DMF Tuning Notes

Susheel Gokhale
SGI

XVM Volume Tuning

Storage Design Notes

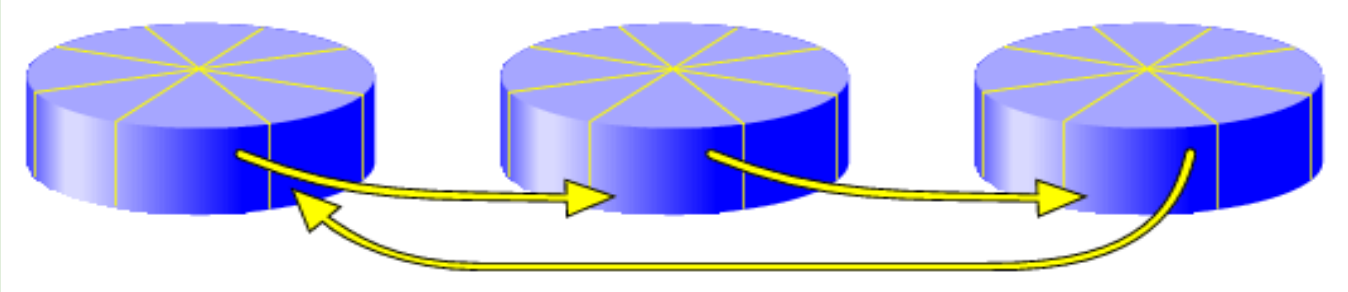
Stripe Alignment

When using RAID LUNs to make XVM volumes and filesystems, it is critical to understand the design of the RAID sub-system and to make sure that the volumes built on top of it are aligned correctly to the RAID Stripe boundary.

Write performance on a mis-aligned volume can be severely affected by forced read-modify-write operations on every write to the disk.

Choosing a 4+1 or 8+1 design for RAID5 will lead to stripe sizes that align naturally with buffer sizes used by the Linux kernel and LSI controller cache which is tuned for 2MB transfers.

RAID sets created with a different number of disks may need careful checking of stripe alignment.

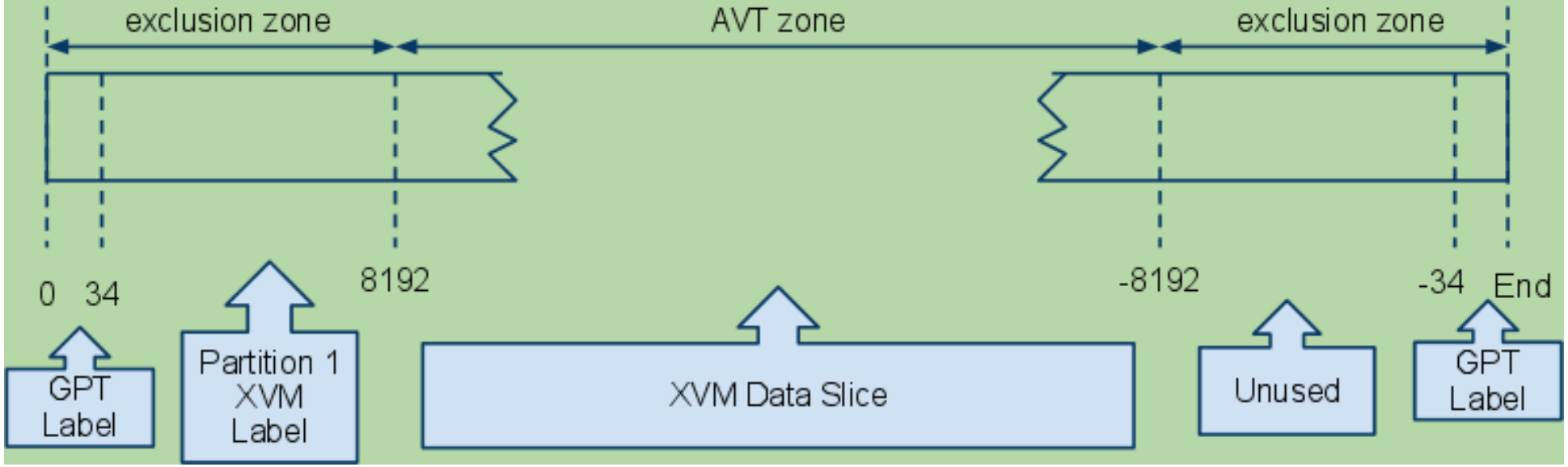


Creating XVM volumes

LUN labeling : GPT and XVM

SGI uses the AVT feature of LSI storage to support transparent multi-path failover. To avoid AVT events during system boot-up and scanning by other hosts on the SAN, LSI reserves a 8MB exclusion zone at the front and end of each LUN device.

We create one partition of size 8MB and use it to keep the XVM label. Rest of the disk can be used for Data.



Creating XVM volumes

Storage Engineering group has created a simple script that automates the process of labeling a disk taking into account the AVT exclusion zone as well as the stripe alignment issue.

```
# xvm-mkgpt.sh  -h

----------------------------------------------------------------
Usage: xvm-mkgpt.sh [-hdgls] [-b stripe ] [-n slice] [name XVM label] <device>

Options:
  -b stripe      The stripe boundery size
  -d             enable debug output
  -h             This help menu
  -g             Only make the GPT label not the XVM
  -l             Work in XVM local domain
  -N             use NEW GPT format.
  -n num_slice   Number of slice(s) to create default=1
  -s             Do not make the XVM slice(s)

The <device> is the device that we want to label
----------------------------------------------------------------
```

Creating XVM volumes

We have an Array with 5+1 RAID5 volumes and 128K (256 block) stripe unit. So we must ensure during the Disk Labeling process that XVM Data area is aligned to a 1280 block stripe boundary.

```
# xvm-mkgpt.sh    -N   -b 1280   spare12   /dev/xscsi/.. /lun12/disc

Going to use NEW XVM label with only 1 Partition
=================================================
Verify that the LUN is unlabled:                    DONE
Trying to access the device:                       DONE

Calculated disk size = 296038400 blocks / 141GB
Will make 1 slice(s) align to 1280 blocks

Zero out data in exclusion zone:                    DONE
Create GPT label:                                 DONE
Create XVM label:                                 DONE
Slice XVM label in 1 slice(s)                       DONE

XVM slice created:
-----------------------------------------------------
phys/spare12            296022016 online,local,accessible
slice/spare12s0         296020480 online,accessible
vol/spare12s0                 0 online,accessible
-----------------------------------------------------
```

Creating XVM volumes

This is what the XVM volume looks like after creating with alignment flags:

```
# xvm show -v phys/spare12

XVM physvol phys/spare12
==========================
size: 296022016 blocks  sectorsize: 512 bytes  state: online,local,accessible
uuid: bacc75a2-1208-4d75-99dd-9789eeba593b
:
Disk has the following XVM label:
:
  secbytes:  512
  label area:  8157 blocks starting at disk block 35 (10 used)
  user area:   296022016 blocks starting at disk block 8192

Physvol Usage:
Start       Length      Name
-----------------------------------------------
0           768         (unused)
768         296020480   slice/spare12s0
296021248   768         (unused)
-------------------------------------------------------------------------
```

New versions of XVM have a '-align' option that is used to ensure that slices created within a volume are always aligned to the stripe boundary

Creating File System

File systems are built on single LUNs or on complex Concat/ Stripe structure in a XVM volume. Stripe alignment has to be correct at the LUN level as well as at the XVM volume level to ensure good performance.

The *mkfs.xfs* command provides 'sunit' and 'swidth' options in the DATA section to address this issue.

When the File system is using a XVM volume built on a STRIPE with proper stripe units specified during creation, the alignment is automatically picked up by mkfs.

```
# xvm show -t -e vol/home
vol/home                    0 online,open,accessible
   subvol/home/data         83850240 online,open,accessible
      stripe/stripe2        83850240 online,tempname,open,accessible (unit size: 1280)
         slice/Home_0s0          41925120 online,open,accessible (Home_0:/dev/xscsi/../lun0/disc)
         slice/Home_1s0          41925120 online,open,accessible (Home_1:/dev/xscsi/../lun1/disc)

# xfs_info /dmf/home
meta-data=/dev/lxvm/home      isize=256    agcount=16, agsize=655200 blks
         =                    sectsz=512   attr=2, nfs4acl=0
data     =                    bsize=4096   blocks=10481280, imaxpct=25
         =                    sunit=160    swidth=320 blks
naming   =version 2           bsize=4096   ascii-ci=0
```

Creating File System

However, if the file system is built on a single disk LUN or a XVM volume built on a single LUN or a CONCAT, the stripe alignment information must be supplied during mkfs using these options.

```
# xvm show -t -e vol/spare
vol/spare                  0 online,open,accessible
   subvol/spare/data       592040960 online,open,accessible
      concat/concat1       592040960 online,tempname,open,accessible
         slice/spare12s0   296020480 online,open,accessible (spare12:/dev/xscsi/../lun12/disc)
         slice/spare13s0   296020480 online,open,accessible (spare13:/dev/xscsi/../lun13/disc)

# xfs_info /dmf/spare
meta-data=/dev/lxvm/spare      isize=256    agcount=16, agsize=4625320 blks
         =                     sectsz=512   attr=2, nfs4acl=0
data     =                     bsize=4096   blocks=74005120, imaxpct=25
         =                     sunit=0      swidth=0 blks
naming   =version 2            bsize=4096   ascii-ci=0
```

The RAID design of these two volumes is identical. But in a CONCAT (or single LUN) based volume the stripe unit information is not present in the XVM volume structure and does not propagate to the mkfs layer.

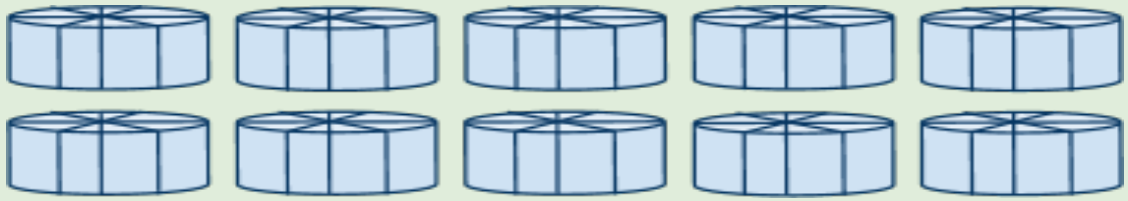
So it must be supplied during creation of the filesystem with explicit options.

```
# mkfs.xfs   -d sunit=1280,swidth=1280    /dev/lxvm/spare
```

File System Design

The XFS file system is divided into multiple AGs or Allocation Groups. The size and number of AGs is calculated based on the size of the file system. Inodes and file data are grouped together within AGs. Conceptually AGs can be thought of as separate disk spindles. XFS is able to parallelize I/O to separate AGs to improve performance.

A new mount option, agskip=N, that tells the XFS allocator to skip over X many AGs when selecting an initial AG for data extent allocations for a new file. This option is particularly useful when the volume layout is a series of concat units - just set N to be the number of AGs in a concat and data allocations will then be load balanced over the spindles.



```
# xfs_info /mnt/data
meta-data = /dev/lxvm/data      isize=256 agcount=64, agsize=114382720 blks
          =                     sectsz=512 attr=0
data      =                     bsize=4096 blocks=7320488320, imaxpct=25
          =                     sunit=160 swidth=160 blks
```

It is a CONCAT with ten disks. So using a mount option of agskip=7 will ensure that every other file is written to the next disk in the concat. This way you can spread the load across all elements of the CONCAT more evenly.

Tape Drive Tuning

Tape Reports

DMF team added a 'tsreport' utility in the toolset some time ago. This utility looks at the sense data gathered by the TS driver for all tape operations. Two standard reports generated :

Daily Drive report : Reports on Cleaning flags raised by any drives in the last 24 Hours
Daily TS report : Reports all drive and volume related activity e.g. Reads/ Writes/ Errors/ Corrections/ Loads/ Unloads etc in the last 24 Hours.

These reports provide a good monitoring tool on the health and behavior of the Tape sub-system.

ibmlto drive LTO-21 indicated it requests or requires cleaning at 2010/12/30 21:52:31
ibmlto drive LTO-12 indicated it requests or requires cleaning at 2010/12/31 04:12:11

Drive stats are presented with various performance reports :

Serial	Type	read	rcorr	written	wcorr	r+w/GB	unloads	loads	CL
10110031EE	ibmlto	169 GB	6	134 GB	253	0.85	15	15	1
10120031EE	ibmlto	59 GB	2	10 GB		0.03	16	17	
10130031EE	ibmlto	298 GB		190 GB	1	0.00	27	26	
10210031EE	ibmlto	438 GB	416	44 GB	1	0.86	24	24	7
10220031EE	ibmlto	140 GB		140 GB	1	0.00	28	29	

Tape Reports

Tape Cleaning

A heads-up on SpectraLogic Library new feature :

Recent versions of BlueScale firmware have added a Auto_clean feature to the library function. A separate Cleaning partition is created in the Library to store cleaning media and this is used to auto_clean any drive that signals Cleaning Required.

A special barcode on a "Maintenance Terapack" entitles you to get the Auto_clean feature license from SpectraLogic.

The TS reports show that the Library now cleans drives automatically.

ibmlto drive LTO-11 at 2011/02/17 08:09:36: Clean Now
ibmlto drive LTO-11 cleared its indication that it requested or required cleaning at 2011/02/17 08:28:22

Tape Mounting Cycles

Sometimes you see multiple load/unloads on the same tape within a short span of time. This is commonly due to an application that is recalling files and processing them sequentially. Each recall potentially leading to a tape load/unload operation.

As a DMF Admin you would advise Users to plan ahead and stage batch recall of the files they need for their next job run. In practice that is not always possible.

Indications of such behavior may be present in the Daily TS report.
Let us look at an example of this behavior in a small time window on a DMF server

```
14:17:15:386-V server    73871-dmatrc process_mount_tape: Mount tape vsn=DMF005, tf_version=4
14:19:46:247-V server    74046-dmatrc process_mount_tape: Mount tape vsn=DMF005, tf_version=4
14:22:35:260-V server    74761-dmatrc process_mount_tape: Mount tape vsn=DMF005, tf_version=4
14:25:38:692-V server    74894-dmatrc process_mount_tape: Mount tape vsn=DMF005, tf_version=4
14:27:41:399-V server    75525-dmatrc process_mount_tape: Mount tape vsn=DMF005, tf_version=4
14:31:20:230-V server    76020-dmatrc process_mount_tape: Mount tape vsn=DMF005, tf_version=4
14:34:13:585-V server    76411-dmatrc process_mount_tape: Mount tape vsn=DMF005, tf_version=4
```

There were 50 files recalled from this tape during the 17 minute window. All files were within a single Zone and close to each other in sequence. If they had been recalled in a batch mode, it would have taken the server one tape load/unload cycle and a few seconds of positioning/ reading time.

There are a few options available to the DMF Admin to help in this situation.

Tape Mounting Cycles

DMF allows a 2 second wait after finishing a read before it asks the Tape to rewind. After a rewind it allows a 5 second wait before it asks the read child to exit and un-mount the tape

Let us look at how the requests were getting handled. A snippet of the moverlog file reveals this sequence of how the recalls progress within one of the mount requests

```
14:31:20:230-V server   76020-dmatrc process_mount_tape: Mount tape vsn=DMF005, tf_version=4
 ---  mount 22 seconds ---
14:31:42:402-V server   76020-dmatrc process_mount_tape: tape available: vsn=DMF005
14:31:42:408-V server   76020-dmatrc queue_chunk: Req=26941334,, zn=246, cn=7554 Added
 --- position and read  4 seconds ---
14:31:46:558-V server   76020-dmatrc send_chunk_done: done: Req=26941334, zn=246, cn=7554, bytes=132608
14:31:48:005-V server   76020-dmatrc tape_rewind: to boi
14:31:48:537-V server   76020-dmatrc queue_chunk: Req=26941335,, zn=246, cn=7559 Added
 --- re-position and read  11 seconds ---
14:31:59:778-V server   76020-dmatrc send_chunk_done: done: Req=26941335,, zn=246, cn=7559, bytes=82836
14:32:00:789-V server   76020-dmatrc queue_chunk: Req=26941336,, zn=246, cn=7567 Added
14:32:00:825-V server   76020-dmatrc send_chunk_done: done: Req=26941336,, zn=246, cn=7567, bytes=302080
14:32:02:014-V server   76020-dmatrc tape_rewind: to boi
14:32:03:161-V server   76020-dmatrc queue_chunk: Req=26941337,, zn=246, cn=7507 Added
 --- re-position and read  9 seconds ---
14:32:12:828-V server   76020-dmatrc send_chunk_done: done: Req=26941337,, zn=246, cn=7507, bytes=120114
14:32:14:007-V server   76020-dmatrc tape_rewind: to boi
14:32:14:901-V server   76020-dmatrc queue_chunk: Req=26941338,, zn=246, cn=7514 Added
 --- re-position and read  11 seconds ---
14:32:25:446-V server   76020-dmatrc send_chunk_done: done: Req=26941338,, zn=246, cn=7514, bytes=86863
14:32:27:002-V server   76020-dmatrc tape_rewind: to boi
 --- rewind and start unmount  7 seconds ---
14:32:34:412-V server   76020-dmatrc tape_unmount: unmounting DMF005 ....
```

Tape Mounting Cycles

If we could ask the 'dmatrc' process to wait a little longer after finishing a read before rewinding, and before un-mounting the tape after re-winding, it would save the server a lot of extra work in re-positioning the tape and possibly avoid a few load/unload cycles over the completion of the current sequence of recalls.

DMF provides two config flags that can help smooth out these kinds of mount storms.

REWIND_DELAY
Specifies the number of seconds an idle tape LS read child (dmatrc) can wait before rewinding. Default is 2 seconds

READ_IDLE_DELAY
Specifies the number of seconds an idle tape LS read child (dmatrc) can wait before being told to exit. Default is 5 seconds

Looking at the behavior in the previous slides we could benefit by increasing the REWIND_DELAY to 10 seconds and the READ_IDLE_DELAY to 30 seconds. It would save a lot of wear and tear on the Tape drives and the Tape Media as well.

Note that these setting are non-binding. If other requests are waiting for a tape drive then the server can intervene and ask the read child process to exit without waiting.