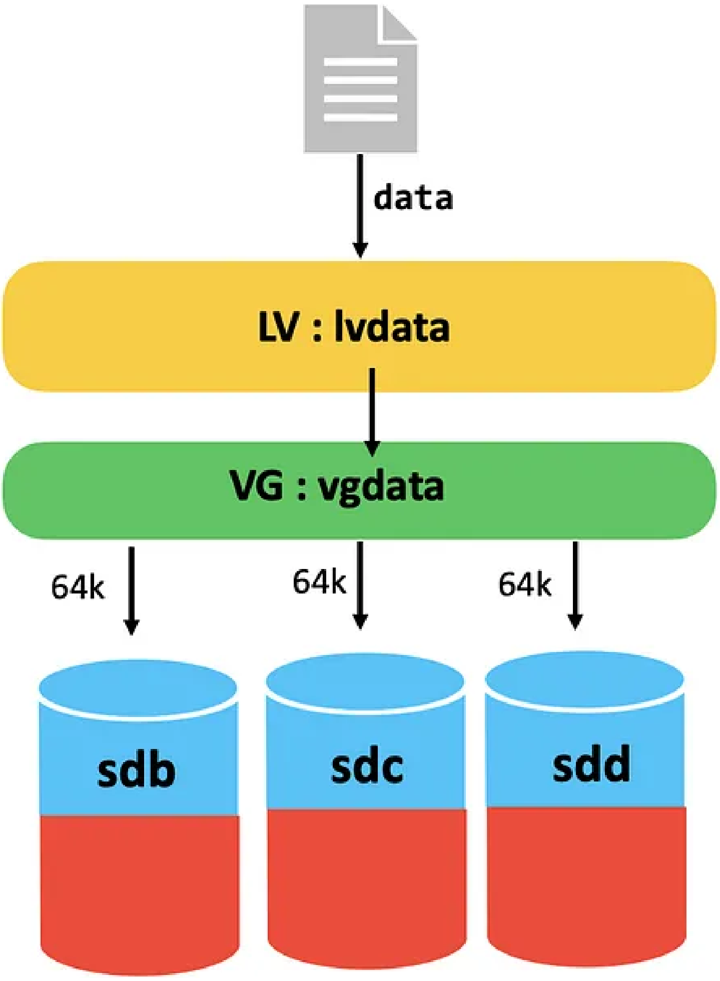
Boosting Linux Storage Performance with LVM Striping

Ahmed Mansouri · Follow

6 min read · Feb 4, 2024

Listen Share

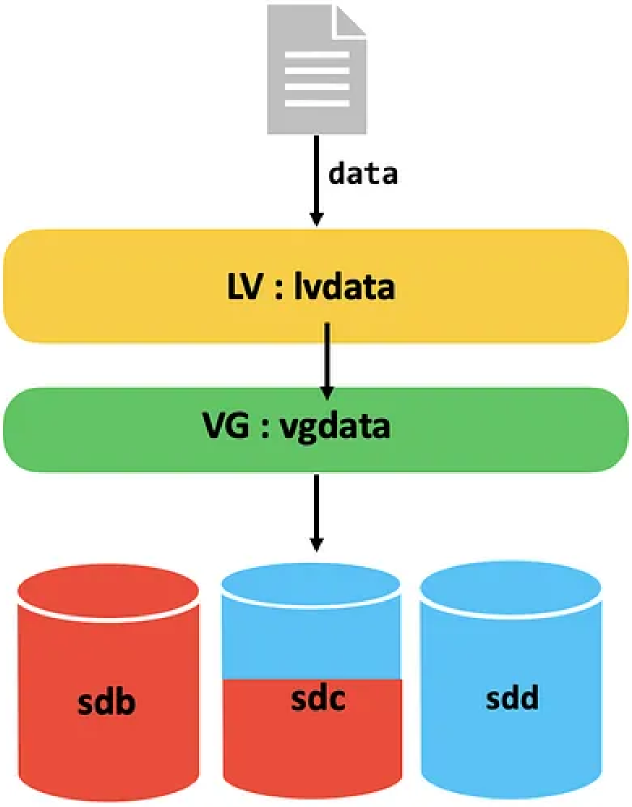


In the realm of storage management, achieving optimal disk performance takes precedence. The need for faster data access, reduced latency, and improved I/O operations and throughout has led to the adoption of advanced techniques. One such technique is `LVM Striping`, a powerful feature of the Logical Volume Manager `LVM` that can significantly enhance volume performance.

I — Linear LVM vs Striping LVM:

I .1 — Linear LVM:

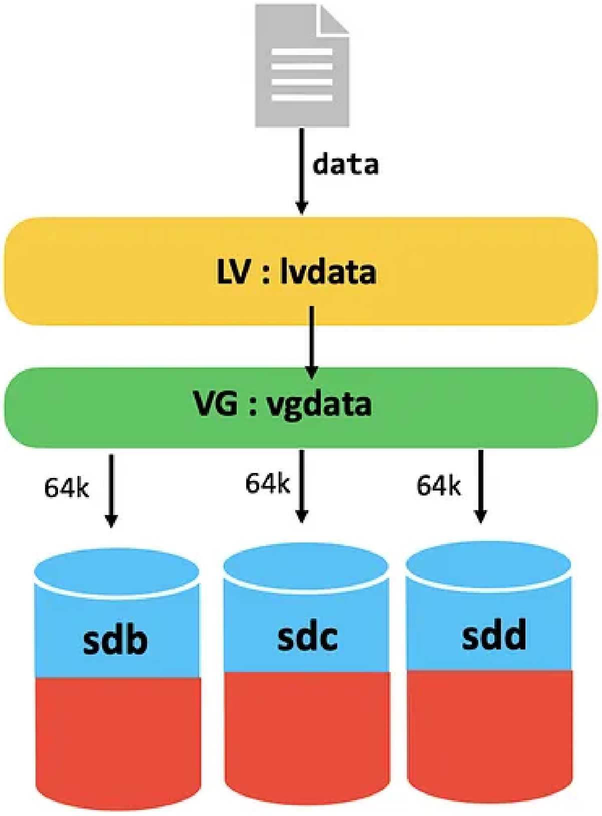
The linear configuration, often considered the **standard LVM** setup, involves adding multiple physical volumes (disks) to a volume group in a linear fashion. This means that data is sequentially stored across these volumes, utilizing one disk before moving on to the next.



While linear LVM provides a straightforward approach to storage expansion, it may not fully exploit the potential for parallel processing and increased throughput.

I .2 — Striping LVM:

In contrast, Striping LVM takes a more advanced approach by distributing data across multiple physical volumes simultaneously. This striping process creates a logical volume that spans the disks, allowing for parallel read and write operations. The result is a substantial boost in performance, making Striping LVM an attractive option for environments with high I/O and throughput demands.



1. **Stripe Creation:**

- The data is divided into segments known as “ `stripes` ”
- Each stripe is written to a specific disk in the striped logical volume `LV` .

2. Parallel Writing of Stripes:

- These stripes are simultaneously written across the multiple physical volumes `PVs` (disks).

3. Balanced Distribution:

- The stripes are distributed evenly across the disks, preventing a single disk from becoming a bottleneck.

4. Enhanced IOPS and Throughput:

- Through the parallel writing of stripes, LVM Striping significantly boosts the overall throughput and performance of the logical volume.

I.3 — Scenario :

Consider a scenario with three disk drives allocated to three physical volumes PVs. If each individual physical volume can achieve a total of `125M/s` as throughput :

- Employing “**LVM Striping**” would result in a volume group capable of `375M/s` .
- In contrast, using “**LVM Linear**”, the throughout remains at `125M/s` , and no matter how many disks we add in LVM.

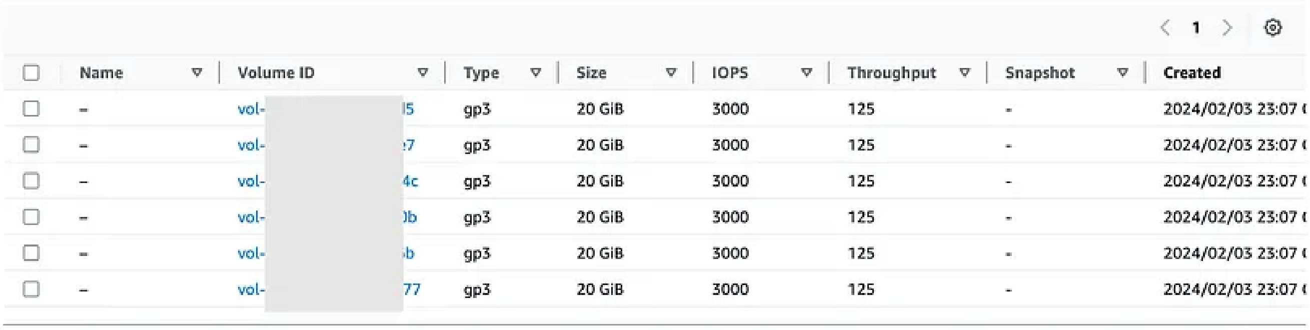
II — Setting up “Linear LVM“ and “Striping LVM”:

In this section, we’ll establish two volume groups (VGs): the initial one for Linear LVM utilizing 3 disks, and the second one for Striping LVM, also with three disks. Following the VG setup, we’ll proceed to create Logical Volumes (LVMs) on each group, then formating and mounting them

II.1 — Initial check

> **Environment :**
>
> — Server : AWS EC2 instance = *m4.10xlarge ,*
>
> — EBS Volumes : *6 x 20G EBS volumes* .
>
> — Each EBS volume has the following characteristics : { Type : *GP3,* IOPS = *3000 ,* Throughput = *`125MiB/s`* }
>
> — OS/image : Amazon Linux 2

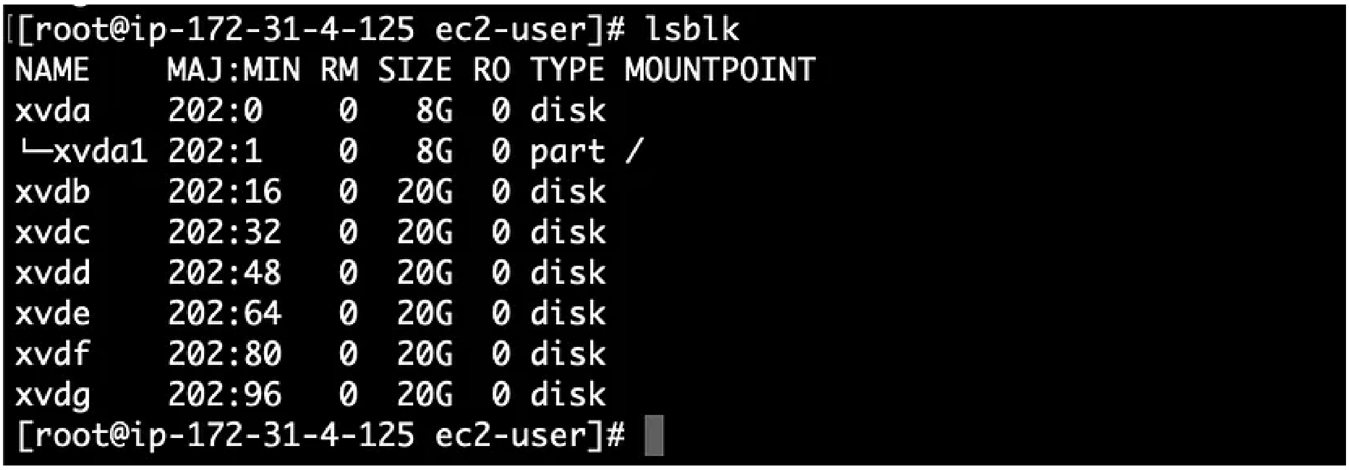
Here the list of the volumes :



☐	Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot	Created
☐	–	vol-[redacted]l5	gp3	20 GiB	3000	125	-	2024/02/03 23:07 (
☐	–	vol-[redacted]7	gp3	20 GiB	3000	125	-	2024/02/03 23:07 (
☐	–	vol-[redacted]4c	gp3	20 GiB	3000	125	-	2024/02/03 23:07 (
☐	–	vol-[redacted]0b	gp3	20 GiB	3000	125	-	2024/02/03 23:07 (
☐	–	vol-[redacted]b	gp3	20 GiB	3000	125	-	2024/02/03 23:07 (
☐	–	vol-[redacted]77	gp3	20 GiB	3000	125	-	2024/02/03 23:07 (

Let’s check our EC2 instance and see our disks

```
# lsblk
```

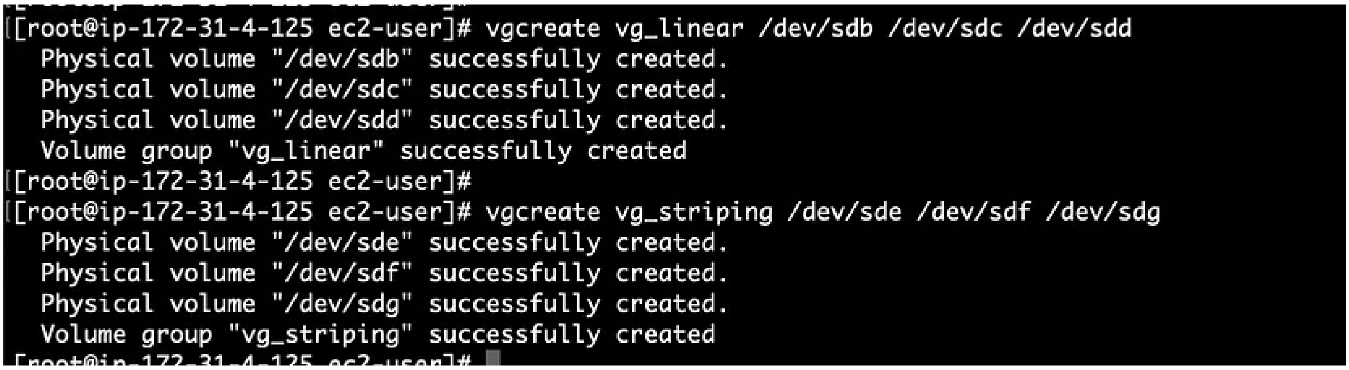


```
[[root@ip-172-31-4-125 ec2-user]# lsblk
 NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
 xvda    202:0    0   8G  0 disk
 └─xvda1 202:1    0   8G  0 part /
 xvdb    202:16   0  20G  0 disk
 xvdc    202:32   0  20G  0 disk
 xvdd    202:48   0  20G  0 disk
 xvde    202:64   0  20G  0 disk
 xvdf    202:80   0  20G  0 disk
 xvdg    202:96   0  20G  0 disk
 [root@ip-172-31-4-125 ec2-user]#
```

II.2 — Setting up the VGs:

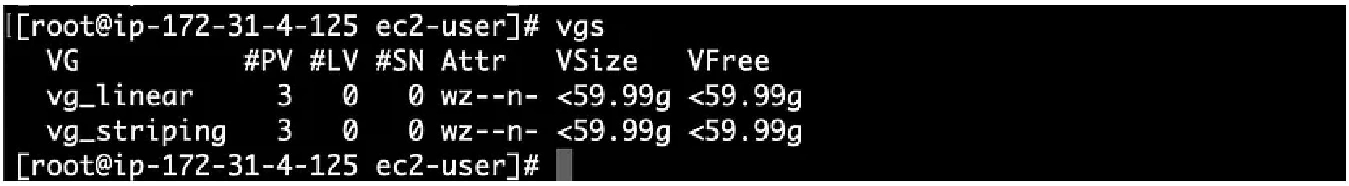
In order to be able to create the LVs, we need first to create the VG for each one of them :

```
# vgcreate vg_linear /dev/sdb /dev/sdc /dev/sdd

# vgcreate vg_striping /dev/sde /dev/sdf /dev/sdg
```



```
[[root@ip-172-31-4-125 ec2-user]# vgcreate vg_linear /dev/sdb /dev/sdc /dev/sdd
   Physical volume "/dev/sdb" successfully created.
   Physical volume "/dev/sdc" successfully created.
   Physical volume "/dev/sdd" successfully created.
   Volume group "vg_linear" successfully created
[[root@ip-172-31-4-125 ec2-user]#
[[root@ip-172-31-4-125 ec2-user]# vgcreate vg_striping /dev/sde /dev/sdf /dev/sdg
   Physical volume "/dev/sde" successfully created.
   Physical volume "/dev/sdf" successfully created.
   Physical volume "/dev/sdg" successfully created.
   Volume group "vg_striping" successfully created
 [root@ip-172-31-4-125 ec2-user]#
```

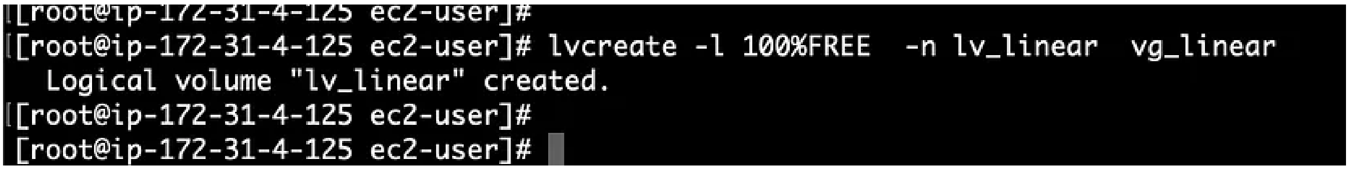
```
# vgs
```



```
[[root@ip-172-31-4-125 ec2-user]# vgs
  VG          #PV #LV #SN Attr   VSize   VFree
  vg_linear     3   0   0 wz--n- <59.99g <59.99g
  vg_striping   3   0   0 wz--n- <59.99g <59.99g
 [root@ip-172-31-4-125 ec2-user]#
```

II.3 —Create a Logical Volume as “Linear LVM”

```
# lvcreate -l 100%FREE  -n lv_linear  vg_linear
```

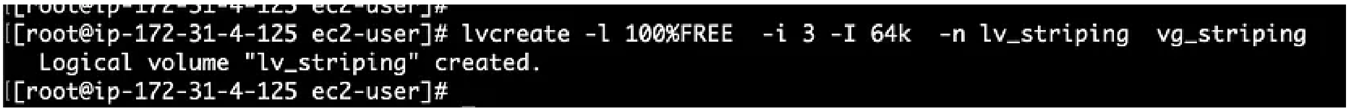


```
[[root@ip-172-31-4-125 ec2-user]#
[[root@ip-172-31-4-125 ec2-user]# lvcreate -l 100%FREE  -n lv_linear  vg_linear
  Logical volume "lv_linear" created.
[[root@ip-172-31-4-125 ec2-user]#
 [root@ip-172-31-4-125 ec2-user]#
```

- `-l 100%FREE` : The size of LV is set to match the entirety of the available free space within the VG `vg_linear` .
- `-n lv_linear` : LV name
- `vg_linear` : Target VG

II.4 — Create a Logical Volume as “Striping LVM”

```
# lvcreate -l 100%FREE  -i 3 -I 64k  -n lv_striping  vg_striping
```



```
[[root@ip-172-31-4-125 ec2-user]#
[[root@ip-172-31-4-125 ec2-user]# lvcreate -l 100%FREE  -i 3 -I 64k  -n lv_striping  vg_striping
  Logical volume "lv_striping" created.
[[root@ip-172-31-4-125 ec2-user]#
```

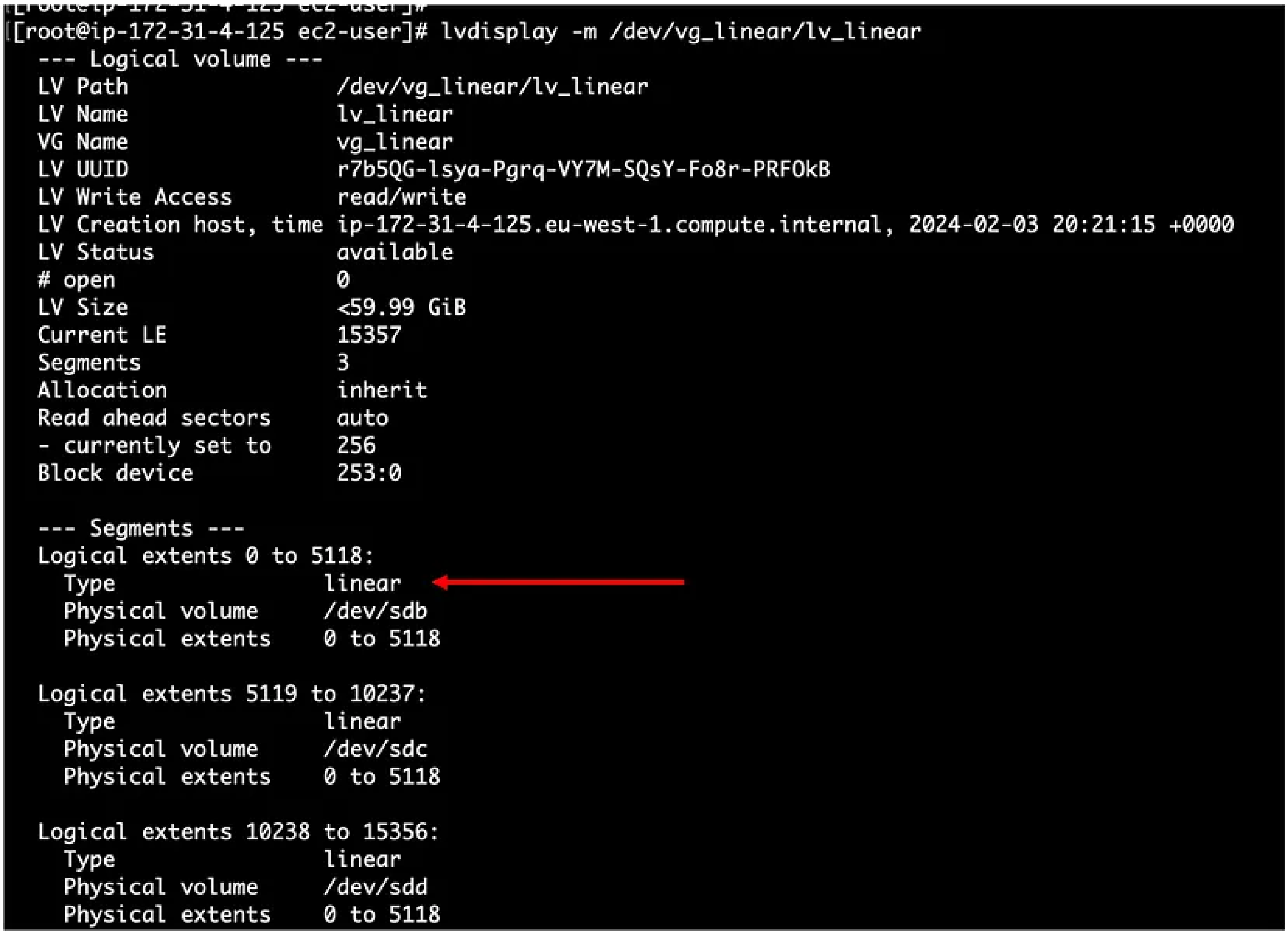
- `-l 100%FREE` : The size of LV is set to match the entirety of the available free space within the VG `vg_striping` .
- `-n lv_striping` : LV name
- `vg_striping` : Target VG
- `-i 3` : stripes Number
- `-I 64k` : stripe size

II.5 — Check the LVs

Let’s check the new LVs:

- Let’s start by `lv_linear`

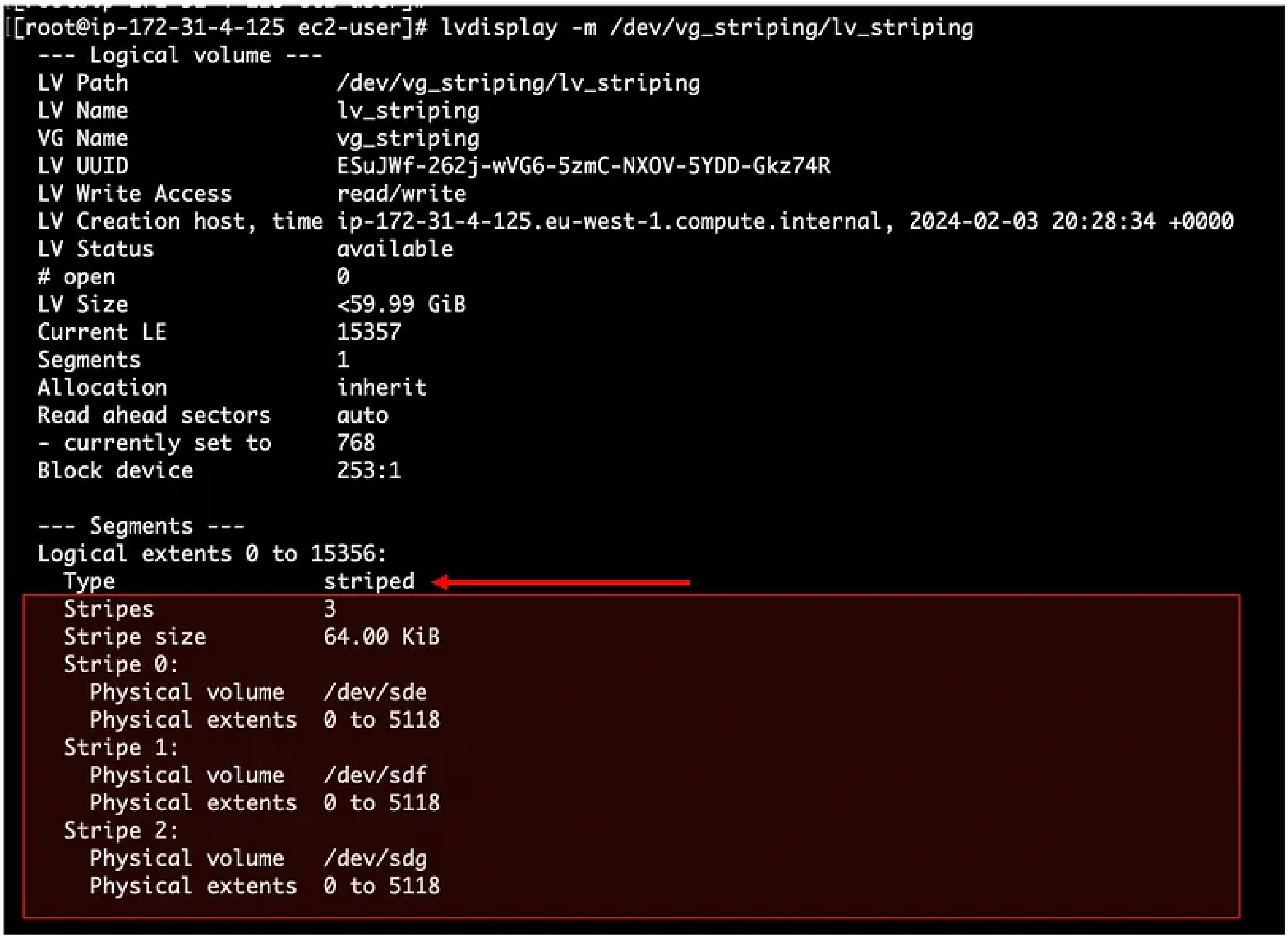
```
# lvdisplay -m /dev/vg_linear/lv_linear
```



```
[root@ip-172-31-4-125 ec2-user]#
[root@ip-172-31-4-125 ec2-user]# lvdisplay -m /dev/vg_linear/lv_linear
  --- Logical volume ---
  LV Path                /dev/vg_linear/lv_linear
  LV Name                lv_linear
  VG Name                vg_linear
  LV UUID                r7b5QG-lsya-Pgrq-VY7M-SQsY-Fo8r-PRFOkB
  LV Write Access        read/write
  LV Creation host, time ip-172-31-4-125.eu-west-1.compute.internal, 2024-02-03 20:21:15 +0000
  LV Status              available
  # open                 0
  LV Size                <59.99 GiB
  Current LE             15357
  Segments               3
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0

  --- Segments ---
  Logical extents 0 to 5118:
    Type                linear
    Physical volume     /dev/sdb
    Physical extents    0 to 5118

  Logical extents 5119 to 10237:
    Type                linear
    Physical volume     /dev/sdc
    Physical extents    0 to 5118

  Logical extents 10238 to 15356:
    Type                linear
    Physical volume     /dev/sdd
    Physical extents    0 to 5118
```

- Let’s check new `lv_striping`

```
# lvdisplay -m /dev/vg_striping/lv_striping
```



```
[root@ip-172-31-4-125 ec2-user]# lvdisplay -m /dev/vg_striping/lv_striping
  --- Logical volume ---
  LV Path                /dev/vg_striping/lv_striping
  LV Name                lv_striping
  VG Name                vg_striping
  LV UUID                ESuJWf-262j-wVG6-5zmC-NXOV-5YDD-Gkz74R
  LV Write Access        read/write
  LV Creation host, time ip-172-31-4-125.eu-west-1.compute.internal, 2024-02-03 20:28:34 +0000
  LV Status              available
  # open                 0
  LV Size                <59.99 GiB
  Current LE             15357
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     768
  Block device           253:1

  --- Segments ---
  Logical extents 0 to 15356:
    Type                striped
    Stripes             3
    Stripe size         64.00 KiB
    Stripe 0:
      Physical volume   /dev/sde
      Physical extents  0 to 5118
    Stripe 1:
      Physical volume   /dev/sdf
      Physical extents  0 to 5118
    Stripe 2:
      Physical volume   /dev/sdg
      Physical extents  0 to 5118
```

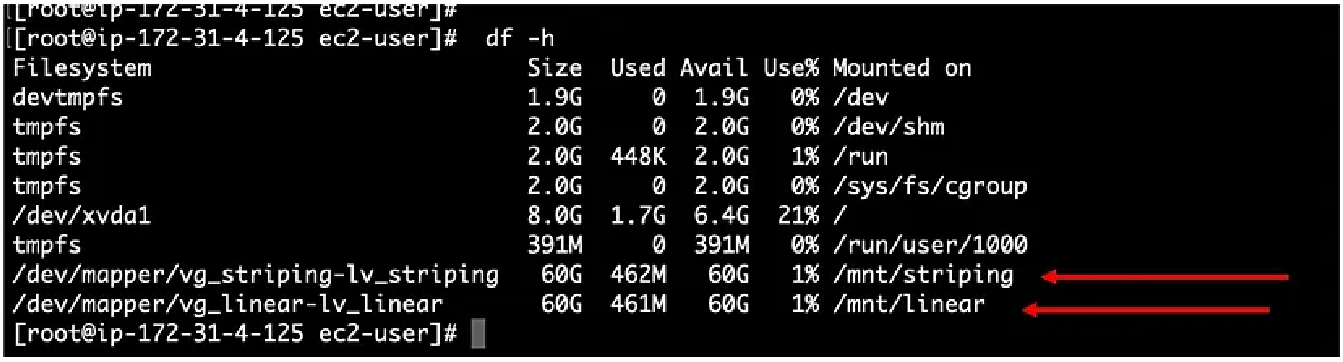
==> Here we can see the different details about the striping that we configured .

II.6 — Formating and mounting the LVs

```
# mkfs.xfs /dev/vg_linear/lv_linear

# mkfs.xfs /dev/vg_striping/lv_striping
```

```
# mkdir /mnt/linear

# mkdir /mnt/striping
```

```
# mount /dev/vg_linear/lv_linear  /mnt/linear/

# mount /dev/vg_striping/lv_striping /mnt/striping/
# df -h
```



```
[[root@ip-172-31-4-125 ec2-user]#
[[root@ip-172-31-4-125 ec2-user]#  df -h
 Filesystem                          Size  Used Avail Use% Mounted on
 devtmpfs                            1.9G     0  1.9G   0% /dev
 tmpfs                               2.0G     0  2.0G   0% /dev/shm
 tmpfs                               2.0G  448K  2.0G   1% /run
 tmpfs                               2.0G     0  2.0G   0% /sys/fs/cgroup
 /dev/xvda1                          8.0G  1.7G  6.4G  21% /
 tmpfs                               391M     0  391M   0% /run/user/1000
 /dev/mapper/vg_striping-lv_striping  60G  462M   60G   1% /mnt/striping
 /dev/mapper/vg_linear-lv_linear      60G  461M   60G   1% /mnt/linear
 [root@ip-172-31-4-125 ec2-user]#
```

==> The LVs are now mounted correctly under the appropriate directories

III — Benchmarks the LVs/disks

In order to benchmark our LVs/disks, we will use the `fio` tool. So let us install it before :

```
# yum install fio -y
```

III .1 — Benchmark "lv_linear"

In order to benchmark the LV, we will use the `fio` config file below that will help us generating traffic for `400M` as throughput :

```
# cat fio_config-1.fio

[global]
ioengine=libaio
runtime=60
time_based
direct=1
rw=write
size=10G
bs=512K
rate=400M
numjobs=16
```

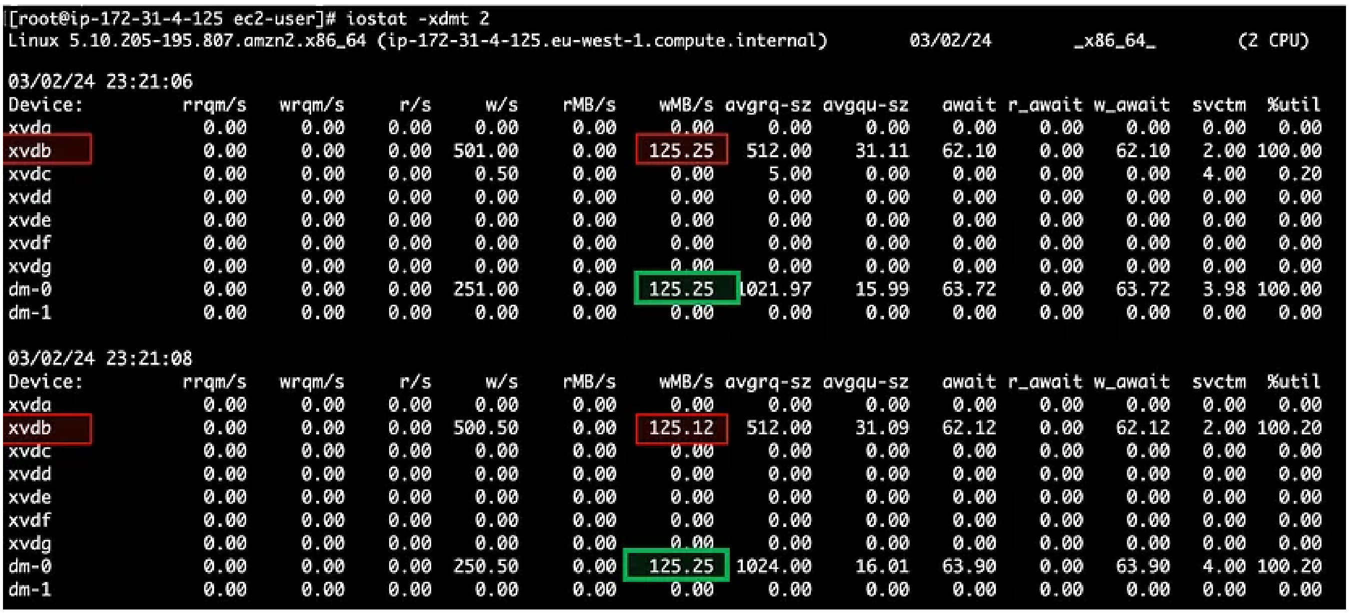
```
[job1]
filename=/mnt/linear/testfile
```

Execute now the `fio` tool as below using the defined file above:

```
# fio fio_config-1.fio
```

- In the same time, run the `iostat` tool to monitor the disks usage in a separate terminal :

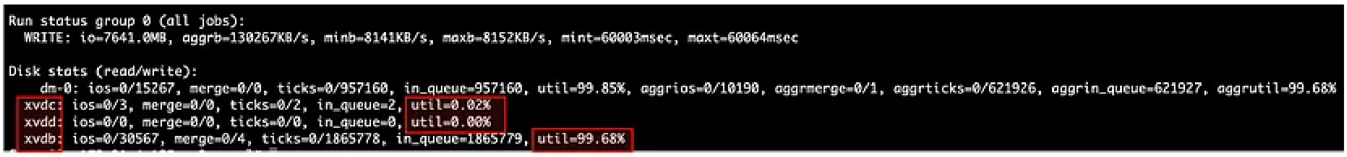
```
# iostat -xdmt 2
```



```
[root@ip-172-31-4-125 ec2-user]# iostat -xdmt 2
Linux 5.10.205-195.807.amzn2.x86_64 (ip-172-31-4-125.eu-west-1.compute.internal)       03/02/24        _x86_64_        (2 CPU)

03/02/24 23:21:06
Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvda              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdb              0.00     0.00    0.00  501.00     0.00   125.25   512.00    31.11   62.10    0.00   62.10   2.00 100.00
xvdc              0.00     0.00    0.00    0.50     0.00     0.00     5.00     0.00    0.00    0.00    0.00   4.00   0.20
xvdd              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvde              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdf              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdg              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dm-0              0.00     0.00    0.00  251.00     0.00   125.25  1021.97    15.99   63.72    0.00   63.72   3.98 100.00
dm-1              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00

03/02/24 23:21:08
Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvda              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdb              0.00     0.00    0.00  500.50     0.00   125.12   512.00    31.09   62.12    0.00   62.12   2.00 100.20
xvdc              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdd              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvde              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdf              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdg              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dm-0              0.00     0.00    0.00  250.50     0.00   125.25  1024.00    16.01   63.90    0.00   63.90   4.00 100.20
dm-1              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
```

==> We can see from the output above that ONLY the first disk `xvdb` is used and that the throughput is equal to `125M` which is the baseline value of one EBS volume itself.

The same thing is showing by the summary given the `fio` tool :



```
Run status group 0 (all jobs):
  WRITE: io=7641.0MB, aggrb=130267KB/s, minb=8141KB/s, maxb=8152KB/s, mint=60003msec, maxt=60064msec

Disk stats (read/write):
    dm-0: ios=0/15267, merge=0/0, ticks=0/957160, in_queue=957160, util=99.85%, aggrios=0/10190, aggrmerge=0/1, aggrticks=0/621926, aggrin_queue=621927, aggrutil=99.68%
  xvdc: ios=0/3, merge=0/0, ticks=0/2, in_queue=2, util=0.02%
  xvdd: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
  xvdb: ios=0/30567, merge=0/4, ticks=0/1865778, in_queue=1865779, util=99.68%
```

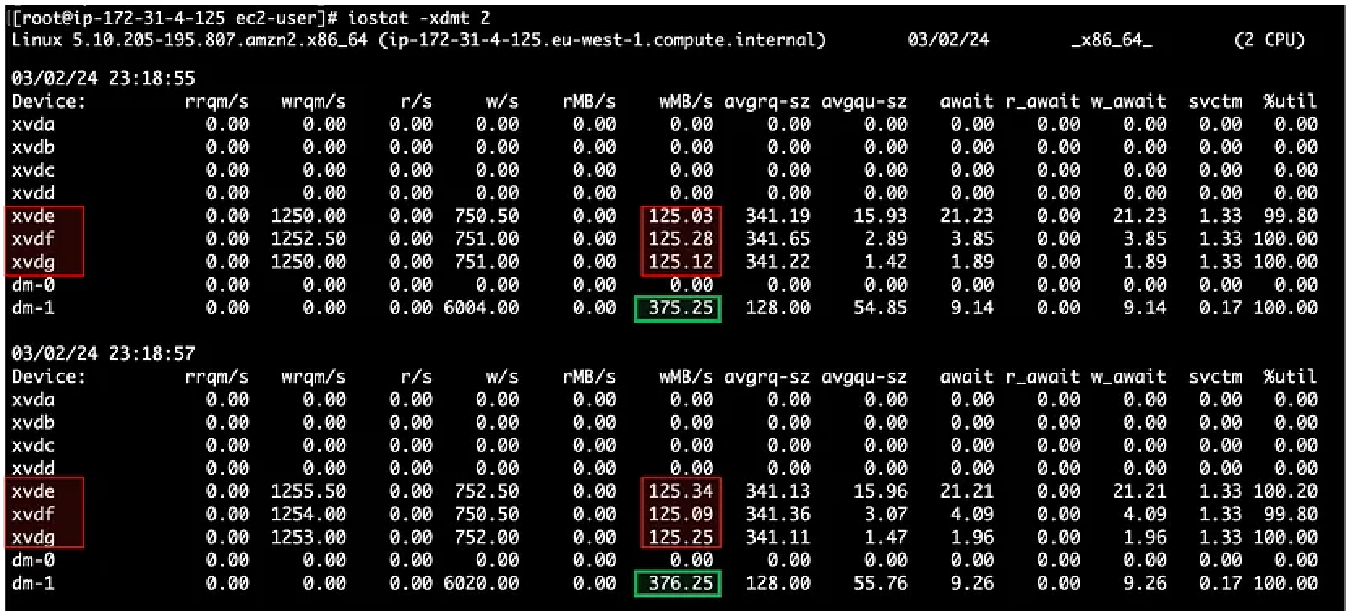
==> WRITE operations were exclusively performed on a single disk.

III .2 — Benchmark "lv_striping"

Use the same `fio` config file above but just change the `filename` parameter :

```
filename=/mnt/striping/testfile
```

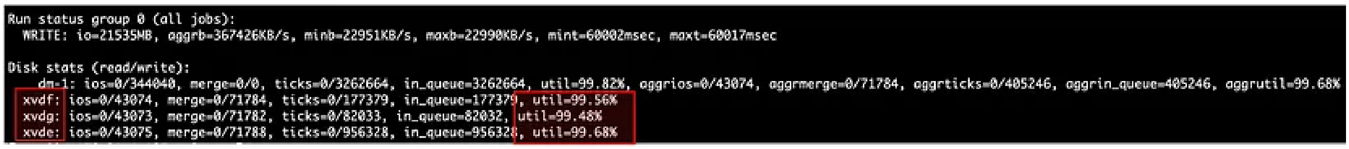
Use `iostat` to monitor the disks



```
[root@ip-172-31-4-125 ec2-user]# iostat -xdmt 2
Linux 5.10.205-195.807.amzn2.x86_64 (ip-172-31-4-125.eu-west-1.compute.internal)       03/02/24        _x86_64_        (2 CPU)

03/02/24 23:18:55
Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvda              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdb              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdc              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdd              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvde              0.00  1250.00    0.00  750.50     0.00   125.03   341.19    15.93   21.23    0.00   21.23   1.33  99.80
xvdf              0.00  1252.50    0.00  751.00     0.00   125.28   341.65     2.89    3.85    0.00    3.85   1.33 100.00
xvdg              0.00  1250.00    0.00  751.00     0.00   125.12   341.22     1.42    1.89    0.00    1.89   1.33 100.00
dm-0              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dm-1              0.00     0.00    0.00 6004.00     0.00   375.25   128.00    54.85    9.14    0.00    9.14   0.17 100.00

03/02/24 23:18:57
Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvda              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdb              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdc              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvdd              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
xvde              0.00  1255.50    0.00  752.50     0.00   125.34   341.13    15.96   21.21    0.00   21.21   1.33 100.20
xvdf              0.00  1254.00    0.00  750.50     0.00   125.09   341.36     3.07    4.09    0.00    4.09   1.33  99.80
xvdg              0.00  1253.00    0.00  752.00     0.00   125.25   341.11     1.47    1.96    0.00    1.96   1.33 100.00
dm-0              0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
dm-1              0.00     0.00    0.00 6020.00     0.00   376.25   128.00    55.76    9.26    0.00    9.26   0.17 100.00
```

==> All 3 disks used by the `lv_striping` participated in the WRITE operations, resulting in a combined throughput equal to the sum of the individual disk throughputs (`125M X 3 = 375M`).

The same thing is showing by the summary given the `fio` tool :



```
Run status group 0 (all jobs):
  WRITE: io=21535MB, aggrb=367426KB/s, minb=22951KB/s, maxb=22990KB/s, mint=60002msec, maxt=60017msec

Disk stats (read/write):
    dm-1: ios=0/344040, merge=0/0, ticks=0/3262664, in_queue=3262664, util=99.82%, aggrios=0/43074, aggrmerge=0/71784, aggrticks=0/405246, aggrin_queue=405246, aggrutil=99.68%
  xvdf: ios=0/43074, merge=0/71784, ticks=0/177379, in_queue=177379, util=99.56%
  xvdg: ios=0/43073, merge=0/71782, ticks=0/82033, in_queue=82032, util=99.48%
  xvde: ios=0/43075, merge=0/71788, ticks=0/956328, in_queue=956328, util=99.68%
```

Conclusion:

LVM Striping stands as a robust solution for organizations seeking to unlock the full potential of their storage infrastructure. By distributing data intelligently across multiple disks, LVM Striping not only boosts performance but also provides a