

原 cgroup从入门到懵圈——cgroup概念

2018年06月26日 20:11:50 享乐主 阅读数: 2431

从今天起，我要开始写博客了。先立个flag在这儿：两周一篇。万事开头难，中间不容易，最后会放弃。请各位看官监督（虽然似乎肯定没有人看）。

之前有接触过cgroup，但东西都是同事在做（羡慕），理解不深。所以这个系列的文档，我假装自己是cgroup菜鸡（其实我不需要假装），从零开始学习cgroup。

cgroup是linux内核实现、用于控制linux系统资源的组件。因此要了解cgroup，首先从引入这个组件的kernel文档中寻找。去到举世闻名的www.kernel.org网站寻找，嘿，找到介绍文档：

Documentation/cgroup-v1/cgroups.txt

1 什么是cgroup？

cgroup，控制组，它提供了一套机制用于控制一组特定进程对资源的使用。cgroup绑定一个进程集合到一个或多个子系统上。

subsystem，子系统，一个通过cgroup提供的工具和接口来管理进程集合的模块。一个子系统就是一个典型的“资源控制器”，用来调度资源或者控制资源使用的上限。其实每种资源就是一个子系统。子系统可以是以进程为单位的任何东西，比如虚拟化子系统、内存子系统。

hierarchy，层级树，多个cgroup的集合，这些集合构成的树叫hierarchy。可以认为这是一个资源树，附着在这上面的进程可以使用的资源上限必须受树上节点（cgroup）的控制。hierarchy上的层次关系通过cgroupfs虚拟文件系统显示。系统允许多个hierarchy同时存在，每个hierarchy包含系统中的部分或者全部进程集合。

cgroupfs是用户管理操纵cgroup的主要接口：通过在cgroupfs文件系统中创建目录，实现cgroup的创建；通过向目录下的属性文件写入内容，设置cgroup对资源的控制；向task属性文件写入进程ID，可以将进程绑定到某个cgroup，以此达到控制进程资源使用的目的；也可以列出cgroup包含的进程pid。这些操作影响的是sysfs关联的hierarchy，对其它hierarchy没有影响。

对于cgroup，其本身的作用只是任务跟踪。但其它系统（比如cpuset，cpuacct），可以利用cgroup的这个功能实现一些新的属性，比如统计或者控制一个cgroup中进程可以访问的资源。举个例子，cpuset子系统可以将进程绑定到特定的cpu和内存节点上。

2 为什么需要cgroup？

这个问题相当于问cgroup重要吗？有哪些地方用到了。回答是重要，又不重要。如果你用到了，那就重要，如果没有用到，那就不重要。呵呵呵~~~~其实挺重要的。cgroup的主要运用是资源跟踪。我接触的场景就是用cgroup控制虚拟机进程或者docker进程可以使用的资源。当你在linux对应用进程做资源访问控制的时候，cgroup就派上用场了。

3 cgroup怎么实现的？

—— 系统中的每个进程（task_struct，后面用task代指）都持有一个指向css_set结构的指针。

—— 一个css_set结构体包含了一组指向cgroup_subsys_state对象的指针（所以一个task可以附加到多个cgroup上），每个cgroup_subsys_state在系统中都有注册。task结构体没有直接指向hierarchy中一个节点（cgroup）的指针。但可以通过其包含的cgroup_subsys_state间接确定。这样设计的原因是cpu对subsystem state的访问很频繁，但涉及到将task绑定到cgroup的操作却不多。task中还有个双向链表cg_list，这个链表维护所有同属于一个css_set的tasks。

—— 用户可以通过cgroupfs文件系统来浏览cgroup hierarchy。

—— 用户可以列出任意一个cgroup上附着的task PID

cgroup在kernel中除了本身功能的实现外，在kernel中还有两处修改：

—— 在kernel启动时对root cgroup的初始化和css_set结构体的初始化。这个在init/main.c文件中实现。

—— 在task的创建（fork）和退出（exit）阶段，对应地将task与css_set进行绑定和解绑。

另外，cgroup为了向用户提供操作接口，特别开发了一个虚拟文件系统类型（cgroupfs），这个文件系统与sysfs，proc类似。cgroupfs是向用户展示cgroup的hierarchy，通知kernel用户对cgroup改动的窗口。挂载cgroupfs时通过选项（-otype）指定要挂载的子系统类型，如果不指定，默认挂载所有的注册的子系统。

如果新挂载的cgroup关联的hierachy与系统中存在的hierarchy完全一样，那么cgroupfs会拒绝挂载。如果没有匹配到相同的hierarchy，但新挂载hierachy声明的资源正在被已经存在的hierarchy使用，挂载会报-EBUSY错误。

当前cgroup还没有实现向已经存在的cgroup hierarchy绑定新子系统的操作，将子系统从cgroup hierarchy解绑也不允许。这些操作在未来也许会支持，但也可能会进一步产生错误恢复的一系列问题。

卸载cgroupfs时，如果它的子cgroupfs还在活动，那么子cgroupfs还是会持续生效。直到所有的子cgroupfs不再活动，卸载cgroupfs才会真正生效。

cgroupfs下不能再挂载其它类型的文件系统。所有对cgroup的查询修改都只通过cgroupfs文件系统来完成。

系统中的所有task，在/proc/pid目录下都有一个名为cgroup的文件，这个文件展示了该task相对cgroupfs 根的路径。通过查看这个文件，可以了解一个进程在cgroup hierarchy的位置。以此得到task可以使用的资源信息。

cgroupfs中目录表示cgroup，每个目录在创建时默认生成如下的属性文件，这些文件描述了cgroup的信息：

—— tasks: 所有附属于这个cgroup的进程ID列表。tasks文件中增加进程ID，表示将进程加入这个cgroup，进程能够使用的资源受cgroup限制。

—— cgroup.procs: 所有附属于这个cgroup线程组ID，将TGID写入这个文件后，TGID所在进程包含的所有线程都加入这个cgroup，这些线程受cgroup限制。

- **PID**: 这是 Linux 中在其命名空间中唯一标识进程而分配给它的一个号码，称做进程ID号，简称PID。在使用 fork 或 clone 系统调用时产生的进程均会由内核分配一个新的唯一的PID值。
- **TGID**: 在一个进程中，如果以CLONE_THREAD标志来调用clone建立的进程就是该进程的一个线程，它们处于一个线程组，该线程组的ID叫做TGID。处于相同的线程组中的所有进程都有相同的TGID；线程组组长的TGID与其PID相同；一个进程没有使用线程，则其TGID与PID也相同。
- **PGID**: 另外，独立的进程可以组成进程组（使用setpgid系统调用），进程组可以简化向所有组内进程发送信号的操作，例如用管道连接的进程处在同一进程组内。进程组ID叫做PGID，进程组内的所有进程都有相同的PGID，等于该组组长的PID。
- **SID**: 几个进程组可以合并成一个会话组（使用setsid系统调用），可以用于终端程序设计。会话组中所有进程都有相同的SID。

—— notify_on_release flag: 标记退出时是否运行release agent

—— release_agent: 制定要运行的release agent的路径，这个属性文件只在cgroup的顶层目录中存在。

以上文件是每个cgroup基本的属性文件，对于不同的子系统，对应的cgroup可能会有其它附加的属性文件，存在于其对应的cgroup目录之下。

通过mkdir命令创建cgroup，通过向目录下的文件写入适当的数值设置修改cgroup的属性。

嵌套的cgroups，指定了层级结构，以此将系统资源划分成嵌套的，动态可变的更小的资源块。

一个进程可以附加到多个不同的cgroup中，只要这些cgroup不在同一个层级树上即可。因为cgroupfs会保证新挂载的cgroup关联的层级树全局唯一。子进程在被创建后默认附加到父进程所在的cgroup，后面用户可以根据需要将其移动到别的cgroup。

当进程从一个cgroup被移动到另一个cgroup。进程的task_struct会获取一个新的css_set指针：如果这个cgroup所在的css_set已经存在就重用这个css_set，否则就新分配一个css_set。kernel会在全局的hash表中查找确认cgroup所属的css_set是否存在。

4 notify_on_release 是做什么的？

如果cgroup中使能notify_on_release，cgroup中的最后一个进程被移除，最后一个子cgroup也被删除时，cgroup会主动通知kernel。接收到消息的kernel会执行release_agent文件中指定的程序。notify_on_release默认是关闭的，release_agent的内容默认为空，子cgroup在创建时会继承父cgroup中notify_on_release和release_agent的属性。所以这两个文件只存在于cgroupfs的顶层目录中。

5 clone_children有什么用？

clone_children仅针对cpu绑定（cpuset），如果clone_children使能，新的cpuset cgroup在初始化时会继承父cgroup的属性。

6 cgroup怎么用？

假设现在要将一个新的任务加入到cgroup，功能是将该任务的进程在指定的cpu上运行，因此我们使用"cpuset"cgroup 子系统，操作的大致步骤如下：

- 1) mount -t tmpfs cgroup_root /sys/fs/cgroup

挂载cgroup根文件系统，类型为tmpfs

2) `mkdir /sys/fs/cgroup/cpuset`

在cgroupfs根目录下创建子cgroup，名为cpuset

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

将名为cpuset的cgroup关联到cpuset子系统

4) 在cpuset目录下创建目录，生成一个子cgroup，属性文件中写入相应内容，设置属性。

5) 启动需要限制的进程，查找其对应的进程ID，将其写入对应的task文件中

以下操作步骤是创建一个名为"Charlie"的cgroup，这个cgroup的资源包含cpu2，cpu3和内存节点1，将shell进程附加到这个cgroup。

```
mount -t tmpfs cgroup_root /sys/fs/cgroup
```

```
mkdir /sys/fs/cgroup/cpuset
```

```
mount -t cgroup cpuset -o cpuset /sys/fs/cgroup/cpuset
```

```
cd /sys/fs/cgroup/cpuset
```

```
mkdir Charlie
```

```
cd Charlie
```

```
echo 2-3 > cpuset.cpus
```

```
echo 1 > cpuset.mems
```

```
echo $$ > tasks
```

```
sh
```

```
cat /proc/self/cgroup
```