Understanding the VS: Coraid's EtherDrive$^{®}$ Storage Virtualization Appliance

ABSTRACT

The *Coraid* VS storage virtualization appliance provides a flexible way to allocate and use ATA-over-Ethernet storage appliances, such as the SR2461 or SR1521. Here we describe the limitations common in raw storage appliances and explain how these problems are addressed by using storage virtualization. Finally we illustrate how to use the VS to perform common operations.

This is a draft copy.
Date: 24 July 2008.

Limitations Of Simple RAID Arrays

Network RAID appliances that combine several disk drives into a single high performance, reliable network storage target are a cost-effective way to provide storage.

The *Coraid* EtherDrive SR2461, for example, can combine twenty-four disk drives into a single large RAID target. It is common to assemble twenty-three 1TB disks in a RAID 5 configuration yielding 22TB of storage with one disk being used as a hot standby. If one of the disks in the RAID fails, the standby is drafted to be the failed disk's replacement.

Network RAID appliances have many advantages. Since the storage is no longer captive to the system to which it is directly attached, it can be shared by numerous servers on the network. Due to the design of Ethernet networking, there is practically no limit to the amount of storage that can be put onto a network.

RAID appliances do have some limitations, though. One is related to the size of storage created using RAID techniques. The RAID storage is a single large chunk which may not be the right size for a particular task. This large volume could be divided using client software to partition the device, but partition allocation is largely static. A partition cannot be grown when the storage beyond its end is already being used. Making a partition smaller would leave a hole which may be of an awkward and unusable size.

Another problem is the issue of redundancy. One reason a *Coraid* SR is so affordable is that it is a non-redundant RAID controller. If the RAID controller fails, then the data is not available until the RAID controller is replaced. The VS can use multiple SR appliances to achieve redundancy in a modular way, so that storage remains available in the event of a single controller failure. The SR itself remains simple and affordable for those customers who do not need this redundancy.

The Network Storage Virtualization Solution

With the advent of the *Coraid* EtherDrive VS, AoE storage networks now have many advantages available in much more expensive Fibre Channel installations. The VS virtualizer appliance creates virtual targets constructed out of available physical RAID unit storage. The physical storage is added to the VS as physical volumes. The virtualizer divides the physical volumes into extents on the order of four million bytes, 4MB, each. All the extents are then added to a pool, called a volume group. Multiple physical volumes can be added to a volume group, and there can be as many groups as needed. These groups can be used to to keep storage segregated by administrative criteria. For example, different volume groups can be used to keep track of the resources to be used by various departments within an organization, separating various ‘clients’ of the system. If desired, all the storage can simply be put into a single volume group.

The extents in these groups are the raw material used to build logical units. On the VS, a logical unit, also called a *LUN*, is a virtual device that is the aggregate of extents drawn from a pool. VS LUNs are completely virtual. The real storage is on the physical volumes. There is no storage to speak of in the VS unit itself; all the storage is in the underlying targets. The VS creates a logical unit by allocating extents from one of the volume group pools to back the LUN. The LUN is then placed online to make it accessible from client initiators.

Since the LUN is made of independent extents, it can be any size. A LUN can be grown and shrunk without moving any data. The underlying storage for a LUN can be spread over several physical volumes. Data can be migrated from one physical volume to another while being used by the LUN. By chunking the storage into extents and allocating a map of them for each LUN, virtualization provides all the flexibility needed.

The virtualizer translates and forwards AoE requests. A client sends a read or write AoE request to the LUN, and the VS looks up in a table the real location of the storage. The request’s logical block address and shelf/slot address are changed to reflect the location on the physical volume. The request is then sent to that physical volume for processing. The AoE response in most cases can return directly to the original requester avoiding having to pass back through the VS.

This leads to the question of where the virtualizer keeps its information about the storage. That is "Where is the map of real storage kept?". Such information is called *metadata*. The VS’s metadata is stored in extents allocated from the same pool as the data. When a physical volument is added to a volume group, extents are allocated from the physical volume to hold the extent table metadata. Each extent entry contains the LUN and the offset in that LUN for the extent the entry represents. Every extent in the physical volume has an extent entry. These entries are used to form the map of logical-to-physical translation mapping for a LUN and to track extent features (such as whether the extent has been written for efficient LUN/PV copying).

Using the *Coraid* EtherDrive VS

The first step when installing a VS is to set its two shelf addresses. The first address -- the admin shelf or simply shelf*P address -- is used to manage the appliance. The shelf address is used for flash based LUNs and for CEC. The second address -- the service shelf or srvshelf -- is used to export the virtual storage LUNs. In environments where only one VS will serve the LUNs, both addresses can be the same value.*

```
VS-1:-1> shelf 63
VS63:-1> srvshelf 163
VS63:163>
```

In the above example the shelf address is set to 63 and the service shelf address is set to 163. Notice that the prompt reflects both these numbers.

The *lsaoe* command is used to see the physical volumes available on the network.

```
VS63:163> lsaoe
TARGET            LENGTH  STATE  PORT          ADDR
8.1           1152.874GB      V     0  0060dd475d90
8.1           1152.874GB      V     1  0060dd475d90
21.1          1000.215GB      V     0  0030488b67ca
21.1          1000.215GB      V     0  0030488b67cb
21.1          1000.215GB      V     1  0030488b67cb
21.1          1000.215GB      V     1  0030488b67ca
61.0          6001.294GB      V     0  00304833f6aa
61.0          6001.294GB      V     1  00304833f6aa
61.1          6001.294GB      V     0  00304833f6aa
61.1          6001.294GB      V     1  00304833f6aa
VS63:163>
```

Each target may appear more than once if there are multiple network interfaces connected on the VS as each interface can see the target. This provides multiple paths to the targets for higher reliability. For network redundancy, two switches can be installed with each interface of the VS plugged into each switch and each interface of target devices plugged into each switch. The VS will load balance across all available local interfaces and all available target addresses for a given shelf.slot.

lsaoe -c displays the config strings set on the targets. The VS uses the config string of an AoE target to store VS configuration information. As shown below, target 21.0 is already claimed for use. The format of the config string is not published as it is subject to change, but the first field is a magic string used to denote this target as a PV, and the second field denotes the srvshelf the target belongs to. As below, target 21.0 is claimed by the VS with srvshelf 77.

```
VS63:163> lsaoe -c
TARGET        LEN  CONFIG
8.1             0
21.1           54  'CoraidPV 77 21.1 -1.-1 0 1 238469 storage 1 53 21.1.0 '
61.0            0
61.1            0
VS63:163>
```

In this case and in all the cases to follow, see the documentation on each individual command for a complete description.

To create a volume group, use the *mkvg* command. A volume group has to be created before any physical volumes can be put into it.

```
VS63:163> mkvg accounting
VS63:163> lsvg
NAME                      LENGTH        AVAILABLE  EXTSZ    PVS
accounting               0.000GB          0.000GB    4MB  00/00
VS63:163>
```

As shown, the volume group is empty. A group without any members won't survive a reboot since its existence is recorded on its constituent physical volumes.

```
VS63:163> mkpv 61.0 accounting
2008.07.23 11:16:55 updating 61.0
VS63:163> lsvg
NAME                      LENGTH        AVAILABLE  EXTSZ    PVS
accounting            6001.289GB       6001.277GB    4MB  01/01
VS63:163> mkpv 61.1 accounting
2008.07.23 11:18:08 updating 61.1
2008.07.23 11:18:09 updating 61.0
VS63:163> lsvg
NAME                      LENGTH        AVAILABLE  EXTSZ    PVS
accounting           12002.579GB      12002.554GB    4MB  02/02
VS63:163> lspv
TARGET         VOLUME GROUP        LENGTH     AVAILABLE          NPE  MIRROR
61.0             accounting    6001.289GB    6001.277GB      1430819
61.1             accounting    6001.289GB    6001.277GB      1430819
VS63:163>
```

In the above example the *mkpv* command was used to add physical volumes to the 'accounting' volume group. The *lsvg* command displays 12TB in the volume group divided up into 4MB extents. The *lspv* command displays information about the added pvs. The NPE field is the number of extents in the physical volume. These extents are the raw material used to build logical units.

You may notice that the AVAILABLE amount for each physical volume is less than the LENGTH amount. This is due to extent allocation for holding the physical volume's metadata. The *lspv -a* command displays additional information about a physical volume, including the location of the metadata extents.

The following shows that the allocated 12MB corresponds to three metadata extents allocated for the PV.

```
VS63:163> lspv -a 61.0
TARGET         VOLUME GROUP        LENGTH     AVAILABLE         NPE  MIRROR
61.0             accounting    6001.289GB    6001.277GB     1430819
        luns:
        metadata: 61.0.1 61.0.2 61.0.3
        nvec=1430819 data=0 written = 0, dirty=0 cow=0 changing=0 metadata=3 scanned=1
VS63:163>
```

We can now look at the *config string* on the physical volumes and see that our VS has claimed the storage.

```
VS63:163> lsaoe -c
TARGET       LEN  CONFIG
8.1            0
21.1          54  'CoraidPV 77 21.1 -1.-1 0 1 238469 storage 1 53 21.1.0 '
61.0          72  'CoraidPV 163 61.0 -1.-1 0 2 1430819 accounting 1 6 61.0.1 61.0.2 61.0.3 '
61.1          72  'CoraidPV 163 61.1 -1.-1 0 2 1430819 accounting 1 6 61.1.1 61.1.2 61.1.3 '
VS63:163>
```

Now, let's build a LUN.

```
VS63:163> mklun 0 750g accounting
2008.07.23 11:28:48 updating 61.1
2008.07.23 11:28:49 updating 61.0
VS63:163> lslun
LUN           LENGTH  ONLINE  LABEL
0          750.000GB     OFF
VS63:163> lsvg accounting
NAME                      LENGTH        AVAILABLE  EXTSZ    PVS
accounting           12002.579GB      11252.550GB    4MB  02/02
VS63:163>
```

We have now made a 750GB logical unit out of extents from the accounting group. We then executed the *lslun* command to see it and then *lsvg* to see that blocks were allocated from the physical volumes.

We still can't access LUN 0 from client initiators. It must first be put online with the *online* command.

```
VS63:163> online 0
2008.07.23 11:30:24 putting lun 0 online
VS63:163> lslun
LUN           LENGTH  ONLINE  LABEL
0          750.000GB      ON
VS63:163>
```

Putting a LUN online triggers an AoE broadcast message announcing the target, in this case shelf 163 and slot zero. There can be up to 255 LUNs in a single shelf address.

To tear everything down, reverse the above procedure using the associated remove commands.

```
VS63:163> offline 0
2008.07.23 11:31:29 Taking lun 0 offline
VS63:163> rmlun 0
Are you sure you want to remove lun 0?  [n]: y
Freeing Lun...
2008.07.23 11:31:33 updating 61.1
2008.07.23 11:31:33 updating 61.0
VS63:163> rmpv 61.1
2008.07.23 11:31:38 updating 61.0
VS63:163> rmpv 61.0
VS63:163> rmvg accounting
VS63:163>
```

Notice that the LUN has to be offline before it can be removed. A physical volume may not be removed if it has allocated extents; all LUNs using a pv must be removed before a physical volume can be removed. We finish by removing the volume group.

It's as simple as that. These few commands do almost everything one would want to do. Read through the command section of this manual to get a better idea of what functions are available.

When Physical Volumes Go Missing

When the makeup of a volume group changes, a generation number is incremented and written out to each physical volume's *config string*. When the VS reboots, it checks these generation numbers to make sure that all the physical volumes are in sync with the group as a whole. If a physical volume is not available when the volume group changes in some way, the failed physical volume's generation number will not change. When the failed physical volume is put back on line, the VS will reject it due to its out-of-date generation number.

To bring the physical volume back into the group, use the *restore* command. This will update the physical volume's generation number, put its extent descriptors back into the pool, and create any LUNs that were on the device. **This can be a very dangerous action and should only be performed if the LUN configuration did not change in any way while the volume was off line. If LUNs were removed and recreated it is possible for extent numbers for a LUN to be duplicated on the old physical volume. Restoring a physical volume to the volume group in this scenario can lead to data corruption.**

Mirroring Physical Volumes

If the applications using the storage require more reliability than provided by the single SR unit, then the SR units can be mirrored to protect both the disk RAIDs and the controllers. The following example shows how to mirror a physical volume using the VS.

```
VS63:163> mkpv 61.0 accounting
2008.07.23 11:50:23 updating 61.0
VS63:163> lsvg accounting
NAME                      LENGTH        AVAILABLE  EXTSZ    PVS
accounting            6001.289GB       6001.277GB    4MB  01/01
VS63:163> mirror 61.0 61.1
2008.07.23 11:50:33 updating 61.0
2008.07.23 11:50:35 started rebuilding: 61.0 -> 61.1
VS63:163>
2008.07.23 11:50:35 mirror rebuild complete: 61.0 -> 61.1
2008.07.23 11:50:35 updating 61.0
VS63:163> lspv
TARGET         VOLUME GROUP        LENGTH     AVAILABLE         NPE  MIRROR
61.0             accounting    6001.289GB    6001.277GB     1430819  61.1
VS63:163>
```

The 6TB 61.0 is now mirrored on target 61.1. The mirror doesn't appear as a volume in the pool; it only shows up as the mirror of the physical volume. In this case, 61.0 is called the primary and 61.1 is referred to as the mirror. LUNs created from extents of a mirrored volume will automatically be synchronously mirrored. The VS ensures synchronization by executing writes to both the primary and the mirror before responding to the AoE initiator. AoE requests for reads are always serviced from the primary.

Mirrors can be added to existing volumes that already have extents allocated and are being used by online LUNs. The metadata keeps track of whether the extents on a PV have been written. Note that on mirror creation in the example above the mirror is brought in sync. As we have not yet allocated any extents from 61.0, there is no work to be done to rebuild the mirror and the rebuild completes immediately. Had we created LUNs and been using them, however, all dirty extents would have been mirrored to 61.1.

If a mirror fails, the VS will just use the primary. If the primary fails the mirror will be relabeled as the primary. In both cases the user will be notified by a log message that the physical volume is now unmirrored.

Mirrors can also be broken on purpose. The *unmirror* command will free the mirror from the primary. The mirror is not added to the volume group, but instead it is just freed from the mirror relation. Its *config string* will be cleared. Removing a mirrored primary from the volume group will cause the physical volume to be automatically unmirrored.

Adding Currently Used Storage to the VS

In some cases the physical volume will already have data stored on it. These units can be added to the VS and used like any other physical storage by using a special allocation command *mklegacy*. This process is special because it simultaneously adds the volume to a volume group and creates a new logical unit with the extents arranged one-to-one with the physical SR. The first extent in the logical unit is the first extent in the physical unit, the second extent in the logical unit is the second extent in the physical unit, and so on. Accessing the logical is the same as if you were accessing the physical.

This, however, means that the metadata for the physical unit or the logical unit cannot be stored on the physical volume. Free extents must be available in the volume group to create a legacy unit. Here is an example of creating a new volume group, adding a pv for metadata storage, and then adding the legacy volume.

```
VS63:163> mkvg existing
VS63:163> mkpv 61.0 existing
2008.07.23 11:58:48 updating 61.0
VS63:163> mklegacy 1 61.1 existing
2008.07.23 11:59:27 updating 61.1
2008.07.23 11:59:27 updating 61.0
VS63:163> lspv
TARGET         VOLUME GROUP        LENGTH     AVAILABLE         NPE  MIRROR
61.0               existing    6001.289GB    6001.260GB     1430819
61.1               existing    6001.289GB       0.000GB     1430819
VS63:163> lslun
LUN           LENGTH  ONLINE  LABEL
1         6001.289GB     OFF
VS63:163>
```

We created a new volume group called *existing*, added a free physical volume 61.0. We then used the *mklegacy* command to simultaneously add 61.1 to the volume group and create a logical unit 1. Note that there are no available extents on 61.1 as they are all, by definition, taken up by the LUN.

There are a couple of unexpected things about storage added to the system in this way. First, one may expect that storage added to the system in a special way will be special forever. It is not. After being created there is nothing special about LUN 1 or the physical volume 61.1. Lun 1, for example, can be shrunk, or grown. It is just an ordinary LUN and an ordinary physical volume. Freed extents will be available for other LUNs.

Legacy physical volumes can also be mirrored. The metadata for LUN 1 and for physical volume 61.1 is in this case on 61.0. To mirror PV 61.1, we need a physical volume that is larger than 61.1 and can hold this metadata at its end. For 4MB extents, this means we need one more extent for every 2.5TB of storage. It should also be noted that the *mklegacy* command marks each extent as written. In the event that the volume is mirrored, every extent will be copied to the mirror.

Lastly, all extents on a PV are defined by the extent size of the volume group the PV is created in. For legacy volumes, this means that up to EXTSZ-1 bytes at the end of the legacy volume may not be exported. It is recommended that users shrink their use of the legacy device by EXTSZ (4MB by default) before executing the mklegacy command to ensure all user data is exported by the legacy LUN.

Updating the Firmware

The VS has a flash boot disk that contains two image areas, A and B. The *setboot* command without arguments will display the default boot image.

```
VS63:163> setboot
will boot from A
VS63:163>
```

In this case we are booting from the first area. To change the default boot image, the *setboot* command is used to specify the area to boot from.

When the system boots it displays a prompt providing the user the chance to change the booting image. **The boot area selected at boot time will not set the default boot area. Only the setboot command sets the default boot area.** If no choice is made in a few seconds, the default image location will be used.

To update the contents of the flash images, flash LUNs are created using the *mklun* command. There is no reason to remove the LUN after update as flash LUNs are not persistent across reboots.

```
VS63:163> mklun 99 flash A
VS63:163> online 99
2008.07.23 12:12:15 putting lun 99 online
VS63:163> lslun
LUN           LENGTH  ONLINE  LABEL
99           0.008GB      ON  flash area A
VS63:163>
```

Now copy the binary from *Coraid* into LUN 99 from an AoE initiator host on your network. For Linux users, the *coraid-update* command should be used. Other UNIX variants can use the *dd* command. The new firmware will be loaded on next boot.

Since there are two areas, it is easy to back-out an update if things go wrong. You can reboot from B if this new code written into location A has some problem. Once the administrator is happy with the code, it should be copied to both areas to avoid future confusion.

Conclusion

This has been a brief overview of the *Coraid* EtherDrive VS network storage virtualizer. The short length of this document bears witness to the simplicity of the system. The manual pages of the individual commands should be studied for a full treatment of what can be done with the VS. It is our sincere hope that we have created a simple yet flexible and powerful appliance that will enable our customers to more effectively use their network storage.