云硬盘

最佳实践

产品文档

腾讯云

文档目录

最佳实践
如何衡量云硬盘的性能

最近更新时间：2018-09-20 15:23:23

腾讯云提供的块存储设备根据类型的不同拥有不同的性能和价格，具体内容可以参考云硬盘的分类，需要注意的是，由于不同应用程序的工作负载不同，若未提供足够的 I/O 请求来充分利用云硬盘时，可能无法达到云硬盘的最大性能。

云硬盘的性能如何衡量？一般使用以下几个指标对存储设备的性能进行描述：

- IOPS：每秒读/写次数，单位为次（计数）。存储设备的底层驱动类型决定了不同的 IOPS。
- 吞吐量：每秒的读写数据量，单位为 MB/s。
- 时延：IO操作的发送时间到接收确认所经过的时间，单位为秒。

FIO 是测试磁盘性能的一个非常好的工具，用来对硬件进行压力测试和验证。建议使用 libaio 的 I/O 引擎进行测试，请用户自行安装 FIO 和 Libaio。

请特别注意：
1. 请不要在系统盘上进行 fio 测试，避免损坏系统重要文件。
2. fio 测试建议在空闲的、未保存重要数据的硬盘上进行，并在测试完后重新制作文件系统。请不要在业务数据硬盘上测试，避免底层文件系统元数据损坏导致数据损坏。
3. 测试硬盘性能时，推荐直接测试裸数据盘（如 /dev/vdb，请确保此时数据盘上无数据）；测试文件系统性能时，推荐指定具体文件测试（如 /data/file）。
4. 若测试裸数据盘，将会破坏原磁盘的数据，如果原磁盘上有文件系统也将破坏。
5. 需确认 /etc/fstab 文件配置项中没有被测盘的挂载配置，否则将导致 CVM 启动失败。

不同场景的测试公式基本一致，只有 3 个参数（读写模式，iodepth，blocksize）的区别。下面举例说明使用 block size 为 4k，iodepth 为 1 来测试顺序读性能的命令。

命令如下：

```
fio --bs=4k --ioengine=libaio --iodepth=1 --direct=1 --rw=read --time_based --runtime=600 --refill_buffers --norandommap --randrepeat=0 --group_reporting --name=fio-read --size=100G --filename=/dev/sdb
```

每个工作负载适合的最佳 iodepth 不同，具体取决于您的特定应用程序对于 IOPS 和延迟的敏感程度。

参数说明：

参数名	参数值	说明
bs	4k	每次请求的块大小为4k，也可以为8k，16k等
ioengine	libaio	IO引擎配置，这里使用linux的异步IO引擎
iodepth	1	请求的IO队列深度
direct	1	direct写
rw	read	读写模式，包括顺序读，顺序写，随机读，随机写，随机读写，顺序读写
time_based	NA	该参数不需要值，只是fio运行基于时间来运行
runtime	600	fio运行时长
refill_buffers	NA	每次提交IO任务会重新填充
norandommap	NA	这个参数在测试随机IO时起作用，默认随机IO也会写入所有的SIZE里描述的块，加了之后就打破了这个限制，有些块可能无法被read/write到，有些则可能IO多次；能够更好的模拟用户场景
randrepeat	0	随机序列是否重复
group_reporting	NA	多个job并发时，打印整个group的统计值
name	fio-read	job的名称
size	100G	IO测试的寻址空间
filename	/dev/sdb	待测试的磁盘设备文件

常见用例如下：

- block=4k iodepth=1 随机读测试，能反映磁盘的时延性能；
- block=128K iodepth=32 能反映峰值吞吐性能；
- block=4k iodepth=32 能反映峰值 IOPS 性能。

下图为 SSD 云硬盘的测试性能截图：

```
[root@VM_62_196_centos fio-2.1.2]# ./fio --bs=4k --ioengine=libaio --iodepth=1 --direct=1 --rw=read --time_based
 --runtime=300 --refill_buffers --norandommap --randrepeat=0 --group_reporting --name=fio-write --size=600G --fi
lename=/dev/vdd
fio-write: (g=0): rw=read, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=1
fio-2.1.2
Starting 1 process
Jobs: 1 (f=1): [R] [100.0% done] [15984KB/0KB/0KB /s] [3996/0/0 iops] [eta 00m:00s]
fio-write: (groupid=0, jobs=1): err= 0: pid=9534: Mon Mar  7 10:09:29 2016
  read : io=4670.2MB, bw=15941KB/s, iops=3985, runt=300001msec
    slat (usec): min=5, max=1235, avg= 7.19, stdev= 1.98
    clat (usec): min=1, max=54940, avg=241.94, stdev=87.55
     lat (usec): min=150, max=54949, avg=249.45, stdev=87.54
    clat percentiles (usec):
     |  1.00th=[  223],  5.00th=[  235], 10.00th=[  237], 20.00th=[  239],
     | 30.00th=[  241], 40.00th=[  241], 50.00th=[  241], 60.00th=[  243],
     | 70.00th=[  243], 80.00th=[  243], 90.00th=[  245], 95.00th=[  249],
     | 99.00th=[  282], 99.50th=[  306], 99.90th=[  406], 99.95th=[  454],
     | 99.99th=[  692]
    bw (KB  /s): min=14208, max=16016, per=100.00%, avg=15944.65, stdev=152.91
    lat (usec) : 2=0.01%, 100=0.01%, 250=95.76%, 500=4.21%, 750=0.01%
    lat (usec) : 1000=0.01%
    lat (msec) : 2=0.01%, 4=0.01%, 10=0.01%, 20=0.01%, 50=0.01%
    lat (msec) : 100=0.01%
  cpu          : usr=0.90%, sys=3.96%, ctx=1197275, majf=0, minf=29
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=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=0.0%, 64=0.0%, >=64=0.0%
     issued    : total=r=1195563/w=0/d=0, short=r=0/w=0/d=0

Run status group 0 (all jobs):
   READ: io=4670.2MB, aggrb=15940KB/s, minb=15940KB/s, maxb=15940KB/s, mint=300001msec, maxt=300001msec

Disk stats (read/write):
  vdd: ios=1194763/0, merge=0/0, ticks=285299/0, in_queue=285004, util=95.08%
```

```
[root@VM_62_196_centos fio-2.1.2]# ./fio --bs=128k --ioengine=libaio --iodepth=32 --direct=1 --rw=randwrite --ti
me_based --runtime=300 --refill_buffers --norandommap --randrepeat=0 --group_reporting --name=fio-randwrite --si
ze=600G --filename=/dev/vdd
fio-randwrite: (g=0): rw=randwrite, bs=128K-128K/128K-128K/128K-128K, ioengine=libaio, iodepth=32
fio-2.1.2
Starting 1 process
Jobs: 1 (f=1): [w] [100.0% done] [0KB/257.4MB/0KB /s] [0/2059/0 iops] [eta 00m:00s]
fio-randwrite: (groupid=0, jobs=1): err= 0: pid=15412: Mon Mar  7 11:18:50 2016
  write: io=77897MB, bw=265882KB/s, iops=2077, runt=300006msec
    slat (usec): min=[redacted]g=15.36, stdev= 5.30
    clat (msec): min=2, max=185, avg=15.36, stdev=18.88
     lat (msec): min=2, max=185, avg=15.38, stdev=18.88
    clat percentiles (msec):
     |  1.00th=[    4],  5.00th=[    5], 10.00th=[    5], 20.00th=[    6],
     | 30.00th=[    7], 40.00th=[    8], 50.00th=[    9], 60.00th=[   11],
     | 70.00th=[   15], 80.00th=[   20], 90.00th=[   30], 95.00th=[   78],
     | 99.00th=[   95], 99.50th=[   97], 99.90th=[  128], 99.95th=[  143],
     | 99.99th=[  169]
    bw (KB  /s): min=73472, max=459264, per=100.00%, avg=266052.80, stdev=109188.76
    lat (msec) : 4=1.76%, 10=58.12%, 20=22.43%, 50=11.53%, 100=5.89%
    lat (msec) : 250=0.27%
  cpu          : usr=6.29%, sys=3.99%, ctx=394871, majf=0, minf=25
  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    : total=r=0/w=623173/d=0, short=r=0/w=0/d=0

Run status group 0 (all jobs):
  WRITE: io=77897MB, aggrb=265881KB/s, minb=265881KB/s, maxb=265881KB/s, mint=300006msec, maxt=300006msec

Disk stats (read/write):
  vdd: ios=0/622853, merge=0/0, ticks=0/9546305, in_queue=9547812, util=100.00%
```

```
[root@VM_62_196_centos fio-2.1.2]# ./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 --size=6
00G --filename=/dev/vdd
fio-randread: (g=0): rw=randread, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=32
fio-2.1.2
Starting 1 process
Jobs: 1 (f=1): [r] [100.0% done] [96000KB/0KB/0KB /s] [24.0K/0/0 iops] [eta 00m:00s]
fio-randread: (groupid=0, jobs=1): err= 0: pid=11278: Mon Mar  7 10:30:05 2016
  read : io=28106MB, bw=95934KB/s, iops=23983, runt=300002msec
    slat (usec): min=0, max=8086, [redacted]tdev=22.01
    clat (usec): min=0, max=143616, avg=1328.32, stdev=2594.76
     lat (usec): min=0, max=143620, avg=1333.21, stdev=2594.82
    clat percentiles (usec):
     |  1.00th=[    0],  5.00th=[  620], 10.00th=[  676], 20.00th=[  732],
     | 30.00th=[  780], 40.00th=[  812], 50.00th=[  852], 60.00th=[  884],
     | 70.00th=[  932], 80.00th=[  996], 90.00th=[ 1160], 95.00th=[ 3568],
     | 99.00th=[13632], 99.50th=[15168], 99.90th=[19328], 99.95th=[20864],
     | 99.99th=[80384]
    bw (KB  /s): min=47112, max=118216, per=100.00%, avg=95938.15, stdev=15209.47
    lat (usec) : 2=1.93%, 4=0.01%, 10=0.01%, 20=0.02%, 50=0.03%
    lat (usec) : 100=0.05%, 250=0.15%, 500=0.40%, 750=20.81%, 1000=56.78%
    lat (msec) : 2=13.00%, 4=2.09%, 10=2.47%, 20=2.18%, 50=0.03%
    lat (msec) : 100=0.04%, 250=0.01%
  cpu          : usr=3.64%, sys=17.43%, ctx=1473109, majf=0, minf=59
  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    : total=r=7195109/w=0/d=0, short=r=0/w=0/d=0

Run status group 0 (all jobs):
   READ: io=28106MB, aggrb=95934KB/s, minb=95934KB/s, maxb=95934KB/s, mint=300002msec, maxt=300002msec

Disk stats (read/write):
  vdd: ios=7195108/0, merge=0/0, ticks=9346310/0, in_queue=9344782, util=100.00%
```

多块弹性云盘构建RAID组

最近更新时间：2017-11-24 11:58:50

RAID（独立磁盘冗余阵列，Redundant Array of Independent Disks）可以将多个磁盘组合起来，构成一个磁盘阵列组，以提高数据的读写性能和可靠性。同时操作系统只会将磁盘阵列组当作一个硬盘来使用。目前RAID有多种等级，以下将介绍RAID0、RAID1、RAID01和RAID10。根据选择版本不同，磁盘阵列组相较于一块容量相当的大硬盘有增强数据集成度、增强容错功能、增加处理量或容量等优势。

以下是不同RAID版本的对比信息：

RAID等级	RAID0	RAID1	RAID01	RAID10
特点	数据分段存放在不同的磁盘中。虚拟盘大小为阵列中盘容量之和	数据被镜像存储在多个磁盘中。虚拟盘大小为阵列中容量最小的盘的容量	对数据先做RAID0，后做RAID1	对数据先做RAID1，后做RAID0
优点	读写都可以并行进行，因此理论的读写速率可以达到单个磁盘的N倍（N为组成RAID0的磁盘个数），但实际上受限于文件大小、文件系统大小等多种因素	单个磁盘的损坏不会导致数据的不可修复，读取速度快	兼顾RAID0和RAID1的优势	
缺点	没有数据冗余，单个磁盘损坏时，在最严重的情况下将有可能导致所有数据的丢失	磁盘利用率最低，写入速度受限于单个磁盘的写入速度	成本相对较高，需要使用至少4块盘	
建议使用场景	对I/O性能要求很高，并且已通过其他方式对数据进行了备份处理或者不需要进行数据备份的情况	对读性能要求较高，并且需要对写入的数据进行备份处理	推荐使用RAID10，因为如果发生单一磁盘的损坏，RAID01会导致同组的磁盘都不可用	

下面介绍如何使用4块腾讯云弹性云盘来构建RAID0阵列。Linux内核提供了md模块在底层管理RAID设备，我们可以使用mdadm工具来调用md模块。

```
[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 ~]#
```

注：请及时对将要到期的弹性云盘进行续费操作，以避免由于弹性云盘到期导致被系统强制隔离对RAID阵列产生影响。

安装mdadm（以CentOS为例）

```
[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!
```

使用mdadm创建RAID0

```
[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.
```

注：创建RAID1、RAID01、RAID10时最好使用相同大小的分区创建RAID，以避免对磁盘空间的浪费。

使用mkfs创建文件系统

```
[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.
```

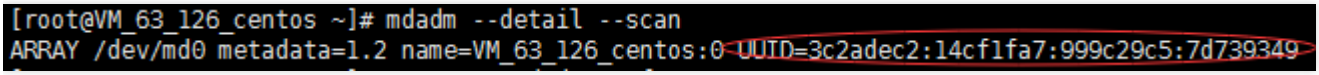
挂载文件系统

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

1 directory, 0 files
```

修改mdadm配置文件

确定文件系统UUID：



执行以下命令修改mdadm配置文件：

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

对于弹性云硬盘，建议写入以下配置：

```
DEVICE /dev/disk/by-id/virtio-弹性云盘1ID-part1
DEVICE /dev/disk/by-id/virtio-弹性云盘2ID-part1
DEVICE /dev/disk/by-id/virtio-弹性云盘3ID-part1
DEVICE /dev/disk/by-id/virtio-弹性云盘4ID-part1
ARRAY 逻辑设备路径 metadata= UUID=
```

本例为：ARRAY /dev/md0 metadata=1.2 UUID=3c2adec2:14cf1fa7:999c29c5:7d739349

多块弹性云盘构建LVM逻辑卷

最近更新时间：2018-06-28 17:22:58

LVM（Logical Volume Manager，逻辑卷管理）通过在硬盘和分区之上建立一个逻辑层，可以将磁盘或分区划分为相同大小的PE（Physical Extents）单元，不同的磁盘或分区可以划归到同一个卷组（VG，Volume Group），在VG上可以创建逻辑卷（LV，Logical Volume），在LV上可以创建文件系统。可以简单的把卷组与磁盘，逻辑卷与分区的概念对应起来。但是相对于直接使用磁盘分区的方式，LVM的重点在于弹性调整文件系统的容量：

- 文件系统不再受限于物理磁盘的大小，可以分布在多个磁盘上：比如您可以购买3个4TB的弹性云盘并使用LVM创建一个将近12TB的超大文件系统
- 可以动态调整逻辑卷大小，不需要重新对磁盘重新分区：当LVM卷组的空间无法满足您的需求时，您可以单独购买弹性云盘并将其挂载在相应的云服务器上，然后参考下边的指引将其添加到LVM卷组中进行扩容操作
  ....

下面介绍如何使用三块腾讯云弹性云硬盘通过LVM创建可以动态调整大小的文件系统。

```
[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
```

创建物理卷（PV）

执行以下命令创建一个物理卷：

```
pvcreate 磁盘路径1 ... 磁盘路径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
```

执行 `pvscan` 、 `lvmdiskscan` 、 `pvs` 、 `pvdisplay 物理卷路径` 等命令查看现在系统中的物理卷：

```
[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
```

创建卷组（VG）

执行以下命令创建卷组：

```
vgcreate [-s 指定PE大小] 卷组名 物理卷路径
```

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

创建完成后可以使用 `vgextend 卷组名 新物理卷路径` 来向卷组中添加新的物理卷：

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

使用 `vgs` 、 `vgdisplay` 等命令查看当前系统中的卷组：

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

创建逻辑卷（LV）

创建出大卷组后，接下来可以开始建立分割区（LV）了，执行以下命令创建逻辑卷：

```
lvcreate [-L 逻辑卷大小][ -n 逻辑卷名称] VG名称
```

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

这里创建了一个8G的名为“lv_0”的逻辑卷。

此时使用 pvs 命令可以发现只有vdc的PE被使用了：

```
[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
```

创建文件系统

执行以下命令在创建好的逻辑卷上创建文件系统：

```
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.
```

使用 mount 命令挂载该文件系统：

```
[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)
```

动态扩展逻辑卷及文件系统大小

当VG容量有剩余时，LV容量可动态扩展。执行以下命令扩展逻辑卷大小：

```
lvextend [-L +/- 增减容量] 逻辑卷路径
```

```
[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
```

这里对名为"lv_0"的逻辑卷扩展了4G大小的空间。

此时使用 pvs 命令可以发现vdc已被完全使用，vdd被使用了2G空间：

```
[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
```

此时只是扩展的逻辑卷的大小，在其之上的文件系统也要随之进行扩展才能使用，这里使用 resize2fs 来扩展文件系统大小：

```
[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
```

此时使用 df 命令可以看到lv_0的大小已被修改为12G了。