanism for protecting and managing the most critical data on the host -- the Operating Environment.

This article will outline the fundamental procedures typically followed in a boot disk encapsulation and the vulnerabilities this default encapsulation introduces. As an alternative to this problematical default, a best practice for VxVM installation, root disk encapsulation and a reference configuration is presented.

Note – The Sun BluePrints book *Boot Disk Management: A Guide for the Solaris™ Operating Environment* (ISBN # 0-13-062153-6) by John S. Howard and David Deeths, released in Fall of 2002, provides updated information about the topics detailed in this article. Visit `http://www.sun.com/books/catalog/howardbp/` for more information.

Deficiencies of the Default Encapsulation

Veritas Volume Manager is so widely used for mission critical data storage on large systems that it is considered by some to be the de facto standard. Nearly universal is its acceptance within the datacenter, it is almost as universally cursed for the way in which it manages the boot device and its mirrors. Many, in fact, go to the extra effort and expense of installing a second volume manager to handle the boot disk only.

VxVM's bad reputation in this area is that it makes fairly simple boot disk operations needlessly complex and arduous. That reputation is not entirely undeserved, but VxVM is not so difficult as it is made out to be. The primary objectives in managing a boot device are *RECOVERABILITY* and *SERVICABILITY*. The VxVM default boot disk configuration compromises both of these objectives in favor of ease of installation.

Encapsulation, the method for placing the boot disk under VxVM management is scripted within the installation tool (`vxinstall`). This is the only practical way in which to encapsulate the boot device so that it may be effectively mirrored. Because the details of the encapsulation process are buried within `vxinstall`, and are not easily changed, the install tool makes a number of simplifying decisions and assumptions. This makes the VxVM boot disk encapsulation process easier for the administrator (no options, no parameters, no confusing questions; just encapsulate and go). Unfortunately, `vxinstall`'s decisions are geared towards the average system and are not designed to maximize recoverability and serviceability. The resulting encapsulation, while easy to install, may be difficult to service and may hinder recovery procedures. In some extreme cases, default-configured boot devices may even completely prohibit recovery.

Two key philosophies will make VxVM boot disk management much easier:

- Do the difficult work at installation time.

The most experienced and effective system administrators take the default VxVM boot disk configuration as merely a starting point from which to begin work. The default setup merely places the disk under management in the most general way. It is not the most resilient, nor the most manageable arrangement. More work is necessary to turn this into a resilient and easily managed boot device.

- Consistency in all things.

A great deal of the problems in recovering or servicing a VxVM boot device stem from the inconsistent configuration produced by a default installation. In a variety of ways, the boot disk is an exception in the world of VxVM. The most notable

exceptions are in the geographic layout of the data. From the private region location to the order in which mirrors are attached to rootdisk volumes, a number of exceptions make it more complex than it really needs to be.

We will present an alternate boot disk configuration that is based upon the desire to make the service events easier, rather than making the installation process as painless as possible. Over the life of the system the boot disk will be installed and configured only once, but it will undoubtedly have multiple service events. Installation time is also usually less expensive than unscheduled downtime.

A Typical Encapsulated Boot Disk

The documented method for placing the boot disk under VxVM control (using `vxinstall` or `vxdiskadm`) will achieve the fundamental goal of setting the boot media under volume management control; the boot disk will be encapsulated. For reliability, this configuration is usually augmented by attaching a simple mirror. The resulting configuration, however, may not be what you had envisioned as you set about to mirror your boot device.

The typical encapsulation attempts to place the boot device under VxVM control, the challenge facing `vxinstall` is that the disk may have already allocated its entire range of cylinders. That is, the boot disk has been partitioned such that there are no free cylinders left on the disk. Because VxVM requires a small amount of space for its private region, it will attempt to steal several cylinders from the swap partitions as it re-partitions the disk during the encapsulation process. The consequences of this are as follows.

No protection for private region

A consequence of placing the private region in the middle of the disk is that the public region cannot be represented by a discrete slice. The Solaris Operating Environment requires that slices must be defined as a start cylinder and a length. No facility exists to map the public region slice as the first few cylinders, skip the private region, then the rest of the disk.

VxVM solves this problem by mapping the public region to the entire disk (much as slice 2), and creating the private region slice in its location somewhere in the middle of the device. The private region contents are now within the address space of BOTH the public and private region. To prevent data volumes from being created out of the space occupied by the private region, VxVM creates a special subdisk “on top of” the section occupied by the private region. This subdisk is named “`rootdiskPriv`” and exists solely to mask off the private region.

Private region in middle of the disk

The standard locations for the private region on a VxVM disk are either at the beginning or at the end of the device. Placing the private region in the middle will reduce the flexibility of the configuration, because the data space must now be segmented into a before- and after- area.

No protection for block zero (The VTOC)

All SCSI disks in the Solaris Operating Environment maintain their Volume Table Of Contents (VTOC) at the first addressable sector on the disk, block zero. The VTOC is absolutely essential for the proper operation of the disk. Without it, slice information is lost and no partition information is available.

Most data formats laid down onto disk know to skip this first sector before writing data. UFS file systems, for example, do not write to block 0 of their supporting slice. VxVM private regions also are aware of the need to preserve block zero, so they offset by one sector as well.

VxVM does not know (nor does it care) what data may someday live on this disk. It must take steps to preserve block zero now in order to ensure that it will have a valid VTOC in the future. VxVM does this by creating a special subdisk on top of the VTOC which will persist even if `rootvol` is moved. This subdisk is named "`rootdisk-B0`." Its only function is to mask off the VTOC block.

In addition to the oddities on the original boot disk, the boot mirror has its own quirks that make this configuration exceptional. The most disturbing of which is the sequence in which the mirrors are attached.

The order that mirrors are attached by `vxdiskadm` is alphabetical, not sequential. At the most fundamental level, this does not matter. This simple mirror accomplishes its main requirement, that the data is mirrored across two spindles which are presumably device-independent. However, because you have a device and its mirror which are not exactly identical, this configuration is confusing and more complex to manage in service procedures. The default boot disk and boot mirror configuration is also more difficult to recover from because it represents yet another way that the boot device and its mirror are exceptions, we prefer a more consistent approach.

Best Practice Boot Disk Configuration

With any architecture, there are trade-offs. The configuration proposed here promotes serviceability and recoverability at the expense of space and cost. This best practice configuration will seem to waste some disk space, among other things. This is not a waste, it is an investment in simplicity and consistency. These two principles will make the configuration much safer and faster to recover should any failure occur. With the escalating cost of downtime, a system which is faster to recover will make up the added cost of installation in the very first outage event.

The proposed configuration was architected for these design characteristics:

SIMPLE -- The configuration should be easy. Any system administrator with a moderate level of experience should be able to look at the configuration briefly to understand what is going on. There should be few, if any, exceptions or special cases to how various aspects of the boot disk are configured.

CONSISTENT -- This is a corollary to “SIMPLE.” The more cookie-cutter we can make the configuration, the more useful an administrator’s experience becomes. An administrator who has gone through a type of recovery, for example, on one system can make that same recovery happen on any other system in the enterprise. Consistency in implementation makes this easier to achieve.

RESILIENT -- The configuration has designed out the possibility of a single hardware error (or device driver error) causing an outage. All hardware elements necessary to support each mirror of the boot device are completely independent of one another. We will tolerate no single points of failure.

Hardware

In truth, the choice of specific hardware is not very important. Almost any type of disk will function quite well as the boot device, although some are better than others. The critical issue on boot disk selection is that whatever the choice, all mirrors hotspares and clone disks be of identical geometry. The easy way to achieve this is to use all identical disks of the same model from the same manufacturer.

The reference configuration proposed here will use the Sun StorEdge™ D1000 array enclosure, partially populated with 9Gb drives. The D1000 array is split according to installation documentation into two distinct SCSI busses and the four drives in the boot configuration are split evenly across these two busses. The D1000 array draws its power from two distinct power sources.

It is important to note that the reliability, availability, and serviceability (RAS) needs of the application and services should drive your hardware decision making process. For example, the Sun StorEdge D1000 array has a common backplane for both busses. This backplane is not a single point of failure (SPOF) as it's failure will not effect both halves. However, as the entire D1000 array enclosure must be powered down to replace this backplane, it does provide a serviceability issue. All such RAS aspects of all products should be considered when making your hardware choices. This reference configuration addresses the "typical" RAS needs, if your application requires a higher RAS level use two D1000 arrays. The two D1000 arrays should still be configured in a split buss configuration and then mirrored across enclosures.

We will only use 4 of the drives, including hotspare and clone disk. Other drive bays may be used in this enclosure for other data needs, but these additional disks should not be in the root disk group (`rootdg`).

The Sun StorEdge D1000 array is our choice for a reference configuration because:

1. "Mature" SCSI Technology

The boot device does not have any high-bandwidth or low-latency I/O requirements. Performance of the boot device is rarely an issue, so it needn't be a faster, more complex type of drive (such as FC-AL). In order to make recovery and maintenance easy, we would like to keep the boot disk on the simplest technology possible. SCSI is very stable as a command set and as a transport. Firmware upgrades to disk devices are few, as compared with newer technologies such as FC-AL.

2. Independent data path and power feeds

The Sun StorEdge D1000 array has a key feature which makes it ideal for boot disk setup. It is possible to split the enclosure into two logical and electrically distinct SCSI busses. These two busses are also served by two independent power supplies within the D1000 array enclosure. One enclosure can then function as two logical enclosures both for data path and for power.

3. Large ship volume

Sun uses the D1000 array enclosure and the disks within it as a building block for several of the Sun StorEdge products (the A3500, to name one). These disks and the enclosure itself have a high ship volume. Any bugs with firmware or issues with hardware will be discovered rapidly and reported by the large customer base. Because of its key position within the product line, we can expect those issues (if there are any) to be addressed very quickly.

4. Flexible

The Sun StorEdge D1000 array enclosure can be used in a variety of situations. It is approved for use in various server cabinets, so it can be deployed in a wider range of circumstances than can other enclosures. We want to maintain a consistent boot disk setup for as many server types as possible. The D1000 array, because it can

operate in the server cabinet for everything from an Ultra Enterprise™ 4500 to a Ultra Enterprise 10000 server, provides the ability to standardize our reference configuration throughout the enterprise.

In order to ensure complete device independence, we need to arrange the data paths to the disks in our Sun StorEdge D1000 array carefully. As mentioned above, we want the enclosure to be split into two SCSI busses. These two busses are to be serviced by two independent power sources as described in the install documentation for the D1000 array.

The two SCSI busses should service no other devices than the disks within this enclosure. Do not extend the bus to include tape drives or other devices, no matter how trivial.

These two busses should be serviced by host adapters installed on separate system boards within the host. This is the easiest way to ensure that the host adapters do not share an internal Bus or other hardware element.

VxVM Configuration

Root Disk Group Contents

As mentioned earlier, there are only four (4) disks used for the boot device and its entourage. These four disks are the only items to be included in the root disk group (`rootdg`). Any data volumes or file system spaces to be created outside of the core OS should reside in other disk groups.

```
istvan# vxdisk -g rootdg list
DEVICE      TYPE     DISK         GROUP     STATUS
c1t0d0s2    sliced   rootdisk     rootdg    online
c1t1d0s2    sliced   hotspare     rootdg    online
c2t8d0s2    sliced   rootmirror   rootdg    online
c2t9d0s2    sliced   clone        rootdg    online
```

Notice that the disk media name for each disk reflects its function. Clear and obvious naming will prevent confusion later on. Also note that `rootdisk` and `rootmirror` are on different controllers. These are the two SCSI host adapters servicing each side of the Sun StorEdge D1000 array.

Volumes in `rootdg`

The standard for how to partition the boot device is often hotly debated. Where you stand on this issue is less important than making a stand. Choose a partition standard in your datacenter that you can deploy on every type of server. Use these guidelines to help establish and define your boot device partitioning standard:

1. Be sure to leave plenty of space on each partition to support the inevitable upgrades and patches.
2. Keep `/usr` on the root partition and break `/var` off into a separate partition only if absolutely necessary. The contents of `/`, `/usr`, and `/opt` are relatively static, their contents rarely change except during upgrades or other service events. As such, there is little need to designate a separate partition for `/usr` or `/opt`.

It is very helpful in recovery situations, especially when VxVM is involved, to keep the `/usr` file system on the root partition. Segregating `/usr` from root can have serviceability implications on VxVM encapsulated devices as all of the VxVM utilities are located in `/usr`. The only VxVM components located in root are the kernel drivers and `vxconfigd`. If ever `/usr` cannot mount, there is precious little that can be done with only root mounted. Co-locating `/usr` on the root file system eliminates this problem.

The reference configuration contains these file systems on `rootdisk`:

Table 1: `rootdisk` file systems

Mount-point	Size (Gb)	Comments
/	4.0	Round down to the nearest cylinder boundary
swap	2.0	Primary swap, round down to the nearest cylinder boundary

Subdisks

There are a number of oddities about an encapsulated boot device that we would like to avoid. Two of these, in particular, are the special masking subdisks created on a typical encapsulated boot disk: `rootdisk-B0` and `rootdiskPriv`. If we can convert the encapsulated boot device into an initialized device, these subdisks are no longer necessary.

The subdisks which compose the various mirrors of each volume are also important for where they are. We want to be more detailed about the volume configuration than simply “this disk mirrors that disk.” We would like to have an exact picture of precisely where on one disk the data resides, and precisely where on that other disk it is mirrored. The goal is to have the two subdisks representing the two copies of some volume’s data to occupy the exact same positions on each disk.

Recovery Configuration

Clone Disks

There are a variety of recovery situations in which it is useful to boot from some sort of failsafe media and still maintain the ability to manipulate VxVM objects. There are a number of ways to achieve this, the most universal being a clone disk. For a more elegant solution consult the various Sun BluePrints™ Online articles detailing the MiniRoot (MR) system for making VxVM available from a failsafe boot image.

The clone disk is a filesystem copy of the core Solaris Operating Environment (OE) which is written to slices, not volumes. The function of the clone disk is to provide a bootable image without depending upon VxVM volumes, yet still maintain the ability to use VxVM utilities and drivers. This allows us to effect a wide range of simple repairs or service procedures without having to unencapsulate the boot device.

Making a clone disk can be complex, one must be exceptionally careful of the target disk so as not to overwrite important data. The choice of data copy utility is also important, the common choice for this sort of activity is `dd(1)`. While `dd` is an acceptable utility to perform this function under certain circumstances, it must be used with far more care than filesystem-aware utilities such as `cpio`, `tar`, or `ufsdump`. Because of the lower potential to do inadvertent harm, we prefer a utility which is aware of and works within the structure of the filesystem. See Appendix A for a script which uses the `ufsdump` and `ufsrestore` utilities to effect the cloned copy.

Once the filesystem copy is made, some modifications must be made to various configuration files on the clone disk, in order to successfully boot from slices. The `/etc/system` file and `/etc/vfstab` must both be modified in order to prevent the clone OS from depending upon VxVM volumes. However, we still want to load the VxVM device drivers.

Sun’s Live Upgrade (LU) product is designed to manage multiple boot environments, provide a safe fall-back boot environment and allow upgrades of the Solaris Operating Environment and unbundled products on alternate boot environments. Further, LU provides a synchronization mechanism to keep user specified files up to date across multiple boot environments. LU will provide a

standard and easily managed mechanism for duplicating, activating and managing multiple boot environments. Once LU is released, it should be used as the preferred mechanism for duplicating and managing multiple boot environments.

Slices

As a secondary precaution to help ensure easy and effective recovery in an emergency, we would like to have some way to access the underlying slices on the root disk even if the clone disk does not function. In order to be able to access the underlying filesystems on the root disk, the subdisks should be mapped to the corresponding disk partition. This is automatically done by VxVM for the root volume, but no others. We need to ensure that all file systems required during a recovery be mapped to partitions so that they may be used in such dire circumstances. Then, in extreme cases where VxVM simply will not function, we may boot off of CD-ROM or over the network and mount the underlying slices which map to the subdisks for each volume.

Resorting to slices to access the file systems is a common recovery tactic, but it often takes longer to perform this action (and recover from it) than it does to correct the original problem. For this reason, we prefer to leave it as a recovery option of last resort in the reference configuration.

OBP Configuration

Creating and maintaining the Open Boot PROM definitions are critical to help ensure easy startup and recovery. Without clear and descriptive device aliases defined and kept up to date, it may be difficult to discern which is the correct boot device. This has implications for recovery attempts in outage situations. Suppose you need to boot from a mirror or the clone disk. What is the full device path to that disk? Are you certain? What if that disk has been replaced or moved?

At the OBP, the host's file systems are inaccessible, so configuration data may not be verified. We must ensure that descriptive device aliases exist ahead of time so that it is quite clear which device to boot from for the situation at hand. We want device aliases for "`rootdisk`", "`rootmirror`", "`hotspare`", and "`clone`" and we want these same names on all systems so as to prevent confusion.

The boot list as defined in the OBP parameter "`boot-device`" should list all possible boot devices that the system should attempt to open when it tries to boot. The reference configuration sets `boot-device` equal to "`rootdisk rootmirror hotspare`". The clone disk should only be used to manually boot in recovery or service situations.

There are several mechanisms to set and control these settings in the OBP environment: the OBP commands `setenv`, `show-disks` and `nvalias`, the Solaris Operating Environment `eeprom` command, the VxVM `vxeeprom` command, and the Solaris Operating Environment `luxadm` command. The implementation section will address the usage and relative merits of some of these commands.

Implementation

Hardware Installation

Install the Sun StorEdge D1000 array per the hardware installation guide, making certain to follow all instructions to split the bus. All cables should be clearly labelled (at both ends) to identify which controller, slot and system board they connect. Depending upon the options purchased with your D1000 array, either install 2 disks per bus or, if the D1000 array was purchased fully populated, select 2 disks from each bus. These disks will be used for the `rootdisk`, `hotspare`, `rootmirror` and `clone` disks. Additionally, these disks should be physically labelled with these disk media names. In the event of physically replacing any of these disks, the labels will allow ease of identification and prevent the inadvertent replacement of the wrong disk.

The Sun StorEdge D1000 array should also be configured with 2 independent power supplies. If possible, these should be powered from 2 separate power sequencers or separate power grids.

To prevent the IO board from being a single point of failure (SPOF), the SCSI controllers should be installed such that each controller is on a separate IO board. However, for ease of administration and maintenance the controllers should be installed in the same slot number on their respective boards. For example, the SCSI controllers are installed in slot 0 of IO board 1 and slot 0 of IO board 3.

When the hardware configuration is completed it should be documented and diagrammed. Remember to keep hard copy of the configuration documentation in an easily accessible location, as there are few things as frustrating as realizing that the configuration information you require is well documented but stored only in electronic format on a system that is currently down or inaccessible.

Vx Commands for installation

After the hardware is installed, we install VxVM and use `vxinstall` to encapsulate the boot disk in the default manner. When running `vxinstall` we want to encapsulate the boot disk only and accept the default disk media name (‘`rootdisk`’) for the encapsulated boot device. Do not encapsulate or initialize any other disks at this time. After the system reboots and the boot disk is encapsulated, we begin the work of re-configuring the default VxVM boot disk encapsulation to our reference configuration. Throughout this example we will be using the Solaris 8 Operating Environment, Veritas Volume Manager 3.0.4 and the disk devices:

Device	Description
/dev/dsk/c1t0d0	rootdisk
/dev/dsk/c1t1d0	hotspare
/dev/dsk/c2t8d0	rootmirror
/dev/dsk/c2t9d0	clone disk

- mirror-remove-remirror

1. Initialize and add the disk to be used for `rootmirror` to `rootdg`:

```
# /usr/lib/vxvm/bin/vxdisksetup -i c2t8d0
# vxdg -g rootdg adddisk rootmirror=c2t8d0
```

2. Then attach the mirrors, in the order we desire:

```
# /etc/vx/bin/vxrootmir rootmirror
# vxassist -g rootdg mirror swapvol rootmirror
```

3. Dissociate `rootdisk` plexes and remove the “special” subdisks:

```
# vxplex -g rootdg dis rootvol-01 swapvol-01
# vxedit -g rootdg -fr rm rootvol-01 swapvol-01
```

Note that if your boot disk partitioning scheme uses separate partitions (such as a separate `/var`), those partitions will also need to be attached to the mirror and then dissociated from the `rootdisk` before proceeding to step 4. Similarly, steps 6 and 7 will need to be adapted to your partitioning scheme.

4. Remove `rootdisk` from `rootdg`:

```
# vxdg -g rootdg rmdisk rootdisk
```

5. Initialize `rootdisk` and add it back into `rootdg`:

```
# /etc/vx/bin/vxdisksetup -i c1t0d0
# vxdg -g rootdg adddisk rootdisk=c1t0d0
```

6. Attach mirrors in correct order:

```
# /etc/vx/bin/vxrootmir rootdisk
# vxassist -g rootdg mirror swapvol rootdisk
```

7. Create the underlying partitions on the primary boot disk:

Use the `/usr/lib/vxvm/bin/vxmksdpart` command to make the underlying partitions corresponding to each of the subdisks on `rootdisk` and `rootmirror`. The `vxmksdpart` command takes four parameters - the slice, subdisk, the partition tag and the partition flags. Solaris Operating Environment is very sensitive to the tags and flags of partitions on the system disk. Please consult the `fmthard` man page for the correct partition tag and flag values for your version of the Solaris Operating Environment. Additionally, the device used for holding a crash dump will need to be re-specified after the underlying partition has been recreated.

For example, with the Solaris 8 Operating Environment and Sun Enterprise Volume Manager™ (SEVM) 3.0.4 software, the following commands would be used to make the underlying partitions for swap on `rootdisk` and `rootmirror`. Additionally, we specify the primary swap partition as the dump device:

```
# /usr/lib/vxvm/bin/vxmksdpart -g rootdg rootdisk-02 1 0x03 0x01
# /usr/lib/vxvm/bin/vxmksdpart -g rootdg rootmirror-02 1 0x03 0x01
# dumpadm -d /dev/dsk/c1t0d0s1
      Dump content: kernel pages
       Dump device: /dev/dsk/c1t0d0s1 (dedicated)
Savecore directory: /var/crash/blackmesa
  Savecore enabled: yes
```

There is no need to create the underlying partition for `/`, it is already there.

8. OpenBoot Prom (OBP) settings:

First, save the current OBP `nvramrc` settings:

```
# eeprom nvramrc >/var/adm/doc/‘date +%Y%m%d‘.eeprom.nvramrc.out
```

(As with most system configuration documentation, it is a good idea to print this file and save the hardcopy with the system configuration binder). Then, determine the full OBP path to the `rootdisk`, `rootmirror` and the `clone` disk:

```
# ls -l /dev/dsk/c[12]t[0189]d0s0
lrwxrwxrwx   1 root  root    41 Jun 12 10:41 /dev/dsk/c1t0d0s0 ->
../../devices/pci@1f,4000/scsi@4/sd@0,0:a
lrwxrwxrwx   1 root  root    41 Jun 12 10:41 /dev/dsk/c1t1d0s0 ->
../../devices/pci@1f,4000/scsi@4/sd@1,0:a
lrwxrwxrwx   1 root  root    41 Jun 12 10:41 /dev/dsk/c2t8d0s0 ->
../../devices/pci@1f,2000/scsi@1/sd@8,0:a
lrwxrwxrwx   1 root  root    41 Jun 12 10:41 /dev/dsk/c2t9d0s0 ->
../../devices/pci@1f,2000/scsi@1/sd@9,0:a
```

Then, make a copy of the saved `nvramrc` definition and edit the copy to add boot aliases for the devices underlying `rootmirror` and `clone`:

```
# cp /var/adm/doc/‘date +%Y%m%d‘.eeprom.nvramrc.out /var/tmp/nv
# vi /var/tmp/nv
# cat /var/tmp/nv
devalias rootdisk /pci@1f,4000/scsi@4/disk@0,0:a
devalias rootmirror /pci@1f,2000/scsi@1/disk@8,0:a
devalias clone /pci@1f,2000/scsi@1/disk@9,0:a
devalias hotspare /pci@1f,4000/scsi@4/disk@1,0:a
```

Then define boot-device to include the new `devalias`', enable execution of the contents of the NVRAM (`use-nvramrc?=true`) and assert the new `nvramrc` definition:

```
# eeprom ”boot-device=rootdisk rootmirror hotspare”
# eeprom ”use-nvramrc?=true”
# eeprom “nvramrc=‘cat /var/tmp/nv‘”
```

The aliases will be available after the next time the system is taken down and the OBP `reset` command is issued. Alternatively, the boot aliases may be set with the `luxadm` command, the VxVM `vxeeprom` command or at the OBP by using the OBP `show-disks` and `nvalias` commands.

Regardless of the method used to define the boot aliases, it is crucial that all of the boot aliases be tested before the system is put into production.

Clone

Setting up the Clone Disk

The clone disk should be reserved exclusively for that purpose. We would like to keep it in VxVM control so as to ensure that it will not be diverted to some other task later on. In order to accomplish this, we borrow a page from the VxVM boot disk setup:

```
# /etc/vx/bin/vxdisksetup -i c2t9d0
# vxdg -g rootdg adddisk clone=c2t9d0
# size=‘vxassist -g rootdg maxsize nmirror=1 clone | \
  awk ’{print $4}’‘
# vxmake sd DO_NOT_USE \
     dmname=clone \
     dmoffset=0 len=$size \
     comment=”Do not delete or relocate this subdisk.”
```

These commands initialize the target disk and add it to the `rootdg` disk group. A large subdisk mapping the entire usable space of the public region is then created with `vxmake`. This subdisk is named “DO_NOT_USE” in the hopes that this name will help remind us not to allocate volumes out of this space.

The `clone` Script

The process to clone the OS filesystems to the target disk is fairly straightforward, although moderately tedious. The best solution for this is to script the entire operation to ensure consistent and safe results. An example script to do this is attached in Appendix A. This script:

1. Verifies the clone disk exists and has no volumes on it
2. Re-partitions it to ensure slice boundaries are in place
3. Copies filesystem data from the core OS file systems to those slices
4. Installs boot blocks on the clone disk
5. Rebuilds the system file and `vfstab` on the clone disk to allow slices-only booting

It is important to keep in mind that this script is presented to demonstrate this technique and contains limited error checking. Further, this script assumes the single-slice / filesystem presented in this document. If you have adopted multiple partitions in your boot environment, the clone script will need to be augmented to

re-partition the clone disk and copy those partitions appropriately. Whether you use the example script or your own, be sure that your clone disk is bootable before placing the system in production. Run the clone script out of `cron` on a weekly basis. As mentioned earlier, once Live Upgrade is released it should be used to clone the boot environment rather than this script.

Disabling Hot Relocation and Enabling Hot Sparing

Finally, we disable VxVM's hot relocation feature in favor of the hot sparing feature. To disable hot relocation and enable hot sparing, kill the running relocation daemon, `vxrelocd`, (warning: do not kill `vxrelocd` if it is currently in the process of performing a relocation). After killing the relocation daemon, start the hot sparing mechanism by typing '`vxsparecheck root &`'. To make these changes permanent, you must edit `/etc/rc2.d/S95vxvm-recover` to comment out the '`vxrelocd root &`' command and uncomment the '`vxsparecheck root &`' line. After editing, the last 10 lines of `/etc/rc2.d/S95vxvm-recover` should look like this:

```
# start the watch daemon.This sends E-mail to the administrator when
# any problems are found.To change the address used for sending problem
# reports, change the argument to vxrelocd.
#vxrelocd root &

# to enable hot sparing instead of hot relocation.
# ( comment out vxrelocd before uncommenting vxspare )

vxsparecheck root &

exit 0
```

Next, add `c1t1d0` as the hotspare for `rootdg`:

```
# /etc/vx/bin/vxdisksetup -i c1t1d0
# vxdg -g rootdg adddisk hotspare=c1t1d0
# vxedit -g rootdg set spare=on hotspare
```

Finally, as we have such a small number of disks in `rootdg`, configure VxVM to store a configuration database and kernel log copy on all of the disks:

```
# vxedit set nconfig=all nlog=all rootdg
```

Service Procedures

Booting from Clone OS

In recovery situations, the clone disk may be booted from the OBP with the command:

```
ok boot clone
```

VxVM is installed on the clone disk but not managing the clone disk. As such, VxVM objects may be referenced directly and recovery procedures initiated on any disk volume.

Replacing Disks and Upgrades

If any disks in the `rootdg` are replaced `vxmksdpart` will need to be run to re-partition the drive after replacement. Also, it is recommended to save the partition map of the boot disk in a safe place. This can be used to help insure that if the root disk is replaced the underlying partitions of the replacement are maintained exactly on the original disk. The VxVM utility `/usr/lib/vxvm/bin/vxprtvtoc` will print the partition information of the specified disk, output from `vxprtvtoc` can be re-directed to a file and also a copy should be printed and kept with the system documentation and maintenance logbook.

To upgrade VxVM, the VxVM supplied `upgrade-start` and `vxunroot` scripts will still function as normal. However, the clone disk gives you an easier upgrade method. You can boot off of the clone disk and mount / on `/a`:

```
# mount /dev/vx/dsk/rootvol /a
```

and then use the `chroot` to remove all of the VxVM packages:

```
# chroot /a pkgrm VRTSvxvm VRTSvmsa VRTSvmdev VRTSvmman VRTSvmdoc
```

and then use `pkgadd` to install the new version of VxVM:

```
# cd /net/swinstall-server/packages/VxVM/3.0.4
# pkgadd -R /a -d . VRTSvxvm VRTSvmsa VRTSvmdev VRTSvmman VRTSvmdoc
```

Documentation

As has been mentioned previously, documentation of your system configuration is necessary for day to day administration, during service events, and during recovery procedures. VxVM provides commands to save and recover the VxVM configuration. In addition to backing up your data on a regular basis, the information required to re-create the configuration of your systems should be regularly backed up as well. The configuration of all of the VxVM disk groups should be saved with the `vxprint` command. To restore a saved VxVM configuration, the `vxmake` command is used. For example, to save the configuration of the disk group `dblogsdg`, use the following command:

```
# vxprint -g dblogsdg -vpshm > \
    /var/adm/doc/vxprint-vpshm.dblodgsdg.out
```

To restore the saved configuration, use:

```
# vxmake -g dblogsdg -d /var/adm/doc/vxprint-vpshm.dblodgsdg.out
```

Consult the `man` pages for `vxprint` and `vxmake` for detailed information on their usage.

Conclusion

This article has presented methodologies for designing and implementing a RAS configuration for a VxVM managed boot disk that emphasizes simplicity, consistency, resilience and recoverability.

By adhering to a well-planned and consistent boot disk configuration, VxVM is well suited to managing all disks including the boot disk. This article has presented the basis for a reference configuration for VxVM managed boot disks that emphasizes recoverability and serviceability over boot disk space utilization.

Appendix A

```
#!/bin/sh
##
## clone
##
## Takes on-line copy of the running OS to a spare “clone” disk.
## Data is copied via ufsdump/restore to partitions/slices, rather
than
## volumes.
##
## Central assumptions:
## 1) A disk in rootdg is called “clone” (case sensitive match)
## 2) This disk has been initialized in the conventional way, with
its
##    private region beginning at cylinder 0.
## 3) No volumes are defined on “clone”
##
TargetDisk=`vxprint -g rootdg -F”%device_tag” clone`
if [ -z “$TargetDisk” ] ; then
    echo “WARNING: No disk found matching the name \”clone\”.”
    echo “Unable to continue.”
    exit 2
fi
vols=`vxprint -Q -g rootdg -e”pl_sd.sd_dm_name == \”clone\”” -
F”%vol”`
if [ -n “$vols” ] ; then
    echo “WARNING: Volumes found on \”clone\” disk.”
    echo “Unable to continue.”
    exit 2
fi
C_path=”/dev/rdsk/${TargetDisk}s2”
B_path=”/dev/dsk/${TargetDisk}s2”
prtvtoc $C_path > /tmp/_clone_$$
privslice=`nawk ‘$2 == “15” {print $1}’ /tmp/_clone_$$`
privstart=`nawk ‘$2 == “15” {print $4}’ /tmp/_clone_$$`
privlen=`nawk ‘$2 == “15” {print $5}’ /tmp/_clone_$$`
pubslice=`nawk ‘$2 == “14” {print $1}’ /tmp/_clone_$$`
pubstart=`nawk ‘$2 == “14” {print $4}’ /tmp/_clone_$$`
publen=`nawk ‘$2 == “14” {print $5}’ /tmp/_clone_$$`
```

```
if [ $privstart -ne 0 ] ; then
    echo “WARNING: Private region is not at the beginning of the
disk.”
    echo “Unable to continue.”
    exit 2
fi

## STEP 1: fmthard the disk
##   We are going to be somewhat draconian here: root and swap
##   are the only slices you get.
cylsize=`nawk ‘$3 == “sectors/cylinder” {print $2}’ /tmp/
_clone_$$`

ROOTSIZE=8388608   # 4Gb, expressed in sectors
SWAPSIZE=4096000   # 2Gb

#... calculate start and length for root, swap, var
#... Must round all lengths to next SMALLER cylinder boundary....
rootslice=0
rootstart=`expr $privstart + $privlen`
rootlen=`expr \( $ROOTSIZE / $cylsize \) \* $cylsize`

swapslice=1
swapstart=`expr $rootstart + $rootlen`
swaplen=`expr \( $SWAPSIZE / $cylsize \) \* $cylsize`

fmthard -d $rootslice:2:0x00:$rootstart:$rootlen $C_path
fmthard -d $swapslice:3:0x01:$swapstart:$swaplen $C_path

## STEP 2: newfs file systems
newfs /dev/rdsk/${TargetDisk}s${rootslice}

## STEP 3: Mount on tmp mount point
## We make it new in the hopes that it is not already in use by
another mount
mkdir -p /_clone/root
mount /dev/dsk/${TargetDisk}s${rootslice} /_clone/root

## STEP 4: Copy data
## ufsdump core_os | ufsrestore on /_clone/root
## Notice that we use the actual mount table rather than slurping
## through /etc/vfstab.  No worries about comments or typos in the
## vfstab that way.
```

```
mount -p | while read Bdev Cdev MP FStype Order AtBoot Options
do
    if [ ! $FStype = “ufs” ] ; then
    continue
    fi
    case $MP in
    /|/usr|/var|/opt)
        cd /_clone/root/$MP
        ufsdump 0uf - $MP | ufsrestore xf -
        cd /
    ;;
    *) continue
    ;;
    esac
done ## While loop

## STEP 5: install boot blocks
/usr/sbin/installboot \
    /usr/platform/`uname -i`/lib/fs/ufs/bootblk \
    /dev/rdsk${TargetDisk}s${rootslice}

## STEP 6: tweak system config files
## /etc/system
mv /_clone/root/etc/system \
   /_clone/root/etc/system.pre_clone
sed -e ‘/vol_rootdev_is_volume/d’ -e ‘/rootdev:/d’ \
    /_clone/root/etc/system.pre_clone > \
    /_clone/root/etc/system
## Create a new /etc/vfstab
mv /_clone/root/etc/vfstab /_clone/root/etc/vfstab.pre_clone
cat > /_clone/root/etc/vfstab <<EOVFSTAB
/dev/dsk/${TargetDisk}s${rootslice} /dev/rdsk/
${TargetDisk}s${rootslice}  /     ufs   1  yes  -
/dev/dsk/${TargetDisk}s${swapslice} -    -     swap  -  no   -
swap                -                     /tmp  tmpfs -  no   -
/proc               -                     /proc proc  -  no   -
EOVFSTAB

## STEP 7: umount tmp space
umount /_clone/root

exit 0
```

Author's Bio: Gene Trantham

Gene Trantham is a Staff Engineer for Enterprise Engineering at Sun Microsystems. Prior to joining Sun, he spent eight years as a Unix® system administrator specializing in storage management and disaster recovery. While at Sun, Gene has spent most of his time in the field, servicing storage and high-end servers at some of Sun's largest accounts.

Author's Bio: John S. Howard

John S. HowaJrd is currently a Staff Engineer in the Enterprise Engineering group at Sun Microsystems in San Diego, California. He has worked as a software engineer and systems administrator for the past 19 years. Prior to Enterprise Engineering, John worked in Enterprise Services as an Area System Support Engineer for five years. As an ASSE, he was responsible for developing and performing Reliability, Accessibility, and Serviceability (RAS) studies of customer datacenters and the development of proactive Enterprise RAS Services. Prior to Sun, John held engineering positions at: The Chicago Board of Trade Clearing Corporation, Datalogics Inc, and Rand McNally. Throughout his career he has developed: pagination and publishing software, loose-leaf publishing systems, extensive SGML systems development, database publishing systems, text editors and WYSIWIG systems, and device drivers