Veritas Krash Kourse: The Who's Who of Vx Land

The Cuddletech Veritas Volume Manager Series

Ben Rockwood, Cuddletech <benr@cuddletech.com>

Revision History

Revision v1.0	August 10th 2002	br
	Initial conversion to DocBook	
Revision v1.1	August 11th 2002	br
	SGML Cleanup	

A broad overview of the Veritas Volume Manager, this course follows the RAID Theory course. The primary focus is to clarify the purpose and use of Veritas objects, the basis for volume building.

Table of Contents

Introduction

Welcome to VERITAS!

This course is the first step to learning the wild and wonderful world of VERITAS Volume Manager. This course is absolutely necessary for you to continue on to the future

courses, and if you already know this stuff it's at least a good review.

Before we start, there are a couple of things you need to know about me (your humble narrator).

Why I'm Doing This

You may, or may not, be curious as to why I'd waste part of my life writing these courses. You may also have already noticed (from the preceding course) that I am not a professional instructor. Well, you're right: I'm not a professional instructor. I am, however, a UNIX systems admin who specializes in Sun Enterprise Storage Systems. I won't bore you with the story of how I got to where I am, but for several months I wanted to learn VERITAS, and no one could teach me more, and I couldn't afford classes. I begged my company to get me materials or a class on the subject to no avail. So, I learned VERITAS the hard way. I downloaded the manuals from Sun and studied them for a couple of weeks, before I got a contract that would let me actually experience VERITAS first hand. The bottom line here is that some of the concepts took days and weeks of rolling over and over in my head before I fully understood them. These documents, and the courses I teach, are my way of helping you get the most out of VERITAS and the VERITAS manuals the FIRST time you read them. Hopefully, you'll be understanding VERITAS in days or hours instead of weeks. This is my way of "giving back" to the community. I am not an "expert", nor is anyone an "expert". I'm just sharing the knowledge I've gained during the last year. My teachings have done well for many others that I've taught, so I hope you'll pick up a lot from these writings.

Now, on with the show!

VERITAS Volume Manager Documentation and Reference

These tutorials directly apply to the following docs:

- VERITAS Volume Manager: Installation Guide (P/N: 100-001121)
- VERITAS Volume Manager: Getting Started Guide (P/N: 100-001123)
- VERITAS Volume Manager: Command Line Interface Administrators Guide (P/N: 100-001124)
- VERITAS Volume Manager: Storage Administrator: Administrators Guide (P/N: 100-000954)
- VERITAS Volume Manager: Administrators Reference Guide (P/N: 100-001125)

These are Sun part numbers. At the time of this writing these documents are not available on the web, and only available with the VERITAS VM 3.0.1 software itself. HOWEVER, you can find the VERITAS VM 2.6 documentation on docs.sun.com. To get the docs (available in HTML or PDF), go to docs.sun.com. Then click on "Browse By Collection". Then search down the list (toward the bottom) until you find "Sun StorEdge Volume Manager 2.6 AnswerBook". Click on that. Then you'll see three titles:

- Sun StorEdge Volume Manager 2.6 User's Guide
- Sun StorEdge Volume Manager 2.6 Storage Administrator User's Guide
- Sun StorEdge Volume Manager 2.6 System Admin Guide

Notice, while you're there, the "Download PDF" option in the top right hand corner; great for printing.

VERITAS Volume Manager: An Overview

Here's an excerpt from the Sun web page (www.sun.com) regarding VERITAS:

> VERITAS Volume Manager software is an advanced, system-level disk and storage array solution that alleviates downtime during system maintenance by enabling easy, online disk administration and configuration. The product also helps ensure data integrity and high availability by offering fast failure recovery and fault tolerant features. VERITAS Volume Manager software provides easy-to-use, online storage management for enterprise computing and emerging Storage Area Network (SAN) environments. Through the support of RAID redundancy techniques, VERITAS Volume Manager software helps protect against disk and hardware failures, while providing the flexibility to extend the capabilities of existing hardware. By providing a logical volume management layer, VERITAS Volume Manager overcomes the physical restriction imposed by hardware disk devices.

So, there you have it. But let me briefly list some of the things they totally missed.

- Support for: Simple, RAID0, RAID1, RAID0+1, RAID1+0, RAID5
- Dynamic Multipathing (DMP): Load balancing and redundant I/O paths to disk arrays supporting multi-controller attachment.

- Online Relayout: VERITAS allows you to change the layout of a VERITAS volume while it is live and mounted. Change a RAID0 to a RAID5 without a second of down-time!
- Snapshoting: Take a snapshot of your data, creating a "shadow" of it which you can use for online backups.
- Hot Relocation: Designate "spares" which will take the place of failed disks on-the-fly.
- Dirty Region Logging (DRL): Volume transaction logs which provide fast recoveries after system crashes.
- More..... more than I can list!

So lets start by building on all that. VERITAS Volume Manager is a product of VERITAS Software Corporation (www.VERITAS.com), which is used by Sun and also sold to Sun customers under a Sun part number. VERITAS is also can "Vx" or "VRTS" (the prefix of VERITAS packages, and VERITAS' ticker symbol). VERITAS Volume Manager is often abbreviated "VxVM". Sun, before VERITAS 3.x, had its own marketing names for VERITAS, the names you'll see and the version numbers are:

- <= Vx 2.5: Sun Enterprise Volume Manager (SEVM)
- = Vx 2.6: Sun StorEdge Volume Manager (Sun SEVM)
- > Vx 3.x: VERITAS Volume Manger (VxVM)

As for OS support:

- SEVM 2.5: Solaris 2.5.1 and older.
- SEVM 2.6: Solaris 2.6 and Solaris 2.5.x
- VxVM 3.0.2: Solaris 7, Solaris 2.6, and Solaris 2.5.x
- VxVM 3.0.3: Solaris 8, Solaris 7, and Solaris 2.6

Let me add, when Sun and/or VERITAS says "OS xxx isn't supported", they don't really mean "Ummm, we haven't really tried it yet", like most other companies in the world. VERITAS actually installs kernel modules into the system, so if you run SEVM 2.6 on

Solaris 7, you'll be running Solaris 2.6 kernel modules in a Solaris 7 kernel. It doesn't work, I tried it! (and Sun was really pissed when they found out I tried it)

VERITAS Volume Manager and Vx Objects

VERITAS is a logical volume manager. The operative word here is "logical". So, in order to keep things in order we have some basic pieces which are used in different ways for building volumes. The better you understand these guys, the better you'll understand VERITAS. Here's the list of objects:

- Disk: This is a normal physical disk with a SCSI id. (c0t0d0...)
- VM Disk (dm): A disk that is put under Vx control.
- Sub Disk (sd): A section of VM disk used to build plexes.
- Plex (pl): A mirror.
- Volume (v): A virtual disk, which can contain data.

Now, lets talk about these objects a bit. A disk is nothing new or magical. When we want to use a disk in VERITAS we need to turn it over to VERITAS control. A disk turned over to VERITAS control is then given a VERITAS name (like disk01). After doing this the disk is no longer available to the system for anything outside of use inside VERITAS. Also, when we turn disks over to VERITAS control we don't turn over a partition (c0t0d0s0), but the whole disk itself (c0t0d0). Now that we've got a VM Disk we then create a Sub Disk from the VM Disk. Think of a subdisk as a VERITAS partition. We could make one big subdisk from the VM Disk and use the whole disk, or we could create a bunch of smaller subdisks. You can divide VM Disks into subdisks however you like. From subdisks (one or more) we create what is called a plex. Plexes are confusing so let's talk about these in some length.

Say the following out loud: "A Plex is a Mirror. A Plex is a Mirror. A Plex is a Mirror." Maybe you should do that a couple times. You may feel silly, but plexes are a sort of cornerstone in VERITAS. A Volume is a container, in VERITAS. The volume is made up of one or more plexes. See the catch? A Plex is a mirror, yes, however you can make a volume with only one plex. Is a volume made with only one plex mirrored? No. We'll explain this more later, but for the time being keep it in your head. So, subdisks are grouped together into a plex. The interesting thing is that in VERITAS the plexes do all the work, so lets say you wanted to create a Striped volume. You would actually create a striped plex, and then attach that striped plex to a volume. The volume doesn't care, it's just a container. See the beauty here? Let's put all this stuff together and build an imaginary volume in VERITAS.

We're going to build a striped (RAID0) volume from 2 9G disks. We'll say that the first disk is c1t0d0, and the second is c1t1d0. First, we need to put them in VERITAS control, so we create VM disks. The VM disks are then named disk01, and disk02. Next, we'll create subdisks using these two disks. Let's use the whole disks and just create 2 subdisks, one for each VM disk. We'll call disk01's subdisk "disk01-01", and disk02's subdisk "disk02-01". Now, it's plex time! We're going to build a striped plex using our two subdisks, which we'll call "myplex". (Don't worry yet about how we create the plex, just get the concepts now.) So now we've got a plex, which contains the subdisks "disk01-01" and "disk02-01". Now, we create a volume named "myvol" using "myplex". And bingo! We've got a striped 18G volume ready to create a file system on! Maybe, if we used the short names mentioned earlier (with the list of objects) and make an outline it'd look something like this:

```
dm      disk01  c1t0d0 <-- VM Disk named "disk01" made from "c1t0d0"
dm      disk02  c1t1d0 <-- VM named "disk02" made from "c1t1d0"

sd      disk01-01 disk01 <-- Subdisk named "disk01-01" made from "disk01"
sd      disk02-01 disk02 <-- Subdisk named "disk02-01" made from "disk02"

pl      myplex  striped <-- Striped Plex named "myplex"
sd      disk01-01       <-- Made using subdisk "disk01-01"
sd      disk02-01       <--   and subdisk "disk02-01"

v       myvol           <-- Volume made from....
pl      myplex striped  <-- the striped plex named "myplex", made from...
sd      disk01-01       <-- Subdisk "disk01-01", and...
sd      disk02-01       <--     "disk02-01"
```

Look OK? Because if it does, take a look at this, real output from VERITAS, from a real volume I created on my test machine:

```
v  myvol        fsgen        ENABLED  35356957 -        ACTIVE   -       -
pl myplex       myvol        ENABLED  35357021 -        ACTIVE   -       -
sd disk01-01    myplex       ENABLED  17678493 0        -        -       -
sd disk02-01    myplex       ENABLED  17678493 0        -        -       -
```

Does that make any sense? Any at all? I hope it does. And if it does, you're ready for VERITAS. We're going to explain more, much more, as we roll along, but at least understand that:

```
        Volumes are made up of plexes.
        Plexes are made up of subdisks.
        Subdisks are made up of VM Disks.
        VM Disks are made up of (real) Disks.

        and,

        Disks can be turned into VM Disks.
        Vm Disks can be turned into Subdisks.
        Subdisks can be grouped into Plexes.
```

```
Plexes can be grouped into a Volume.
```

Good. One more note about plexes before we move on. Here's the groovy thing about plexes. Because plexes are "mirrors", we can mirror the volume we built earlier simply by building another plex identical to the first one, using two more subdisks (which means we need 2 more vmdisks, etc,etc). Once we build the second plex, we attach it to the volume (myvol) and presto chango! We're mirrored! This is really kool... mirrors are something magical, they are a part of everything. If you have only one mirror you may see yourself, but you'll need a second mirror to see yourself holding the mirror. Hmmm?

VERITAS Object Creation

In VERITAS there are two ways to create objects. There's the "vxmake" way and the "vxassist" way. Here's the difference...

VxMake is an object creation tool that allows you to specifically build objects, one at a time, by your specific specs. The methods we mentioned before (add a disk, make a VM disk, create a subdisk, then...) would be an example of how we use vxmake. Piece by piece, we'd build out components, and assemble them. This is kool because there is no "guess work", meaning we are explicitly saying "build this!" and it's built. We'll learn more about vxmake, in detail, in the next course.

VxAssist is the other way. As you've been reading this course, have you said "this sure looks like a lot of work!"? Well, here's your answer: vxassist. When we want to build a volume, but we don't want to specify every little detail, we could add disks to make VM disks and then use vxassist to do the rest. So instead of using the "build this, and only this!" philosophy of vxmake, we turn to the vxassist mind set that sounds something like "OK, here are some disks, build me a RAID5 that can house 10G of data... I don't care how you do it, just hurry up." Vxassist is handy for a jam, or when you really just don't care exactly how things are set up. Again, in our following course we'll discuss object creation at length using vxassist.

Don't get any ideas yet about which one is better. When we get to the courses applicable to these tools and you see how they work with your own eyes (yup, you'll see lots of examples) then you can get defensive about your favorite. But also remember, you can use them both interchangeably in some cases, and often with use both. Keep your eyes open for these guys though, because they are very different ways of doing things.

Special Notes on RAID0 and RAID5: Layouts

I want to briefly discuss layouts. This is a topic will delve into more thoroughly later when we talk about plex creation, but because the "VERITAS Getting Started Guide" talks about them, I'll give you the highlights. This stuff may seem pretty basic now, but the more you work will VERITAS the more you will come to appreciate layouts.

Layout can be thought of as the defining difference between the different types of RAID. Simple RAID's don't really have a layout, you simple write linearly starting at the beginning of the first disk until you hit the end, at this point you start writing the second disk, and so on. Striped RAID's have special type of layout, however. Layouts use the following terms:

- stripe width (stwidth): The Amount of data to write to a disk before moving to the next. (Generally 32k by default)
- columns (ncolumn): A subdisk.

In the case of a Striped RAID, each subdisk would have a column number. A RAID0 with 4 disks, would have 4 columns. The first disk would be column 0 (remember: everything starts with 0), the second would be column 1, the third would be column 2, and the fourth disk would be column 3. Now, when we build a striped RAID we specify both stripe width (which VERITAS calls "stwidth"), and the number of columns (which VERITAS calls "ncolumns") which is the number of subdisks that are going to have data on them. So, as mentioned earlier, we've got 4 disks, and we know the column numbers for each subdisk. If we assigned a stripe width of 32k, and we wrote some data to the volume, we'd write the data 32k at a time, writing each new 32k chunk to the next column, and wrapping around from the 4th column (column 3) to the 1st column (column 0), untill all the data is written. Refer to the "RAID Theory" course for more on this. If you've got this down, but feel foggy, just roll with it, it'll click later as we build plexes.

Now, RAID5 is very similar because it's striped, it's just a different type of stripe. RAID5 uses a layout called "Left Symmetric Layout", or more commonly as simply "RAID5 layout". It would take me too much time to explain this, so check the book. Read through the "RAID Theory" course if you need help, and then go back to the book. You may have to look twice, but you'll get it.

Volumes and Devices

So now that you understand how a volume is built, and you understand a little about how it works lets talk about how you use them.

As mentioned earlier, a volume is a container. A "bit bucket" is you will. The advantage of a volume over a normal disk partition, is that volumes aren't static. You can grow, shrink, move, rearrange, and manipulate volumes in ways you just can't with a partition. You can also do all this in a mounted and live environment with no reboots. When you don't have enough disk space in a filesystem you can just grow the volume that the filesystem is on, and then grow the filesystem. Very easy.

But... you're saying to yourself, "how do I actually use a volume?" Well, that's easy

enough. You use a volume just like a disk block device. Once VERITAS is installed and set up, you'll see a new directory in /dev. You'll see /dev/vx/, which is where all of your volume block devices are. If you created a volume named "myvol", and wanted to "newfs" it, you could do it like this:

```
# newfs /dev/vx/rdsk/myvol
```

or you could mount it like this:

```
# mount /dev/vx/dsk/myvol /myvol
```

Kool huh? Nothing magical. Poke around in the /dev/vx file tree and see where things are, everything is fairly well named, and self explanatory. If you're new to device trees, then just know that you can access your volumes from /dev/vx/dsk/ as though it were a normal disk.

> See how simple and logical VERITAS is? This is want makes VERITAS so kool. There's no voodoo whoodoo. Everything is clearly laid out, and organized. I love VxVM and I'm hoping that you're starting to as well.

Kernels and Daemons

One thing that separates VERITAS from other volume managers is its integration with the host operating system. VERITAS actually utilizes kernel modules, which are the backbone of the application. You can see some of the driver loads in /etc/system. You'll see several driver forceloads. This is important to note because on most systems you'll see "WARNING: Forceload of dev/XXX Failed!" messages at boot up. Don't worry about these, it just means that VERITAS is trying to load a driver for a piece of hardware you don't have.

I wanted to talk about VERITAS Installation, but I'm going to leave that to the book, but be VERY careful on VERITAS and Solaris upgrades to a system using VERITAS. Remember that because VERITAS uses kernel modules, that if you should wipe out and/or replace the kernel you could loose VERITAS.... which means you'll get a very intimate session alone with the books and the system while you learn how to recover botched VERITAS installations. Please, be informed about this, I've killed a couple machines this way, and luckily they were all "test" machines and not production servers.

VERITAS also has several daemons running all the time to keep watch over your volumes, and to ensure smooth operation. To look for them try typing "ps -ef | grep vx"

on a machine running VERITAS. You generally will see 3 common daemons, which are:

- vxiod : The VERITAS IO Manager, which managers reads and writes to volumes.
- vxconfigd : The Vx Object Configuration Manager.
- vxrelocd : The Vx Hot Swap Manager, which reacts to disk failures with spares.

Maybe, by now your asking "Hurry up, just tell me where the VERITAS configuration file is!". Whelp, there isn't one! Everything we do to VERITAS is watch by the vxconfigd. The vxconfigd then updates VERITAS's object database, and keeps watching. As for the other daemons, they are pretty self explanatory. Vxiod makes sure that data is written to the volume the way it's supposed to be (so if the volume is striped, vxiod makes sure the data is written to the stripe the way we told it too, etc..). Vxrelocd just looks for disk failures and if it finds one it puts a spare (if there is one) in it's place.

Intro to Advanced Features

As we wrap this up, lets talk about VERITAS's truly special, and more "advanced" features. Please don't for an instant think this is the complete and definitive list. Rather, this is a small list of features that I really think separate VxVM from the other volume managers available. All of these topics will be covered in detail in the "Advanced VERITAS Theory" course later on.

- Dynamic Multi-Pathing: Also know as DMP, this feature allows for arrays which can be attached to multiple controllers (such as the Sun A5000 family of arrays) to leverage them for both load balancing and redundancy. So if your disks are attached to both c1 and c2, and your using a 100Mb/s fiber connection, with DMP you'll access the disks at 200Mb/s. And if c1 dies, for any reason, you still have access to the disk through c2, which means zero downtime for your volume. DMP can really save your butt if there is a failure.
- Hot Spares: You can allocate VM Disks as spares, so that if a disk in one of your plexes dies for some reason the spare will immediately take it's place and again, zero downtime. You can have as many hot spares as you like, and VM Disks allocated as spares can are independent of any particular disk group, plex or volume, so you don't need to have different spares for different volumes. Spares are managed by the vxrelocd as mentioned earlier.
- Dirty Region Logging: Also known as DRL, this feature can be thought of as a volumes diary. Before writes are made to a DRL'ed volume VERITAS makes a note of the write in the DRL, marking that region "dirty". After the note is made in the log,

the write occurs. The idea here is that if the system crashes, when the system comes back online it won't know what writes were in progress. So, if a DRL log exists for the volume, it checks the log and then sync's dirty regions from the other mirror(s). In this way, we don't have to worry about corrupted data. DRL only applies to mirrored volumes, even though DRL can be added to unmirrored volumes (it just won't do anything useful).

- Snapshotting: Users of NetApp Filers will be familiar with this. Snapshotting is used for online backups. When you "snapshot" a volume, you are saying "I want to create a static picture of what the volume looks like right now". Then once the snapshot is created it appears as a new temporary volume, which can be mounted and backed up. Once the backup is done you "release" the snapshot. The kool thing is that while the snapshot exists the volume is still read and writable to everyone else. Zero downtime backups!

OK, I'm going to leave the list at that for now. Don't get frustrated if your thinking "Hey, that's kool! But your not telling me anything useful!", because all of that will come later in the "Advanced VERITAS Theory" course, where I will talk about these topics and more in detail. For the meantime, just know that they exists and can be used.

The Wrap Up

Congratulations! You now are ready to actually start playing with a VERITAS system and have a good idea of what's actually going on. The next course is split into two camps, one for VxMake and one for VxAssist. Choose either, but you should read both. At this point you should read, or re-read the "VERITAS Getting Started Guide" for you 3.x users, and for SEVM 2.6 or older users, read or re-read the first chapter of your SEVM manual. Make sure that everything makes sense, and then continue on.

Feel free to comment on this tutorial to benr@cuddletech.com