CPU隔离——ubuntu中进程和处理器亲和性和vCPU的绑定

cpu 隔离

启动宿主机时隔离出两个逻辑CPU专门供客户机使用。在Linux内核启动的命令行加上“isolcpus=”参数，可以实现CPU的隔离，让系统启动后普通进程默认都不会调度到被隔离的CPU上执行。下面测试，在四核心的ubuntu 系统中隔离cpu2和cpu3

```
root@map-VirtualBox:~# grep "processor" /proc/cpuinfo
processor       : 0
processor       : 1
processor       : 2
processor       : 3
```

方法1：ubuntu 启动的引导项保持在/boot/grub/grub.cfg 中，可以通过开机加载引导项时编译，

或者

方法2：在系统中通过 /etc/default/grub 修改 **编辑 /etc/default/grub 文件 ，在 quiet splash 后面加上 isolcpus=2,3**

```
root@map-VirtualBox:~# vi /etc/default/grub

# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash isolcpus=2,3"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"
```

编辑完成并保存后，回到终端，执行命令“**update-grub**”。其将自动依照刚才编辑的配置文件（/etc/default/grub）生成为引导程序准备的配置文件（/boot/grub/grub.cfg）

```
root@map-VirtualBox:~# update-grub
Generating grub configuration file ...
Warning: Setting GRUB_TIMEOUT to a non-zero value when GRUB_HIDDEN_TIMEOUT is set is no longer supported.
Found linux image: /boot/vmlinuz-3.19.0-25-generic
Found initrd image: /boot/initrd.img-3.19.0-25-generic
Found memtest86+ image: /memtest86+.elf
Found memtest86+ image: /memtest86+.bin
done
root@map-VirtualBox:~#
```

连续输出了各个引导项之后，输出“done”即已完成生成过程，查看 **/boot/grub/grub.cfg**，已经添加成功

```
export linux_gfx_mode
menuentry 'Ubuntu' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux-si
mple-5e752fea-5cdd-432c-8a45-e0dc0a4b6541' {
        recordfail
        load_video
        gfxmode $linux_gfx_mode
        insmod gzio
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-bareme
tal=ahci0,msdos1  1ec8e7be-160b-4906-9545-ec74fe36965d
        else
          search --no-floppy --fs-uuid --set=root 1ec8e7be-160b-4906-9545-ec74fe36965d
        fi
        linux   /vmlinuz-3.19.0-25-generic root=/dev/mapper/ubuntu--vg-root ro  quiet splash isolcpus=2,3 $v
t_handoff
        initrd  /initrd.img-3.19.0-25-generic
}
submenu 'Advanced options for Ubuntu' $menuentry_id_option 'gnulinux-advanced-5e752fea-5cdd-432c-8a45-e0dc0a
4b6541' {
        menuentry 'Ubuntu, with Linux 3.19.0-25-generic' --class ubuntu --class gnu-linux --class gnu --clas
s os $menuentry_id_option 'gnulinux-3.19.0-25-generic-advanced-5e752fea-5cdd-432c-8a45-e0dc0a4b6541' {
                recordfail
                load_video
                gfxmode $linux_gfx_mode
                insmod gzio
                insmod part_msdos
                insmod ext2
                set root='hd0,msdos1'
                if [ x$feature_platform_search_hint = xy ]; then
                  search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hin
t-baremetal=ahci0,msdos1  1ec8e7be-160b-4906-9545-ec74fe36965d
                else
                  search --no-floppy --fs-uuid --set=root 1ec8e7be-160b-4906-9545-ec74fe36965d
                fi
                echo    'Loading Linux 3.19.0-25-generic ...'
                linux   /vmlinuz-3.19.0-25-generic root=/dev/mapper/ubuntu--vg-root ro  quiet splash isolcpu
s=2,3 $vt_handoff
                echo    'Loading initial ramdisk ...'
                initrd  /initrd.img-3.19.0-25-generic
        }
```

重启系统

```
root@map-VirtualBox:~# ps -eLo psr | grep 0 | wc -l
280
root@map-VirtualBox:~# ps -eLo psr | grep 1 | wc -l
239
root@map-VirtualBox:~# ps -eLo psr | grep 2 | wc -l
5
root@map-VirtualBox:~# ps -eLo psr | grep 3 | wc -l
5
root@map-VirtualBox:~# ps -eLo psr | grep 4 | wc -l
0
```

从上面的命令行输出信息可知，cpu0和cpu1上分别有280和239个线程在运行，而cpu2和cpu3上都分别只有5个线程在运行，机器只有四核心所以cpu4不存在。

隔离了cpu2和cpu3 为什么还有进程

查看cpu2,cpu3 上的进程

```
root@map-VirtualBox:~# ps -eLo ruser,pid,ppid,lwp,psr,args | awk '{if($5==2)print $0}
root         20     2    20   2 [watchdog/2]
root         21     2    21   2 [migration/2]
root         22     2    22   2 [ksoftirqd/2]
root         23     2    23   2 [kworker/2:0]
root         24     2    24   2 [kworker/2:0H]
root@map-VirtualBox:~# ps -eLo ruser,pid,ppid,lwp,psr,args | awk '{if($5==3)print $0}
root         27     2    27   3 [watchdog/3]
root         28     2    28   3 [migration/3]
root         29     2    29   3 [ksoftirqd/3]
root         30     2    30   3 [kworker/3:0]
root         31     2    31   3 [kworker/3:0H]
root@map-VirtualBox:~#
```

根据输出信息中cpu2和cpu3上运行的线程信息（也包括进程在内），分别有migration进程（用于进程在不同CPU间迁移）、两个kworker进程（用于处理workqueues）、ksoftirqd进程（用于调度CPU软中断的进程），这些进程都是内核对各个CPU的一些守护进程，而没有其他的普通进程在cup2和cpu3上运行，说明对其的隔离是生效的。

另外，简单解释一下上面的一些命令行工具及其参数的意义。ps命令显示当前系统的进程信息的状态，它的“-e”参数用于显示所有的进程，“-L”参数用于将线程（LWP，light-weight process）也显示出来，“-o”参数表示以用户自定义的格式输出（其中“psr”这列表示当前分配给进程运行的处理器编号，“lwp”列表示线程的ID，“ruser”表示运行进程的用户，“pid”表示进程的ID，“ppid”表示父进程的ID，“args”表示运行的命令及其参数）。结合ps和awk工具的使用，是为了分别将在处理器cpu2和cpu3上运行的进程打印出来。

KVM 虚拟机绑定 CPU

我在devstack 部署的环境中测试，启动一个虚拟机

```
stack@map-VirtualBox:~/devstack$ nova list
+--------------------------------------+------+--------+------------+-------------+-
| ID                                   | Name | Status | Task State | Power State |
+--------------------------------------+------+--------+------------+-------------+-
| 5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e | b    | SHUTOFF | -          | Shutdown    |
```

```
+--------------------------------------+------+--------+------------+-------------+-
stack@map-VirtualBox:~/devstack$ nova start b
Request to start server b has been accepted.
stack@map-VirtualBox:~/devstack$ nova list
+--------------------------------------+------+--------+------------+-------------+--
| ID                                   | Name | Status | Task State | Power State | N
+--------------------------------------+------+--------+------------+-------------+--
| 5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e | b    | ACTIVE | -          | Running     | p
+--------------------------------------+------+--------+------------+-------------+--
stack@map-VirtualBox:~/devstack$ virsh list --all
 Id    Name                           State
----------------------------------------------------
 2     instance-00000003              running

stack@map-VirtualBox:~/devstack$ ps -ef|grep instance-00000003
libvirt+  5006     1 85 15:06 ?        00:00:17 /usr/bin/qemu-system-x86_64 -name ins
stack     5058  4280  0 15:06 pts/25   00:00:00 grep --color=auto instance-00000003
stack@map-VirtualBox:~/devstack$
```

查看vCPU 的qemu 进程，执行 ps -eLo ruser,pid,ppid,lwp,psr,args | grep qemu | grep -v grep

多执行几次，可以看到，第五列的值是变动的，它是基于cpu 时间片的机制，不停切换的

绑定cpu，先切换到root 用户

```
stack@map-VirtualBox:~/devstack$ su -
Password:
root@map-VirtualBox:~# taskset -p 0x4 5006
pid 5006's current affinity mask: f
pid 5006's new affinity mask: 4
root@map-VirtualBox:~# taskset -p 0x4 5009
```

```
pid 5009's current affinity mask: f
pid 5009's new affinity mask: 4
root@map-VirtualBox:~# taskset -p 0x4 5011
pid 5011's current affinity mask: f
pid 5011's new affinity mask: 4
root@map-VirtualBox:~# taskset -p 0x8 5011
pid 5011's current affinity mask: 4
pid 5011's new affinity mask: 8
root@map-VirtualBox:~#
```

三条命令分别为 ：

绑定代表整个虚拟机的QEMU进程，使其运行在cpu2上

绑定虚拟机 第一个vCPU的QEMU进程，使其运行在cpu2上

绑定虚拟机 第一个vCPU的QEMU进程，使其运行在cpu3上

再次 查看vCPU 的qemu 进程，执行 ps -eLo ruser,pid,ppid,lwp,psr,args | grep qemu | grep -v grep

多执行几次，可以看到，第五列的值是固定的

绑定成功 ！

```
root@map-VirtualBox:~# ps -eLo ruser,pid,ppid,lwp,psr,args | grep qemu | grep -v grep
libvirt+  5006     1  5006  2 /usr/bin/qemu-system-x86_64 -name instance-00000003 -S -machine pc-i440fx-t
 5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e -smbios type=1,manufacturer=OpenStack Foundation,product=OpenStack N
-1fd39aaa432e,family=Virtual Machine -no-user-config -nodefaults -chardev socket,id=charmonitor,path=/var/
r,mode=control -rtc base=utc -no-shutdown -boot strict=on -kernel /opt/stack/data/nova/instances/5e4aaac7-
0-92a1-1fd39aaa432e/ramdisk -append root=/dev/vda console=tty0 console=ttyS0 no_timer_check -device piix3-
ab8d-45d0-92a1-1fd39aaa432e/disk,if=none,id=drive-virtio-disk0,format=qcow2,cache=none -device virtio-blk-
 file=/opt/stack/data/nova/instances/5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e/disk.config,if=none,id=drive-ide
1,id=ide0-1-1 -netdev tap,fd=24,id=hostnet0 -device virtio-net-pci,netdev=hostnet0,id=net0,mac=fa:16:3e:c8
5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e/console.log -device isa-serial,chardev=charserial0,id=serial0 -charde
en-us -device cirrus-vga,id=video0,bus=pci.0,addr=0x2 -device virtio-balloon-pci,id=balloon0,bus=pci.0,add
libvirt+  5006     1  5009  2 /usr/bin/qemu-system-x86_64 -name instance-00000003 -S -machine pc-i440fx-t
 5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e -smbios type=1,manufacturer=OpenStack Foundation,product=OpenStack N
-1fd39aaa432e,family=Virtual Machine -no-user-config -nodefaults -chardev socket,id=charmonitor,path=/var/
r,mode=control -rtc base=utc -no-shutdown -boot strict=on -kernel /opt/stack/data/nova/instances/5e4aaac7-
0-92a1-1fd39aaa432e/ramdisk -append root=/dev/vda console=tty0 console=ttyS0 no_timer_check -device piix3-
ab8d-45d0-92a1-1fd39aaa432e/disk,if=none,id=drive-virtio-disk0,format=qcow2,cache=none -device virtio-blk-
 file=/opt/stack/data/nova/instances/5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e/disk.config,if=none,id=drive-ide
1,id=ide0-1-1 -netdev tap,fd=24,id=hostnet0 -device virtio-net-pci,netdev=hostnet0,id=net0,mac=fa:16:3e:c8
5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e/console.log -device isa-serial,chardev=charserial0,id=serial0 -charde
en-us -device cirrus-vga,id=video0,bus=pci.0,addr=0x2 -device virtio-balloon-pci,id=balloon0,bus=pci.0,add
libvirt+  5006     1  5011  3 /usr/bin/qemu-system-x86_64 -name instance-00000003 -S -machine pc-i440fx-t
 5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e -smbios type=1,manufacturer=OpenStack Foundation,product=OpenStack N
-1fd39aaa432e,family=Virtual Machine -no-user-config -nodefaults -chardev socket,id=charmonitor,path=/var/
r,mode=control -rtc base=utc -no-shutdown -boot strict=on -kernel /opt/stack/data/nova/instances/5e4aaac7-
0-92a1-1fd39aaa432e/ramdisk -append root=/dev/vda console=tty0 console=ttyS0 no_timer_check -device piix3-
ab8d-45d0-92a1-1fd39aaa432e/disk,if=none,id=drive-virtio-disk0,format=qcow2,cache=none -device virtio-blk-
 file=/opt/stack/data/nova/instances/5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e/disk.config,if=none,id=drive-ide
1,id=ide0-1-1 -netdev tap,fd=24,id=hostnet0 -device virtio-net-pci,netdev=hostnet0,id=net0,mac=fa:16:3e:c8
5e4aaac7-ab8d-45d0-92a1-1fd39aaa432e/console.log -device isa-serial,chardev=charserial0,id=serial0 -charde
en-us -device cirrus-vga,id=video0,bus=pci.0,addr=0x2 -device virtio-balloon-pci,id=balloon0,bus=pci.0,add
root@map-VirtualBox:~#
```

来源： http://www.itkeyword.com/doc/1365895376935810x730/linux-kernel-cpu-kvm