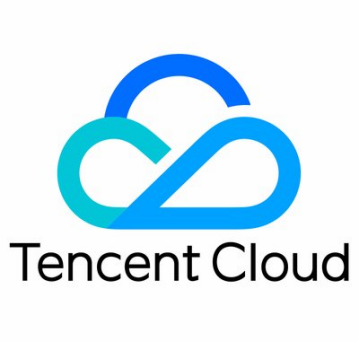
Cloud Block Storage
Best Practice



Service Notice

This document provides an overview of the as-is details of Tencent Cloud's products and services in their entirety or part. The descriptions of certain products and services may be subject to adjustments from time to time.

The commercial contract concluded by you and Tencent Cloud will provide the specific types of Tencent Cloud products and services you purchase and the service standards. Unless otherwise agreed upon by both parties, Tencent Cloud does not make any explicit or implied commitments or warranties regarding the content of this document.

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

Best Practice
Measuring Cloud Disk Performance

Last updated: 2023-09-19 21:34:12

Important Notes

Warning

- This document uses FIO as the testing tool, **please refrain from conducting FIO tests on the system disk to prevent damage to crucial system files.**
- **To prevent data corruption due to damage to the underlying file system metadata, please refrain from conducting tests on the business data disk. Use a cloud disk on a test machine that does not store business data for stress testing, and ensure to create a snapshot in advance to safeguard your data.**
- Please ensure that there are **no** mount configurations for the disk under test in the `/etc/fstab` file configuration items, as this could lead to a failure in starting the cloud server.

Metrics

The block storage devices provided by Tencent Cloud have different performances and prices depending on their type. For detailed information, please refer to Cloud Disk Types. As different applications have different workloads, the maximum performance of a cloud disk may not be achieved if sufficient I/O requests are not provided to fully utilize the cloud disk.
The performance of cloud disks is generally measured using the following metrics:

- IOPS: The number of read/write operations per second, measured in counts. The underlying driver type of the storage device determines the different IOPS.
- Throughput: read/written data volume per second, in MB/s.
- Latency: Time from I/O operation sending to receiving, in microseconds.

Test Tool

FIO is a tool for testing disk performance, used for stress testing and verification of hardware. This document uses FIO as an example. When using FIO, it is recommended to use the libaio I/O engine for testing. Please refer to Tool Installation to complete the installation of FIO and libaio.

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 that you specify the specific file (such as /data/file) for testing.

Tool Installation

1. Refer to Logging in to Linux Instance Using Standard Login Method (Recommended) to log in to the CVM instance. This section uses a CVM instance running the CentOS 7.6 operating system as an example.
2. Run the following command to check whether the cloud disk is 4KiB-aligned.

```
fdisk -lu
```

As depicted below, if the Start value in the returned results is divisible by 8, it indicates that the disk is 4 KiB-aligned. If not, please ensure the disk is 4 KiB-aligned before proceeding with the test.

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

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
```

Upon completion of the installation, please refer to the test examples to commence the performance testing of the cloud disk.

Test Example

The test formulas for different scenarios are essentially the same, with only the rw, iodepth, and bs (block size) parameters differing. For instance, each workload is suited to a different optimal iodepth, depending on your specific application's sensitivity to IOPS and latency.
Parameters:

Parameter	Note	Sample Value
bs	The block size for each request. Values include 4k, 8k, and 16k, among others.	4k
ioengine	I/O Engine. It is recommended to use the asynchronous I/O engine of Linux.	libaio
iodepth	Queue depth of an I/O request.	1
direct	Specify the direct mode. • True (1) denotes the specification of the O_DIRECT identifier, bypassing the I/O cache for direct data writing. • False (0) indicates that the O_DIRECT identifier is not specified. Default is True (1).	1
read	Read/Write modes. The values include sequential read (read), sequential write (write), random read (randread), random write (randwrite), mixed random read/write (randrw), and mixed sequential read/write (rw, readwrite).	read
time_based	Specify the use of time mode. There is no need to set this parameter value as long as FIO operates based on time.	N/A
runtime	Specifies the test duration, which is the FIO runtime.	100
refill_buffers	FIO will refill the I/O buffer with each submission. The default setting is to only populate and reuse the data initially.	N/A
norandommap	During random I/O, FIO will overwrite each block of the file. If this parameter is provided, a new offset will be chosen without reviewing the I/O history.	N/A
randrepeat	Determines whether the random sequence can be repeated. True (1) indicates that the random sequence can be repeated, while False (0) signifies that the random sequence cannot be repeated. The default 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.	1G
filename	Test object, which is the name of the disk to be tested.	/dev/sdb
numjobs	The number of concurrent threads is set to 1 by default. When the performance of the disk under test is high, it is recommended to increase the numjobs count (such as 2 or 4) to intensify the load.	1

Common use cases are as follows:

bs = 4k iodepth = 1: Random read/write test, which can reflect the latency performance of the disk

```
fio -bs=4k -ioengine=libaio -iodepth=1 -numjobs=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
```

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

> **Note:**
> Ensure to create a snapshot before executing the command.

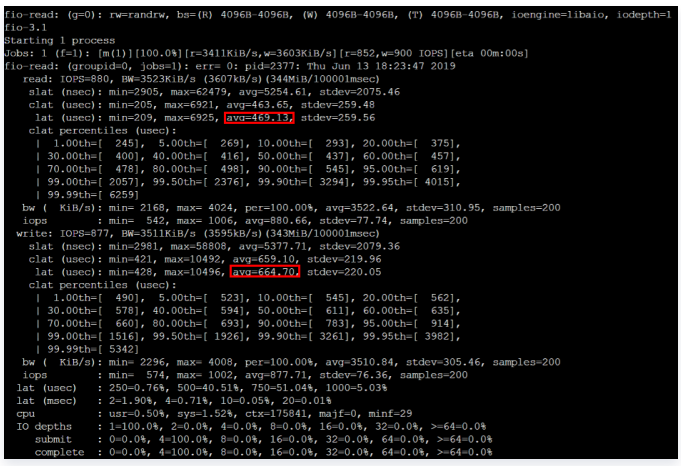
```
fio -bs=4k -ioengine=libaio -iodepth=1 -numjobs=1 -direct=1 -rw=randwrite -time_based -runtime=600 -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-randwrite-lat -size=10G -filename=/dev/vdb
```

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

> **Note:**
> Ensure to create a snapshot before executing the command.

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

The following figure shows the command output:



bs = 128k iodepth = 32: Sequential read/write test, which can reflect the throughput performance of the disk

> **Reminder:**
> - **Please refrain from conducting FIO tests on the system disk to prevent damage to crucial system files.**
> - **To prevent data corruption due to damage to the underlying file system metadata, please refrain from conducting tests on the business data disk. Use a cloud disk on a test machine that does not store business data for stress testing, and ensure to create a snapshot in advance to safeguard your data.**

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

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

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

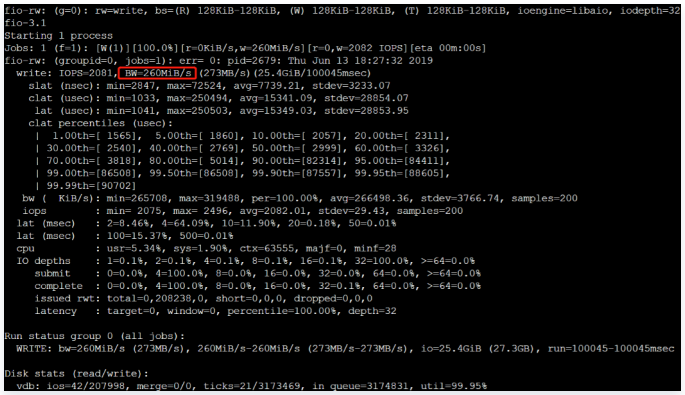
> **Note:**
> Ensure to create a snapshot before executing the command.

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

The following figure shows the command output:



bs = 4k iodepth = 32: Random read/write test, which can reflect the IOPS performance of the disk

> **Reminder:**
> - **Please refrain from conducting FIO tests on the system disk to prevent damage to crucial system files.**
> - **To prevent data corruption due to damage to the underlying file system metadata, please refrain from conducting tests on the business data disk. Use a cloud disk on a test machine that does not store business data for stress testing, and ensure to create a snapshot in advance to safeguard your data.**

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

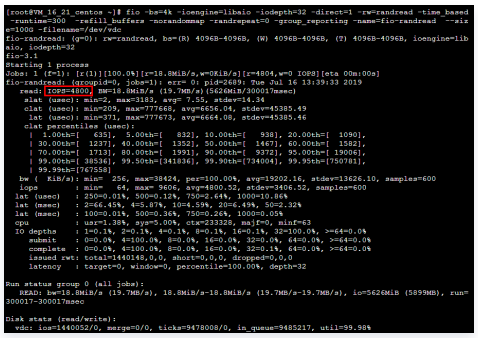
```
fio -bs=4k -ioengine=libaio -iodepth=32 -numjobs=4 -direct=1 -rw=randread -time_based -runtime=600 -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-randread-iops -size=10G -filename=/dev/vdb
```

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

> **Note:**
> Ensure to create a snapshot before executing the command.

```
fio -bs=4k -ioengine=libaio -iodepth=32 -numjobs=4 -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 an SSD cloud disk, as illustrated below:



Building LVM Logic Volumes with Multiple Elastic Cloud Disks

Last updated: 2023-09-19 21:38:15

Introduction to LVM

Logical Volume Manager (LVM) creates a logical layer over your disks or partitions to divide them into physical extent (PE) units with the same size. This categorizes disks or partitions into a volume group (VG), on which you can create a logical volume (LV), and then create file systems on the LV.
Different from direct disk partitioning, LVM allows elastic scaling of file system.

- The file system is not limited by the size of a physical disk. Instead, it can be distributed among 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 elastic cloud disk, attach it to your CVM instance, and add it to the LVM VG to expand capacity.

Building LVM

Note
This article illustrates how to create a dynamically resizable file system using LVM with three elastic cloud disks, as shown in the following figure:

```
[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 a physical volume (PV)

1. Log in to the Linux CVM as the root user. See Logging in to Linux CVM.
2. Run the following command to create a PV.

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

Taking `/dev/vdc`, `/dev/vdd`, and `/dev/vde` as examples, execute the following:

```
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. Run the following command to view physical volumes of the system.

```
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 a volume group (VG)

1. Run the following command to create a VG.

```
vgcreate [-s <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 the prompt "Volume group “<Volume Group Name>” successfully created" appears, it indicates that the volume group has been created successfully.

- Then you can run the following commands to add a new PV to the VG.

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

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

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

- After the volume group is created, you can run commands such as `vgs` and `vgdisplay` to view the volume group information in the current system, as shown below:

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

Step 3: Create a logical volume (LV)

1. Run the following command to create a LV.

   ```
   lvcreate [-L <size of logical volume>][ -n <name of logical volume>] <name of VG>
   ```

   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:

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

   Note

   Run the `pvs` command to see that only `/dev/vdc` has used 8GB, as shown below:

   ```
   [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
   ```

Step 4: create and mount a file system

1. Run the following command to create a file system on an existing LV.

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

2. Run the following command to create the mount node directory `/vg0` .

```
mkdir /vg0
```

3. Run the following command to mount the file system.

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

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

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

Step 5: resize the logical volume and file system dynamically

> **Note**
> The LV capacity can only be dynamically expanded when there is remaining capacity in the VG. After expanding the LV capacity, the size of the file system created on that LV must also be expanded.

1. Run the following command to extend the LV.

```
lvextend [-L +/- <Increase/Decrease Capacity>] <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:

```
[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
```

> **Note**
> Run the `pvs` command to see that `/dev/vdc` is now fully utilized and `/dev/vdd` has used 2GB, as shown below:
>
> ```
> [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. Run the following command to extend the file system.

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

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

```
[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
```

After the extension, run the following command to check whether the LV capacity is 12 GB.

```
df -h
```

Expanding MBR Cloud Disks to Greater Than 2 TB

Last updated: 2023-09-19 21:44:14

Scenario

When your cloud disk has an MBR partition with a created file system and has been expanded to greater than 2 TB, the file system cannot be expanded to greater than 2 TB. This document describes how to convert the MBR partition to the GPT partition to implement the expansion.

Supports and Limits

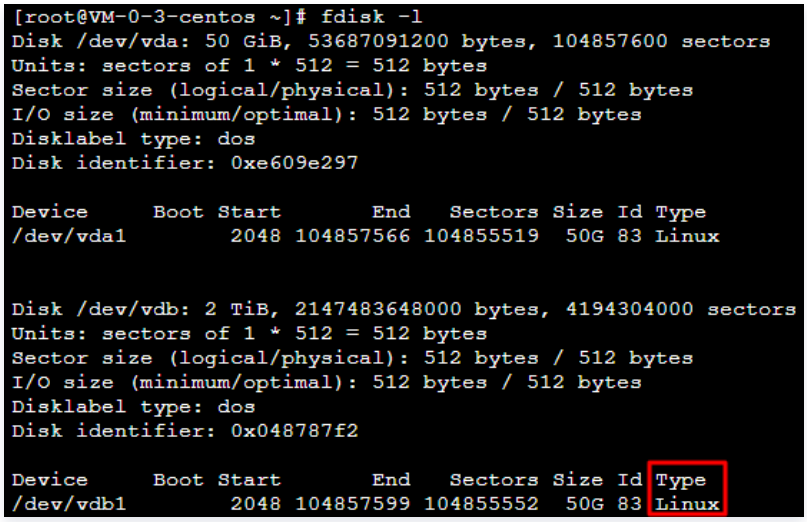
- To convert the partition format, you need to replace the original partition. The original partition data will not be deleted in normal cases. However, as the original partition needs to be unmounted, online businesses will be affected.
- Please proceed with caution as any misoperation may lead to data loss or anomalies. It is advised to create a snapshot for the corresponding cloud disk to back up the data. For more information, see Creating Snapshots. If data loss occurs due to misoperation, you can roll back the snapshot to recover the data.

Instructions

1. Log in to the cloud server. For more information, see Logging in to Linux Instance Using Standard Login Method (Recommended).
2. Run the following command to check whether the partition format is MBR.

```
fdisk -l
```

If the following result is shown (which may vary by operation system), the partition format is MBR.



3. Run the following command to unmount the partition.

```
umount <mount point>
```

Taking the `/data` mount point as an example, run the following command:

```
umount /data
```

4. Run the following command to view the unmount result.

```
lsblk
```

If the original partition MOUNTPOINT is displayed as empty, it indicates that the unmounting task has been successful. This document uses the `/dev/vdb1` partition as an example, and the result is shown in the following figure .

```
[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. Run the following command to enter the parted partition tool:

```
parted <disk path>
```

This guide uses the disk path `/dev/vdb` as an example. Proceed as follows:

```
parted /dev/vdb
```

6. Enter `p` and press **Enter** to view the current partition information. The returned information should resemble the following:

```
[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. Enter `rm partition number` and press **Enter** to delete the last partition to be replaced. In this example, there is only one partition, so you can enter `rm 1` and press **Enter** to delete partition 1.
8. Enter `p` and press **Enter** to view the current partition information and check whether the last partition has been deleted.
9. Enter `mklabel GPT` and press **Enter** to repartition using the GPT partition format.
10. After confirming the prompt, type `Yes` and press **Enter**, as shown in the following figure:

```
(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
```

11. Enter `mkpart primary 2048s 100%` and press **Enter** to create a partition. Here, `2048s` represents the initial disk capacity, and `100%` represents the end capacity of the disk. This is for reference only, and you can plan the number and capacity of disk partitions according to your business needs.

> **Note**
> Data may be lost in the following cases:
> - The configured initial capacity differs from the original partition capacity.
> - The configured maximum capacity is smaller than the original partition capacity before the expansion.

12. Enter `p` and press **Enter** to check if the new partition has been successfully replaced. If the returned information is similar to the following figure, it indicates that the replacement has been successful:

```
(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
```

13. Enter `q` and press **Enter** to exit the parted partition tool.

14. Execute the following command to mount the partition.

```
mount <partition path> <mount point>
```

This document uses the partition path `/dev/vdb1` and the mount point `/data` as examples. Execute the following command:

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

15. Execute the corresponding command to extend the file system.

Expanding the EXT File System

Run the following command to extend the EXT file system.

```
resize2fs /dev/corresponding partition
```

Assuming the partition path is `/dev/vdb1`, execute the following:

```
resize2fs /dev/vdb1
```

Extending the XFS File System

Run the following command to extend the XFS file system.

```
xfs_growfs /dev/corresponding partition
```

Assuming the partition path is `/dev/vdb1`, execute the following:

```
xfs_growfs /dev/vdb1
```

16. Refer to Setting Up Automatic Mounting at Startup to set up automatic partition mounting.

At this point, the configuration to convert MBR partition to GPT partition has been completed. You can run the `df -h` command to view the partition information.