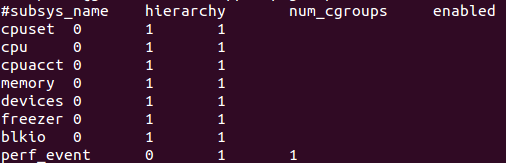
cgroups

1. cgroups 简介

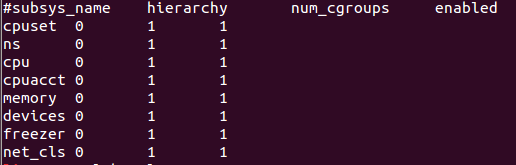
cgroups是control groups的简称，它是Linux内核级别支持的一种系统资源分配机制。

2. cgroups 的子系统（subsystems）

cgroups包含了多个孤立的子系统，每一个子系统代表了单一的系统资源。目前，redhat默认支持10个子系统，但默认只挂载了8个，ubuntu 12.04 默认支持8个子系统，但默认只挂载了5个。



*Ubuntu 12.04.2 LTS 支持的子系统*



*SUSE Enterprise Linux Server 11 支持的子系统（目前测试环境只有这个系统支持）*

2.1 redhat enterprise 6 中可用的子系统及其描述：

blkio – 这个子系统为块设备设定输入/输出限制，比如物理设备（磁盘，固态硬盘，USB 等等）。

cpu – 这个子系统使用调度程序提供对CPU的cgroups任务访问。

cpuacct – 这个子系统自动生成cgroups中任务所使用的CPU报告。

cpuset – 这个子系统为cgroups中的任务分配独立CPU（在多核系统）和内存节点。

devices – 这个子系统可允许或者拒绝cgroups中的任务访问设备。

freezer – 这个子系统负责挂起或者恢复cgroups中的任务。

memory – 这个子系统设定cgroups中任务使用的内存限制， 并且自动生成由那些任务使用的内存资源报告。

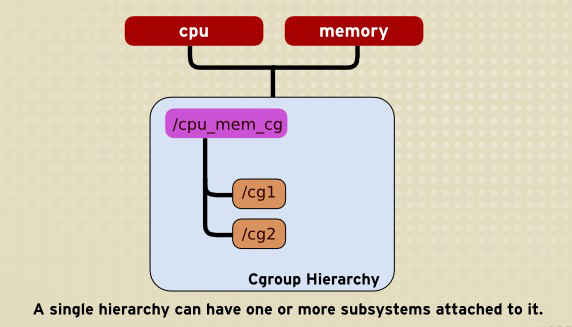
net\_cls – 这个子系统使用等级标识符（classid）标记网络数据包，可允许Linux流量控制程序（tc）识别从具体cgroups中生成的数据包。

ns – 名称空间（namespace）子系统。

3. 子系统（subsystem）、层级（hierarchy）、控制群组（control groups）和任务（tasks）的关系

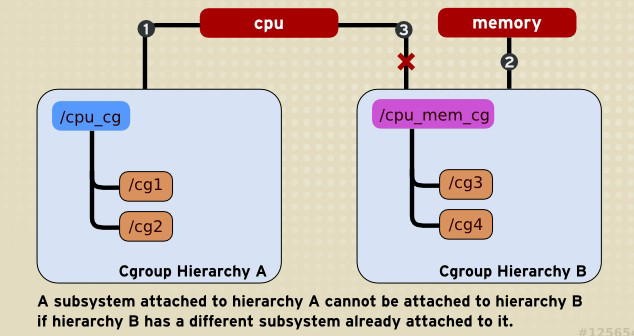
cgroups 在内核代码里有如下4点主要限制，只要遵循这些限制关系，就可以成功的配置cgroups工作，但要科学有效的配置cgroups，还需要合理的层级设计和资源分配。一般来说，就像Linux操作系统本身，一个层级（Linux /sbin/init）挂载所有子系统，在这个层级下建立不同的控制组群以分配系统资源。也可以是一个子系统挂载一个层级，再进行系统资源分配，这种模式是libcgroup管理的默认模式，同时，lxc也是在这个架构基础上分配各个子系统资源的。

3.1 一个层级可以有一个或者多个子系统附加在其之上。



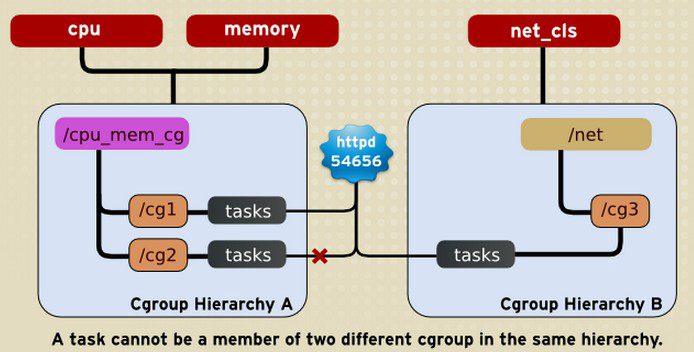
结果是，CPU和memory子系统（或者其他任何子系统）都可以挂载到一个单独的层级，只要这些子系统没有挂载到有其他子系统已经挂载了的层级。

3.2 一个子系统（比如CPU）不能挂载到一个已经挂载了其他子系统的层级。



结果是，CPU子系统不可以挂载到2个不同的层级，如果其中一个层级已经有memory子系统挂载在上面。然而，任何一个子系统都可以挂载到2个层级，条件是这2个层级都只挂载了这个单一的子系统。

3.3 每次在系统中创建新的层级时，该系统中的所有任务都是那个层级的默认cgroup（root cgroup）的初始成员。对于用户创建的任何单一层级，系统中的每个任务都可以是这个层级中唯一一个cgroup的成员。单一任务可以存在于多个cgroup中，只要每个cgroup都在不同的层级中即可。如果某个任务成为了同一个层级中第二个cgroup的成员，那么他会自动被那个层级的第一个cgroup删除。一个任务永远不会同时位于同一个层级中的不同cgroup中。

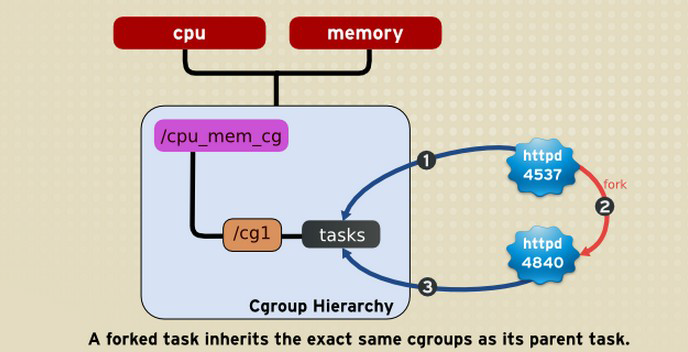


结果是（如上图），如果CPU和memory子系统都挂载到名为cpu\_and\_mem的这个层级上，且net\_cls子系统是挂载到net这个层级上，那么运行的httpd进程可以是cpu\_and\_mem中任意cgroup的成员，是也是net中任意一个cgroup的成员。

例如，httpd进程所在cpu\_and\_mem中的cgroup可将其CPU时间限制为分配给其他进程时间的一半，并将其内存用量限制为最多1GB。另外，net中的cgroup还可将其传输速率限制为最大30MB/秒。

初次创建层级时，系统中每个任务都至少是一个cgroup的成员，即root cgroup。因此每当使用cgroup时，每个系统任务总是至少在一个cgroup中。

3.4 系统中的任意进程（任务）为自己分支创建子进程（任务）。该子任务自动成为其父进程所在cgroup的成员。然后可根据需要将该子任务移动到不同的cgroup中，但开始时，他总是继承父任务的cgroup。



cpu\_and\_mem层级中，cg1控制群组被约束为一半CPU使用和1GB内存。那么当httpd进程将自身分成几个分支时，他自动成为cg1控制群组的成员。

此后，父进程和子进程会相互独立，更改某个任务所属cgroup不会影响到另外一个。总之，所有子任务总是继承父任务的cgroup成员关系，但之后可随意更改或者删除。

4. 资源管理实施

4.1 Linux内核的实施

一个新的文件系统（cgroup）需要被装载，这个文件系统用来浏览、修改cgroups资源分配。Linux内核文档中明确指出，cgroups没有添加新的系统调用（System Calls），所有的操作都通过修改cgroup文件系统来实现。比如新建一个cgroup，就可以通过'mkdir'这个系统调用或者shell command来实现。

因此，每一个cgroup都以一个系统文件夹（directory）的形式出现，并且有以下文件来形容这个cgroup；

- tasks: 这个文件包括所有任务的Process ID。这个表不保证顺序。当任务pid写入这个列表的时候，这个任务就被加入到这个控制群组里了。

- cgroup.procs: 这个文件之包括任务组的ID。

- notify\_on\_release flag: 这个标志着是否触发release\_agent。

- release\_agent: 这个指明了释放通知的路径。

4.1.1 Linux内核cgroups的基本应用

1) mount -t tmpfs cgroup\_root /sys/fs/cgroup

挂载相应的文件系统，注意type最好是tmpfs，这种虚拟内存模式。这个在支持cgroups的Linux系统中，这个路径已经存在了，所以不需要额外mkdir，直接挂载就可以了。

2) mkdir /sys/fs/cgroup/cpuset

这样就简单的建立了一个层级。

3) mount -t cgroup -o cpuset cpuset /sys/fs/cgroup/cpuset

现在就把cpuset这个子系统，挂载到建立好的这个层级上了。

4) cd /sys/fs/cgroup/cpuset && mkdir chinasofti && cd chinasofti

这样就在cputset这个层级下建立了一个名为chinasofti的控制群组。

5) echo 0-2,8 > cpuset.cpus

这就意味在这个控制群组中，所有任务只能访问CPU 0,1,2,8

6) echo $$ >> tasks

现在就把我们目前shell的进程加了进去，那么我们现在在这个shell启动的程序，ls啊，或者eclipse，或者是tomcat，都只能使用0,1,2,8 CPU了。

4.2 libcgroup工具

像其他内核特征一样，虽然cgroups没有增加任何新的系统调用，但还是可以通过libcgroup这个工具来实现管理的。

redhat系统工具安装：

yum install libcgroup

ubuntu系统工具安装：

sudo apt-get install cgroup-bin

这个工具的使用在不同Linux系统上的安装都大同小异，也可以直接到sourceforge checkout 源代码进行make install，如果系统默认系统打包服务不支持的话。

一般来说，启动这个服务

service cgconfig start|stop|restart|status

也可以到 /etc/init.d/cgconfig 这里找到这个脚本来启动服务

4.2.1 cgconfig实施

cgconfig的配置文件一般放置在

/etc/cgconfig.conf

这个文件的配置主要由mount和group两个section组成。

（1) Mount section的基本语法如下：

mount {

<controller> = <path>;

...

}

controller:就是内核子系统的名称

path：就是该系统的挂载点

例如：

mount {

cpuset = /sys/fs/cgroup/cpuset;

}

那么里这个配置就相当于内核级别对文件系统操作的第二和第三步。

（2) group section的语法如下：

group <name> {

[<permissions>]

<controller> {

<param name> = <param value>;

...

}

...

}

name: 指定cgroup的名称

permissions: 可选项，指定cgroup对应的挂载点文件系统的权限，root用户拥有所有权限。

controller: 子系统的名称

param name 和 param value: 子系统的属性和其属性值。

（3）具体点的一个例子，包括对权限的使用

mount { ## 定义需要创建的cgroup子系统及其挂载点，这里创建cpu与cpuacct（统计）两个cgroup子系统

cpu = /mnt/cgroups/cpu;

cpuacct = /mnt/cgroups/cpu;

}

group daemons/www { ## 定义daemons/www(web服务器进程)组

perm { ## 定义这个组的权限

task {

uid = root;

gid = webmaster;

}

admin {

uid = root;

gid = root;

}

}

cpu { ## 定义cpu子系统的属性及其值，即属于词组的任务的权重为1000

cpu.shares = 1000;

}

}

group daemons/ftp { ## 定义daemons/ftp(ftp进程)组

perm {

task {

uid = root;

gid = ftpmaster;

}

admin {

uid = root;

gid = root;

}

}

cpu { ## 定义词组的任务的权重为500

cpu.shares = 500;

}

}

这样一组配置就等同于执行了如下一组shell命令：

mkdir /mnt/cgroups/cpu

mount -t cgroup -o cpu,cpuacct cpu /mnt/cgroups/cpu

mkdir /mnt/cgroups/cpu/daemons

mkdir /mnt/cgroups/cpu/daemons/www

chown root:root /mnt/cgroups/cpu/daemons/www/\*

chown root:webmaster /mnt/cgroups/cpu/daemons/www/tasks

echo 1000 > /mnt/cgroups/cpu/daemons/www/cpu.shares

mkdir /mnt/cgroups/cpu/daemons/ftp

chown root:root /mnt/cgroups/cpu/daemons/ftp/\*

chown root:ftpmaster /mnt/cgroups/cpu/daemons/ftp/tasks

echo 500 > /mnt/cgroups/cpu/daemons/ftp/cpu.shares

4.2.2 为cgroups分配进程的几种方法

1. 直接写入相应cgroup的tasks文件

echo 1234 > /sys/fs/cgroup/cpuset/group-web/tasks

1. 使用cgclassify工具

cgclassify -g cpuset:group-web 1234

1. 使用cgexec启动需要分配的程序

cgexec -g cpuset:group-web /opt/tomcat/bin/startup.sh

1. cgred守护进程

cgred是一个守护进程，他是根据/etc/cgrules.conf来配置如果将任务移动到各cgroup中的，基本语法很简单，只有如下两种：

user hierarchies control\_group

user:[process name] hierarchies control\_group

<user>可以是：

- 用户名

- 组群名，但要遵循Linux组群语法特征@group

- 通配符\*，表示所有用户，或者组

- %，意思就是“同上”

<process name>是可选配的，可以是：

- 进程的名字

- 进程的绝对路径

<hierarchies>可以是：

- 逗号分隔的层级（注意不能有空格）

- \*代表所有层级

<control\_group>则是：

- 在这个层级的相对路径 (e.g. /usergroup/staff)

5. 附录

5.1 lxc – Linux Container

这个Linux容器顾名思义就是在Linux系统下实现若干个相互隔离的子Linux系统。类似于虚拟机，但不同的是他不会虚拟硬件，而是通过Linux内核的cgroups功能来实现宿主系统的资源共享。这样在性能上会有提升，但反之也只能隔离建立基于Linux内核的系统（容器，lxc terminology）。