Subject: Re: [RFC] cpuset update_cgroup_cpus_allowed
Posted by David Rientjes on Tue, 16 Oct 2007 06:21:08 GMT
View Forum Message <> Reply to Message

On Mon, 15 Oct 2007, Paul Jackson wrote:

> My solution may be worse than that. Because set_cpus_allowed() will
> fail if asked to set a non-overlapping cpumask, my solution could never
> terminate. If asked to set a cpusets cpus to something that went off
> line right then, this I'd guess this code could keep looping forever,
> looking for cpumasks that didn't match, and then not noticing that it
> was failing to set them so as they would match.
>

Why can't you just add a helper function to sched.c:

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

And then change each task's cpus_allowed via that function instead of
set_cpus_allowed() directly?

You don't need to worry about making the task->cpuset->cpus_allowed
assignment a critical section because common_cpu_mem_hotplug_unplug() will
remove any hot-unplugged cpus from each cpuset's cpus_allowed in the
hierarchy.

Your loop will still need to be reworked so that cgroup_iter_{start,end}()
are not reinvoked unnecessarily and you rely only on cgroup_iter_next()
returning NULL to determine when you've gone through the entire list.
There's no need to go back and check the cpus_allowed of tasks you've
already called set_cpus_allowed() on either directly or indirectly via my
helper function above.

  David