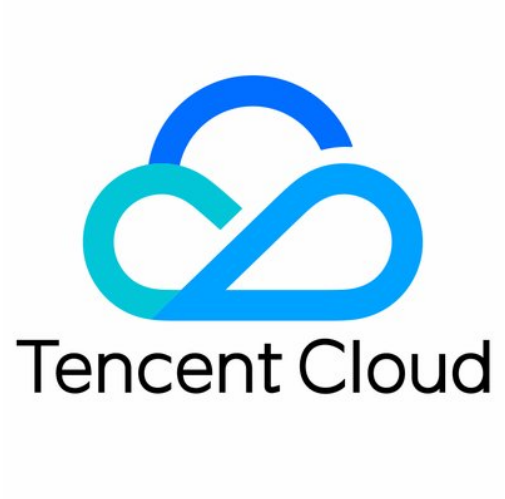
Cloud Block Storage

Cloud Hard Disk Performance

Product Documentation



Contents

Cloud Hard Disk Performance
Measuring cloud disk performance

Last updated : 2020-05-14 11:15:28

Metrics

Tencent Cloud CBS devices vary in performance and price by type. For more information, see Cloud Disk Types. Because different applications have different workloads, if the number of I/O requests is low, the cloud disk may not play its full performance.
The following metrics are generally used to measure the performance of a cloud disk:

- IOPS: read/write count per second. IOPS varies by the underlying drive type of the storage device.
- Throughput: read/written data volume per second, unit in MB/s.
- Latency: time elapsed from sending an I/O operation to receiving an acknowledgement (in seconds).

Testing Tool

FIO is a tool for testing disk performance. It is used to perform stress test and verification on hardware. This document uses FIO as an example.
We recommend that you use FIO together with libaio's I/O engine to perform the test. Please install FIO and libaio with reference to Tool Installation.

- **To avoid damaging important files in the system, do not perform FIO test on the system disk**.
- **To avoid data corruption caused by corruption of the metadata of the underlying file system, do not perform the test on the business data disk. **
- Ensure the `/etc/fstab` file **DOES NOT contain** the mount configuration of the disk to be tested. Otherwise, CVM may fail to launch.

Recommended Test Objects

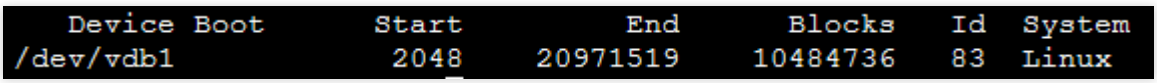
- We recommend that you perform FIO test on empty disks that do not store important data, and re-create the file system after completing the test.
- When testing disk performance, we recommend that you directly test raw data disks (such as /dev/vdb).
- When testing file system performance, we recommend you specify the specific file test (such as /data/file).

Tool Installation

1. Log in to the CVM as instructed in Log in to Linux Instance Using Standard Login Method. Here, take the CVM running CentOS 7.6 OS as an example.
2. Run the following command to check whether the cloud disk is 4KiB-aligned.

   ```
   fdisk -lu
   ```

   As shown below, if the Start value in the command output is divisible by 8, then the disk is 4KiB-aligned. Otherwise, complete 4KiB alignment before testing.

3. Run the following commands in sequence to install the testing tools, FIO and libaio.

   ```
   yum install libaio -y
   ```

   ```
   yum install libaio-devel -y
   ```

   ```
   yum install fio -y
   ```

   Once completed, start testing the cloud disk performance as instructed in the test example below.

Test Example

The testing formulas for different scenarios are basically the same, except the `rw`, `iodepth`, and `bs` (block size) parameters. For example, the optimal `iodepth` for each workload is different as it depends on the sensitivity of your application to the IOPS and latency.

Parameter description:

Parameter	Remarks	Sample value

bs	Block size per request. Valid values include 4k, 8k, and 16k.	4k
ioengine	I/O engine. We recommend that you use Linux’s async I/O engine.	libaio
iodepth	Queue depth of an I/O request.	1
direct	Specifies direct mode. • True (1) indicates that the O_DIRECT identifier is specified, the I/O cache will be ignored, and data will be written directly. • False (0) indicates that the O_DIRECT identifier is not specified. The default is True (1).	1
rw	Read and write mode. Valid values include read, write, randread, randwrite, randrw, and rw, readwrite.	read
time_based	Specifies that the time mode is used. As long as FIO runs based on the time, it is unnecessary to set this parameter.	N/A
runtime	Specifies the test duration, which is the FIO runtime.	600
refill_buffers	FIO will refill the I/O buffer at every submission. The default setting is to fill the I/O buffer only at the start and reuse the data.	N/A
norandommap	When performing random I/O operations, FIO overwrites every block of the file. If this parameter is set, a new offset will be selected without viewing the I/O history.	N/A
randrepeat	Specifies whether the random sequence is repeatable. True (1) indicates that the random sequence is repeatable. False (0) indicates that the random sequence is not repeatable. The default value is True (1).	0
group_reporting	When multiple jobs are concurrent, statistics for the entire group are printed.	N/A
name	Name of the job.	fio-read
size	Address space of the I/O test.	100GB
filename	Test object, which is the name of the disk to be tested.	/dev/sdb

Common use cases are as follows:

- **bs=4k iodepth=1**: random read/write test, which can reflect the latency performance of a disk. Run the following command to test the random read latency of the disk:

```
fio -bs=4k -ioengine=libaio -iodepth=1 -direct=1 -rw=randread -time_based -runtime=600 -refill
_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-randread-lat --size=10G -filena
me=/dev/vdb
```

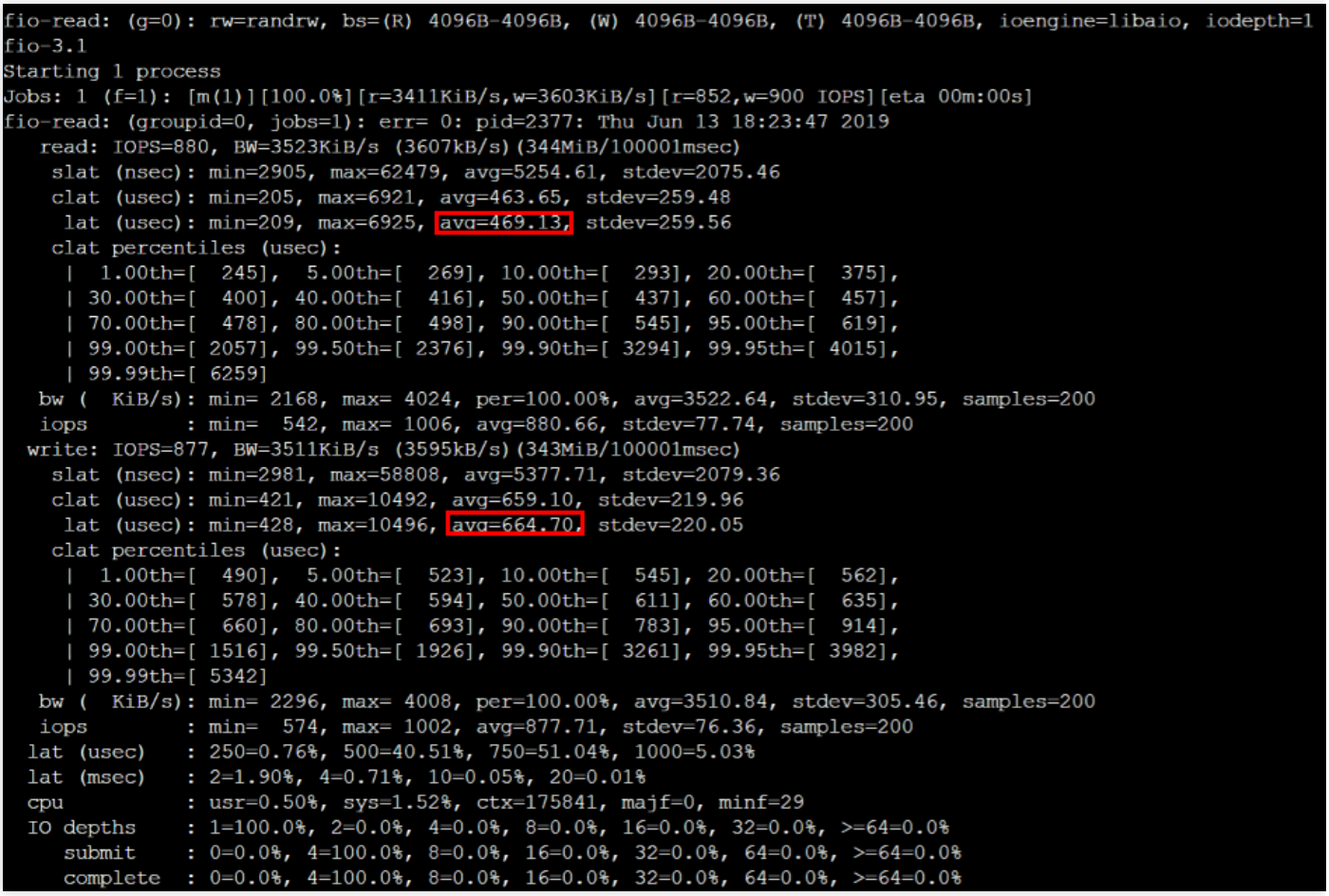
  Run the following command to test the random write latency of the disk:

```
fio -bs=4k -ioengine=libaio -iodepth=1 -direct=1 -rw=randwrite -time_based -runtime=600 -refil
l_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-randwrite-lat --size=10G -file
name=/dev/vdb
```

  Run the following command to test the random hybrid read and write latency performance of an SSD cloud disk:

```
fio --bs=4k --ioengine=libaio --iodepth=1 --direct=1 --rw=randrw --time_based --runtime=100 --
refill_buffers --norandommap --randrepeat=0 --group_reporting --name=fio-read --size=1G --file
name=/dev/vdb
```

The following figure shows the command output:



- **bs=128k iodepth = 32**: sequential read/write test, can reflect the throughput performance of the disk.
  Run the following command to test the sequential read throughput bandwidth:

```
fio -bs=128k -ioengine=libaio -iodepth=32 -direct=1 -rw=read -time_based -runtime=600 -refill_
buffers -norandommap -randrepeat=0 -group_reporting -name=fio-read-throughput --size=10G -file
name=/dev/vdb
```

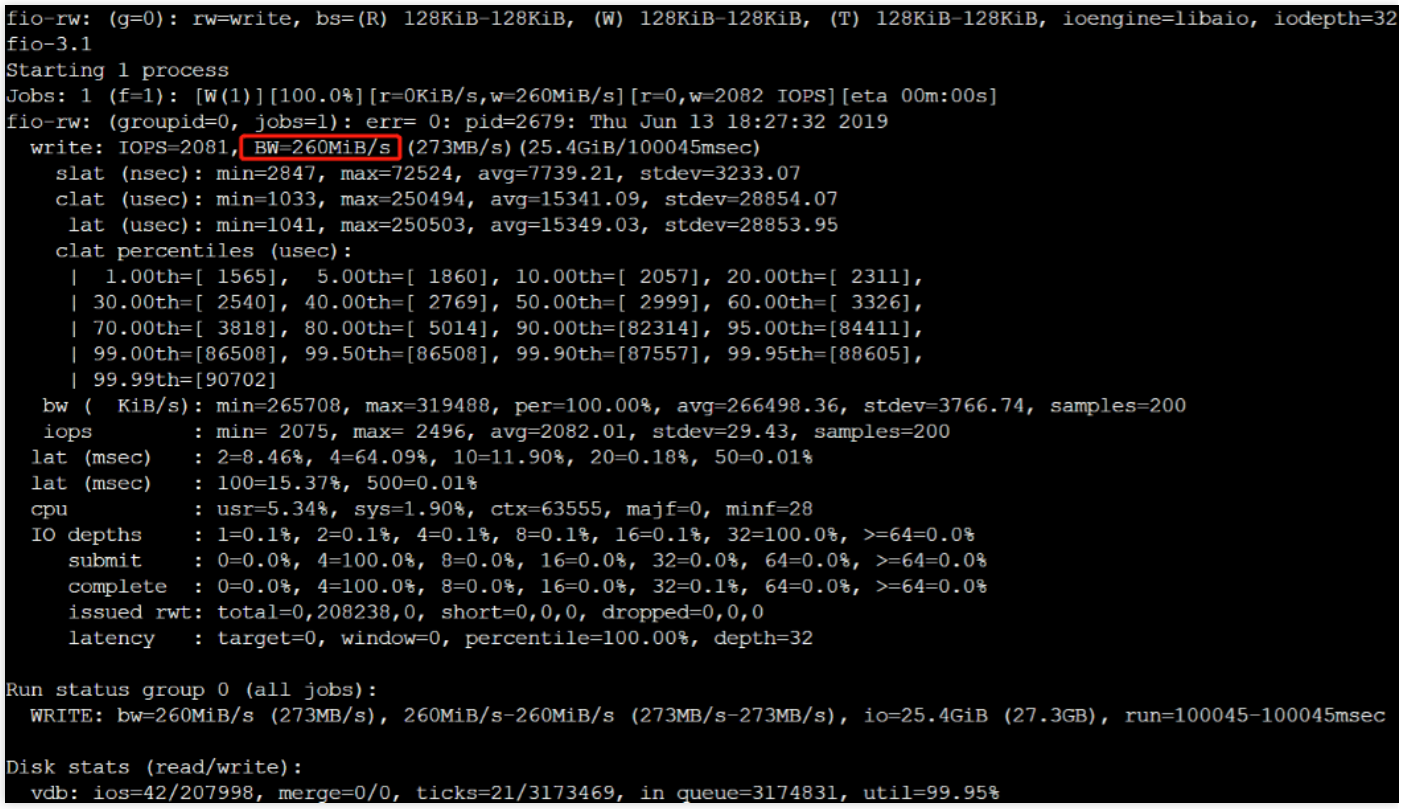
  Run the following command to test the sequential write throughput bandwidth:

```
fio -bs=128k -ioengine=libaio -iodepth=32 -direct=1 -rw=write -time_based -runtime=600 -refill
_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-write-throughput --size=10G -fi
lename=/dev/vdb
```

Run the following command to test the sequential read throughput performance of an SSD cloud disk:

```
fio --bs=128k --ioengine=libaio --iodepth=32 --direct=1 --rw=read --time_based --runtime=100 --refill_buffers --norandommap --randrepeat=0 --group_reporting --name=fio-rw --size=1G --filename=/dev/vdb
```

The following figure shows the command output:



- **bs=4k iodepth=32**: random read/write test, can reflect the IOPS performance of the hard disk. Run the following command to test the random read IOPS of the disk:

```
fio -bs=4k -ioengine=libaio -iodepth=32 -direct=1 -rw=randread -time_based -runtime=600 -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-randread-iops --size=10G -filename=/dev/vdb
```

Run the following command to test the random write IOPS of the disk:

```
fio -bs=4k -ioengine=libaio -iodepth=32 -direct=1 -rw=randwrite -time_based -runtime=600 -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-randwrite-iops --size=10G -filename=/dev/vdb
```

Test the random read IOPS performance of the SSD cloud disk. This is shown in the following figure:

```
[root@VM_16_21_centos ~]# fio -bs=4k -ioengine=libaio -iodepth=32 -direct=1 -rw=randread -time_based
 -runtime=300  -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-randread  --siz
e=100G -filename=/dev/vdc
fio-randread: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=lib
aio, iodepth=32
fio-3.1
Starting 1 process
Jobs: 1 (f=1): [r(1)][100.0%][r=18.8MiB/s,w=0KiB/s][r=4804,w=0 IOPS][eta 00m:00s]
fio-randread: (groupid=0, jobs=1): err= 0: pid=2689: Tue Jul 16 13:39:33 2019
   read: IOPS=4800, BW=18.8MiB/s (19.7MB/s)(5626MiB/300017msec)
    slat (usec): min=2, max=3183, avg= 7.55, stdev=14.34
    clat (usec): min=209, max=777668, avg=6656.04, stdev=45385.49
     lat (usec): min=371, max=777673, avg=6664.08, stdev=45385.46
    clat percentiles (usec):
     |  1.00th=[   635],  5.00th=[   832], 10.00th=[   938], 20.00th=[  1090],
     | 30.00th=[  1237], 40.00th=[  1352], 50.00th=[  1467], 60.00th=[  1582],
     | 70.00th=[  1713], 80.00th=[  1991], 90.00th=[  9372], 95.00th=[ 19006],
     | 99.00th=[ 38536], 99.50th=[341836], 99.90th=[734004], 99.95th=[750781],
     | 99.99th=[767558]
   bw (  KiB/s): min=  256, max=38424, per=100.00%, avg=19202.16, stdev=13626.10, samples=600
   iops        : min=   64, max= 9606, avg=4800.52, stdev=3406.52, samples=600
  lat (usec)   : 250=0.01%, 500=0.12%, 750=2.64%, 1000=10.86%
  lat (msec)   : 2=66.45%, 4=5.87%, 10=4.59%, 20=6.49%, 50=2.32%
  lat (msec)   : 100=0.01%, 500=0.36%, 750=0.26%, 1000=0.05%
  cpu          : usr=1.38%, sys=5.00%, ctx=233328, majf=0, minf=63
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=100.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.1%, 64=0.0%, >=64=0.0%
     issued rwt: total=1440148,0,0, short=0,0,0, dropped=0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=32

Run status group 0 (all jobs):
   READ: bw=18.8MiB/s (19.7MB/s), 18.8MiB/s-18.8MiB/s (19.7MB/s-19.7MB/s), io=5626MiB (5899MB), run=
300017-300017msec

Disk stats (read/write):
  vdc: ios=1440052/0, merge=0/0, ticks=9478008/0, in_queue=9485217, util=99.98%
```

Building Up RAID Groups

Last updated : 2020-08-31 09:42:54

Introduction to RAID

RAID (Redundant Array of Independent Disks) combines multiple disks into a disk array in order to improve data read and write performance and reliability. Meanwhile, the operating system will use the disk array as a single hard disk. RAID has multiple levels at present. The following will introduce RAID0, RAID1, RAID01 and RAID10. Depending on the level of RAID, the disk array offers improvement benefits in data integration, fault tolerance, and throughput or capacity compared with a large hard disk with considerable capacity.

Note:

- please renew your elastic cloud disk that is about expire to avoid impact on your RAID array due to forced isolation of the disk.
- We recommend you use partitions of the same size when creating RAID 1, RAID 01, and RAID 10 for minimal waste of disk capacity.

The following is a comparison of different RAID levels:

- **Pros**: Read and write can be synchronized.
  Theoretically, the read and write rate can be N times that of a single disk (N is the number of disks in RAID0), despite practical limitations such as file size, file system size, etc.
- **Cons**: No data redundancy. If a single disk is damaged, it is likely to cause all data lost in the most serious cases.

- **Pros**:
  - Fast read.
  - High data reliability. Damage to a single disk will not result in unrepairable data.
- **Cons**:
  - Low disk utilization.
  - The write speed is limited by that of a single disk.

RAID Level	Description	Pros and Cons	Scenarios (Recommended)

RAID 0	Storage mode: Data are striped and stored in different disks. Virtual disk size: the combined capacity of all disks in the array.	Require a higher level of I/O performance, and have backed up data through other means or there is no need for data backup.	
RAID 1	Storage mode: Data are stored through image memory into disks. Virtual disk size: depends on the capacity of the disk with the smallest one in the array.	Require high read performance and backups of written data.	
RAID 01	First deal with data through RAID0, then RAID1.	• **Pros**: Take into both RAID0 and RAID1 advantages. • **Cons**: ◦ Costs are relatively high and it is essential to use at least 4 disks. ◦ The damage of a single disk will make the other disks unavailable in the same array.	-
RAID 10	Build RAID 1 with multiple disks, and then build RAID 0 with more than one RAID 1.	• **Pros**: Take into both RAID0 and RAID1 advantages. • **Cons**: Costs are relatively high and it is essential to use at least 4 disks.	

Building RAID

The following describes how to build RAID 0 on CentOS by using 4 Tencent Cloud elastic cloud disks. The operation may vary for different operating systems (OS) and RAID levels. Since this

document is for reference only, for detailed instructions and differences, please see the product documentation for a specific OS or relevant RAID documents.

Linux kernel provides a MD module that manages RAID devices, so you may create RAID 0 by directly calling this module using the mdadm tool.

```
[root@VM_63_126_centos ~]# fdisk -l | grep /dev/vd | grep Linux | grep -v vda
/dev/vdc1               1       20805    10485688+  83  Linux
/dev/vdc2           20806       27046     3145464   83  Linux
/dev/vdd1               1       20805    10485688+  83  Linux
/dev/vde1               1       20805    10485688+  83  Linux
/dev/vdf1               1       20805    10485688+  83  Linux
[root@VM_63_126_centos ~]#
```

1. [log In to the Linux CVM] (https://intl.cloud.tencent.com/document/product/213/5436) as root user.
2. Run the following command to install mdadm.

```
yum install mdadm -y
```

success result is as shown below:

```
[root@VM_63_126_centos ~]# yum install mdadm -y
Loaded plugins: fastestmirror, security
Setting up Install Process
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package mdadm.x86_64 0:3.3.2-5.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch                   Version                 Repository            Size
================================================================================
Installing:
 mdadm                  x86_64                 3.3.2-5.el6             os                   345 k

Transaction Summary
================================================================================
Install       1 Package(s)

Total download size: 345 k
Installed size: 800 k
Downloading Packages:
mdadm-3.3.2-5.el6.x86_64.rpm                                | 345 kB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : mdadm-3.3.2-5.el6.x86_64                                    1/1
  Verifying  : mdadm-3.3.2-5.el6.x86_64                                    1/1

Installed:
  mdadm.x86_64 0:3.3.2-5.el6

Complete!
```

3. Run the following command to create RAID 0 using mdadm.

```
mdadm --create /dev/md0 --level=0 --raid-devices=4 /dev/vd[cdef]1
```

success result is as shown below:

```
[root@VM_63_126_centos ~]# mdadm --create /dev/md0 --level=0 --raid-devices=4 /dev/vd[cdef]1
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

4. Run the following command to create a file system using mkfs (take EXT3 file system as an example).

```
mkfs.ext3 /dev/md0
```

success result is as shown below:

```
[root@VM_63_126_centos ~]# mkfs.ext3 /dev/md0
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=128 blocks, Stripe width=512 blocks
2621440 inodes, 10477056 blocks
523852 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
320 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624

Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 24 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

5. Run the following commands in sequence to mount the file system.

```
mkdir md0/
mount /dev/md0 md0/
tree md0
```

success result is as shown below:

```
[root@VM_63_126_centos ~]# mount /dev/md0 md0/
[root@VM_63_126_centos ~]# tree md0
md0
`-- lost+found

1 directory, 0 files
```

6. Run the following command to view the file system details and record its UUID (take `c5d8a204:c28853ba:3882e9f8:62d078de` as an example) as shown below:

```
mdadm --detail --scan
```

```
[root@VM_163_144_centos ~]# mdadm --detail --scan
ARRAY /dev/md0 metadata=1.2 name=VM_163_144_centos:0 UUID=c5d8a204:c28853ba:3882e9f8:62d078de
```

7. Run the following command to edit mdadm configuration file.

```
vi /etc/mdadm.conf
```

8. In Edit mode, enter your configuration information.
   It is recommended to write the following configuration for elastic cloud disks:

```
DEVICE /dev/disk/by-id/virtio-ID of elastic cloud disk 1-part1
DEVICE /dev/disk/by-id/virtio-ID of elastic cloud disk 2-part1
DEVICE /dev/disk/by-id/virtio-ID of elastic cloud disk 3-part1
DEVICE /dev/disk/by-id/virtio-ID of elastic cloud disk 4-part1
ARRAY logical device path metadata= UUID=
```

   Assume the logical device path is /dev/md0 and metadata is 1.2, and you should enter:

```
DEVICE /dev/disk/by-id/virtio-ID of elastic cloud disk 1-part1
DEVICE /dev/disk/by-id/virtio-ID of elastic cloud disk 2-part1
DEVICE /dev/disk/by-id/virtio-ID of elastic cloud disk 3-part1
DEVICE /dev/disk/by-id/virtio-ID of elastic cloud disk 4-part1
ARRAY /dev/md0 metadata=1.2 UUID=3c2adec2:14cf1fa7:999c29c5:7d739349
```

9. Press Esc, enter `:wq` to save and exit.

Building LVM Logic Volumes with Multiple Elastic Cloud Disks

Last updated : 2020-05-12 15:28:37

Introduction to LVM

Logical Volume Manager (LVM) divides your disks or partitions into units of physical extents (PEs) with the same size by creating a logical layer over the disks and partitions. It works by combining different disks or partitions into the same volume group (VG), so that you can create logical volumes (LVs) within the VG, and file systems on the LVs.
Compared with using disk partitions directly, LVM supports scaling your file systems elastically.

- The file system is no longer limited by the size of physical disk. Instead, it can be distributed across multiple disks:
  For example, you can purchase 3 elastic cloud disks of 4 TB, and use LVM to create a massive file system up to 12 TB.
- You can resize the LVs dynamically instead of repartitioning your disks.
  When the LVM VG capacity cannot meet your needs, you can purchase a separate elastic cloud disk to mount on your CVM instance, and scale your VG by adding the cloud disk to it.

Building LVM

The following example uses 3 elastic cloud disks to create a dynamically resizable file system through LVM:

```
[root@VM_63_126_centos ~]# fdisk -l | grep vd | grep -v vda | grep -v vdb
Disk /dev/vdc: 10.7 GB, 10737418240 bytes
Disk /dev/vdd: 10.7 GB, 10737418240 bytes
Disk /dev/vde: 10.7 GB, 10737418240 bytes
```

Step 1 Create physical volume (PV)

1. log In to the Linux CVM as root user.
2. Run the following command to create PVs:

```
pvcreate <disk path 1> ... <disk path N>
```

Take `/dev/vdc`, `/dev/vdd` and `/dev/vde` as an example, then run:

```
pvcreate /dev/vdc /dev/vdd /dev/vde
```

The following figure shows the command output when the creation is successful:

```
[root@VM_63_126_centos ~]# pvcreate /dev/vdc /dev/vdd /dev/vde
  Physical volume "/dev/vdc" successfully created
  Physical volume "/dev/vdd" successfully created
  Physical volume "/dev/vde" successfully created
```

3. View physical volumes in the current system with the

```
lvmdiskscan | grep LVM
```

```
[root@VM_63_126_centos ~]# lvmdiskscan  | grep LVM
  /dev/vdc   [      10.00 GiB] LVM physical volume
  /dev/vdd   [      10.00 GiB] LVM physical volume
  /dev/vde   [      10.00 GiB] LVM physical volume
  3 LVM physical volume whole disks
  0 LVM physical volumes
```

Step 2 Create volume group (VG)

1. Run the following command to create a volume group:

```
vgcreate [-s <specified PE size>] <VG name> <PV path>.
```

Assume you want to create a VG named “lvm_demo0”, then run

```
vgcreate lvm_demo0 /dev/vdc /dev/vdd
```

The following figure shows the command output when the creation is successful:

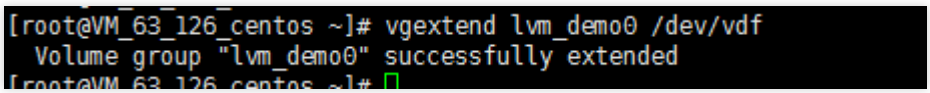
```
[root@VM_63_126_centos ~]# vgcreate lvm_demo0 /dev/vdc /dev/vdd
  Volume group "lvm_demo0" successfully created
```

When "Volume group" "successfully created" is displayed, your VG has been created successfully.

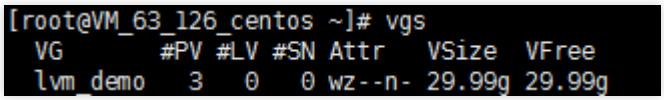
- Once created, you can add a new PV to the VG with the

```
vgextend VG name New PV path
```

The following figure shows the command output when the operation is successful:



- Once created, you can run `vgs`, `vgdisplay` or other commands to view the VG(s) in the current system, as shown below:



Step 3 Create logical volume (LV)

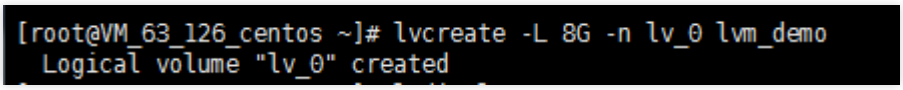
1. Run the following command to create a LV:

```
lvcreate [-L <LV size>][-n <LV name>] <VG name>.
```

Assume you want to create an 8 GB logical volume named "lv_0", then run

```
lvcreate -L 8G -n lv_0 lvm_demo0.
```

The following figure shows the command output when the creation is successful:



Run `pvs`, and you can see that the 8 GB comes solely from the `/dev/vdc` as shown below:

Step 4 Create and mount file system

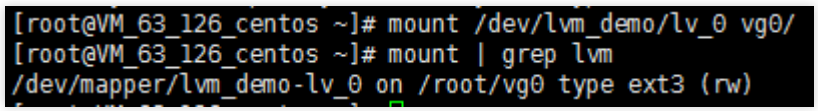
1. Create a file system on an existing LV using the command

```
mkfs.ext3 /dev/lvm_demo0/lv_0.
```

2. Mount the file system with the

```
mount /dev/lvm_demo0/lv_0 vg0/.
```

The following figure shows the command output when the mount is successful:



Step 5 Perform dynamic scaling on your logical volume and file system

LVs can be scaled dynamically only when the VG capacity is not fully occupied. When you scale a LV's capacity, you also need to scale the file system created on this LV.

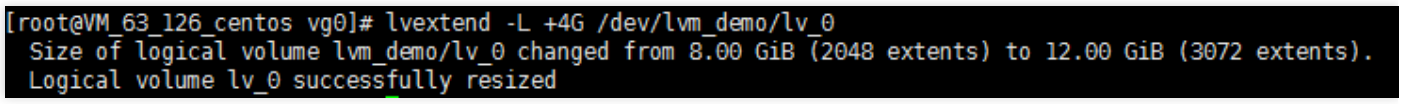
1. Scale a LV by running this command

```
lvextend [-L +/- <capacity amount>] <LV path>.
```

Assume you want to scale up the capacity of LV named "lv_0" by 4 GB, then run

```
lvextend -L +4G /dev/lvm_demo0/lv_0.
```

The following figure shows the command output when the scaling is successful:

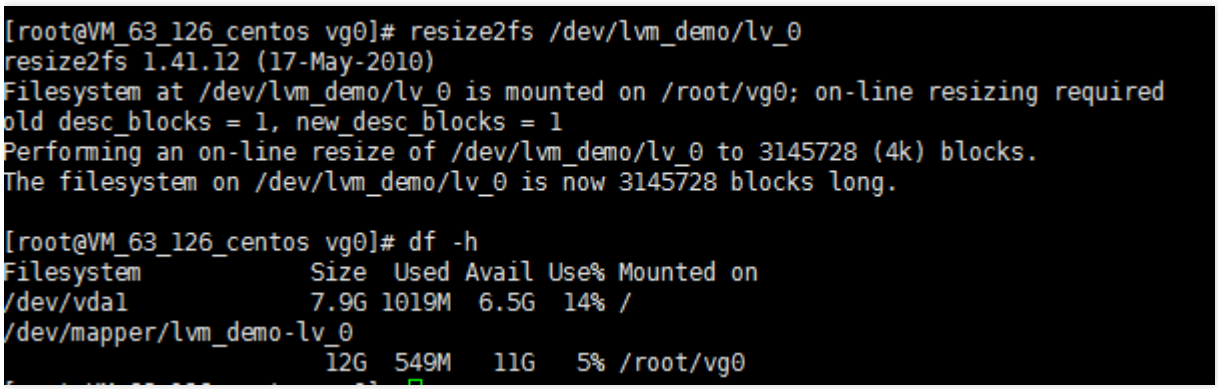


Run `pvs`, and you can find that `/dev/vdc` has been fully used, and 2 G has been used from `/dev/vdd` as shown below:

2. Scale the file system with the command

```
resize2fs /dev/lvm_demo0/lv_0.
```

The following figure shows the command output when the scaling is successful:



Once completed, you can run the following command to see if the capacity of your LV has been changed to 12 GB:

```
df -h
```