Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Paul Menage on Tue, 23 Oct 2007 06:06:54 GMT
View Forum Message <> Reply to Message

On 10/22/07, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
> On Mon, Oct 22, 2007 at 05:49:39PM -0700, Paul Menage wrote:
> > +static u64 cpu_usage_read(struct cgroup *cgrp, struct cftype *cft)
> > +{
> > + struct task_group *tg = cgroup_tg(cgrp);
> > + int i;
> > + u64 res = 0;
> > + for_each_possible_cpu(i) {
> > + unsigned long flags;
> > + spin_lock_irqsave(&tg->cfs_rq[i]->rq->lock, flags);
>
> Is the lock absolutely required here?

I'm not sure, I was hoping you or Ingo could comment on this. But some kind of locking seems to required at least on 32-bit platforms, since sum_exec_runtime is a 64-bit number.

>
> Hmm .. I hope the cgroup code prevents a task group from being destroyed while
> we are still reading a task group's cpu usage. Is that so?

Good point - cgroups certainly prevents a cgroup itself from being freed while a control file is being read in an RCU section, and prevents a task group from being destroyed when that task group has been read via a task's cgroups pointer and the reader is still in an RCU section, but we need a generic protection for subsystem state objects being accessed via control files too.

Using cgroup_mutex is certainly possible for now, although more heavy-weight than I'd like long term. Using css_get isn't the right approach, I think - we shouldn't be able to cause an rmdir to fail due to a concurrent read.

Paul

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