The Cuddletech Veritas Volume Manager Series

Volume Kreation: VxAssist & Ways to spend more time cuddling

Ben Rockwood
CDLTK
Cuddletech

benr@cuddletech.com

VxAssist is a versital and easy tool for creating and manipulating Veritas Volume Manager volumes. This course is an introduction to the tool with numberous examples of it's usage, syntax, and indepth looks at the results.

Introduction

JUST DO IT ALREADY!!!

VxMake is certainly the best way to build things. You know what you want, you know how you want it, and you want it done right. Right? What if you don't k now exactly what you want. Even worse, what if you don't care! Thats where vxassist comes in. You can directly build volumes without building the sub-obje cts and give vxassist just as much or little info as you like. The trade off? The less you tell vxassist the more it decides for you. This can be really b ad. Kinda like telling your kids to put away the groceries without telling them specifically where to put things... you might find places to put things in your kitchen you didn't know were there.

There is a time and place for everything. Often this actually has to do with what you're using and how much you care. On a Sun A5200 you might want to put only one volume per disk. You might want each disk in Array A to be perfectly mirrored to a disk in the same slot in Array B. But you might also be using a D1000 where you don't care what goes where. For these reasons you may want to carefully decide which tool you should use for building. We'll talk about a number of reasons and ways in which to use the tool. But remember that analogy of the kids in your kitchen, it's pretty close to letting vxassist decide whe re to dump all your data.

VxAssist: Why it's the Lazy Way

This course directly correlates to its sister course "Volume Kreation: The VxMake Way". There we saw that we can acheive a huge ammount of control over our volume creations by building each volume object by object untill we have a well humming bit bucket for our data to live. But all too often we just need the data quickly, or we don't care about the nitty gritty details, or, more likely, we simply don't have the time neccisary to broad over the details of each volume. What's needed is a simple, quick method of creation. Not to mention the fact that when we build each of our objects by hand we leave ourselves open to accidental errors which may be difficult to catch. Chief ammong all other reasons this is why vxAssist should be carefully considered as an alternative (if used properly!). A simple example of the dangers of manual creation: when you create a volume from the ground up you'll quickly realize the need for a shell that has command history (such as the Borne Again SHell). And you'll be humming along recalling the last command, changing the neccisary bits and building along. At some point, I guarrenty you'll accidently name an object wrong doing this. And that little innocent typo, that may or may not cause an error, will cause you headaches. But, with that said, there are times and places for both. For instance, if you simply want a plex to use you'll need to vxMake it yourself, and therefore you need a good foundation in using it. Furthermore, you can not properly understand the Volume Manager without having a good solid background in vxMake.

But what makes VxAssist such a "lazy" method. It's the fact that you can, in one line, tell the Volume Manager to create you a, say, 20G volume on disks A, B, and C, and bang! with only that information you've got a functional usable ready for filesystem volume. Or you could even go so far as to only tell VxAssist to create you a 20G volume named "newvol" and not even tell it which disks to use, it'll just find the space in it's configured disks and go off on it's merry way. Whatever you lack to tell VxAssist about the volume your creating it will descide for itself, but as we mentioned before, the less you descide the less control you have and the more likely you are to get something you didn't want. In the examples we'll go through below we'll build far more complex volumes than we did in the VxMake course, but we'll put heavy emphisis in using as much information as we can to get the job done right, the first time. Let's hop into it.

Volume Lessons Background

Let's take a quick look at the equipment we'll be using for our examples below. I'm using a Sun Ultra1 200E with 128M of memeory. For our test disks we'll be using a Sun A5100 Fibre Channel disk array with 14 9G disks. This will provide enough disks to build some much larger structures than we did in the vxMake course using only 4 disks. We'll be using the UFS file system, VERITAS Volume Manger version 3.0.3 on Solaris8. Here a quick peek at our disks and system for your reference:

```
# uname -a
SunOS gaff 5.8 Generic_108528-15 sun4u sparc SUNW,Ultra-1
# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
       0. c0t0d0 <SUN4.2G cyl 3880 alt 2 hd 16 sec 135>
          /sbus@1f,0/SUNW,fas@e,8800000/sd@0,0
       1. c0t1d0 <SUN4.2G cyl 3880 alt 2 hd 16 sec 135>
```

```
          /sbus@1f,0/SUNW,fas@e,8800000/sd@1,0
       2. c1t0d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w2200002037096efd,0
       3. c1t1d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020372d0f69,0
       4. c1t2d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370971e8,0
       5. c1t3d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w2200002037097752,0
       6. c1t4d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370970f3,0
       7. c1t5d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370d44c8,0
       8. c1t6d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370d3926,0
       9. c1t16d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370e0b08,0
      10. c1t17d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370e85f8,0
      11. c1t18d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370d3bef,0
      12. c1t19d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370d44ee,0
      13. c1t20d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w220000203714322b,0
      14. c1t21d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370971df,0
      15. c1t22d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370d44d2,0
Specify disk (enter its number): ^D
#
```

You'll notice that SCSI controller 0 (c0) is the onboard SCSI controller for our internal 4.2G disks and SCSI controller 1 (c1) is our disks in the A5100. The split in the number of targets (c1t6 then we skip to c1t16) is due to the way A5100's dish out SCSI target numbers. Let's take another look at how we can add these disks into Volume Manager (VM) control, but using a diffrent method than we used in the VxMake course.

The Pregame Show: Disk Groups and vxdiskadd

In the VxMake tutorial we added disks using the helpful "vxdiskadm" ncurses app. Something you'll find is that most of the options you are given via "vxdiskadm" are not actually a part of the app, but rather it simply acts like a hub for them. This is really helpful for thoughs of us with bad memories who just can't remember the name or the args, but sometimes a quicker method is nice. Therefore we'll add in our disks this time using the "vxdiskadd" command. But before we do that, let's talk about Disk Groups (dg's), a topic we avoided for simplicity sake in the VxMake tutorial, but mentioned in the Krash Kourse.

Disk Groups are a method to segregate VM objects. Each system is required to have at least one disk group named "rootdg". You must have at least one disk in rootdg, whether you use it or not. The rootdg is used to store some fundimental information about the VM on the system so it's not an option. However

we can create as many other disk groups as we like. Now, you might wonder why we don't simply do everything in rootdg all the time, like we did in the vxMake tutorial. There are two good reasons: first is that each disk group is "self contained", if you add 5 disk to a disk group and create a volume you ONLY have those 5 disks to work with (unless you add more to the DG, obviously) and therefore can be used as a sort of logical seperation from the rest of your storage subsystem. The second reason is that you can import and deport disk groups. When you import a disk group all volumes, disks, plexes, and any other objects in that disk group become usable by the system. When you deport a disk group all the objects and the disk group itself vanish from the system. If this seems vague, think of a file system; you create a filesystem on a disk, create files on it, etc. When you mount the filesystem all the contents are avalible and ready to use, the system is fully aware of that disks filesystem. But when you umount the filesystem suddenly all that data sort of "disappears" back into the darkness of it's disks... the data is there, you know it is, but unless that file system was noted in the /etc/vfstab a passerby wouldn't even know it exsisted. This is similar to the concept of disk groups. To illistrate why import/deporting of disk gropus is so powerful, lets again draw an analogy to a filesystem on disk: when you umount a filesystem on disk you suddenly have the option of taking that disk out of the system, putting it in another and by knowing what type of filesystem it is, and it's device number you can re-mount that filesystem on a completely diffrent system, which can be a real asset! You can do the same with disk groups! You can deport a disk group full of volumes, then move the disks with that disk group (you must move ALL of them, if you break up the disks it won't import) to another system where you simply import the disk group, a start all the volumes. You can see how useful that could be. For this reason, in this tutorial we will do all our example using an added options to almost all VxVM commands (the -g option) which specifies the disk group to work with. If you do not specify the disk group you are working with rootdg is assumed.

So let's start setting up our test enviorment for this course. First let's add our disks. We're going to add all disks in the A5100 (all 14) and we'll be adding them to a new disk group. We'll need to check the disks via "format", then use "vxdiskadd" to add them and put them in a disk group. Here we go.....

```
# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
       0. c0t0d0 <SUN4.2G cyl 3880 alt 2 hd 16 sec 135>
          /sbus@1f,0/SUNW,fas@e,8800000/sd@0,0
       1. c0t1d0 <SUN4.2G cyl 3880 alt 2 hd 16 sec 135>
          /sbus@1f,0/SUNW,fas@e,8800000/sd@1,0
       2. c1t0d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w2200002037096efd,0
       3. c1t1d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020372d0f69,0
       4. c1t2d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370971e8,0
       5. c1t3d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w2200002037097752,0
       6. c1t4d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370970f3,0
       7. c1t5d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370d44c8,0
       8. c1t6d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w22000020370d3926,0
       9. c1t16d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
```

There we have it, from start to finish, from SCSI disks to a usable volume with filesystem and it's even mounted. You'll notice that the device path is in the form: /usr/vx/(r)dsk/<diskgroup>/<volumename>. Everything else should be familar to you.

From this point on I will limit down the ammount of output, such as only showing the volume information in the vxprint output instead of all of it, and we won't cover creation of filesystems since you already know how to do that and every volume (regardless of how it's created) is created in a similar fashion as above. With that said, let's get onto the next lesson!

Volume Lesson 2: Striped Volumes

VxAssist will, by default, create concatinated (simple) volumes. If we had descided to create a 40G volumes in the first lession it would have simply created a 40G concat using up as many disks as it needed. But simple volumes really don't give us the speed and power that we really want from a volume, so in this lesson we'll get just alittle more specific with vxassist and great a large (40G) striped volume. Let's look at the syntax:

```
vxassist -g <diskgroup> -U <usagetype> make <volname> <size> \
 layout=<layouttype> stwidth=<width> ncolumn=<#> <disk1> \
 <disk2> <disk3> ....
```

Alright, we discused the first several options, and the others you'll recall from the VxMake course when you made striped plexes. "layout" specifies the layout type for the volume we're building (concat, stripe, mirror, RAID5, etc). The options "stwidth" and "ncolumn" are stripe specific options defining the width of the stripe unit for each column. The remaining options specify the vmdisks we want vxassist to use in building this volumes.

It should be noted that we can use as many or as few of these options as we like. For instance, we can make the layout "stripe" but not specify the stripe width or number of columns. We can specify the stripe width and stripe layout, but not the number of columns or disks to use. You should exersies some common sense when omiting options, thinking of what vxassist is going to need to be told to make it's choice, but without being told it will do the best it can to meet your request. In the following examples we'll build a striped volume using only the "layout" option, and letting vxassist figure the rest out.

```
# vxassist -g cuddledg -U fsgen make stripevol 40g layout=stripe
# vxprint -g cuddledg -hrt
DG NAME         NCONFIG      NLOG     MINORS   GROUP-ID
DM NAME         DEVICE       TYPE     PRIVLEN  PUBLEN   STATE
V  NAME         USETYPE      KSTATE   STATE    LENGTH   READPOL   PREFPLEX
PL NAME         VOLUME       KSTATE   STATE    LENGTH   LAYOUT    NCOL/WID MODE
SD NAME         PLEX         DISK     DISKOFFS LENGTH   [COL/]OFF DEVICE   MODE
SV NAME         PLEX         VOLNAME  NVOLLAYR LENGTH   [COL/]OFF AM/NM    MODE

 (..... removed ..........)

v  stripevol    fsgen        ENABLED  ACTIVE   83886080 SELECT    stripevol-01
pl stripevol-01 stripevol    ENABLED  ACTIVE   83907654 STRIPE    7/128    RW
sd cuddle-f0-01 stripevol-01 cuddle-f0 0       11986758 0/0       c1t0d0   ENA
sd cuddle-f1-01 stripevol-01 cuddle-f1 0       11986758 1/0       c1t1d0   ENA
sd cuddle-f2-01 stripevol-01 cuddle-f2 0       11986758 2/0       c1t2d0   ENA
```

```
sd cuddle-f3-01 stripevol-01 cuddle-f3 0       11986758 3/0       c1t3d0   ENA
sd cuddle-f4-01 stripevol-01 cuddle-f4 0       11986758 4/0       c1t4d0   ENA
sd cuddle-f5-01 stripevol-01 cuddle-f5 0       11986758 5/0       c1t5d0   ENA
sd cuddle-f6-01 stripevol-01 cuddle-f6 0       11986758 6/0       c1t6d0   ENA
```

VxAssist descided that to create our volume it would use 7 disks, use a 64k stripe width (128 sectors). Because it doesn't need the full size of each vmdisk it created 7 subdisks that are 11986758 sectors in length (5.9G). There this really comes to be diffrent than if we had used vxmake to create our own striped volumes is that more than likely we'd have used 5 disks and used the full length of each disk, creating a volume about 45G in size, but vxassist instead gets as close to the desired volume size as possible. This, in and of itself, has advantages.

Let's build another volume, but this time specify the stripe width, the number of columns and the vmdisks to use.

```
# vxassist -U fsgen -g cuddledg make stripevol2 40g layout=stripe stwidth=128k \
ncolumn=5 cuddle-f0 cuddle-f1 cuddle-f2 cuddle-f3 cuddle-f4
# vxprint -g cuddledg -hrt
DG NAME         NCONFIG      NLOG     MINORS   GROUP-ID
DM NAME         DEVICE       TYPE     PRIVLEN  PUBLEN   STATE
V  NAME         USETYPE      KSTATE   STATE    LENGTH   READPOL   PREFPLEX
PL NAME         VOLUME       KSTATE   STATE    LENGTH   LAYOUT    NCOL/WID MODE
SD NAME         PLEX         DISK     DISKOFFS LENGTH   [COL/]OFF DEVICE   MODE
SV NAME         PLEX         VOLNAME  NVOLLAYR LENGTH   [COL/]OFF AM/NM    MODE

 (..... removed ......)

v  stripevol2   fsgen        ENABLED  ACTIVE   83886080 SELECT    stripevol2-01
pl stripevol2-01 stripevol2  ENABLED  ACTIVE   83903943 STRIPE    5/256    RW
sd cuddle-f0-01 stripevol2-01 cuddle-f0 0      16780743 0/0       c1t0d0   ENA
sd cuddle-f1-01 stripevol2-01 cuddle-f1 0      16780743 1/0       c1t1d0   ENA
sd cuddle-f2-01 stripevol2-01 cuddle-f2 0      16780743 2/0       c1t2d0   ENA
sd cuddle-f3-01 stripevol2-01 cuddle-f3 0      16780743 3/0       c1t3d0   ENA
sd cuddle-f4-01 stripevol2-01 cuddle-f4 0      16780743 4/0       c1t4d0   ENA
#
```

In this example you can see that I told vxassist exactly what I wanted, and that's exactly what I got. I hope that these two examples help you see just how vxassist can intelligently "fill in the gaps", but may not do what we assume. If you assumed that vxassist would build a 5 disk stripe for our 40g volumes you might have been supprised when you got a 7 disks stripe. This helps illustrate that if you descide to be indiffrent about a layout descision you should be sure about it, otherwise you'll be disappointed in what you get.

Because in this course concepts are much easier to grasp we'll go even further than we could in the VxMake course. Let's next jump into building mirrored volumes and RAID0+1 volumes using vxassist.

Volume Lesson 3: RAID1 and RAID0+1 Volumes

With the same ease we used to create simple and striped volumes we'll now build mirrored volumes. We'll start with a simple 10G mirrored volume so that we can see how a RAID1 volume will look when using 2 disks for each mirror. But first, let's examine the syntax:

```
vxassist -g <diskgroup> -U <usagetype> make <volname> <size> \
 layout=<layouttype>  <disk1> <disk2> <disk3> ...
```

This syntax is looking real repeditive isn't it. In this case we'll use the same options as usual, but we'll specify the layout as "mirror-concat":

```
# vxassist -U fsgen -g cuddledg make mirrorvol3 10g layout=mirror-concat
# vxprint -g cuddledg -hrt
DG NAME         NCONFIG      NLOG     MINORS   GROUP-ID
DM NAME         DEVICE       TYPE     PRIVLEN  PUBLEN   STATE
V  NAME         USETYPE      KSTATE   STATE    LENGTH   READPOL   PREFPLEX
PL NAME         VOLUME       KSTATE   STATE    LENGTH   LAYOUT    NCOL/WID MODE
SD NAME         PLEX         DISK     DISKOFFS LENGTH   [COL/]OFF DEVICE   MODE
SV NAME         PLEX         VOLNAME  NVOLLAYR LENGTH   [COL/]OFF AM/NM    MODE

 (..... removed ......)

v  mirrorvol    fsgen      ENABLED  ACTIVE   20971520 SELECT    -
pl mirrorvol-01 mirrorvol  ENABLED  ACTIVE   20975031 CONCAT    -        RW
sd cuddle-f4-01 mirrorvol-01 cuddle-f4 0      3296538  0         c1t4d0   ENA
sd cuddle-r4-01 mirrorvol-01 cuddle-r4 0      17678493 3296538   c1t20d0  ENA
pl mirrorvol3-02 mirrorvol  ENABLED  ACTIVE   20975031 CONCAT    -        RW
sd cuddle-f5-01 mirrorvol-02 cuddle-f5 0      3296538  0         c1t5d0   ENA
sd cuddle-r3-01 mirrorvol-02 cuddle-r3 0      17678493 3296538   c1t19d0  ENA
```

The beauty here is that the mirroring is done all in one faul swoop. You can go so far as to even use an optional parameter after specifing the layout to vxassist which will create multiple mirrors! To do this use the option: "nmirror=X" where X is the number of mirrors (plexes) you want.

It should be noted, that you can also create a volume that isn't mirrored and then mirror it later using the following syntax:

```
vxassist -g <diskgroup> mirror <volname>
```

This will create a mirror that is (hopefully) identical to the exsisting plex. Here's an example of creating a 10g striped volumes and then mirroring it afterwards:

```
# vxassist -g cuddledg -U fsgen make stripevol 10g layout=stripe
# vxprint -g cuddledg -hrt stripevol
 (... removed ...)
v  stripevol    fsgen        ENABLED  ACTIVE   20971520 SELECT    stripevol-01
pl stripevol-01 stripevol    ENABLED  ACTIVE   20989653 STRIPE    7/128    RW
sd cuddle-f0-01 stripevol-01 cuddle-f0 0       2998485  0/0       c1t0d0   ENA
sd cuddle-f1-01 stripevol-01 cuddle-f1 0       2998485  1/0       c1t1d0   ENA
sd cuddle-f2-01 stripevol-01 cuddle-f2 0       2998485  2/0       c1t2d0   ENA
sd cuddle-f3-01 stripevol-01 cuddle-f3 0       2998485  3/0       c1t3d0   ENA
sd cuddle-f4-01 stripevol-01 cuddle-f4 0       2998485  4/0       c1t4d0   ENA
```