

# 容器技术基石：Linux namespace 和 cgroups，运维了解一下

点击关注  马哥Linux运维



关注蓝字



收获每日技术干货

匠 心 精 神 良 心 教 育

先放结论，namespace 是用来做资源隔离，cgroup 是用来做资源限制。

## Namespace

先说Namespace，虚拟技术基本要求就是资源隔离，简单的说就是我独占当前所有的资源。比如我在 8080 端口起 web 服务器，不用担心其他进程端口占用。Linux 自带 namespace 就能达到这个目的。namespace 从2002 开始开发到现在已经快20年的历史了，到现在一共有6种 namespace：

- mnt，文件系统
- pid，进程
- net，网络
- ipc，系统进程通信
- uts，hostname
- user，用户

可以通过三个系统调用的方式

- clone，创建新的进程和新的namespace，新创建的进程 attach 到新创建的 namespace
- unshare，不创建新的进程，创建新的 namespace 并把当前进程 attach 上
- setns，attach 进程到已有的 namespace 上

shell 也提供了一个和系统调用同名的 `unshare` 命令可以非常简单的创建 namespace。

```
1 sudo unshare --fork --pid --mount-proc bash
```

这样创建了一个新的 PID namespace 并在里面运行了 `bash`。我们看看当前 namespace 的进程

```
root@ubuntu-xenial:/# ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  19940  3840 pts/0    S   05:33   0:00 bash
root       15  0.0  0.0  36056  3288 pts/0    R+  05:40   0:00 ps aux
root@ubuntu-xenial:/#
```

在这个 namespace 里，就只有两个进程了。

## Cgroups

cgroups 是 control groups 控制组的意思，可以通过文件系统来访问这些信息。一般 cgroups 挂载在 `/sys/fs/cgroup`

```
vagrant@ubuntu-xenial:/sys/fs/cgroup$ ls -al
total 0
drwxr-xr-x 13 root root 340 Jan 17 05:32 .
drwxr-xr-x 10 root root  0 Jan 17 05:32 ..
dr-xr-xr-x  5 root root  0 Jan 17 05:32 blkio
lrwxrwxrwx  1 root root 11 Jan 17 05:32 cpu -> cpu,cpuacct
dr-xr-xr-x  5 root root  0 Jan 17 05:32 cpu,cpuacct
lrwxrwxrwx  1 root root 11 Jan 17 05:32 cpuacct -> cpu,cpuacct
dr-xr-xr-x  2 root root  0 Jan 17 05:32 cpuset
dr-xr-xr-x  5 root root  0 Jan 17 05:32 devices
dr-xr-xr-x  2 root root  0 Jan 17 05:32 freezer
dr-xr-xr-x  2 root root  0 Jan 17 05:32 hugetlb
dr-xr-xr-x  5 root root  0 Jan 17 05:32 memory
lrwxrwxrwx  1 root root 16 Jan 17 05:32 net_cls -> net_cls,net_prio
dr-xr-xr-x  2 root root  0 Jan 17 05:32 net_cls,net_prio
lrwxrwxrwx  1 root root 16 Jan 17 05:32 net_prio -> net_cls,net_prio
dr-xr-xr-x  2 root root  0 Jan 17 05:32 perf_event
dr-xr-xr-x  5 root root  0 Jan 17 05:32 pids
dr-xr-xr-x  5 root root  0 Jan 17 05:32 systemd
```

知乎 @Vincent

内核会读取这些信息来调度资源分配给每个进程。比如我要限制进程占用CPU的时间。我用 Go 写了一个模拟高 CPU 的代码。

```

1 func IsPrime(value int) bool {
2     for i := 2; i <= int(math.Floor(float64(value)/2)); i++
3     {
4         if value%2 == 0
5         {
6             return false
7         }
8     }
9     return true
10 }
11
12 func main() {
13     for i := 0; i < 999999999; i++
14     {
15         fmt.Printf("%v is prime: %v\n", i,
16             IsPrime(i))
17     }
18 }

```

我创建两个 CPU 的 cgroups

```

1 sudo cgcreate -g
2 cpu:cpulimited
sudo cgcreate -g
cpu:/lesscpulimited

```

cpu.shares 是给内核为每个进程决定 CPU 计算资源，默认值是1024。给 cpulimited 设置为 512，lesscpulimited 保留默认值，那么在这两个组的进程会以1 : 2的比例占用CPU。

```

1 sudo cgset -r cpu.shares=512 cpulimited

```

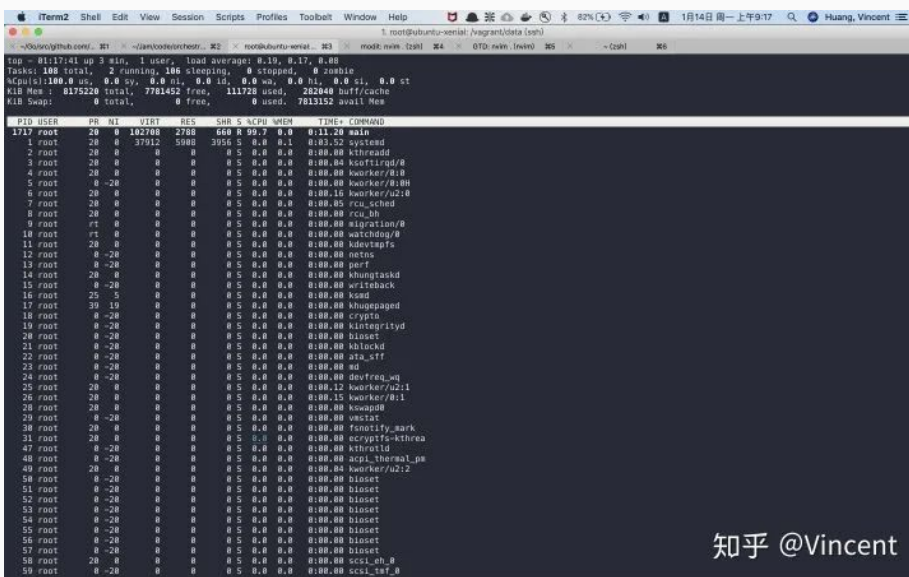
```
vagrant@ubuntu-xenial:/sys/fs/cgroup/cpu/cpulimited$ ls
cgroup.clone_children  cgroup.procs  cpu.cfs_quota_us  cpu.cfs_period_us  cpu.shares  cpu.stat  cpucact.stat  cpucact.usage  cpucact.usage_percpu  notify_on_release  tasks
vagrant@ubuntu-xenial:/sys/fs/cgroup/cpu/cpulimited$ cat cpu.shares
512
vagrant@ubuntu-xenial:/sys/fs/cgroup/cpu/cpulimited$
```

我们来验证一下。

在 cpulimited 起一个进程

```
1 sudo cgexec -g cpu:cpulimited ./main > /dev/null
&
```

可以看到独占了 100% 的 CPU，在 cpulimited 再起一个进程



```
top - 01:17:41 up 3 min, 1 user, load average: 0.19, 0.17, 0.08
Tasks: 186 total, 2 running, 186 sleeping, 0 stopped, 0 zombie
MiB Mem : 8175220 total, 7781452 free, 111728 used, 262040 buff/cache
KiB Swap: 0 total, 0 free, 0 used, 7813152 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME	COMMAND
1	root	20	0	102760	2720	608	S	0.0	0.0	0:11:50	main
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	systemd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kernelthread
4	root	20	0	0	0	0	S	0.0	0.0	0:00.00	scsi_tiered0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	worker/0:0
6	root	20	0	0	0	0	S	0.0	0.0	0:00.15	worker/0:8
7	root	20	0	0	0	0	S	0.0	0.0	0:00.05	rcu_sched
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	rtmon/0
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	udevdsfs
12	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	netns
13	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	perf
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
15	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	writeback
16	root	25	5	0	0	0	S	0.0	0.0	0:00.00	kssd
17	root	30	10	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	cryptd
19	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	integrityd
20	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	binset
21	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	khlockd
22	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	ata_sff
23	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	sd
24	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	dmefreq_wq
25	root	20	0	0	0	0	S	0.0	0.0	0:00.12	worker/0:1
26	root	20	0	0	0	0	S	0.0	0.0	0:00.15	worker/0:11
28	root	20	0	0	0	0	S	0.0	0.0	0:00.00	ksmcpd
29	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	ext4
30	root	20	0	0	0	0	S	0.0	0.0	0:00.00	fnotify_mark
31	root	20	0	0	0	0	S	0.0	0.0	0:00.00	ccryptfs-kthrea
47	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	khlockd
48	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	acpi_thermal_ps
49	root	20	0	0	0	0	S	0.0	0.0	0:00.00	worker/u2:2
50	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	binset
51	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	binset
52	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	binset
53	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	binset
54	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	binset
55	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	binset
56	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	binset
57	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	binset
58	root	20	0	0	0	0	S	0.0	0.0	0:00.00	ccci_wq_0
59	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	ccci_wq_0

两个进程都在 cpulimited，各占50%的 CPU。在 lesscpulimited 起一个进程

```

1 root@ubuntu-xenial:~$ top
top - 01:17:53 up 4 min, 1 user, load averages: 0.22, 0.24, 0.11
Tasks: 109 total, 3 running, 106 sleeping, 0 stopped, 0 zombie
%cpu(s): 99.3 us, 0.7 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 817520 total, 777048 free, 113124 used, 282448 buff/cache
KiB Swap: 0 total, 0 free, 0 used, 7609752 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  %CPU  MEM%   TIME+  COMMAND
1717 root        20   0 102708 3568 660 R 49.8  0.0  0:20.91 main
1721 root        20   0 102708 2088 660 R 49.0  0.0  0:02.99 main
1 root        20   0 37912 5968 5 0.0 0.1 0:03.52 systemd
2 root        20   0 0 0 0 0 0.0 0.0 0:00.00 kthreadd
3 root        20   0 0 0 0 0 0.0 0.0 0:00.04 ksrtirqd/0
4 root        20   0 0 0 0 0 0.0 0.0 0:00.00 kworker/0:0
5 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 kworker/0:0H
6 root        20   0 0 0 0 0 0.0 0.0 0:00.15 kworker/u2:0
7 root        20   0 0 0 0 0 0.0 0.0 0:00.05 rcu_sched
8 root        20   0 0 0 0 0 0.0 0.0 0:00.00 rcu_bh
9 root        rt    0 0 0 0 0 0.0 0.0 0:00.00 expiration/0
10 root       rt    0 0 0 0 0 0.0 0.0 0:00.00 watchdog/0
11 root        20   0 0 0 0 0 0.0 0.0 0:00.00 kdevtmpfs
12 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 netns
13 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 perf
14 root        20   0 0 0 0 0 0.0 0.0 0:00.00 khungtaskd
15 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 writeback
16 root        25  5 0 0 0 0 0.0 0.0 0:00.00 ksmd
17 root        30  19 0 0 0 0 0.0 0.0 0:00.00 khupgaged
18 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 crypto
19 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 kintegrityd
20 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
21 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 kblockd
22 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 ata_sff
23 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 md
24 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 devfreq_wq
25 root        20   0 0 0 0 0 0.0 0.0 0:00.12 kworker/u2:1
26 root        20   0 0 0 0 0 0.0 0.0 0:00.15 kworker/0:1
28 root        20   0 0 0 0 0 0.0 0.0 0:00.00 kswapd0
29 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 unstat
30 root        20   0 0 0 0 0 0.0 0.0 0:00.00 fnotify_mark
31 root        20   0 0 0 0 0 0.0 0.0 0:00.00 ecryptfs-kthrea
47 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 kthrotld
48 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 acpi_thermal_pm
49 root        20   0 0 0 0 0 0.0 0.0 0:00.04 kworker/u2:2
50 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
51 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
52 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
53 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
54 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
55 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
56 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
57 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
58 root        20   0 0 0 0 0 0.0 0.0 0:00.00 ccsl_ch_0
59 root        20   0 0 0 0 0 0.0 0.0 0:00.00 ccsl_ch_0
```

知乎 @Vincent

1 sudo cgexec -g cpu:lesscpulimited ./main > /dev/null

&

```

1 vagrant@ubuntu-xenial:~$ top
top - 08:20:36 up 2:17, 2 users, load averages: 2.13, 0.55, 0.19
Tasks: 110 total, 4 running, 106 sleeping, 0 stopped, 0 zombie
%cpu(s): 100.0 us, 0.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 817520 total, 741936 free, 124700 used, 638484 buff/cache
KiB Swap: 0 total, 0 free, 0 used, 7791280 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  %CPU  MEM%   TIME+  COMMAND
9087 root        20   0 102708 3568 660 R 16.8  0.0  0:21.55 main
9091 root        20   0 102708 2088 660 R 16.9  0.0  0:03.77 main
9083 root        20   0 102708 2264 660 R 16.6  0.0  0:05.92 main
8772 vagrant    20   0 93184 5816 3944 S 0.3 0.1 0:00.30 sshd
1 root        20   0 37688 5968 5 0.0 0.1 0:04.76 systemd
2 root        20   0 0 0 0 0 0.0 0.0 0:00.00 kthreadd
3 root        20   0 0 0 0 0 0.0 0.0 0:00.11 ksrtirqd/0
5 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 kworker/0:0H
7 root        20   0 0 0 0 0 0.0 0.0 0:00.18 rcu_sched
8 root        20   0 0 0 0 0 0.0 0.0 0:00.00 rcu_bh
9 root        rt    0 0 0 0 0 0.0 0.0 0:00.00 expiration/0
10 root       rt    0 0 0 0 0 0.0 0.0 0:00.06 watchdog/0
11 root        20   0 0 0 0 0 0.0 0.0 0:00.00 kdevtmpfs
12 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 netns
13 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 perf
14 root        20   0 0 0 0 0 0.0 0.0 0:00.00 khungtaskd
15 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 writeback
16 root        25  5 0 0 0 0 0.0 0.0 0:00.00 ksmd
17 root        30  19 0 0 0 0 0.0 0.0 0:00.00 khupgaged
18 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 crypto
19 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 kintegrityd
20 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
21 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 kblockd
22 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 ata_sff
23 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 md
24 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 devfreq_wq
25 root        20   0 0 0 0 0 0.0 0.0 0:00.27 kworker/u2:1
26 root        20   0 0 0 0 0 0.0 0.0 0:00.00 kswapd0
29 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 unstat
30 root        20   0 0 0 0 0 0.0 0.0 0:00.00 fnotify_mark
31 root        20   0 0 0 0 0 0.0 0.0 0:00.00 ecryptfs-kthrea
47 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 kthrotld
48 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 acpi_thermal_pm
50 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
51 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
52 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
53 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
54 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
55 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
56 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
57 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 bioct
58 root        20   0 0 0 0 0 0.0 0.0 0:00.00 ccsl_ch_0
59 root        0 -20 0 0 0 0 0.0 0.0 0:00.00 ccsl_tf_0
```

知乎 @Vincent

两个 cpulimited 进程的 CPU 之和 与 一个 lesscpulimited 进程的 CPU 差不多就是 1 : 2的关系。

来源 : <https://zhuanlan.zhihu.com/p/55099839>

文章转载：高效运维

(版权归原作者所有，侵删)





马霄教育  
匠心精神 · 真心品质



# 运维核心技能全get

## 精选优质课程

授课老师：王晓春

### 轻松玩转ansible，实现企业级运维自动化



课时：6h+  
原价：298  
限时拼团 ¥ 0.02元

- ansible 介绍和架构
- Ansible Playbook介绍
- Ansible Playbook其他高级用法



### iptables防火墙从入门到精通



课时：6h+  
原价：9.9  
限时拼团 ¥ 0.02元

- iptables的扩展模块用法
- iptables实现网络防火墙
- iptables实现NAT原理和实战



### LNMP网站架构实战训练营



课时：5h+  
原价：69  
限时拼团 ¥ 0.02元

- 实现LNMP个人博客站点实操
- 自动化部署LNMP企业站实战
- 10倍效率实现Docker网站架构





点击下方“阅读原文”查看更多

阅读原文