edsionte's TechBlog No Pains, No Gains

Cgroup简介-资源控制的实现

2013年4月25日 由 edsionte 留言 »

Cgroup自身通过文件系统的形式在内核中实现，通过对子系统配置文件的读写即可完成对进程组资源的控制。不过，cgroup对各种资源的实际控制则分布到整个内核代码中。下面从CPU、内存和I/O三个方面说明Cgroup对资源的控制过程。

1 CPU控制

Cgroup对进程组使用CPU的限制是通过cpu和cpuset两个子系统来完成的。cpu子系统主要限制进程的时间片大小，cpuset子系统可为进程指定cpu和内存节点。

1) cpu子系统

cpu子系统主要分布在内核的调度系统中，通过该子系统中的cpu.shares文件可对进程组设置权重。

根据CFS的原理，一个进程的权重越大，那么它的被调度的可能性就越大。那么进程组的权重如何在CFS中体现？CFS将进程和进程组视为同一个调度体，并用sched_entity结构来表示，每个结构中包含该调度体的权重以及虚拟运行时间等。

因此，用户可通过CPU子系统中的cpu.shares文件来控制进程组对CPU的使用。

2) cpuset子系统

通过cpuset子系统中的cpuset.cpus和cpuset.mems可对进程组设定可访问的CPU和内存节点。内核使用cpuset结构来描述cpuset子系统的

属性信息，其中该结构中的cpus_allowed和mems_allowed两个字段分别保存上述两个文件的值。同时，进程描述符中也有cpus_allowed和mems_allowed两个字段，其值与cpuset结构保持同步。

cpuset限制进程所能访问的CPU主要通过两方面。首先体现在进程的创建，如果父进程新建子进程时没有指定CLONE_STOPPED标志，则父进程将调用wake_up_new_task()将子进程状态设置为TASK_RUNNING，并将其加入就绪队列。为子进程选择就绪队列的具体工作则由select_task_rq()完成，其内部实现将涉及CPU的检查操作，即在cpus_allowed所指定的cpu范围内为当前进程分配CPU。

其次，当调度器在调度一个进程时，也要通过select_task_rq()进行同样的对比选择。这样就可以保证cgroup实例中的进程只在cpus_allowed所限定的cpu中运行。

cpuset进程所能访问内存节点的限制表现在物理内存的分配过程中。Linux内核将物理内存在逻辑上分为node、zone和page，内核通过alloc_pages()来实现物理内存的分配工作。alloc_pages()的主要工作是在所有物理内存中选择node，再在当前node中选择zone，最终在zone中分配一个物理页框。其中，在选择node的过程中会进行mems_allowed的判断过程。

2 内存控制

cgroup对内存的控制通过memory子系统完成，其控制作用主要体现在对内存使用量的限制，同时为当前cgroup生成一份内存使用情况报告。

在具体实现的过程中，cgroup通过内核中的resource counter机制实现内存的限制。resource counter相当于一个通用的资源计数器，在内核中通过res_counter结构来描述。该结构可用于记录某类资源的当前使用量、最大使用量以及上限等信息。具体描述如下：

```
1  struct res_counter {
2      /*
3       * the current resource consumption level
4       */
5      unsigned long long usage;
6      /*
7       * the maximal value of the usage from the counter creation
8       */
9      unsigned long long max_usage;
10     /*
11      * the limit that usage cannot exceed
12      */
```

```
    unsigned long long limit;
    /*
     * the limit that usage can be exceed
     */
    unsigned long long soft_limit;
    /*
     * the number of unsuccessful attempts to consume the resource
     */
    unsigned long long failcnt;
    /*
     * the lock to protect all of the above.
     * the routines below consider this to be IRQ-safe
     */
    spinlock_t lock;    /*
     * Parent counter, used for hierarchial resource accounting
     */
    struct res_counter *parent;
};
```

内核使用mem_cgroup结构描述cgroup中内存的使用信息，其内部嵌入了res_counter结构。因此res_counter中的每个字段则表示对内存使用量的记录。用户态下memory子系统所导出的配置文件与该结构中的字段互相对应，比如mem.limit_in_bytes表示当前cgroup可使用内存的最大上线，该文件与res_counter结构中的limit字段对应。也就是说，当用户在用户态向mem.limit_in_bytes文件写入值后，则res_counter中的limit字段相应更新。

内核对res_counter进行操作时有三个基本函数：res_counter_init()对res_counter进行初始化；当分配资源时，res_counter_charge()记录资源的使用量，并且该函数还会检查使用量是否超过了上限，并且记录当前资源使用量的最大值；当资源被释放时，res_counter_uncharge()则减少该资源的使用量。

cgroup对内存资源的限制主要是将上述三个函数分布到内存的分配单元中，比如，系统发生缺页异常时，由于页表项未分配而申请内存时，由于页缓存而分配内存等。

3 块I/O控制

Cgroup中通过blkio子系统完成对块设备I/O的控制。具体的控制主要通过blkio.weight文件在用户态设定当前进程组访问块I/O的权重，也就是控制进程组占有I/O的时间。

blkio子系统对块I/O的控制代码主要分布在I/O调度算法中，目前内核中默认的调度算法为CFQ（完全公平队列），该算法与进程调度算法CFS比较类似。

- 上一篇: Cgroup简介-子系统与层级
- 下一篇: CFS中的虚拟运行时间

发表在 Linux容器技术
Tags: cgroup 容器 虚拟化
若要跟踪这篇文章的任何更新，你可以使用 RSS 2.0 Feed. 你可以直接转到文章底部进行评论，Pinging目前已关闭

发表评论

姓名 (required)

电子邮件（不会被公开） (required)

站点

请回答问题: 3 + 7 = ?

发表评论

返回顶端