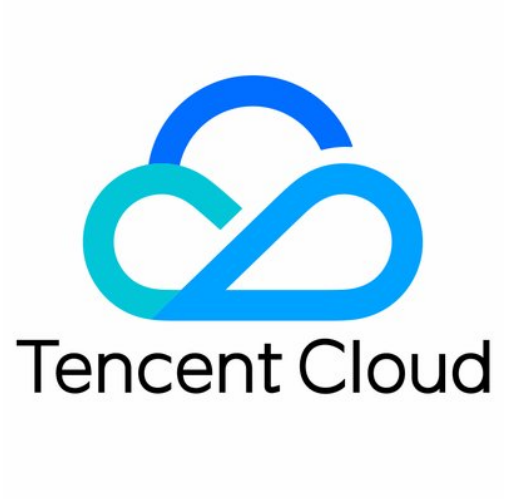
Cloud Block Storage

Cloud Hard Disk Performance

Product Introduction



Contents

Cloud Hard Disk Performance Building Up RAID Groups

Last updated：2017-11-24 11:58:50

RAID (Redundant Array of Independent Disks) combines multiple disks to form a disk array in order to improve data read and write performance and reliability. Meanwhile, the operating system will treat the disk array as a hard disk to use. RAID has a variety of grades at present. The following will introduce RAID0, RAID1, RAID01 and RAID10. Depending on the version of RAID, the disk array domains in enhancing data integration, enhancing fault tolerance, and increasing throughput or capacity compared with a large hard disk with considerable capacity.

The following is a comparison of different RAID versions:

RAID version	RAID0	RAID1	RAID01	RAID10
Features	Data is stored on different disks in segments. The size of virtual disk is the sum capacity of the disks in the array	The data is stored through image memory into disks. The size of virtual disk depends on the capacity of the disk with the smallest one in the array	First deal with data through RAID0, then RAID1	First deal with data through RAID1, then RAID0
Advantages	Read and write can be synchronized, thus the theoretical read and write rate can reach N times faster than a single disk (N is the number of disks in RAID0). But in fact, it is limited by file size, file system size and other factors	Damage to a single disk will not lead data irreversible, read fast	Take into both RAID0 and RAID1 advantages	
Disadvantages	No data redundancy. If a single disk is damaged, it is likely to cause all data lost in the most serious cases	Disk utilization rate is minimal and write speed is limited by that of a single disk	Costs are relatively high and it is essential to use at least 4 disks	
Recommended Using Scenario	Require a higher level of I/O performance, and has backed up data through other means or there is no need for data backup	Require a high level of read performance, and it is essential to back up the written data	RAID10 is recommended because RAID01 will cause disks in the same group	

			unavailable if a single disk is corrupted

The following describes how to use four Tencent Cloud elastic cloud disks to build RAID0 array. Linux kernel provides RAID device which is managed by md module in the bottom level. We can use mdadm tool to call md module.

```
[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 ~]#
```

Note: Please renew fees for elastic cloud disk about to expire in order to prevent the elastic cloud disk from being enforced isolation by the system, resulting in impacts on the RAID array.

Installing mdadm (take CentOS as an example)

```
[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!
```

Creating RAID0 with mdadm

```
[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.
```

Note: When creating RAID1, RAID01, and RAID10, it is best to create RAID with partitions of the same size to avoid wasting disk space.

Using mkfs to Create File System

```
[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.
```

Mounting File System

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

1 directory, 0 files
```

Modifying mdadm Configuration File

Determine UUID of the file system:

```
[root@VM_63_126_centos ~]# mdadm --detail --scan
ARRAY /dev/md0 metadata=1.2 name=VM_63_126_centos:0 UUID=3c2adec2:14cf1fa7:999c29c5:7d739349
```

Execute commands below to modify mdadm configuration files:

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

It is recommended to write the following configuration for elastic cloud disk:

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

In this case: ARRAY /dev/md0 metadata=1.2 UUID=3c2adec2:14cf1fa7:999c29c5:7d739349

Building Up LVM Logic Volumes

Last updated：2018-06-28 17:22:58

By creating a logical layer over the hard disk and partition, Logical Volume Manager (LVM) divides the disk or partition into physical extents (PE) with the same size. Different disks or partitions can be grouped into the same volume group (VG). A logical volume (LV) can be created on VG, and file system can be created on LV. The concept of VG can be simply linked with disk, and the concept of LV can be simply linked with partition. Compared with using the disk partition directly, LVM focus on adjusting the capacity of the file system elastically:

- The file system is no longer limited by the size of the physical disk. Instead, it can be distributed across multiple disks: For example, you can buy 3 elastic cloud disks with 4TB and use LVM to create an extra-large file system of nearly 12TB.
- You can dynamically adjust the size of LV instead of repartitioning the disk: When the LVM VG space cannot meet your needs, you can purchase an elastic cloud disk separately and mount it on the corresponding CVM, and then refer to the instructions below to add it to the LVM VG to expand the capacity.

  ....

The following describes how to use three Tencent Cloud elastic cloud disks to create a file system via LVM which its size can be adjusted dynamically.

```
[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
```

Creating Physical Volume (PV)

Execute the following command to create a physical volume (PV):

```
pvcreate disk path 1 ... disk path N
```

```
[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
```

Execute `pvscan`, `lvmdiskscan`, `pvs`, `pvdisplay physical volume path` and other commands to view the physical volumes in current system:

```
[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
```

Creating Volume Group (VG)

Execute the following commands to create a volume group (VG):

vgcreate [-s specifies PE size] volume **group** name physical volume path

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

After the creation is completed, you can add new physical volumes to the volume group with the `vgextend volume group name new physical volume path`

```
[root@VM_63_126_centos ~]# vgextend lvm_demo0 /dev/vdf
  Volume group "lvm_demo0" successfully extended
```

Use `vgs`, `vgdisplay` and other commands to view volume groups in the current system:

```
[root@VM_63_126_centos ~]# vgs
  VG       #PV #LV #SN Attr   VSize  VFree
  lvm_demo   3   0   0 wz--n- 29.99g 29.99g
```

Creating Logical Volume (LV)

After creating a large volume group, you can start building the logical volume (LV). Execute the following command to create a logical volume:

lvcreate [-L logical volume size][-n logical volume name] VG name

```
[root@VM_63_126_centos ~]# lvcreate -L 8G -n lv_0 lvm_demo
  Logical volume "lv_0" created
```

Here, we created an 8G logical volume named "lv_0".

We can find that only PE in vdc has been used by executing `pvs` command:

```
[root@VM_63_126_centos ~]# pvs
  PV         VG       Fmt  Attr PSize  PFree
  /dev/vdc   lvm_demo lvm2 a--  10.00g  2.00g
  /dev/vdd   lvm_demo lvm2 a--  10.00g 10.00g
  /dev/vde   lvm_demo lvm2 a--  10.00g 10.00g
```

Creating File System

Execute the following command to create a file system on an established logical volume:

```
mkfs
```

```
[root@VM_63_126_centos ~]# mkfs.ext3 /dev/lvm_demo/lv_0
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
524288 inodes, 2097152 blocks
104857 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2147483648
64 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

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

This filesystem will be automatically checked every 27 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Use the `mount` command to mount the file system:

```
[root@VM_63_126_centos ~]# mount /dev/lvm_demo/lv_0 vg0/
[root@VM_63_126_centos ~]# mount | grep lvm
/dev/mapper/lvm_demo-lv_0 on /root/vg0 type ext3 (rw)
```

Dynamically Expand the Size of Logical Volume and File System

If VG is left with surplus capacity, the LV capacity can be dynamically expanded. Execute the following command to expand the size of logical volume:

> lvextend [-L +/- increase **or** decrease capacity] logical volume path

```
[root@VM_63_126_centos vg0]# lvextend -L +4G /dev/lvm_demo/lv_0
  Size of logical volume lvm_demo/lv_0 changed from 8.00 GiB (2048 extents) to 12.00 GiB (3072 extents).
  Logical volume lv_0 successfully resized
```

Here, 4G capacity has been expanded for the logical volume named "lv_0".

We can find that vdc has been fully used and 2G has been used for vdd by executing `pvs` command:

```
[root@VM_63_126_centos vg0]# pvs
  PV         VG       Fmt  Attr PSize  PFree
  /dev/vdc   lvm_demo lvm2 a--  10.00g     0
  /dev/vdd   lvm_demo lvm2 a--  10.00g  7.99g
  /dev/vde   lvm_demo lvm2 a--  10.00g 10.00g
```

Now, we have only expanded the size of logical volume, and the file system should also be expanded according to the logical volume before using. Here, we can use `resize2fs` to expand the size of the file system:

```
[root@VM_63_126_centos vg0]# resize2fs /dev/lvm_demo/lv_0
resize2fs 1.41.12 (17-May-2010)
Filesystem at /dev/lvm_demo/lv_0 is mounted on /root/vg0; on-line resizing required
old desc_blocks = 1, new_desc_blocks = 1
Performing an on-line resize of /dev/lvm_demo/lv_0 to 3145728 (4k) blocks.
The filesystem on /dev/lvm_demo/lv_0 is now 3145728 blocks long.

[root@VM_63_126_centos vg0]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/vda1             7.9G 1019M  6.5G  14% /
/dev/mapper/lvm_demo-lv_0
                       12G  549M   11G   5% /root/vg0
```

Now, we can find that the size of lv_0 has been modified to 12G by executing `df` command.