RAID 5 Installations on Linux and Creating File System

Puri Ganesh D.
Assistant Professor
AVCOE Sangamner.
A'Nagar Maharashtra

Puri Dinesh D.
Assistant Professor
SSBTCOE Bambhori
Jalgaon Maharashtra

Wackchaure Manoj A.
Assistant Professor
AVCOE Sangamner.
A'Nagar Maharashtra

ABSTRACT
Redundant Array of Independent (originally Inexpensive) Disks or RAID can be set up using hardware or software. Hardware RAID is more expensive, but offers better performance. Software RAID is cheaper and easier to manage, but it uses your CPU and your memory. Where ten years ago nobody was arguing about the best choice being hardware RAID, this has changed since technologies like mdadm, lvm and even zfs focus more on manageability. The workload on the CPU for software RAID used to be high, but CPU's have gotten a lot faster. [1]

1. INTRODUCTION
RAID is basically for two things. First is to increase the performance by striping data across multiple drives, thus it can allow multiple drives to supply or take data stream simultaneously. Second it can replicate data across multiple drives to supply so it reduces the risk associated with single failed disk. It assumes two things. First one is mirroring in which data blocks are reproduced bit for bit on several different drives, and parity schemes in which one or more drives contain an error-correcting check-sum of the blocks on the remaining data drives.

2. SOFTWARE RAID 5
Linux requires you to create one or more partitions. A partition's geometry and size is usually defined by a starting and ending cylinder(sometimes by sector). Partitions can be of type primary (maximum four), extended(maximum one) or logical (contained within the extended partition). Each partition has a type field that contains a code. This determines the computers operating system or the partitions file system.[2]

Table 2.1 Partition's type and naming

Partition Type	naming
Primary (max 4)	1-4
Extended (max 1)	1-4
Logical	5

The hard disk devices are named /dev/hdx or /dev/sdx with x depending on the hardware configuration. Next is the partition number, starting the count at 1. Hence the four (possible) primary partitions are numbered 1 to 4. Logicalpartition counting always starts at 5. Thus /dev/hda2 is the second partition on the first ATA hard disk device, and /dev/hdb5 is the first logical partition on the second ATA hard disk device. Same for SCSI, /dev/sdb3 is the third partition on the second SCSI disk.

Table 2.2 Partition's name and device

Partition	device
/dev/hda1	first primary partition on /dev/hda
/dev/hda2	second primary or extended partition on /dev/hda
/dev/sda5	first logical drive on /dev/sda
/dev/sdb6	second logical drive on /dev/sdb

For software RAID we require physical partitions with equal size. For ex. We take sdb disk which is partitioned into sdb1,sdb2,sdb3 of equal size. we create a partition with fdisk on /dev/sdb.[3] First we start the fdisk tool with dev/sdb as argument. While creating partitions care should be taken not to partition the wrong disk. Device contains neither a valid DOS partition table, nor Sun, SGI. Building a new DOS disk label. Changes will remain in memory only, until you decide to write them. After that, of course, the previous content won't be recoverable. Inside the fdisk tool, we can issue the p command to see the current disks partition. After restoring a master boot record with dd, we need to force the kernel to reread the partition table with partprobe. After running partprobe, the partitions can be used again.

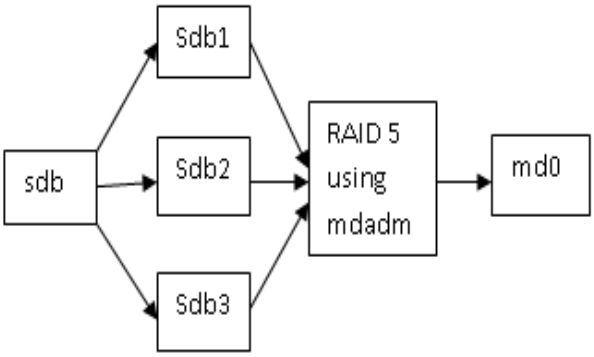

Fig 1: Software RAID-5 using mdadm

In the above fig.1 software RAID-5 is shown which is basically for mirroring as well as performance. In this three physical drives of single SCSI disk are used. Using fdisk tool in linux sdb is partitioned into physical parts. Using mdadm tool RAID-5 level is created. The array of physical disks is created as md0. In software RAID existing disk and CPU is used. It can be extended for next levels of RAID.[4]

3. INSTALLATION OF RAID 5
For Hardware RAID 5 we require to disks of specific configuration for ex. three hard disks of 500 GB size each. It is expensive to get such disks. For experimental purpose we will take USB disks of small size and extend the experiment

for large size. In this installation two USB disks of size 2.4GB are taken for experiment.[5]

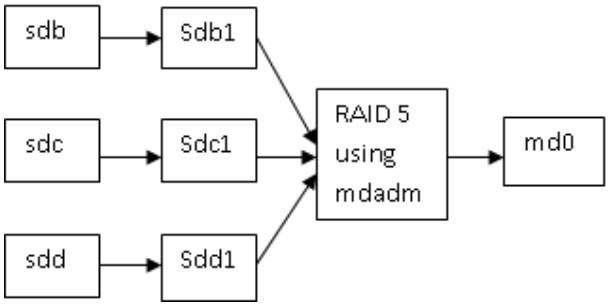

Fig 2: Procedure for RAID-5 using mdadm

In above fig 2 three disks are partitioned. First partition of each disk is of size 2.4 GB. Steps for partitioning are given below. Results are obtained on fedora 18 os and core 2 Duo 64 bit machine.

3.1 Steps for Installation

3.1.1 Using fdisk for partition

[root@CGLAB ~]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.22.1).
Changes will remain in memory only, until you decide to write them.[6]
Be careful before using the write command.
Command (m for help): n
Partition type:
p primary (1 primary, 0 extended, 3 free)
e extended
Select (default p): p
Partition number (1-4, default 2): 1
Partition 1 is already defined. Delete it before re-adding it.
Command (m for help): d
Selected partition 1
Partition 1 is deleted
Command (m for help): n
Partition type:
p primary (0 primary, 0 extended, 4 free)
e extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-15820799, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-15820799, default 15820799): 5000000
Partition 1 of type Linux and of size 2.4 GiB is set.
In the same way partition for second disk sdc can be created of size 2.4 GB.

3.1.2 Change Partition type fd

The next step is to create a partition of type fd on every disk. The fd type is to set the partition as Linux RAID auto detect.
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): fd
Changed system type of partition 1 to fd (Linux raid autodetect)
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.

3.1.3 verify all partitions

Now all disks are ready for raid 5, so we have to tell the system what to do with these disks.
[root@CGLAB ~]# fdisk -l 2> /dev/null | grep raid
/dev/sdb1 2048 5000000 2498976+ fd Linux raid autodetect
/dev/sdc1 2048 5000000 2498976+ fd Linux raid autodetect

3.1.4 Create the RAID 5

Issue the command mdadm with the correct parameters.
[root@CGLAB ~]# mdadm --create /dev/md0 --chunk=64 --level=5 --raid-devices=2 /dev/sdb1 /dev/sdc1
mdadm: /dev/sdc1 appears to be part of a raid array:
level=raid0 devices=0 ctime=Thu Jan 1 05:30:00 1970
mdadm: partition table exists on /dev/sdc1 but will be lost or meaningless after creating array
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.

3.1.5 How fdisk -l sees RAID 5

[root@CGLAB ~]# fdisk -l /dev/md0
Disk /dev/md0: 2556 MB, 2556821504 bytes, 4993792 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 65536 bytes / 65536 bytes.

3.1.6 Check Entry in mdstat

[root@CGLAB ~]# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdc1[2] sdb1[0]
2496896 blocks super 1.2 level 5, 64k chunk, algorithm 2 [2/1] [U_]
[==============>.....] recovery = 74.1% (1851300/2496896) finish=2.1min speed=5056K/sec
unused devices: <none>

3.1.7 Detail information on active RAID 5 Device

[root@CGLAB ~]# mdadm --detail /dev/md0
/dev/md0:
Version : 1.2
Creation Time : Wed Nov 27 15:19:40 2013
Raid Level : raid5
Array Size : 2496896 (2.38 GiB 2.56 GB)
Used Dev Size : 2496896 (2.38 GiB 2.56 GB)
Raid Devices : 2
Total Devices : 2
Persistence : Superblock is persistent
Update Time : Wed Nov 27 15:28:01 2013
State : clean
Active Devices : 2
Working Devices : 2
Failed Devices : 0
Spare Devices : 0
Layout : left-symmetric
Chunk Size : 64K
Name : CGLAB:0 (local to host CGLAB)
UUID : 572e8de9:f27814e0:8800be41:1061c424
Events : 18

Number	Major	Minor	RaidDevice	State	
0	8	17	0	active sync	/dev/sdb1
2	8	33	1	active sync	/dev/sdc1

4. CREATING FILE SYSTEM ON RAID5

4.1 Common file systems

4.1.1 ext2 and ext3

Once the most common Linux file systems is the ext2 (the second extended) file system. A disadvantage is that file system checks on ext2 can take a long time. So ext2 is being replaced by ext3 on most Linux machines. They are essentially the same, except for the journaling which is only present in ext3. Journaling means that changes are first written to a journal on the disk. The journal is flushed regularly, writing the changes in the file system. Journaling keeps the file system in a consistent state, so we don't need a file system check after an unclean shutdown or power failure.

we can create these file systems with the /sbin/mkfs or /sbin/mke2fs commands. Use mke2fs -j to create an ext3 file system. We can convert an ext2 to ext3 with tune2fs -j. We can mount an ext3 file system as ext2, but then we lose the journaling. So it need to run mkinitrd if we are booting from this device.[1][2]

4.1.2 ext4

Since 2008 the newest incarnation of the ext file system is ext4 is available in the Linux kernel. ext4 support larger files (up to 16 terabyte) and larger file systems than ext3 (and many more features).[7]

4.1.3 Vfat

The vfat file system exists in a couple of forms : fat12 for floppy disks, fat16 on ms-dos, and fat32 for larger disks. The Linux vfat implementation supports all of these, but vfat lacks a lot of features like security and links. The fat disks can be read by every operating system, and are used a lot for digital cameras, USB sticks and to exchange data between different OS on a home user's computer.[7]

4.2 Creating ext4 file system on RAID 5

Now we put ext4 file system on /dev/md0 which is created for RAID 5

```
[root@CGLAB ~]# mkfs.ext4 /dev/md0
mke2fs 1.42.5 (29-Jul-2012)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=16 blocks, Stripe width=16 blocks
156160 inodes, 624224 blocks
31211 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=641728512
20 block groups
32768 blocks per group, 32768 fragments per group
7808 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912
Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

4.3 Mounting file system on mount point

We will mount the filesystem on mount point /root/test where contents of RAID drive are visible. Following fig. 3 explain the steps.[5]

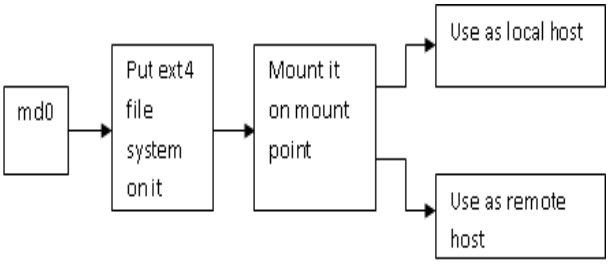

Fig 3: Creating file system and mounting RAID 5

```
[root@CGLAB ~]# mount /dev/md0 /root/test
[root@CGLAB ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
rootfs                 50G   12G   36G  25% /
devtmpfs              927M     0  927M   0% /dev
tmpfs                 938M  184K  938M   1% /dev/shm
tmpfs                 938M  2.7M  935M   1% /run
tmpfs                 938M     0  938M   0% /sys/fs/cgroup
/dev/mapper/fedora-root   50G   12G   36G  25% /
tmpfs                 938M  792K  937M   1% /tmp
/dev/sda2             485M   51M  410M  11% /boot
/dev/mapper/fedora-home   46G  575M   43G   2% /home
/dev/sda1                 49G   16G   33G  33% /run/media/root/D6DC85C0DC859AFF
/dev/md0              2.4G   68M  2.2G   3% /root/test
```

Underlined line shows it is mounted on /root/test

Following step shows how the file system used for storage. So put some files on md0 drive it is visible on mount pt.[6]

```
[root@CGLAB test1]# ls
lost+found  osa total assignment  QP.odt  raje.c  raj.l
```

5. FAULT TOLERANCE IN RAID 5

We will see what happens when one of the disk failed in RAID 5. In our experiment we will set sdc1 disk faulty and observe the log status. It continues to work fine but in degraded mode. Following step shows this.

```
[root@CGLAB ~]# mdadm /dev/md0 -f /dev/sdc1

mdadm: set /dev/sdc1 faulty in /dev/md0

[root@CGLAB ~]# tail /var/log/messages

Nov 27 17:32:04 CGLAB kernel: [13362.597647] md: md0 still in use.

Nov 27 17:32:04 CGLAB kernel: [13362.797729] md: md0 still in use.

Nov 27 17:42:21 CGLAB kernel: [13979.614641] md/raid:md0: Disk failure on sdc1, disabling device.

Nov 27 17:42:21 CGLAB kernel: [13979.614641] md/raid:md0: Operation continuing on 1 devices.
```

It can be checked in /proc/mdstat file as follows.

```
[root@CGLAB ~]# cat /proc/mdstat

Personalities : [raid6] [raid5] [raid4]

md0 : active raid5 sdc1[2](F) sdb1[0]

2496896 blocks super 1.2 level 5, 64k chunk, algorithm 2 [2/1] [U_]

unused devices: <none>
```

5.1 Hot removing faulty disk

In RAID 5 faulty disks can be hot removed without affecting the operation keeping all data intact on remaining disks in RAID. Following step shows it.[1]

```
[root@CGLAB ~]# mdadm /dev/md0 -r /dev/sdc1
mdadm: hot removed /dev/sdc1 from /dev/md0
[root@CGLAB ~]# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdb1[0]
2496896 blocks super 1.2 level 5, 64k chunk, algorithm 2 [2/1] [U_]
unused devices: <none>
```

5.2 Hot adding disk in RAID 5

After removing fault we can add disk without affecting operation. We can add other disk of same configuration and same size partition. Following step shows it.[1]

```
[root@CGLAB ~]# mdadm /dev/md0 -a /dev/sdc1
mdadm: added /dev/sdc1
[root@CGLAB ~]# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdc1[2] sdb1[0]
2496896 blocks super 1.2 level 5, 64k chunk, algorithm 2 [2/1] [U_]
[>.....................] recovery = 1.0% (25724/2496896) finish=8.0min speed=5144K/sec
unused devices: <none>
```

6. DRAWBACKS OF RAID 5

RAID 5 does not replace regular off-line backups. It protects the system against the failure of one disk. It does not protect against the accidental deletion of files. It does not protect against other failures, fires, hackers, or any number of other hazards. Second write performance is not good. Whenever a random block is written, at least one data block and the parity block for that stripe must be updated. The RAID system doesn't know what the new parity block should contain until it has read the old parity block and the old data. Each random write therefore expands into four operations: two reads and two writes.[1]

7. REFERENCES

[1] Evi Nemeth., Garth Snyder, Trent R. Hein., Ben Whaley 2012. Unix and Linux System Administration.

[2] Paul Cobbaut. 2013 Linux System Administration.

[3] Maurice J Bach. The design of the UNIX OS

[4] Seymour lipschutz, Marc Lars Lipson 2010 Discrete Mathematics.

[5] Ellis Horowitz, Sartaz Sahni, Sanguthevar Rajasekaran 2007 Fundamentals of computer Algorirhms

[6] Kenneth Rosen, Douglas Host, Rachel Klee, James Farber, Richard Rosinski 2010 The Complete Reference Unix.

[7] Terry Collings, Kurt Wall 2012 Redhat Linux Networking and System Administration.