Subject: Re: [PATCH] task containersv11 add tasks file interface fix for cpusets
Posted by David Rientjes on Wed, 10 Oct 2007 20:59:44 GMT
View Forum Message <> Reply to Message

On Wed, 10 Oct 2007, Paul Menage wrote:

> On 10/6/07, David Rientjes <rientjes@google.com> wrote:
> >
> > It can race with sched_setaffinity(). It has to give up tasklist_lock as
> > well to call set_cpus_allowed() and can race
> >
> > cpus_allowed = cpuset_cpus_allowed(p);
> > cpus_and(new_mask, new_mask, cpus_allowed);
> > retval = set_cpus_allowed(p, new_mask);
> >
> > and allow a task to have a cpu outside of the cpuset's new cpus_allowed if
> > you've taken it away between cpuset_cpus_allowed() and set_cpus_allowed().
>
> cpuset_cpus_allowed() takes callback_mutex, which is held by
> update_cpumask() when it updates cs->cpus_allowed. So if we continue
> to hold callback_mutex across the task update loop this wouldn't be a
> race. Having said that, holding callback mutex for that long might not
> be a good idea.
>

Moving the actual use of set_cpus_allowed() from the cgroup code to
sched.c would probably be the best in terms of a clean interface. Then
you could protect the whole thing by sched_hotcpu_mutex, which is
expressly designed for migrations.

Something like this:

```
 struct cgroup_iter it;
 struct task_struct *p, **tasks;
 int i = 0;

 cgroup_iter_start(cs->css.cgroup, &it);
 while ((p = cgroup_iter_next(cs->css.cgroup, &it))) {
  get_task_struct(p);
  tasks[i++] = p;
 }
 cgroup_iter_end(cs->css.cgroup, &it);

 while (--i >= 0) {
  sched_migrate_task(tasks[i], cs->cpus_allowed);
  put_task_struct(tasks[i]);
 }
```

kernel/sched.c:

```
 void sched_migrate_task(struct task_struct *task,
    cpumask_t cpus_allowed)
{
  mutex_lock(&sched_hotcpu_mutex);
  set_cpus_allowed(task, cpus_allowed);
  mutex_unlock(&sched_hotcpu_mutex);
}
```

It'd be faster to just pass **tasks to a sched.c function with the number of tasks to migrate to reduce the contention on sched_hotcpu_mutex, but then the put_task_struct() is left dangling over in cgroup code. sched_hotcpu_mutex should be rarely contended, anyway.

David

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers