目 录

1 CPU 配置 **3**
1.1 -smp 参数项 . . . 3
1.2 查看 cpu 配置 . . . 3
1.2.1 在客户机中查看 cpu 信息 . . . 3
1.2.2 使用 qemu 监控客户机 cpu 信息 . . . 4
1.3 -cpu 参数项 . . . 4
1.4 vCPU 的绑定 . . . 5
1.4.1 隔离宿主机 CPU . . . 5
1.4.2 绑定客户机 vCPU . . . 6

2 内存配置 **8**
2.1 -m 参数项 . . . 8
2.2 查看内存信息 . . . 8
2.3 EPT 扩展页表 . . . 8
2.4 -mem-path 参数项 . . . 9

3 存储配置 **10**
3.1 与存储相关的参数项 . . . 10
3.2 -drive 参数项 . . . 10
3.3 -boot 参数项 . . . 11
3.4 查看存储设备的信息 . . . 12
3.5 qemu-img 工具 . . . 12

4 网络配置 **14**
4.1 使用网桥模式 . . . 14
4.2 使用 NAT 模式 . . . 14
4.3 使用用户模式 . . . 14
4.4 其他网络选项 . . . 14

5 显示配置 **15**
5.1 使用 SDL . . . 15
5.2 使用 VNC . . . 15

5.3 使用非图形模式 15
5.4 其他显示选项 15

1 CPU 配置

1.1 -smp 参数项

qemu-system-x86_64 命令行中，”-smp” 参数可以用来配置客户机的 SMP 系统，具体参数如下：

```
qemu-system-x86_64 -smp n[,maxcpus=cpus][,cores=cores][,threads=threads][,sockets=
    sockets]
```

各个选项介绍如下：

n	用于设置客户机中使用的逻辑 PCU 数量
maxcpus	用于设置客户机中最大可能被使用的 CPU 数量
cores	用于设置每个 CPU socket 上的 core 数量
threads	用于设置每个 CPU core 上的线程数
sockets	用于设置客户机中看到的总的 CPU socket 数量

例子如下：

```
qemu-system-x86_64 -smp 4,maxcpus=8,sockets=2,cores=2,threads=2 ubuntu1604 -vnc
    127.0.0.1:2
```

1.2 查看 cpu 配置

1.2.1 在客户机中查看 cpu 信息

使用如下命令可以输出 cpu 当前的信息：

```
cat /proc/cpuinfo
```

这里介绍一下 cat 命令：

三大功能	1. 一次显示整个文件: cat filename 2. 从键盘创建一个文件: cat > filename 3. 将几个文件合并为一个文件: cat file1 file2 > file
参数	-n 或–number: 由 1 开始对所有输出的行数编号 -b 或–number-nonblank: 和-n 相似，只不过对于空白行不编号 -s 或–squeeze-blank: 当遇到有连续两行以上的空白行，就代换为一行的空白行

1.2.2 使用 qemu 监控客户机 cpu 信息

使用 qemu-system-x86_64 命令时，加上”-monitor stdio”，即可使用 monitor command 监控客户机使用情况，比如在联网情况下输入如下命令：

```
qemu-system-x86_64 ubuntu1604.img -vnc 127.0.0.1:2 -monitor stdio
```

此时，就开始 monitor command 来监控客户机。可以在 qemu monitor 中使用如下命令查询 cpu 状态：

```
info cpus
```

1.3 -cpu 参数项

qemu-system-x86_64 命令行中，”-cpu” 参数可以用来查看 qemu 所支持 cpu 模型，或者指定客户机的 CPU 模型。具体使用如下：

```
// 查看qemu所支持的cpu模型
qemu-system-x86_64 -cpu ?
// 指定客户机中的cpu模型
qemu-system-x86_64 -cpu cpu_model
```

qemu 支持的 cpu 模型如下所示：

```
pengsida@psd:~$ qemu-system-x86_64 -cpu ?

(process:3543): GLib-WARNING **: /build/glib2.0-7IO_Yw/glib2.0-2.48.1/./glib
/gmem.c:483: custom memory allocation vtable not supported
x86       Opteron_G4  AMD Opteron 62xx class CPU
x86       Opteron_G3  AMD Opteron 23xx (Gen 3 Class Opteron)
x86       Opteron_G2  AMD Opteron 22xx (Gen 2 Class Opteron)
x86       Opteron_G1  AMD Opteron 240 (Gen 1 Class Opteron)
x86      SandyBridge  Intel Xeon E312xx (Sandy Bridge)
x86          Westmere  Westmere E56xx/L56xx/X56xx (Nehalem-C)
x86           Nehalem  Intel Core i7 9xx (Nehalem Class Core i7)
x86            Penryn  Intel Core 2 Duo P9xxx (Penryn Class Core 2)
x86            Conroe  Intel Celeron_4x0 (Conroe/Merom Class Core 2)
x86              n270  Intel(R) Atom(TM) CPU N270   @ 1.60GHz
x86            athlon  QEMU Virtual CPU version 1.2.50
x86          pentium3
x86          pentium2
x86           pentium
x86               486
x86           coreduo  Genuine Intel(R) CPU           T2600  @ 2.16GHz
x86             kvm32  Common 32-bit KVM processor
x86            qemu32  QEMU Virtual CPU version 1.2.50
x86             kvm64  Common KVM processor
x86          core2duo  Intel(R) Core(TM)2 Duo CPU     T7700  @ 2.40GHz
x86            phenom  AMD Phenom(tm) 9550 Quad-Core Processor
x86            qemu64  QEMU Virtual CPU version 1.2.50
```

如果不加”cpu” 参数启动客户机时，采用”qemu64” 作为默认的 cpu 模型。

1.4 vCPU 的绑定

vCPU 就是客户机的虚拟 cpu，vCPU 相当于宿主机中一个普通的 qemu 线程。可以使用 taskset 工具将 vCPU 线程绑定到特定的 cpu 上执行。

在实际应用中，如果想要为客户提供客户机使用，并且要求不受宿主机中其他客户机的影响，就需要将 vCPU 绑定到特定的 cpu 上。步骤如下：

1. 启动宿主机时隔离出特定的 CPU 专门供一个客户机使用。
2. 启动客户机，将其 vCPU 绑定到宿主机特定的 CPU 上。

1.4.1 隔离宿主机 CPU

在 grub 文件中 Linux 内核启动的命令行加上”isolcpus” 参数，就可以实现 CPU 的隔离。这里介绍一下”isolcpus” 参数项：

功能	将相应的 CPU 从调度算法中隔离出来
参数选项	isolcpus= cpu_number[,cpu_number,...]

向 grub 文件中添加”isolcpus” 参数的命令如下所示：

```
sudo vi /boot/grub/grub.cfg
/menuentry
```

然后在插入模式下，在 initrd 参数前一行写入：

```
isolcpus = cpu\_number1[,cpu\_number2,...]
```

如下图所示：

```
menuentry 'Ubuntu' --class ubuntu --class gnu-linux --class gnu --class os $menu
entry_id_option 'gnulinux-simple-84ca4c1b-ff58-4c6d-8808-45e3dc5f4e63' {
        recordfail
        load_video
        gfxmode $linux_gfx_mode
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_gpt
        insmod ext2
        set root='hd0,gpt2'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,gpt2 --hint-ef
i=hd0,gpt2 --hint-baremetal=ahci0,gpt2  84ca4c1b-ff58-4c6d-8808-45e3dc5f4e63
        else
          search --no-floppy --fs-uuid --set=root 84ca4c1b-ff58-4c6d-8808-45e3dc
5f4e63
        fi
        linux   /boot/vmlinuz-4.9.0-rc8+ root=UUID=84ca4c1b-ff58-4c6d-8808-45e3d
c5f4e63 ro  quiet splash $vt_handoff
        isolcpus = 2,3
        initrd  /boot/initrd.img-4.9.0-rc8+
}
-- 插入 --                                                    151,2         39%
```

重启电脑以后就将相应的 cpu 隔离出调度算法了。

使用如下命令可以查看 cpu 上执行的进程和线程总数，用于检查 CPU 是否成功被隔离。

```
ps -eLo psr | grep cpu\_number | wc -l
```

下面分别介绍命令中的 ps 和 wc：

ps	用于显示当前系统的进程信息的状态
参数项	-e: 用于显示所有进程 -L: 用于显示所有线程 -o: 用于以特定的格式输出信息,psr 指定输出分配给进程运行的处理器编号

wc	该命令统计给定文件中的字节数、字数、行数
参数项	-c: 统计字节数 -l: 统计行数 -w: 统计字数

假如成功隔离了 cpu2，就会看到在 cpu 上执行的进程和线程数非常少。

1.4.2 绑定客户机 vCPU

使用 taskset 命令就可以将 vCPU 绑定到特定的 CPU 上。taskset 命令的使用如下所示：

```
taskset -p mask pid
```

这里介绍一下 taskset 命令：

taskset	将进程绑定到特定的 CPU 上
参数项	-p: 将已经创建的进程绑定到 CPU 上 mask: 用于指定 CPU 的掩码，mask 第几位为 1 就代表第几号 CPU pid: 进程号，用于指定进程

比如，如果想把进程号为 3963 的进程绑定到 cpu2 和 cpu3 上，就使用如下命令：

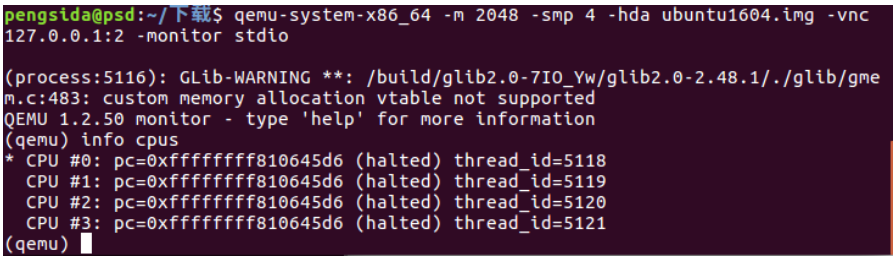
```
taskset -p 0x6 3963
```

0x6 二进制位 1100，代表 cpu2 和 cpu3。而 3963 指定了进程号为 3963 的进程。

如此一来，如果想把 vCPU 绑定到宿主机的 cpu 上，只要知道 vCPU 的进程号就行了。可以在 qemu monitor 中使用如下命令查询 vCPU 的进程号：

```
info cpus
```

如下所示：



可以看到，图中 vCPU 的进程号分别是 5118、5119、5120 和 5121。

2 内存配置

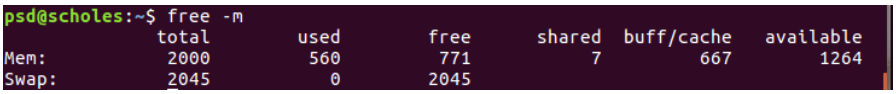
2.1 -m 参数项

-m megs	设置客户机的内存位 megsMB 大小 默认单位为 MB，加上”M” 或”G” 可以指定单位 不设置-m 参数，客户机内存默认为 128MB

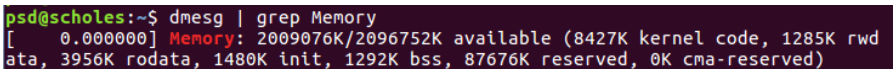
2.2 查看内存信息

linux 下有两个命令可以用于查看内存信息。

第一个是 free -m，如下图所示：



第二个是 dmesg。不过因为 dmesg 存放着内核开机信息，信息量比较多，需要用 grep 命令来筛选，如下图所示：



2.3 EPT 扩展页表

EPT 扩展页表是 Intel 的第二代硬件虚拟化技术，是针对内存管理单元的虚拟化扩展。

在 Linux 系统中，可以通过如下命令确定系统是否支持 EPT 功能：

```
grep ept /proc/cpuinfo
```

可以通过如下命令确定 KVM 是否打开了 EPT 功能：

```
cat /sys/module/kvm_intel/parameters/ept
```

在加载 kvm_intel 模块时，可以通过设置 ept 的值来打开 EPT。

```
modprobe kvm_intel ept=0 // ept代表关闭EPT功能
```

如果 kvm_intel 模块已经处于加载状态，则需要先写在这个模块，在重新加载时加入所需的参数设置。如下所示：

```
rmmod kvm_intel
modprobe kvm_intel ept=1
```

2.4 -mem-path 参数项

qemu-kvm 提供了”-mem-path” 参数项用于将 huge page 的特性应用到客户机上。

huge page 是大小超过 4KB 的内存页面，它可以让地址转换信息减少，节约页表所占用的内存数量，在整体上提升系统的性能。

可以使用如下命令查看系统中 huge page 的信息，如下所示：

```
cat /proc/meminfo | grep HugePages
```

可以通过以下几步让客户机使用 huge page：

(1) 在宿主机中挂载 hugetlbfs 文件系统，命令如下所示：

```
sudo mount -t hugetlbfs hugetlbfs /dev/hugepages
```

这里介绍一下 mount 命令：

标准格式	mount -t type device dir
功能	让内核将在 device 上的文件系统挂载到目录 dir 下，文件系统类型是 type
参数项	-t: 指定文件系统的类型

所以之前命令就是将 hugetlbfs 类型的文件系统挂载到/dev/hugepages 上。

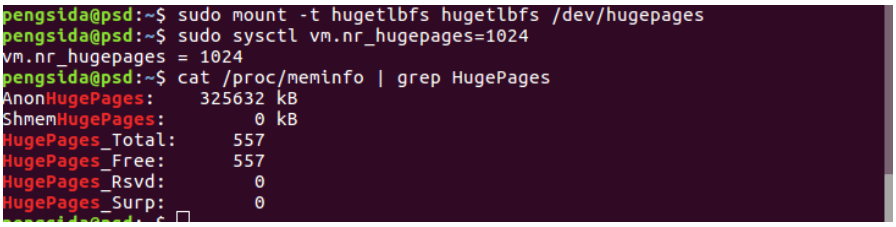
(2) 设置 hugepage 的数量，命令如下所示：

```
sudo sysctl vm.nr_hugepages=num
```

(3) 启动客户机时使用”-mem-path” 参数让客户机使用 hugepage 的内存，如下所示：

```
qemu-system-x86_64 ubuntu1604.img -mem-path /dev/hugepages
```

上述过程的实际操作如下图所示：



3 存储配置

3.1 与存储相关的参数项

qemu-system-x86_64 命令行中，主要有如下的参数来配置客户机的存储：

-hda file	将 file 镜像文件作为客户机中的第一个 IDE 设备。 也可以将宿主机中的一个硬盘作为-hda 的 file 参数来使用。 如果文件名包含”,”，那么书写 file 时需要使用两个逗号，如“-hda my,,file”
-hdb file	将 file 镜像文件作为客户机中的第二个 IDE 设备。
-hdc file	将 file 镜像文件作为客户机中的第三个 IDE 设备。
-hdd file	将 file 镜像文件作为客户机中的第四个 IDE 设备。
-fda file	将 file 镜像文件作为客户机中的第一个软盘设备。
-fdb file	将 file 镜像文件作为客户机中的第二个软盘设备。
-cdrom file	将 file 镜像文件作为客户机中的光盘 CD-ROM。
-mtdblock file	将 file 镜像文件作为客户机自带的一个 Flash 存储器。
-sd file	将 file 镜像文件作为客户机中的 SD 卡。
-pflash file	将 file 镜像文件作为客户机中的并行 Flash 存储器。

3.2 -drive 参数项

qemu-system-x86_64 使用“-drive”参数来定义一个存储驱动器。对“-drive”参数项的介绍如下：

标准格式	-drive option[,option[,option[,...]]]
以下是参数选项	
file=file	使用 file 文件作为镜像文件加载到客户机的驱动器中。
if=interface	指定驱动器使用的接口类型。
bus=bus	设置驱动器在客户机中的总线编号。
unit=unit	设置驱动器在客户机中的单元编号。
index=index	设置在同一种接口的驱动器中的索引编号。
media=media	设置驱动器中媒介的类型。
snapshot=on/off	设置是否启用“-snapshot”选项。 当 snapshot 启用时，qemu 不会把磁盘数据的更改写回到镜像文件中。 可以在 QEMU monitor 中使用“commit”命令强制将磁盘数据的更改保存到镜像文件中。

cache=cache	设置宿主机对块设备数据访问中的 cache 情况。 参数有“none”、“writeback”和“writethrough”。 “writethrough”，默认值，该模式下调用 write 写入数据的同时将数据写入磁盘缓存和后端块设备。 “writeback”，该模式下调用 write 写入数据时只将数据写入到数据缓存中，只有在数据被换出缓存时才将修改的数据写到后段块设备中。 “none”，关闭缓存。
aio=aio	选择异步 IO 的方式。 参数有“threads”和“native”。 “threads”，让一个线程池去处理异步 IO。 “native”，只适用于“cache=none”的情况，使用 linux 原生的 AIO。
format=format	指定使用的磁盘格式。
serial=serial	指定分配给设备的序列号。
addr=addr	分配给驱动器控制器的 PCI 地址。 只适用于“if=virtio”的情况。
id=name	设置该驱动器的 ID。
readonly=on/off	设置该驱动器是否只读。

3.3 -boot 参数项

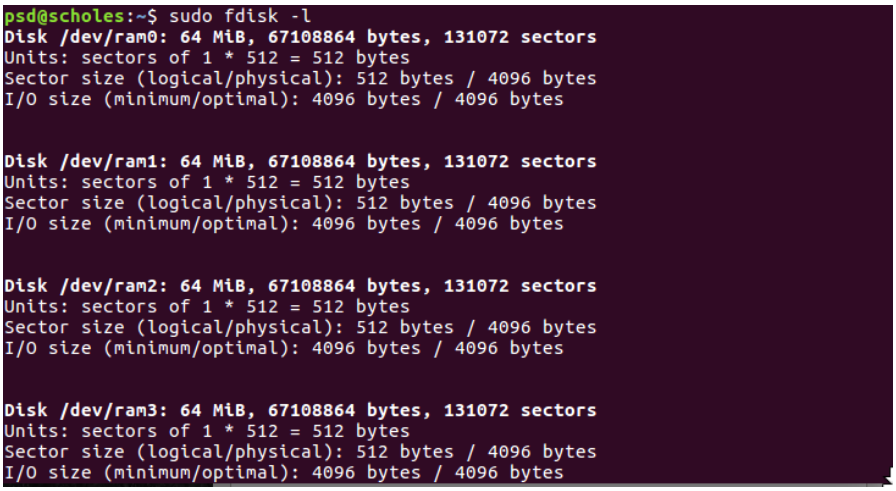
qemu-system-x86_64 使用“-boot”参数来指定各种存储设备在客户机中的启动顺序。对“-boot”参数项的介绍如下：

标准格式如下： -boot [order=drives][,once=drives][,menu=on/off][,splash=splashfile][,splash-time=sp-time]	
以下是参数选项	
order=drives	设置存储设备启动顺序 “drives”有如下参数： “a”表示第一个软驱；“b”表示第二个软驱 “c”表示第一个硬盘；“d”表示 CD-ROM 光驱 “n”表示从网络启动 比如“order=cd”表示先启动硬盘，在启动光驱
once=drives	设置第一次启动的启动顺序，在系统重启后该设置无效
menu=on/off	设置交互式的启动菜单选项 (需要客户机 BIOS 支持)

splash=splashfile	只适用于“menu=on”的情况。将名为 splashfile 的图片作为 logo 传递给 BIOS 显示
splash-time=sp-time	只适用于“menu=on”的情况。sp-time 是 BIOS 显示 logo 的时间，单位是 ms

3.4 查看存储设备的信息

在 linux 下，”fdisk -l” 命令可以列出电脑的磁盘分区情况，如下图所示：



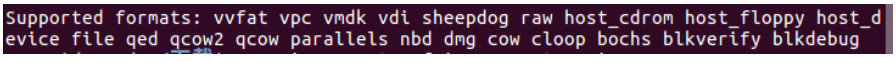
3.5 qemu-img 工具

qemu-img 工具是 QEMU 的磁盘管理工具，它的命令行基本用法如下所示：

```
qemu-img command [command options]
```

它支持的命令如下所示：

1. check [-f fmt] filename
   对磁盘镜像文件进行一致性检查，查找镜像文件中的错误。
   参数-f fmt 用于指定文件的格式，如果没有该参数项，qemu-img 将自动检测文件格式。
   qemu-img 支持如下所示的文件格式：



2. create [-f fmt] [-o options] filename [size]
   创建一个格式为 fmt，大小为 size，文件名为 filename 的镜像文件。
   参数-o options 用于对文件的各种功能进行设置。使用“-o ?”可以查询某种格式文件支持哪些选项，如下所示：

```
pengsida@psd:~/下载$ qemu-img create -f qcow2 -o ? temp.qcow2
Supported options:
size             Virtual disk size
compat           Compatibility level (0.10 or 1.1)
backing_file     File name of a base image
backing_fmt      Image format of the base image
encryption       Encrypt the image
cluster_size     qcow2 cluster size
preallocation    Preallocation mode (allowed values: off, metadata)
lazy_refcounts   Postpone refcount updates
```

3. commit [-f fmt] filename
   将 filename 文件中的更改提交到后端支持镜像文件中。后端支持镜像文件可以在创建磁盘镜像文件时指定，命令如下：

```
qemu-img -f qcow2 -o backing_file=ubuntu1604.img ubuntu1604.qcow2
```

4. convert [-c][-f fmt][-O output_fmt][-o options]filename[filename2[...]]output_filename
   将 fmt 格式的 filename 镜像文件根据 options 选项转换为 output_fmt 格式的 output_filename 镜像文件。
   参数-c 表示对输出的镜像文件进行压缩。
   参数-o options 用于指定各种选项，和 create 命令中的“-o options”参数一样。

5. info [-f fmt] filename
   展示 filename 镜像文件的信息。

6. snapshot [-l|-a snapshot|-c snapshot|-d snapshot]filename
   参数“-l”用于列出镜像文件中所有的快照。
   参数“-a snapshot”用于让镜像文件使用某个快照。
   参数“-c snapshot”用于创建一个快照。
   参数“-d snapshot”用于删除一个快照。

7. rebase [-f fmt][-t cache][-p][-u]-b backing_file[-F backing_file]filename
   改变镜像文件的后端镜像文件，只有 qcow2 和 qed 格式支持 rebase 命令。
   参数“-b backing_file”用于指定文件作为镜像文件的后端镜像文件。
   参数“-F backing_fmt”用于指定后端镜像文件的格式。

8. resize filename [+|-]size
   改变镜像文件的大小，“+”和“-”分别表示增加和减少镜像文件的大小。

4 网络配置

4.1 -net 参数项

qemu-kvm 命令行中“-net”参数项的格式如下所示：

标准格式: -net nic[,vlan=n][,macaddr=mac][,model=type][,name=name][,addr=addr][,vectors=v]

4.2 使用网桥模式

4.3 使用 NAT 模式

4.4 使用用户模式

4.5 其他网络选项

5 显示配置

5.1 使用 SDL

5.2 使用 VNC

5.3 使用非图形模式

5.4 其他显示选项