**09-调度模块**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **所有者** | **修改时间** | **创建时间** |
| [性能工具](https://cqbk0n8o1h.feishu.cn/wiki/ZpUewe8f7ir46LkVrRaccmk0n4g) | 李2017 | 2024-05-14 09:19 | 2024-05-14 09:18 |

1. **cpu管理**

1.1 **使用cpulimit工具**

这是一个可以控制进程CPU使用的工具。你可以指定百分比形式的CPU使用率的上限。

|  |
| --- |
| Bash sudo cpulimit -p <pid> -l <CPU使用率百分比> |

1.2 **使用cgroups控制组**

正如之前提到的，cgroups提供了对进程和进程组资源限制的广泛支持。例如，你可以如下设置CPU的使用份额（shares）：

|  |
| --- |
| Bash sudo cgcreate -g cpu:/mycgroup sudo cgset -r cpu.shares=512 mycgroup sudo cgclassify -g cpu:mycgroup <PID> |

1.2.1 **文件夹位置**

/sys/fs/cgroup/

1.2.2 **cgroup 独自建立目录**

1.2.2.1 **挂载**

|  |
| --- |
| Bash #准备需要的目录 dev@ubuntu:~$ mkdir cgroup && cd cgroup dev@ubuntu:~/cgroup$ mkdir demo  #由于name=demo的cgroup树不存在，所以系统会创建一颗新的cgroup树，然后挂载到demo目录 dev@ubuntu:~/cgroup$ sudo mount -t cgroup -o none,name=demo demo ./demo  #挂载点所在目录就是这颗cgroup树的root cgroup，在root cgroup下面，系统生成了一些默认文件 dev@ubuntu:~/cgroup$ ls ./demo/ cgroup.clone\_children cgroup.procs cgroup.sane\_behavior notify\_on\_release release\_agent tasks  #cgroup.procs里包含系统中的所有进程 dev@ubuntu:~/cgroup$ wc -l ./demo/cgroup.procs 131 ./demo/cgroup.procs |

注意卸载 umount ./demo

1.2.2.2 **删除**

|  |
| --- |
| Bash #创建子cgroup很简单，新建一个目录就可以了 dev@ubuntu:~/cgroup$ cd demo dev@ubuntu:~/cgroup/demo$ sudo mkdir cgroup1  #在新创建的cgroup里面，系统默认也生成了一些文件，这些文件的意义和root cgroup里面的一样 dev@ubuntu:~/cgroup/demo$ ls cgroup1/ cgroup.clone\_children cgroup.procs notify\_on\_release tasks  #新创建的cgroup里没有任何进程和线程 dev@ubuntu:~/cgroup/demo$ wc -l cgroup1/cgroup.procs 0 cgroup1/cgroup.procs dev@ubuntu:~/cgroup/demo$ wc -l cgroup1/tasks 0 cgroup1/tasks  #每个cgroup都可以创建自己的子cgroup，所以我们也可以在cgroup1里面创建子cgroup dev@ubuntu:~/cgroup/demo$ sudo mkdir cgroup1/cgroup11 dev@ubuntu:~/cgroup/demo$ ls cgroup1/cgroup11 cgroup.clone\_children cgroup.procs notify\_on\_release tasks  #删除cgroup也很简单，删除掉相应的目录就可以了 dev@ubuntu:~/cgroup/demo$ sudo rmdir cgroup1/ rmdir: failed to remove 'cgroup1/': Device or resource busy #这里删除cgroup1失败，是因为它里面包含了子cgroup，所以不能删除， #如果cgroup1包含有进程或者线程，也会删除失败  #先删除cgroup11，再删除cgroup1就可以了 dev@ubuntu:~/cgroup/demo$ sudo rmdir cgroup1/cgroup11/ dev@ubuntu:~/cgroup/demo$ sudo rmdir cgroup1/ |

1.2.2.3 **添加进程**

创建新的cgroup后，就可以往里面添加进程了。注意下面几点：

* 在一颗cgroup树里面，一个进程必须要属于一个cgroup。
* 新创建的子进程将会自动加入父进程所在的cgroup。
* 从一个cgroup移动一个进程到另一个cgroup时，只要有目的cgroup的写入权限就可以了，系统不会检查源cgroup里的权限。
* 用户只能操作属于自己的进程，不能操作其他用户的进程，root账号除外。

|  |
| --- |
| Bash #--------------------------第一个shell窗口---------------------- #创建一个新的cgroup dev@ubuntu:~/cgroup/demo$ sudo mkdir test dev@ubuntu:~/cgroup/demo$ cd test  #将当前bash加入到上面新创建的cgroup中 dev@ubuntu:~/cgroup/demo/test$ echo $$ 1421 dev@ubuntu:~/cgroup/demo/test$ sudo sh -c 'echo 1421 > cgroup.procs' #注意：一次只能往这个文件中写一个进程ID，如果需要写多个的话，需要多次调用这个命令  #--------------------------第二个shell窗口---------------------- #重新打开一个shell窗口，避免第一个shell里面运行的命令影响输出结果 #这时可以看到cgroup.procs里面包含了上面的第一个shell进程 dev@ubuntu:~/cgroup/demo/test$ cat cgroup.procs 1421  #--------------------------第一个shell窗口---------------------- #回到第一个窗口，运行top命令 dev@ubuntu:~/cgroup/demo/test$ top #这里省略输出内容  #--------------------------第二个shell窗口---------------------- #这时再在第二个窗口查看，发现top进程自动和它的父进程（1421）属于同一个cgroup dev@ubuntu:~/cgroup/demo/test$ cat cgroup.procs 1421 16515 dev@ubuntu:~/cgroup/demo/test$ ps -ef|grep top dev 16515 1421 0 04:02 pts/0 00:00:00 top dev@ubuntu:~/cgroup/demo/test$  #在一颗cgroup树里面，一个进程必须要属于一个cgroup， #所以我们不能凭空从一个cgroup里面删除一个进程，只能将一个进程从一个cgroup移到另一个cgroup， #这里我们将1421移动到root cgroup dev@ubuntu:~/cgroup/demo/test$ sudo sh -c 'echo 1421 > ../cgroup.procs' dev@ubuntu:~/cgroup/demo/test$ cat cgroup.procs 16515 #移动1421到另一个cgroup之后，它的子进程不会随着移动  #--------------------------第一个shell窗口---------------------- ##回到第一个shell窗口，进行清理工作 #先用ctrl+c退出top命令 dev@ubuntu:~/cgroup/demo/test$ cd .. #然后删除创建的cgroup dev@ubuntu:~/cgroup/demo$ sudo rmdir test |

1.2.2.4 **权限**

|  |
| --- |
| Bash #创建一个新的cgroup，并修改他的owner dev@ubuntu:~/cgroup/demo$ sudo mkdir permission dev@ubuntu:~/cgroup/demo$ sudo chown -R dev:dev ./permission/  #1421原来属于root cgroup，虽然dev没有root cgroup的权限，但还是可以将1421移动到新的cgroup下， #说明在移动进程的时候，系统不会检查源cgroup里的权限。 dev@ubuntu:~/cgroup/demo$ echo 1421 > ./permission/cgroup.procs  #由于dev没有root cgroup的权限，再把1421移回root cgroup失败 dev@ubuntu:~/cgroup/demo$ echo 1421 > ./cgroup.procs -bash: ./cgroup.procs: Permission denied  #找一个root账号的进程 dev@ubuntu:~/cgroup/demo$ ps -ef|grep /lib/systemd/systemd-logind root 839 1 0 01:52 ? 00:00:00 /lib/systemd/systemd-logind #因为该进程属于root，dev没有操作它的权限，所以将该进程加入到permission中失败 dev@ubuntu:~/cgroup/demo$ echo 839 >./permission/cgroup.procs -bash: echo: write error: Permission denied #只能由root账号添加 dev@ubuntu:~/cgroup/demo$ sudo sh -c 'echo 839 >./permission/cgroup.procs'  #dev还可以在permission下创建子cgroup dev@ubuntu:~/cgroup/demo$ mkdir permission/c1 dev@ubuntu:~/cgroup/demo$ ls permission/c1 cgroup.clone\_children cgroup.procs notify\_on\_release tasks  #清理 dev@ubuntu:~/cgroup/demo$ sudo sh -c 'echo 839 >./cgroup.procs' dev@ubuntu:~/cgroup/demo$ sudo sh -c 'echo 1421 >./cgroup.procs' dev@ubuntu:~/cgroup/demo$ rmdir permission/c1 dev@ubuntu:~/cgroup/demo$ sudo rmdir permission |

1.2.2.5 **任务**

**cgroup.procs vs tasks**

|  |
| --- |
| Bash #创建两个新的cgroup用于演示 dev@ubuntu:~/cgroup/demo$ sudo mkdir c1 c2  #为了便于操作，先给root账号设置一个密码，然后切换到root账号 dev@ubuntu:~/cgroup/demo$ sudo passwd root dev@ubuntu:~/cgroup/demo$ su root root@ubuntu:/home/dev/cgroup/demo#  #系统中找一个有多个线程的进程 root@ubuntu:/home/dev/cgroup/demo# ps -efL|grep /lib/systemd/systemd-timesyncd systemd+ 610 1 610 0 2 01:52 ? 00:00:00 /lib/systemd/systemd-timesyncd systemd+ 610 1 616 0 2 01:52 ? 00:00:00 /lib/systemd/systemd-timesyncd #进程610有两个线程，分别是610和616  #将616加入c1/cgroup.procs root@ubuntu:/home/dev/cgroup/demo# echo 616 > c1/cgroup.procs #由于cgroup.procs存放的是进程ID，所以这里看到的是616所属的进程ID（610） root@ubuntu:/home/dev/cgroup/demo# cat c1/cgroup.procs 610 #从tasks中的内容可以看出，虽然只往cgroup.procs中加了线程616， #但系统已经将这个线程所属的进程的所有线程都加入到了tasks中， #说明现在整个进程的所有线程已经处于c1中了 root@ubuntu:/home/dev/cgroup/demo# cat c1/tasks 610 616  #将616加入c2/tasks中 root@ubuntu:/home/dev/cgroup/demo# echo 616 > c2/tasks  #这时我们看到虽然在c1/cgroup.procs和c2/cgroup.procs里面都有610， #但c1/tasks和c2/tasks中包含了不同的线程，说明这个进程的两个线程分别属于不同的cgroup root@ubuntu:/home/dev/cgroup/demo# cat c1/cgroup.procs 610 root@ubuntu:/home/dev/cgroup/demo# cat c1/tasks 610 root@ubuntu:/home/dev/cgroup/demo# cat c2/cgroup.procs 610 root@ubuntu:/home/dev/cgroup/demo# cat c2/tasks 616 #通过tasks，我们可以实现线程级别的管理，但通常情况下不会这么用， #并且在cgroup V2以后，将不再支持该功能，只能以进程为单位来配置cgroup  #清理 root@ubuntu:/home/dev/cgroup/demo# echo 610 > ./cgroup.procs root@ubuntu:/home/dev/cgroup/demo# rmdir c1 root@ubuntu:/home/dev/cgroup/demo# rmdir c2 root@ubuntu:/home/dev/cgroup/demo# exit exit |

1.2.3 **orin实际操作**

1、找到合适的CPU子系统目录  
你需要知道你的系统CGroup的CPU子系统挂载在。可以通过找 /procounts 文件来实，或使用 mount 和 grep 命令来查找，例如：

|  |
| --- |
| Bash mount | grep cgroup | grep cpu |



Orin 在 /sys/fs/cgroup/cpu目录操作

2、创建新的cgroup  
你可以在CPU子系统的目录中创建一个新的目录来创建一个新的cgroup。比如说，如果CPU子系统挂载在 /sys/fs/cgroup/cpu，你可以执行：

|  |
| --- |
| Bash sudo mkdir /sys/fs/cgroup/cpu/mygroup |

3、设置CPU限制  
你可以通过向cpu.shares文件写入来设定CPU份额限制。例如，为了设定mygroup的CPU限制，你可以执行：

|  |
| --- |
| Bash echo 512 | sudo tee /sys/fs/cgroup/cpu/mygroup/cpu.shares |

这里的512是一个相对值，其会与系统中所有其他cgroup的cpu.shares值一起考虑来计算实际的CPU时间分配。

如果你想要设定一个更严格的CPU时间限制，你还可以使用cpu.cfs\_quota\_us和cpu.cfs\_period\_us参数。

|  |
| --- |
| Bash echo 100000 | sudo tee /sys/fs/cgroup/cpu/mygroup/cpu.cfs\_period\_us echo 20000 | sudo tee /sys/fs/cgroup/cpu/mygroup/cpu.cfs\_quota\_us |

4、添加进程到cgroup  
将进程的PID添加到tasks文件来将其移至新的cgroup：

|  |
| --- |
| Bash echo <PID> | sudo tee -a /sys/fs/cgroup/cpu/mygroup/tasks |

进程退出自动消失

请注意上述在多线程限制的是多线需要将线程号写入

1.3 **nice工具**

使用nice和renice命令调整进程优先级

这些命令不会限制CPU使用的百分比，但会影响进程在CPU资源竞争中的优先级。nice用于启动新进程，而renice用于已经运行的进程：

|  |
| --- |
| Bash nice -n <优先级> <命令> 优先级范围 -20（最高优先级）到19（最低优先级） renice -n <优先级> -p <PID> |

1.4 **taskset**

使用taskset命令指定CPU亲和性

可以使用taskset命令将进程绑定到特定的CPU或CPU集，这样进程就只能在指定的CPU上运行，从而实际限制它的CPU使用率。

|  |
| --- |
| Bash taskset -cp <CPU范围> <PID> |

1.5 **systemd**

使用systemd进行资源管理

如果系统使用systemd作为启动管理器，你也可以通过配置服务文件来限制服务或进程的CPU使用。在服务的.service文件中，可以使用CPUQuota指令：

|  |
| --- |
| TOML [Service] CPUQuota=50% |

方案可行

1.6 **线程优先级**

对于新启动的进程，可以使用 nice 命令指定启动时的 nice 值：

|  |
| --- |
| Bash nice -n -2 command |

上面的命令会启动一个新的 command 进程，其 nice 值为 -2。

对于已经运行的进程，可以使用 renice 命令来修改其 nice 值：

|  |
| --- |
| Bash renice -n -2 -p <pid> |

1.7 **链接地址**

资料cgroup操作

1. [A.2. cpu Red Hat Enterprise Linux 7 | Red Hat Customer Portal](https://access.redhat.com/documentation/zh-cn/red_hat_enterprise_linux/7/html/resource_management_guide/sec-cpu)
2. [创建并管理cgroup](https://segmentfault.com/a/1190000007241437)

2. **gpu操作**

1. lspci 命令： lspci 是用来列出系统中所有 PCI 设备的命令。通常，GPU 会作为 PCI 设备出现在这个列表中。

|  |
| --- |
| Bash lspci | grep -i vga 查找所有的 VGA 兼容控制器 lspci | grep -i nvidia 如果是 NVIDIA GPU，可以直接搜索 lspci | grep -i amd 如果是 AMD GPU，可以直接搜索 |

1. lshw 命令： lshw 是一个显示设备硬件配置详细信息的工具。它可以显示更详细的硬件信息，包括制造商、型号等。

|  |
| --- |
| Bash sudo lshw -numeric -C display 查看显示适配器硬件信息，可能需要 root 权限 |

1. nvidia-smi / amdconfig 命令： 如果安装了相应的驱动程序，可以使用制造商提供的工具来查看 GPU 信息。
2. 对于 NVIDIA GPU：

|  |
| --- |
| Bash nvidia-smi 显示 NVIDIA GPU 的状态和信息 |

1. 对于 AMD GPU：

|  |
| --- |
| Bash amdconfig --adapter=all --list-adapters 显示 AMD GPU 的信息，可能需要根据实际安装的驱动版本进行调整 |

1. 查看 /proc 和 /sys 文件系统： /proc 和 /sys 文件系统包含了很多关于系统硬件和驱动的实时信息。

|  |
| --- |
| Bash cat /proc/driver/nvidia/gpus/\*/information 查看 NVIDIA GPU 信息，如果有的话 |