Subject: Re: [PATCH] task containersv11 add tasks file interface fix for cpusets
Posted by Paul Menage on Wed, 10 Oct 2007 20:46:36 GMT
View Forum Message <> Reply to Message

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.

A cleaner solution might be to drop callback_mutex after updating
cs->cpus_allowed in update_cpumask() and then make sched_setaffinity()
do a post-check:

```
 cpus_allowed = cpuset_cpus_allowed(p);
 again:
 cpus_and(new_mask, new_mask, cpus_allowed);
 retval = set_cpus_allowed(p, new_mask);
 if (!retval) {
  /* Check for races with cpuset updates */
  cpus_allowed = cpuset_cpus_allowed(p);
  if (!cpus_subset(new_mask, cpus_allowed)) {
   /*
    * We raced with a change to cpuset update,
    * and our cpumask is now outside the
    * permitted cpumask for the cpuset.  Since a
    * change to the cpuset's cpus resets the
    * cpumask for each task, do the same thing
    * here.
    */
   new_mask = cpus_allowed;
   goto again;
  }
 }
```

Paul

_____

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