

目 录

1 KVM 扩容内存、CPU、磁盘	2
1.1 查看虚拟机状态	2
1.2 KVM 扩容内存	2
1.2.1 设置当前使用内存 “currentMemory”	2
1.2.2 设置最大可使用内存 “memory”	3
1.3 KVM 调整 CPU 数量	4
1.3.1 调整 CPU 数量需要知道的事	6
1.3.2 使用 qemu-agent 动态调整 CPU 数量	7
1.4 KVM 扩展磁盘	9
1.4.1 添加 qcow2 磁盘加入虚拟机	9
1.4.2 直接扩展 qcow2 磁盘	11

1 KVM 扩容内存、CPU、磁盘

1.1 查看虚拟机状态

使用 “dominfo <domain-name>” 可以查看虚拟机的状态，其中就可以看到虚拟机当前的内存，如下图所示：

```
virsh # dominfo devstack
Id: 1
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: 关闭
CPU: 4
最大内存: 1048576 KiB
使用的内存: 1048576 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
```

1.2 KVM 扩容内存

1.2.1 设置当前使用内存 “currentMemory”

KVM 只能在虚拟机的运行状态下扩展虚拟机的当前可使用内存，命令如下：

```
1 virsh setmem <domain-name> <count>
```

使用例子如下图所示：

```
virsh # dominfo devstack
Id: 2
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: running
CPU: 4
CPU 时间: 25.8s
最大内存: 1048576 KiB
使用的内存: 1048576 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
安全性标签: libvirt-c0f5a648-02e6-4514-936d-fd2ca4ae27dd (enforcing)

virsh # setmem devstack 800M

virsh # dominfo devstack
Id: 2
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: running
CPU: 4
CPU 时间: 26.1s
最大内存: 1048576 KiB
使用的内存: 819200 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
安全性标签: libvirt-c0f5a648-02e6-4514-936d-fd2ca4ae27dd (enforcing)
```

如果想永久性地更改虚拟机的“currentMemory”，命令如下：

```
1 virsh setmem <domain-name> <count> --config
```

需要注意的是，虚拟机还有另一个属性，就是最大可使用的内存“memory”，而当前可使用内存是“currentMemory”。这里 setmem 命令是设置“currentMemory”，而“currentMemory”的值不能大于“memory”，否则会出错，如下图所示：

```
virsh # setmem devstack 2G
错误： 无效参数：无法将内存设置为高于最大内存
```

1.2.2 设置最大可使用内存“memory”

KVM 只能在虚拟机的关机状态下调整虚拟机的最大可使用内存，命令如下：

```
1 virsh setmaxmem <domain-name> <count>
```

使用例子如下图所示：

```
virsh # dominfo devstack
Id: -
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: 关闭
CPU: 4
最大内存: 1048576 KiB
使用的内存: 1048576 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0

virsh # setmaxmem devstack 2G

virsh # dominfo devstack
Id: -
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: 关闭
CPU: 4
最大内存: 2097152 KiB
使用的内存: 1048576 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
```

需要注意的是，必须在虚拟机的关机状态下调整虚拟机的最大可使用内存，否则会报错，如下图所示：

```
virsh # setmaxmem devstack 2G
错误： 不能改变最大内存大小
错误： 所需操作无效：无法在活跃的域中创新定义内存最大值
```

还有一点需要知道, 如果调整“memory”小于“currentMemory”, 那么“currentMemory”也会随之改变, 如下图所示:

```
virsh # setmaxmem devstack 500M
virsh # dominfo devstack
Id: -
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: 关闭
CPU: 4
最大内存: 512000 KiB
使用的内存: 512000 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
```

1.3 KVM 调整 CPU 数量

KVM 可以设置虚拟机最多可以使用的 CPU 数量, 命令如下:

```
1 virsh setvcpus <domain-name> --maximum <count> --config
```

设置这个值以后, 需要重启虚拟机才能生效, 命令如下:

```
1 destroy <domain-name>
2 start <domain-name>
```

KVM 只能在虚拟机的运行状态下调整虚拟机的 CPU 数量, 命令如下:

```
1 virsh setvcpus <domain-name> <count>
```

使用例子如下图所示:

```
virsh # dominfo devstack
Id: -
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: 关闭
CPU: 4
最大内存: 4194304 KiB
使用的内存: 512000 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0

virsh # start devstack
域 devstack 已开始

virsh # setvcpus devstack 8

virsh # dominfo devstack
Id: 6
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: running
CPU: 8
CPU 时间: 23.3s
最大内存: 4194304 KiB
使用的内存: 512000 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
安全性标签: libvirt-c0f5a648-02e6-4514-936d-fd2ca4ae27dd (enforcing)
```

如果想永久性地更改虚拟机的 CPU 数量，命令如下：

```
1 virsh setvcpus <domain-name> <count> --config
```

需要注意的是，CPU 个数只能调大，不能调小，如下图所示：

```
virsh # dominfo devstack
Id: 6
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: running
CPU: 8
CPU 时间: 23.3s
最大内存: 4194304 KiB
使用的内存: 512000 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
安全性标签: libvirt-c0f5a648-02e6-4514-936d-fd2ca4ae27dd (enforcing)

virsh # setvcpus devstack 4
错误: 内部错误: 无法更改这个域的 vcpu 计数
```

如果想减小 CPU 的个数，可以使用 “setvcpus <domain-name> --maximum <count> --config” 命令，然后重启虚拟机，如下图所示：

```
virsh # setvcpus devstack 4

virsh # setvcpus devstack --maximum 2 --config

virsh # dominfo devstack
Id: 7
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: running
CPU: 4
CPU 时间: 14.6s
最大内存: 4194304 KiB
使用的内存: 512000 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
安全性标签: libvirt-c0f5a648-02e6-4514-936d-fd2ca4ae27dd (enforcing)

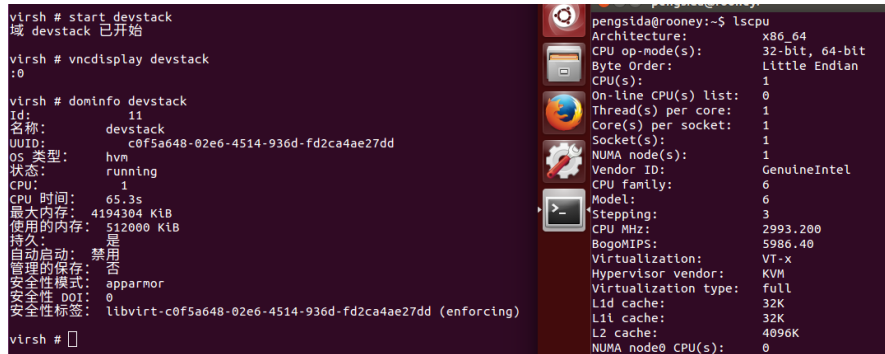
virsh # destroy devstack
域 devstack 被删除

virsh # dominfo devstack
Id: 7
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: 关闭
CPU: 2
最大内存: 4194304 KiB
使用的内存: 512000 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
```

1.3.1 调整 CPU 数量需要知道的事

虽然 KVM 可以动态地增加虚拟机 CPU 的数量，但是虚拟机增加的 CPU 其实是处于“offline”状态。

举个例子，虚拟机 CPU 本身状态如下图所示：



```

virsh # start devstack
域 devstack 已开始

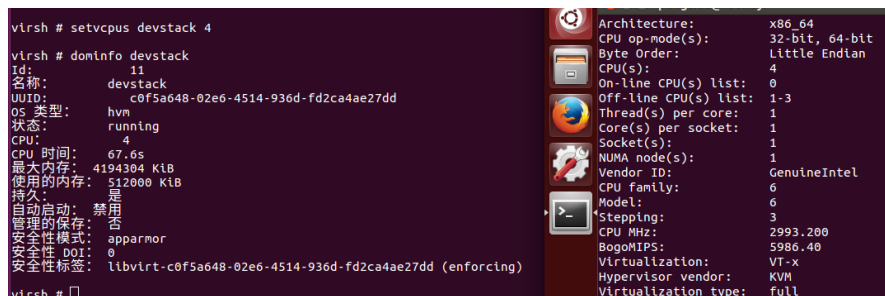
virsh # vncdisplay devstack
:0

virsh # dominfo devstack
Id: 11
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: running
CPU: 1
CPU 时间: 65.3s
最大内存: 4194304 KiB
使用的内存: 512000 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
安全性标签: libvirt-c0f5a648-02e6-4514-936d-fd2ca4ae27dd (enforcing)
virsh #
  
```

```

pengsida@rooney:~$ lscpu
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 1
On-line CPU(s) list: 0
Thread(s) per core: 1
Core(s) per socket: 1
Socket(s): 1
NUMA node(s): 1
Vendor ID: GenuineIntel
CPU family: 6
Model: 6
Stepping: 3
CPU MHz: 2993.200
BogoMIPS: 5986.40
Virtualization: VT-x
Hypervisor vendor: KVM
Virtualization type: full
L1d cache: 32K
L1i cache: 32K
L2 cache: 4096K
NUMA node0 CPU(s): 0
  
```

随后使用“setvcpus”命令设置为 4 个 cpu，结果虽然是分配了 4 个 cpu，但是虚拟机中增加的 3 个 cpu 处于“offline”状态，如下图所示：



```

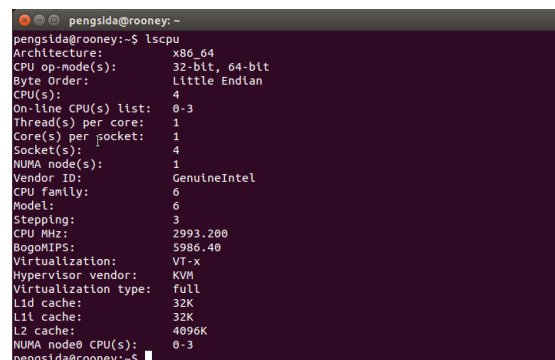
virsh # setvcpus devstack 4

virsh # dominfo devstack
Id: 11
名称: devstack
UUID: c0f5a648-02e6-4514-936d-fd2ca4ae27dd
OS 类型: hvm
状态: running
CPU: 4
CPU 时间: 67.6s
最大内存: 4194304 KiB
使用的内存: 512000 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
安全性标签: libvirt-c0f5a648-02e6-4514-936d-fd2ca4ae27dd (enforcing)
virsh #
  
```

```

pengsida@rooney:~$ lscpu
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 4
On-line CPU(s) list: 0
Off-line CPU(s) list: 1-3
Thread(s) per core: 1
Core(s) per socket: 1
Socket(s): 1
NUMA node(s): 1
Vendor ID: GenuineIntel
CPU family: 6
Model: 6
Stepping: 3
CPU MHz: 2993.200
BogoMIPS: 5986.40
Virtualization: VT-x
Hypervisor vendor: KVM
Virtualization type: full
  
```

只有虚拟机重启之后，这三个 cpu 才会投入使用，如下图所示：



```

pengsida@rooney:~$ lscpu
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 4
On-line CPU(s) list: 0-3
Thread(s) per core: 1
Core(s) per socket: 1
Socket(s): 1
NUMA node(s): 1
Vendor ID: GenuineIntel
CPU family: 6
Model: 6
Stepping: 3
CPU MHz: 2993.200
BogoMIPS: 5986.40
Virtualization: VT-x
Hypervisor vendor: KVM
Virtualization type: full
L1d cache: 32K
L1i cache: 32K
L2 cache: 4096K
NUMA node0 CPU(s): 0-3
pengsida@rooney:~$
  
```

1.3.2 使用 qemu-agent 动态调整 CPU 数量

首先在宿主机安装 qemu-agent:

```
1 sudo apt install qemu-guest-agent
```

然后在虚拟机的 xml 配置文件的 <devices> 中增加如下内容:

```
1 <channel type='unix'>
2   <source mode='bind' path='/var/lib/libvirt/qemu/f16x86_64.agent' />
3   <target type='virtio' name='org.qemu.guest_agent.0' />
4 </channel>
```

如下图所示:

```
<devices>
  <emulator>/usr/bin/qemu-system-x86_64</emulator>
  <disk type='file' device='disk'>
    <driver name='qemu' type='qcow2' />
    <source file='/home/sidapeng/kvm/openstack/devstack.qcow2' />
    <target dev='hda' bus='ide' />
  </disk>
  <interface type='network'>
    <source network='default' />
  </interface>
  <interface type='network'>
    <source network='default' />
  </interface>
  <input type='mouse' bus='ps2' />
  <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0' keymap='en-us' />
  <channel type='unix'>
    <source mode='bind' path='/var/lib/libvirt/qemu/f16x86_64.agent' />
    <target type='virtio' name='org.qemu.guest_agent.0' />
  </channel>
</devices>
```

然后在虚拟机中也安装 qemu-agent:

```
1 sudo apt install qemu-guest-agent
```

这样设置以后, 就可以动态地增加和减少 CPU 数量了。

动态地减少 CPU 数量的命令如下:

```
1 virsh setvcpus --live --guest <domain-name> <count>
```

使用例子如下图所示:

```
virsh # dominfo devstack
Id: 1
名称: devstack
UUID: bd981dc3-d0fe-4606-a707-06aef67d9643
OS 类型: hvm
状态: running
CPU: 4
CPU 时间: 88.9s
最大内存: 1048576 KiB
使用的内存: 1048576 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
安全性标签: libvirt-bd981dc3-d0fe-4606-a707-06aef67d9643

virsh # setvcpus --live --guest devstack 2
```

Byte Order:	Little Endian
CPU(s):	4
On-line CPU(s) list:	0,3
Off-line CPU(s) list:	1,2
Thread(s) per core:	1
Core(s) per socket:	1
Socket(s):	2
NUMA node(s):	1
Vendor ID:	GenuineIntel
CPU family:	6
Model:	6
Stepping:	3
CPU MHz:	2993.200
BogoMIPS:	5986.40
Virtualization:	VT-x
Hypervisor vendor:	KVM
Virtualization type:	full
Lid cache:	32K

如果要保存这个设置，可以再输入如下语句：

```
1 virsh setvcpus <domain-name> <count> --config
2 # 如，setcpus devstack 2 --config
```

需要知道的是，这里实现的并不是把 CPU 数量减少了，而是将虚拟机中 2 个 CPU 设置为“offline”。如果在 virsh 中查看虚拟机的 CPU 数量，会发现和原来一样，如下图所示：

```
virsh # dominfo devstack
Id: 1
名称: devstack
UUID: bd981dc3-d0fe-4606-a707-06aef67d9643
OS 类型: hvm
状态: running
CPU: 4
CPU 时间: 96.6s
最大内存: 1048576 KiB
使用的内存: 1048576 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
安全性标签: libvirt-bd981dc3-d0fe-4606-a707-06aef67d9643 (enforcing)
```

不过这一点我觉得不必计较，因为将 CPU 设置为“offline”以后，就相当于没有使用这两个 CPU 了。

使用 qemu-agent 动态增加 CPU 数量的命令如下：

```
1 virsh setvcpu <domain-name> <count>
2 virsh setvcpu --live --guest <domain-name> <count>
```

使用例子如下图所示：

```
virsh # setvcpus --live --guest devstack 4
virsh # dominfo devstack
Id: 1
名称: devstack
UUID: bd981dc3-d0fe-4606-a707-06aef67d9643
OS 类型: hvm
状态: running
CPU: 4
CPU 时间: 105.8s
最大内存: 1048576 KiB
使用的内存: 1048576 KiB
持久: 是
自动启动: 禁用
管理的保存: 否
安全性模式: apparmor
安全性 DOI: 0
安全性标签: libvirt-bd981dc3-d0fe-4606-a707-06aef67d9643 (enforcing)
virsh #
```

```
Byte Order: Little Endian
CPU(s): 4
On-line CPU(s) list: 0-3
Thread(s) per core: 1
Core(s) per socket: 1
Socket(s): 4
NUMA node(s): 1
Vendor ID: GenuineIntel
CPU family: 6
Model: 6
Stepping: 3
CPU MHz: 2993.200
BogoMIPS: 5986.40
Virtualization: VT-x
Hypervisor vendor: KVM
Virtualization type: full
L1d cache: 32K
L1i cache: 32K
L2 cache: 4096K
NUMA node0 CPU(s): 0-3
```

这样子来增加 CPU 数量，相当于先增加 CPU 数量，然后使用 qemu-agent 将这些 CPU 设置为“online”状态。

如果想保存这个设置，可以再输入如下语句：

```
1 virsh setvcpus <domain-name> <count> --config
```

其实上述语句也就是加一个“--config”参数选项，用于修改下一次的启动配置。

1.4 KVM 扩展磁盘

KVM 扩展磁盘的方式有两种：

1. 添加 qcow2 磁盘加入虚拟机。
2. 直接扩展 qcow2 磁盘。

1.4.1 添加 qcow2 磁盘加入虚拟机

使用这个方法需要注意的是，IDE 磁盘不支持热拔插，其他磁盘可以。如果想添加 IDE 磁盘，只能静态添加。

步骤如下：

1. 使用 qemu-img 工具创建一块 qcow2 磁盘：

```
1  qemu-img create -f qcow2 file.qcow2 <size>
2  # 如, qemu-img create -f qcow2 attach.qcow2 50G
```

2. 向虚拟机中动态添加一块 qcow2 磁盘：

```
1  virsh attach-disk <domain-name> <source> <target> --config
2  # 使用--config选项是为了永久添加这块磁盘
3  # 如, virsh attach-disk devstack /home/sidapeng/kvm/openstack/attach.qcow2
    hdb --subdriver qcow2 --config
```

3. 到上个步骤，虚拟机的磁盘其实已经扩展成功了，我们可以使用如下语句查看虚拟机拥有的磁盘：

```
1  virsh domblklist <domain>
2  # 如, virsh domblklist devstack
```

到虚拟机中查看，如下图所示：

```
pengsida@rooney:~$ sudo fdisk -l

Disk /dev/sda: 53.7 GB, 53687091200 bytes
255 heads, 63 sectors/track, 6527 cylinders, total 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
Disk identifier: 0x000dfef8

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1  *        2048     102762495     51380224   83   Linux
/dev/sda2                102764542     104855551     1045505    5   Extended
/dev/sda5                102764544     104855551     1045504   82   Linux swap / Solaris

Disk /dev/sdb: 53.7 GB, 53687091200 bytes
255 heads, 63 sectors/track, 6527 cylinders, total 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
Disk identifier: 0x00000000

Disk /dev/sdb doesn't contain a valid partition table
```

可以看出，虚拟机的磁盘空间确实扩大了，但是还需要进行分区。

4. 对虚拟机进行分区扩容：

```
1 sudo fdisk /dev/sdb # 从上图可知，新增的是/dev/sdb，它还没分区
```

具体操作如下图所示：

```
Command (m for help): n
Partition type:
 p   primary (0 primary, 0 extended, 4 free)
 e   extended
Select (default p): p
Partition number (1-4, default 1):
Using default value 1
First sector (2048-104857599, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-104857599, default 104857599):
Using default value 104857599

Command (m for help): wq
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

再使用“fdisk -l”语句，就可以看到多了/dev/sdb1 分区，如下图所示：

```
pengsida@rooney:~$ sudo fdisk -l

Disk /dev/sda: 53.7 GB, 53687091200 bytes
255 heads, 63 sectors/track, 6527 cylinders, total 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
Disk identifier: 0x000dfef8

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1 *         2048        102762495   51380224    83   Linux
/dev/sda2           102764542   104855551    1045505     5   Extended
/dev/sda5           102764544   104855551    1045504    82   Linux swap / Solaris

Disk /dev/sdb: 53.7 GB, 53687091200 bytes
22 heads, 22 sectors/track, 216647 cylinders, total 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
Disk identifier: 0x36bcab04

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1           2048        104857599   52427776    83   Linux
```

然后再对/dev/sdb1 进行格式化，当前文件系统为 ext4，所以使用 mkfs.ext4 进行格式化：

```
1 sudo mkfs.ext4 /dev/sdb1
```

最后将这个分区挂载到/mnt 下：

```
1 sudo mount -t ext4 /dev/sdb1 /mnt
```

到此，虚拟机可以说是正式扩容了，如下图所示：

```
pengsida@grooney:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            485M  4.0K  485M   1% /dev
tmpfs           100M  868K   99M   1% /run
/dev/sda1       49G   3.8G   42G   9% /
none            4.0K   0     4.0K   0% /sys/fs/cgroup
none            5.0M   0     5.0M   0% /run/lock
none            496M  152K  496M   1% /run/shm
none            100M   36K  100M   1% /run/user
/dev/sdb1       50G   52M   47G   1% /mnt
```

mount 挂载分区在系统重启之后需要重新挂载，可以通过修改/etc/fstab 文件使得挂载永久生效，如下图所示：

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options>          <dump> <pass>
# / was on /dev/sda1 during installation
UUID=a64bf9e3-ae1d-4208-94ea-4dd09d87a47e /          ext4    errors=remount-ro 0    1
# swap was on /dev/sda5 during installation
UUID=e01c2830-c47d-4b9f-b5de-f0857e988c6c none        swap    sw              0    0
/dev/sdb1 /mnt ext4 defaults 0 0
```

1.4.2 直接扩展 qcow2 磁盘

使用这个方法需要注意的是，如果虚拟机的 qcow2 磁盘拥有快照，那么是无法使用 qemu-img 工具扩展这个磁盘的。还有就是，这个方法无法动态扩容。

步骤如下：

1. 直接扩展虚拟机的 qcow2 磁盘：

```
1  qemu-img resize file.qcow2 +<size>
2  # 比如, qemu-img resize devstack.qcow2 +50G
```

2. 重启虚拟机，使用 “sudo fdisk -l” 命令，就可以看到原先的磁盘多了 50G。
3. 对磁盘进行分区，和上一小节的第四步骤一样，在此就不再详述。