

云硬盘 最佳实践 产品文档



腾讯云

【版权声明】

©2013-2022 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

最佳实践

如何衡量云硬盘的性能

多块弹性云硬盘构建 LVM 逻辑卷

MBR 分区云硬盘扩容至大于 2TB

最佳实践

如何衡量云硬盘的性能

最近更新时间：2022-09-29 17:31:51

操作前须知（重要）

警告

- 本文使用 FIO 作为测试工具，**请不要在系统盘上进行 FIO 测试，避免损坏系统重要文件。**
- **为避免底层文件系统元数据损坏导致数据损坏，请不要在业务数据盘上进行测试。请使用测试机器未存放业务数据的云硬盘进行压测，并提前 [创建快照](#) 保障您的数据安全。**
- 请确保 `/etc/fstab` 文件配置项中**没有**被测硬盘的挂载配置，否则将导致云服务器启动失败。

衡量指标

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，详细信息请参考 [云硬盘类型](#)。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。

一般使用以下指标衡量云硬盘的性能：

- 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：I/O 操作的发送时间到接收确认所经过的时间，单位为微秒。

测试工具

FIO 是测试磁盘性能的工具，用来对硬件进行压力测试和验证，本文以 FIO 为例。

使用 FIO 时，建议配合使用 libaio 的 I/O 引擎进行测试。请参考 [工具安装](#) 完成 FIO 和 libaio 的安装。

测试对象建议

- 建议在空闲的、未保存重要数据的硬盘上进行 FIO 测试，并在测试完后重新制作被测硬盘的文件系统。
- 测试硬盘性能时，建议直接测试裸数据盘（如 `/dev/vdb`）。

- 测试文件系统性能时，推荐指定具体文件测试（如 /data/file）。

工具安装

1. 参考 [使用标准登录方式登录 Linux 实例（推荐）](#) 登录云服务器，本文以 CentOS 7.6 操作系统的云服务器为例。
2. 执行以下命令，查看云硬盘是否4KiB对齐。

```
fdisk -lu
```

如下图所示，若返回结果中的 **Start** 值能被8整除即是4KiB对齐。否则请完成4KiB对齐后再进行测试。

Device	Boot	Start	End	Blocks	Id	System
/dev/vdb1		2048	20971519	10484736	83	Linux

3. 依次执行以下命令，安装测试工具 FIO 和 libaio。

```
yum install libaio -y

yum install libaio-devel -y

yum install fio -y
```

安装完成后，请参考测试示例开始云硬盘性能测试。

测试示例

不同场景的测试公式基本一致，只有 **rw**、**iodepth** 和 **bs**（block size）三个参数的区别。例如，每个工作负载适合最佳 **iodepth** 不同，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	说明	取值样例
bs	每次请求的块大小。取值包括4k、8k及16k等。	4k
ioengine	I/O 引擎。推荐使用 Linux 的异步 I/O 引擎。	libaio
iodepth	请求的 I/O 队列深度。	1
direct	指定 direct 模式。	1

	<ul style="list-style-type: none">• True（1）表示指定 O_DIRECT 标识符，忽略 I/O 缓存，数据直写。• False（0）表示不指定 O_DIRECT 标识符。 <p>默认为 True（1）。</p>	
rw	读写模式。取值包括顺序读（read）、顺序写（write）、随机读（randread）、随机写（randwrite）、混合随机读写（randrw）和混合顺序读写（rw，readwrite）。	read
time_based	指定采用时间模式。无需设置该参数值，只要 FIO 基于时间来运行。	N/A
runtime	指定测试时长，即 FIO 运行时长。	600
refill_buffers	FIO 将在每次提交时重新填充 I/O 缓冲区。默认设置是仅在初始时填充并重用该数据。	N/A
norandommap	在进行随机 I/O 时，FIO 将覆盖文件的每个块。若给出此参数，则将选择新的偏移量而不查看 I/O 历史记录。	N/A
randrepeat	随机序列是否可重复，True（1）表示随机序列可重复，False（0）表示随机序列不可重复。默认为 True（1）。	0
group_reporting	多个 job 并发时，打印整个 group 的统计值。	N/A
name	job 的名称。	fio-read
size	I/O 测试的寻址空间。	100GB
filename	测试对象，即待测试的磁盘设备名称。	/dev/sdb

常见用例如下：

展开全部

bs = 4k iodepth = 1：随机读/写测试，能反映硬盘的时延性能

展开&收起

警告

- 请不要在系统盘上进行 FIO 测试，避免损坏系统重要文件。
- 为避免底层文件系统元数据损坏导致数据损坏，请不要在业务数据盘上进行测试。请使用测试机器未存放业务数据的云硬盘进行压测，并提前 [创建快照](#) 保障您的数据安全。

执行以下命令，测试硬盘的随机读时延。

```
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 -filename=/dev/vdb
```

执行以下命令，测试硬盘的随机写时延。

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

执行以下命令，测试 SSD 云硬盘的随机混合读写时延性能。

```
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 --filename=/dev/vdb
```

测试结果如下图所示：

```
fio-read: (g=0): rw=randrw, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=1
fio-3.1
Starting 1 process
Jobs: 1 (f=1): [m(1)][100.0%][r=3411KiB/s,w=3603KiB/s][r=852,w=900 IOPS][eta 00m:00s]
fio-read: (groupid=0, jobs=1): err= 0: pid=2377: Thu Jun 13 18:23:47 2019
  read: IOPS=880, BW=3523KiB/s (3607kB/s) (344MiB/100001msec)
    slat (nsec): min=2905, max=62479, avg=5254.61, stdev=2075.46
    clat (usec): min=205, max=6921, avg=463.65, stdev=259.48
      lat (usec): min=209, max=6925, avg=469.13, stdev=259.56
    clat percentiles (usec):
      | 1.00th=[ 245], 5.00th=[ 269], 10.00th=[ 293], 20.00th=[ 375],
      | 30.00th=[ 400], 40.00th=[ 416], 50.00th=[ 437], 60.00th=[ 457],
      | 70.00th=[ 478], 80.00th=[ 498], 90.00th=[ 545], 95.00th=[ 619],
      | 99.00th=[ 2057], 99.50th=[ 2376], 99.90th=[ 3294], 99.95th=[ 4015],
      | 99.99th=[ 6259]
    bw ( KiB/s): min= 2168, max= 4024, per=100.00%, avg=3522.64, stdev=310.95, samples=200
    iops        : min=  542, max= 1006, avg=880.66, stdev=77.74, samples=200
  write: IOPS=877, BW=3511KiB/s (3595kB/s) (343MiB/100001msec)
    slat (nsec): min=2981, max=58808, avg=5377.71, stdev=2079.36
    clat (usec): min=421, max=10492, avg=659.10, stdev=219.96
      lat (usec): min=428, max=10496, avg=664.70, stdev=220.05
    clat percentiles (usec):
      | 1.00th=[ 490], 5.00th=[ 523], 10.00th=[ 545], 20.00th=[ 562],
      | 30.00th=[ 578], 40.00th=[ 594], 50.00th=[ 611], 60.00th=[ 635],
      | 70.00th=[ 660], 80.00th=[ 693], 90.00th=[ 783], 95.00th=[ 914],
      | 99.00th=[ 1516], 99.50th=[ 1926], 99.90th=[ 3261], 99.95th=[ 3982],
      | 99.99th=[ 5342]
    bw ( KiB/s): min= 2296, max= 4008, per=100.00%, avg=3510.84, stdev=305.46, samples=200
    iops        : min=  574, max= 1002, avg=877.71, stdev=76.36, samples=200
  lat (usec)   : 250=0.76%, 500=40.51%, 750=51.04%, 1000=5.03%
  lat (msec)   : 2=1.90%, 4=0.71%, 10=0.05%, 20=0.01%
  cpu          : usr=0.50%, sys=1.52%, ctx=175841, 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%
```

bs = 128k iodepth = 32：顺序读/写测试，能反映硬盘的吞吐性能

展开&收起

警告

- 请不要在系统盘上进行 FIO 测试，避免损坏系统重要文件。
- 为避免底层文件系统元数据损坏导致数据损坏，请不要在业务数据盘上进行测试。请使用测试机器未存放业务数据的云硬盘进行压测，并提前 [创建快照](#) 保障您的数据安全。

执行以下命令，测试硬盘的顺序读吞吐带宽。

```
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 -filename=/dev/vdb
```


执行以下命令，测试硬盘的顺序写吞吐带宽。

```
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 -filename=/dev/vdb
```

执行以下命令，测试 SSD 云硬盘的顺序读吞吐性能。

```
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
```

测试结果如下图所示：

```
fio-rw: (g=0): rw=write, bs=(R) 128KiB-128KiB, (W) 128KiB-128KiB, (T) 128KiB-128KiB, ioengine=libaio, iodepth=32
fio-3.1
Starting 1 process
Jobs: 1 (f=1): [W(1)][100.0%][r=0KiB/s,w=260MiB/s][r=0,w=2082 IOPS][eta 00m:00s]
fio-rw: (groupid=0, jobs=1): err= 0: pid=2679: Thu Jun 13 18:27:32 2019
  write: IOPS=2081, BW=260MiB/s (273MB/s) (25.4GiB/100045msec)
    slat (nsec): min=2847, max=72524, avg=7739.21, stdev=3233.07
    clat (usec): min=1033, max=250494, avg=15341.09, stdev=28854.07
    lat (usec): min=1041, max=250503, avg=15349.03, stdev=28853.95
    clat percentiles (usec):
      | 1.00th=[ 1565], 5.00th=[ 1860], 10.00th=[ 2057], 20.00th=[ 2311],
      | 30.00th=[ 2540], 40.00th=[ 2769], 50.00th=[ 2999], 60.00th=[ 3326],
      | 70.00th=[ 3818], 80.00th=[ 5014], 90.00th=[82314], 95.00th=[84411],
      | 99.00th=[86508], 99.50th=[86508], 99.90th=[87557], 99.95th=[88605],
      | 99.99th=[90702]
    bw ( KiB/s): min=265708, max=319488, per=100.00%, avg=266498.36, stdev=3766.74, samples=200
    iops        : min= 2075, max= 2496, avg=2082.01, stdev=29.43, samples=200
    lat (msec)  : 2=8.46%, 4=64.09%, 10=11.90%, 20=0.18%, 50=0.01%
    lat (msec)  : 100=15.37%, 500=0.01%
    cpu         : usr=5.34%, sys=1.90%, ctx=63555, majf=0, minf=28
    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 rwts: total=0,208238,0, short=0,0,0, dropped=0,0,0
    latency     : target=0, window=0, percentile=100.00%, depth=32

Run status group 0 (all jobs):
  WRITE: bw=260MiB/s (273MB/s), 260MiB/s-260MiB/s (273MB/s-273MB/s), io=25.4GiB (27.3GB), run=100045-100045msec

Disk stats (read/write):
  vdb: ios=42/207998, merge=0/0, ticks=21/3173469, in queue=3174831, util=99.95%
```

bs = 4k iodepth = 32：随机读/写测试，能反映硬盘的 IOPS 性能

展开&收起

警告

- 请不要在系统盘上进行 FIO 测试，避免损坏系统重要文件。

- 为避免底层文件系统元数据损坏导致数据损坏，请不要在业务数据盘上进行测试。请使用测试机器未存放业务数据的云硬盘进行压测，并提前 [创建快照](#) 保障您的数据安全。

执行以下命令，测试硬盘的随机读 IOPS。

```
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
```

执行以下命令，测试硬盘的随机写 IOPS。

```
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
```

测试 SSD 云硬盘的随机读 IOPS 性能。如下图所示：

```
[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%
```

多块弹性云硬盘构建 LVM 逻辑卷

最近更新时间：2022-09-29 17:31:51

LVM 简介

逻辑卷管理（Logical Volume Manager，LVM）通过在硬盘和分区之上建立一个逻辑层，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。

相较于直接使用磁盘分区的方式，LVM 的优势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中。

例如，您可以购买3块4TB的弹性云硬盘并使用 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硬盘并挂载到相应的云服务器上，然后将其添加到 LVM 卷组中进行扩容操作。

构建 LVM

说明

本文以使用3块弹性云硬盘通过 LVM 创建可动态调整大小的文件系统为例。如下图所示：

```
[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
```

步骤 1 创建物理卷 PV

- 以 root 用户 [登录 Linux 云服务器](#)。
- 执行以下命令，创建一个物理卷（Physical Volume，PV）。

```
pvcreeate <磁盘路径1> ... <磁盘路径N>
```

本文以 `/dev/vdc`、`/dev/vdd` 和 `/dev/vde` 为例，则执行：

```
pvccreate /dev/vdc /dev/vdd /dev/vde
```

创建成功则如下图所示：

```
[root@VM_63_126_centos ~]# pvccreate /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. 执行以下命令，查看现在系统中的物理卷。

```
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
```

步骤 2 创建卷组 VG

1. 执行以下命令，创建 VG。

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

本文以创建一个名为“lvm_demo0”的卷组为例，则执行：

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

创建成功则如下图所示：

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

当提示“Volume group “<卷组名>” successfully created”时，表示卷组创建成功。

- 卷组创建完成后，可执行以下命令，向卷组中添加新的物理卷。

```
vgextend 卷组名 新物理卷路径
```

添加成功则如下图所示：

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

- 卷组创建完成后，可执行 `vgs`、`vgdisplay` 等命令查看当前系统中的卷组信息。如下图所示：

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

步骤 3 创建逻辑卷 LV

1. 执行以下命令，创建 LV。

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

本文以创建一个8GB的名为“lv_0”的逻辑卷为例，则执行：

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

创建成功则如下图所示：

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

说明

执行 `pvs` 命令，可查看到此时只有 `/dev/vdc` 被使用了8GB。如下图所示：

```
[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
```

步骤 4 创建并挂载文件系统

1. 执行以下命令，在创建好的逻辑卷上创建文件系统。

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

2. 执行以下命令，创建挂载节点目录 `/vg0`。

```
mkdir /vg0
```

3. 执行以下命令，挂载文件系统。

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

挂载成功则如下图所示：

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

步骤 5 动态扩展逻辑卷及文件系统大小

注意

仅当 VG 容量有剩余时，LV 容量可动态扩展。扩展 LV 容量后，需一并扩展创建在该 LV 上的文件系统的大小。

1. 执行以下命令，扩展逻辑卷大小。

```
lvextend [-L +/- <增减容量>] <逻辑卷路径>
```

本文以向逻辑卷“lv_0”扩展4GB容量为例，则执行：

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

扩展成功则如下图所示：

```
[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
```

说明

执行 `pvs` 命令，可查看到此时 `/dev/vdc` 已被完全使用，`/dev/vdd` 被使用了2GB。如下图所示：

```
[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
```

2. 执行以下命令，扩展文件系统。

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

扩展成功则如下图所示：

```
[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
```

扩展成功后，可执行以下命令，查看逻辑卷的容量是否变为12GB。

```
df -h
```


MBR 分区云硬盘扩容至大于 2TB

最近更新时间：2022-09-29 17:31:51

操作场景

当您的云硬盘在已有 MBR 分区并已创建文件系统的情况下，已扩容至大于2TB，此时 MBR 分区下的文件系统已无法扩容至大于2TB，请参考本文将 MBR 分区形式转换为 GPT 分区形式。

注意事项

- 转换分区表格式需替换原有分区，适当操作不会删除原有分区的数据，但需要先卸载原有分区，会一定程度影响线上业务运行。
- 请谨慎操作，误操作可能会导致数据丢失或异常。请给对应云硬盘创建快照，完成数据备份。详情请参见 [创建快照](#)。如出现误操作导致数据丢失，则可回滚快照进行数据恢复。

操作步骤

- 登录云服务器，详情请参见 [使用标准登录方式登录 Linux 实例（推荐）](#)。
- 执行以下命令，查看分区类型是否为 MBR。

```
fdisk -l
```

若结果如下图所示（根据操作系统不同略有不同），则说明为 MBR 分区形式。

```
[root@VM-0-3-centos ~]# fdisk -l
Disk /dev/vda: 50 GiB, 53687091200 bytes, 104857600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe609e297

Device            Boot Start          End  Sectors  Size Id Type
/dev/vda1          2048 104857566 104855519   50G 83 Linux

Disk /dev/vdb: 2 TiB, 2147483648000 bytes, 4194304000 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x048787f2

Device            Boot Start          End  Sectors  Size Id Type
/dev/vdb1          2048 104857599 104855552   50G 83 Linux
```

3. 执行以下命令，卸载分区。

```
umount <挂载点>
```

本文挂载点以 /data 为例，则执行：

```
umount /data
```

4. 执行以下命令，查看分区的卸载结果。

```
lsblk
```

若原分区 MOUNTPOINT 已显示为空，则表明卸载任务成功。本文以 /dev/vdb1 分区为例，返回结果如下图所示

```
[root@VM-0-3-centos ~]# lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0         11:0    1 184.1M  0 rom
vda         253:0    0    50G  0 disk
└─vda1      253:1    0    50G  0 part /
vdb         253:16   0     2T  0 disk
└─vdb1      253:17   0    50G  0 part
```

5. 执行以下命令，进入 parted 分区工具。

```
parted <磁盘路径>
```

本文以磁盘路径以 `/dev/vdb` 为例，则执行：

```
parted /dev/vdb
```

6. 输入 `p` 并按 **Enter**，查看当前分区信息。回显信息类似如下图：

```
[root@VM-5-94-centos ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 2201GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number   Start    End      Size    Type     File system  Flags
  1       1049kB   53.7GB   53.7GB   primary  ext4
```

7. 输入 `rm 分区号` 并按 **Enter**，删除待替换的末尾分区。

本文示例中仅有1个分区，可输入 `rm 1` 并按 **Enter**，删除分区1。

8. 输入 `p` 并按 **Enter**，查看当前分区信息，观察末尾分区是否已删除。

9. 输入 `mklabel GPT` 并按 **Enter**，使用 GPT 分区形式重新划分分区。

10. 在确认提示后输入 `Yes`，按 **Enter**。如下图所示：

```
(parted) rm 1
(parted) mklabel GPT
Warning: The existing disk label on /dev/vdb will be destroyed and all data on this disk will be lost. Do you want to continue?
Yes/No? Yes
```

1. 输入 `mkpart primary 2048s 100%`，按 **Enter**，创建分区。

其中，`2048s` 表示硬盘初始容量，`100%` 表示磁盘截止容量，此处仅供参考，您可以根据业务需要自行规划磁盘分区数量及容量。

注意

以下情况可能导致数据丢失：

- 选择的初始容量与原分区不一致。
- 选择的截止容量小于扩容前的原分区容量的值。

2. 输入 `p` 并按 **Enter**，查看新分区是否替换成功。回显信息类似如下图所示，则表示已成功替换：

```
(parted) mkpart primary 2048s 100%
(parted) p
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 2201GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
  1      1049kB  2201GB  2201GB                primary
```

3. 输入 `q` 并按 **Enter**，退出 parted 分区工具。

4. 执行以下命令，挂载分区。

```
mount <分区路径> <挂载点>
```

本文以分区路径以 `/dev/vdb1`，挂载点 `/data` 为例，则执行：

```
mount /dev/vdb1 /data
```

5. 执行对应命令，扩容文件系统。

- 扩容 ext 文件系统
- 扩容 xfs 文件系统

执行以下命令，扩容 ext 文件系统。

```
resize2fs /dev/对应分区
```

本文以分区路径以 `/dev/vdb1`，则执行：

```
resize2fs /dev/vdb1
```

6. 参考 [设置开机自动挂载](#) 操作，设置分区自动挂载。

至此，已完成 MBR 分区转 GPT 分区配置，您可执行 `df -h` 命令查看分区信息。