Implementation of Hardware RAID and LVM-based Large Volume Storage on Global Data Center System of International GNSS Service

Daekyu Lee*, Sungki Cho*, Jonguk Park* and Pilho Park*

* Korea Astronomy and Space Science Institute, Daejeon, Korea
(Tel : +82-42-865-3245; E-mail: picchi@kasi.re.kr)

Abstract: High performance and reliability of the storage system to handle a very large amount of data has been become very important. Many techniques have been applied on the various application systems to establish very large capacity storage that satisfy the requirement of high I/O speed and physical or logical failure protection. We applied RAID and LVM to construct a storage system for the global data center which needs a very reliable large capacity storage system. The storage system is successfully established and equipped on the latest Linux application server.

Keywords: : RAID, LVM, Linux, Large Capacity storage

1. INTRODUCTION

Recently, Korea Astronomy and Space Science Institute (KASI) has been developed and launched IGS(International GNSS Service) Global Data Center(GDC). IGS organizes the network of more than 380 international GPS stations and their data. IGS GDC receives and archives all the international GPS data and IGS products from the early 90's. GDC opens its data server for downloading all the archived IGS data for the international engineering and scientific user communities. Currently, the amount of data in GDC is more than 3 terabytes and GDC archives data everyday continuously.

To handle and store the large amount of data, a stable large capacity storage system and an appropriate control software system are required. The reliability and flexibility of the storage system can be enhanced by RAID(Redundant Array of Independent Disks) and LVM(Logical Volume Manager). The large capacity storage system can be categorized into the three systems: DAS(Direct Attached Storage), NAS(Network Attached Storage) and SAN(Storage Area Network). In DAS, storages are directly connected into a single server. Networked workstations or clients access to DAS only through the server attached on storage. DAS can be controlled by the network operating system of the server attached on DAS. DAS does not provide remote connectivity or common storage for various platforms. NAS is a dedicated shared storage solution that attaches to a network topology, becoming immediately and transparently available as a network resource for all clients. NAS is platform- and operating system independent. A NAS device is typically a stand alone and high-performance single-purpose system. The advantage of NAS over DAS is performance and connectivity. For the case of adding storage, NAS solution is simpler and less expensive, while it is dependent on network bandwidth and single point of failure. A Storage Area Network (SAN) is a network for storage subsystems connected to one or more servers. SAN connectivity is accomplished by using a high-speed protocol such as Fibre Channel or iSCSI(Internet SCSI). SAN is independent on the Local Area Network (LAN). SAN is a flexible and scaleable storage system. Also, SAN supports heterogeneous server access and storage mirroring in a remote location to ensure data integrity in case of disaster. However, the cost of SAN is very high [2].

As stated above, there are various methods to improve a reliability and efficiency of a storage system. Also, the many researches and developments are on going to improve a performance of the storage system. It is noted that the high performance of a storage system can be achieved only when the appropriate methods are applied to the system. We selected DAS for GDC storage system that equipped Redhat Enterprise Linux ES4(released in February, 2005) as an operating system(OS). Many storage systems have been established by using the various techniques. However, it is very hard to find the case that based on the latest version of server hardware and OS.

In this paper, we present the implementation of RAID and LVM for the high performance storage system with the latest server and OS.

2. RAID

RAID stands for "Redundant Array of Inexpensive Disks", and is meant to be a way of creating a fast and reliable disk-drive subsystem out of individual disks. The basic idea of RAID is to combine multiple small and independent disk drives into an array of disk drives which yields performance exceeding that of a Single Large Expensive Drive (SLED). Additionally, this array of drives appears to the computer as a single logical storage unit or drive [1-2].

The Mean Time Between Failure (MTBF) of the array will be equal to the MTBF of an individual drive, divided by the number of drives in the array. Because of this, the MTBF of an array of drives would be too low for many application requirements. However, disk arrays can be made fault-tolerant by redundantly storing information in various ways [3-4].

Fundamental to RAID is "striping", a method of concatenating multiple drives into one logical storage unit. Striping involves partitioning each drive's storage space into stripes which may be as small as one sector (512 bytes) or as large as several megabytes. These stripes are then interleaved round-robin, so that the combined space is composed alternately of stripes from each drive.

In data intensive environments and single-user systems which access large records, small stripes (typically one 512-byte sector in length) can be used so that each record will span across all the drives in the array, each drive storing part of the data from the record. This causes long record accesses to be performed faster, since the data transfer occurs in parallel on multiple drives [3].

There are a variety of different types and implementations of RAID, each with its own advantages and disadvantages. For example, there are levels such as 0, 1, 2, 3, 4, 5, 6, 7, 10, 50, 0+1 and others. Most, but not all levels of RAID offer redundancy against disk failure. Of those that offer redundancy, RAID 1 and RAID 5 are the most popular. RAID-1 offers better performance, while RAID 5 provides for more efficient use of the available storage space. Another level, especially RAID level 0 is often combined with RAID level 1(RAID 0+1). Of those that offer non-redundancy, RAID-0 is the most popular. Here we explain for four case levels with pictures. Other things omit. Also RAID is classified into two large groups by hardware RAID and software it [3-5].

2.1 RAID Levels

(1) RAID 0

RAID Level 0 is not redundant, hence does not truly fit the "RAID" acronym. In level 0, data is split across drives, resulting in higher data throughput. Since no redundant information is stored, performance is very good, but the failure of any disk in the array results in data loss. This level is commonly referred to as striping, as shown in Fig. 1 [3-6].

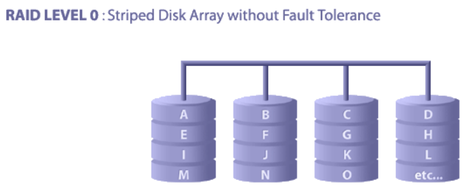

Fig. 1 RAID 0

(2) RAID 1

RAID Level 1 provides redundancy by writing all data to two or more drives. The performance of a level 1 array tends to be faster on reads and slower on writes compared to a single drive, but if either drive fails, no data is lost. This is a good entry-level redundant system, since only two drives are required; however, since one drive is used to store a duplicate of the data, the cost per megabyte is high. This level is commonly referred to as mirroring, as shown in Fig. 2 [3-6].

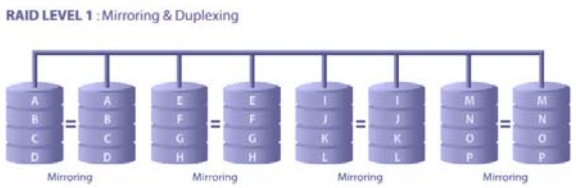

Fig. 2 RAID-1

(3) RAID 5

RAID Level 5 is similar to level 4, but distributes parity among the drives. This can speed small writes in multiprocessing systems, since the parity disk does not become a bottleneck. Because parity data must be skipped on each drive during reads, however, the performance for reads tends to be considerably lower than a level 4 array. RAID 5's principle advantage over mirroring is that it offers redundancy and protection against single-drive failure, while offering far more storage capacity when used with three or more drives. The cost per megabyte is the same as for level 4. This level is shown in Fig. 3 [3-6].

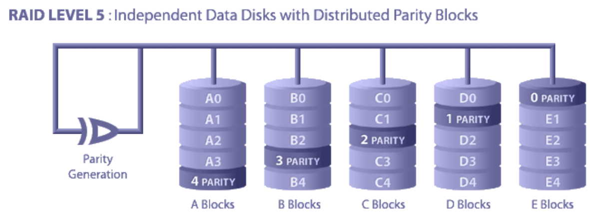

Fig 3 RAID 5

(4) RAID 0+1

As shown in Fig. 4, RAID 0+1 is implemented as a mirrored array whose segments are RAID 0 arrays. RAID 0+1 has the same fault tolerance as RAID level 5 and has the same overhead for fault-tolerance as mirroring alone. High I/O rates are achieved thanks to multiple stripe segments. Excellent solution for sites that need high performance but are not concerned with achieving maximum reliability. But RAID 0+1 is not to be confused with RAID 10. A single drive failure will cause the whole array to become, in essence, a RAID Level 0 array. This level is very expensive and high overhead. All drives must move in parallel to proper track lowering sustained performance. And level 0+1 has very limited scalability at a very high inherent cost [5].

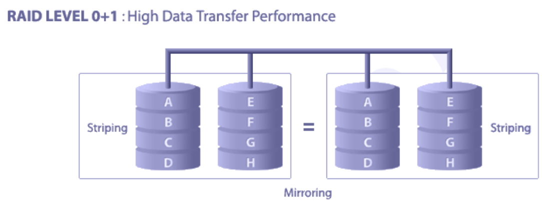

Fig. 4 RAID 0+1

2.2 Hardware RAID vs. Software RAID

There are two possible RAID approaches: Hardware RAID and Software RAID. The hardware-based system manages the RAID subsystem independently from the host and presents to the host only a single disk per RAID array. Software RAID implements the various RAID levels in the kernel disk (block device) code. It offers the cheapest possible solution [9].

Just like any other application, software-based arrays occupy host system memory, consume CPU cycles and are operating system dependent. By contending with other applications that are running concurrently for host CPU cycles and memory, software-based arrays degrade overall server performance. Also, unlike hardware-based arrays, the performance of a software-based array is directly dependent on server CPU performance and load.

Except for the array functionality, hardware-based RAID schemes have very little in common with software-based implementations. Since the host CPU can execute user applications while the array adapter's processor simultaneously executes the array functions, the result is true hardware multi-tasking. Hardware arrays also do not occupy any host system memory, nor are they operating system dependent.

Hardware arrays are also highly fault tolerant. Since the array logic is based in hardware, software is not required to boot. Some software arrays, however, will fail to boot if the boot drive in the array fails. For example, an array implemented in software can only be functional when the array software has been read from the disks and is memory-resident. What happens if the server can't load the array software because the disk that contains the fault tolerant software has failed? Software-based implementations commonly require a separate boot drive, which is not included in the array [3, 6].

3. LVM

LVM is a Logical Volume Manager for the Linux operating system as shown in Fig. 5. Logical volume management provides a higher-level view of the disk storage on a computer system than the traditional view of disks and partitions. This gives the system administrator much more flexibility in allocating storage to applications and users. Storage volumes created under the control of the logical volume manager can be resized and moved around almost at will, although this may need some upgrading of file system tools. The logical volume manager also allows management of storage volumes in user-defined groups, allowing the system administrator to deal with sensibly named volume groups such as “development” and “sales” rather than physical disk names such as “sda” and “sdb”. According to existing research, it can take advantage of actively LVM because degradation is little even if use LVM [8].

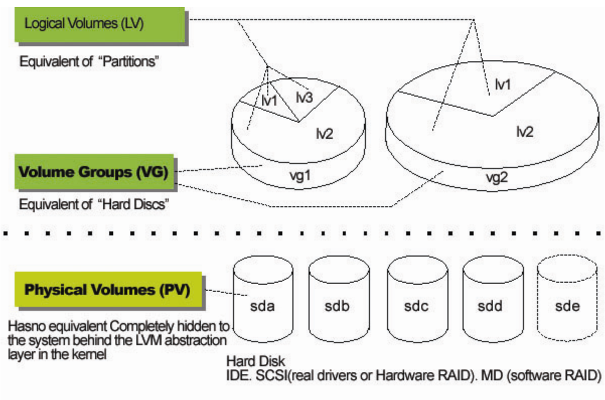

Fig. 5 Logical Volume Manager

There are now two version of LVM for Linux:
LVM 2 is the latest and greatest version of LVM for Linux. LVM 2 is almost completely backward compatible with volumes created with LVM 1. The exception to this is snapshots. Users must remove snapshot volumes before upgrading to LVM 2. LVM 2 uses the device mapper kernel driver. Device mapper support is in the 2.6 kernel tree and there are patches available for current 2.4 kernels. LVM 1 is a mature product that has been considered stable for a couple of years. The kernel driver for LVM 1 is included in the 2.4 series kernels, but this does not mean that all 2.4.x kernel is up to date with the latest version of LVM.

The Volume Group(VG) is the highest level abstraction used within the LVM. It gathers together a collection of Logical Volumes and Physical Volumes into one administrative unit. A Physical Volume(PV) generally refers to the hard disk partitions or a device that looks (logically) similar to a hard disk partition such as a RAID device. One or many physical volumes make up a Logical Volume(LV) . In LVM, a logical volume is similar to a hard disk partition in non-LVM systems. The logical volume can contain a file-system e.g. /home or /usr. Each physical volume is divided chunks of data, known as physical extents(PE), these extents have the same size as the logical extents for the volume group. Each logical volume is split into chunks of data, known as logical extents(LE). The extent size is the same for all logical volumes in the volume group [4, 7-8].

4. STORAGE SYSTEM COMBINED HAREWARE RAID 5 AND LVM 2

4.1 Hardware and Software Spec. of Storage System

Hardware used in implementation of the system combined hardware RAID 5 and LVM 2 is as follows. There are server and storage device. The server model is PowerEdge 2850 by DELL Inc.. Its specification is as shown in Table 1.

Table 1 Spec. of server used in implementation.

Device Name	Specification
Processor - 2×Intel Xeon™	3.4GHz/1M, EM64T, 800MHz FSB
SCSI Controller - LSI53C1030	Dual channel U320 SCSI
Internal RAID Card	PERC4e/Di with 256MB cache
NIC - Intel 82541	Dual embedded Gigabit NICs
Memory	2GB

The storage model is PowerVault 220S by DELL Inc. as shown in Table 2. It has applied the RAID 1.

Table 2 Spec. of storage used in implementation.

Device Name	Specification
RAID controller card	Leverages PERC 4 Dual channel and Quad Channel
Drive Bays	Up to 14 1” LVD Ultra U320 SCSI
HDD	(146GB×14) ×4=8TB

PowerVault 220S SCSI enclosures can be attached to one server in a 14-drive joined-bus configuration.

Software used in implementation of this system is as shown in Table 3. Here, LVM2 has been included on the operating system. The storage has applied hardware RAID 5.

Table 3 Software used in implementation

Software	Version
Operating System	Redhat Enterprise Linux ES4
Linux Kernel	2.6.9
LVM2	1.0.8
OS File System	ext3
Storage File System	xfs

4.2 Implementation of Storage System

The Volume Group Descriptor Area (VGDA) functions similar to the partition table for LVM. It is stored at the beginning of each physical volume. The VGDA consists of the following information : one PV descriptor, one VG descriptor, the LV descriptors and several PE descriptors.

When the system boots, the LVs and the VGs are activated and the VGDA is loaded into memory. The VGDA helps to identify where the LVs are actually stored. When the system wants to access the storage device, the mapping mechanism (constructed with the help of VGDA) is used to access the actual physical location to perform I/O operation [7].

Before installing LVM, there are some prerequisites : kernel should have compiled and the LVM module configured because of a kernel limitation of capacity per block device. How to apply with LVM is as follows.

(1) Configure the kernel

This can be done as follows :

```
# cd /usr/src/kernels/2.6.9
# make menuconfig
```

under the submenu :
Device drivers→Block device→

enable the following a option :
[*] Support for Large Block Devices

where * denotes the selection sign.
And under the submenu :
Device drivers→SCSI device support→SCSI low-level drivers→

enable the following three options :
[*] LSI Logic New Generation RAID Device Drivers
<M> LSI Logic Management Module (New Driver)
<M> LSI Logic MegaRAID Driver (New Driver)

where M represents the module sign.
Also under the submenu :
File systems→

enable the following a option :
<M> XFS filesystem support

(2) Check the mount of disk space free on your drive

```
# df -h
Filesystem    Size   Used Avail Use% Mounted on
/dev/sda5     131G   8.9G  115G   8% /
/dev/sda1      99M    20M   75M  21% /boot
/dev/sdb1     683G      0  683G   0% /gdcdata1
/dev/sdc1     683G      0  683G   0% /gdcdata2
…
/dev/sdi1     683G      0  683G   0% /gdcdata8
```

(3) Change LVM partition type on your hard disk

Use fdisk or any other partition utility to change the LVM partition type. The partition type of linux LVM is 8e.

```
# fdisk /dev/sdb
  press p (to print the partition table)
  and   t (to change partiton's system id)
```

After the change of the Linux LVM partition type. Print the partition table. It will look something like this :

```
Device Boot   Start    End      Blocks    Id   System
/dev/sdb1 *      1    89173   716282091   8e   Linux LVM
```

Others is a similar to above.

(4) Create physical volumes

```
# pvcreate /dev/sdb1
 pvcreate -- -physical volume "/dev/sdb1" successfully created
…
# pvcreate /dev/sdi1
 pvcreate -- -physical volume "/dev/sdi1" successfully created
```

The above command creates a volume group descriptor at the start of the partition.

(5) Create volume groups

Create a new volume group and add the two physical volumes to it in the following way.

```
# vgcreate VolGroup00 /dev/sdb1 … /dev/sdi1
vgcreate- -- INFO: using default physical extent size 4 MB
vgcreate- -- INFO: maximum logical volume size is 255.9Gigabyte
vgcreate- -- doing automatic backup of volume group "VolGroup00"
vgcreate- -- volume group "VolGroup00" successfully created and
             activated
```

This will create a volume group named VolGroup containing the physical volumes from /dev/sdb1 to /dev/sdi1. We can also specify the extent size with this command if the extent size of 4MB is not suitable for our purpose.
Activate the volume groups using the command

```
# vgchange -ay VolGroup00
```

The command "vgdisplay" is used to see the details regarding the volume groups created on system.

```
# vgdisplay
```

(6) Create logical volumes

The lvcreate command is used to create logical volumes in volume groups.

```
# lvcreate –L 5.33T –n LogVol00 VolGroup00
```

(7) Create a file system

Now you need to build a filesystem on this logical volume. We have chosen to make the xfs journalling filesystem on the logical volume.

```
# mkfs.xfs    /dev/VolGroup00/LogVol00
```

Mount the newly created filesystem using the mount command.

```
# mount -t xfs /dev/VolGroup00/LogVol00 /gdcdata
```

(8) Add entries to /etc/fstab

Add the following entry to /etc/fstab so that the filesystem is mounted at boot.

```
/dev/VolGroup00/LogVol00 /gdcdata xfs defaults 1 2
```

Copy the recompiled kernel if you have not replaced your original kernel with it yet so u have the option of using LVM or not using it.

(9) Verify VG and LV

```
# vgdisplay
--- Volume group ---
VG Name                 VolGroup00
Format                  lvm2
Metadata Areas          8
Metadata Sequence No    2
VG Access               read/write
VG Status               resizable
…
VG Size                 5.34 TB
PE Size                 4.00 MB
Total PE                1398968
Alloc PE / Size         1397228 / 5.33 TB
Free   PE / Size        1740 / 6.80 GB

# lvdisplay
```

```
--- Logical volume ---
LV Name                /dev/VolGroup00/LogVol00
VG Name                VolGroup00
…
LV Size                5.33 TB
Current LE             1397228
…
```

As we can see from the above discussion, LVM is quite extensible and straightforward to use. After the volume groups have been set up, it is easy to resize logical volumes as per requirements. As in steps above, total size of LV is 5.33TB. Total size was reduced because of RAID 5.

5. CONCLUSION

We composed storage system of vast capacity by using hardware RAID5 and LVM with the latest Linux kernel. The storage system is currently used for the data server of IGS Global Data Center in KASI. The presented storage system in this paper was composed by using DAS. More flexibility of storage size and protection may be achieved by NAS and SAN system.

ACKNOWLEDGMENTS

This work was supported by the Korea Research Council of Fundamental Science & Technology.

REFERENCES

[1] David A Patterson, Garth Gibson, Randy H Katz, "A Case for Redundant Arrays of Inexpensive Disks(RAID)," *Proc. of the ACM SIGMOD international conference on Management of data*, pp. 109~116, 1998.

[2] Technical Support, SMS Data Products Group. Inc., http://www.sms.com/data.htm.

[3] http://www.staff.uni-mainz.de/neuffer/scsi/what_is_raid.html.

[4] The Linux Documnetation Project, http://www.tldp.org/HOWTO/Software-RAID-HOWTO.html.

[5] Advanced Computer & Network Corporation, "Get to Know RAID," http://www.acnc.com/html.

[6] AJ Lewis, "LVM HOWTO," http://www.tldp.org/HOWTO/LVM-HOWTO/, Red hat, Inc., 2004.

[7] The Linux Gazette, http://www.linuxgazette.com/issue84/vinayak.html.

[8] Michael Hasenstein, "WHITEPAPER The Logical Volume Manager (LVM)," SuSE, Inc., 2001.

[9] Red Hat Linux Customization Guide, Red Hat, Inc., http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/custom-guide/index.html, 2003.