Subject: Re: [PATCH, v6 3/3] cgroups: introduce timer slack controller
Posted by Kirill A. Shutsemov on Mon, 14 Feb 2011 22:39:39 GMT
View Forum Message <> Reply to Message

On Mon, Feb 14, 2011 at 06:01:06PM +0100, Thomas Gleixner wrote:
> B1;2401;0cOn Mon, 14 Feb 2011, Kirill A. Shutemov wrote:
>
> > On Mon, Feb 14, 2011 at 03:00:03PM +0100, Thomas Gleixner wrote:
> > > On Mon, 14 Feb 2011, Kirill A. Shutsemov wrote:
> > > > From: Kirill A. Shutemov <kirill@shutemov.name>
> > > >
> > > > Every task_struct has timer_slack_ns value. This value uses to round up
> > > > poll() and select() timeout values. This feature can be useful in
> > > > mobile environment where combined wakeups are desired.
> > > >
> > > > cgroup subsys "timer_slack" implement timer slack controller. It
> > > > provides a way to group tasks by timer slack value and manage the
> > > > value of group's tasks.
> > >
> > > I have no objections against the whole thing in general, but why do we
> > > need a module for this? Why can't we add this to the cgroups muck and
> > > compile it in?
> >
> > It was easier to test and debug with module.
> > What is wrong with module? Do you worry about number of exports?
>
> Not only about the number. We don't want exports when they are not
> techically necessary, i.e. for driver stuff.

Ok, I'll drop module support.

```
> > > > +static int cgroup_timer_slack_check(struct notifier_block *nb,
> > > > +  unsigned long slack_ns, void *data)
> > > > +{
> > > > + struct cgroup_subsys_state *css;
> > > > + struct timer_slack_cgroup *tslack_cgroup;
> > > > +
> > > > + /* XXX: lockdep false positive? */
> > >
> > >   What? Either this has a reason or not. If it's a false positive then
> > >   it needs to be fixed in lockdep. If not, ....
> >
> > I was not sure about it. There is similar workaround in freezer_fork().
>
> I don't care about workarounds in freezer_work() at all. The above
> question remains and this is new code and therefor it either needs to
> hold rcu_read_lock() or it does not.
```

I'll recheck everything once again.

```
> > > > + rcu_read_lock();
> > > > + css = task_subsys_state(current, timer_slack_subsys.subsys_id);
> > > > + tslack_cgroup = container_of(css, struct timer_slack_cgroup, css);
> > > > + rcu_read_unlock();
> > > > +
> > > > + if (!is_timer_slack_allowed(tslack_cgroup, slack_ns))
> > > > +  return notifier_from_errno(-EPERM);
> > >
> > >   If the above needs rcu read lock, why is the acess safe ?
> > >
> > > > + return NOTIFY_OK;
> > >
> > > > +/*
> > > > + * Adjust ->timer_slack_ns and ->default_max_slack_ns of the task to fit
> > > > + * limits of the cgroup.
> > > > + */
> > > > +static void tslack_adjust_task(struct timer_slack_cgroup *tslack_cgroup,
> > > > +  struct task_struct *tsk)
> > > > +{
> > > > + if (tslack_cgroup->min_slack_ns > tsk->timer_slack_ns)
> > > > +  tsk->timer_slack_ns = tslack_cgroup->min_slack_ns;
> > > > + else if (tslack_cgroup->max_slack_ns < tsk->timer_slack_ns)
> > > > +  tsk->timer_slack_ns = tslack_cgroup->max_slack_ns;
> > > > +
> > > > + if (tslack_cgroup->min_slack_ns > tsk->default_timer_slack_ns)
> > > > +  tsk->default_timer_slack_ns = tslack_cgroup->min_slack_ns;
> > > > + else if (tslack_cgroup->max_slack_ns < tsk->default_timer_slack_ns)
> > > > +  tsk->default_timer_slack_ns = tslack_cgroup->max_slack_ns;
> > >
> > >
> > >   Why is there not a default slack value for the whole group ?
> >
> > I think it breaks prctl() semantic. default slack value is a value on
> > fork().
>
> cgroups break a lot of semantics.
```

I don't know what "a lot of semantics" you mean, but it's not a reason
to add more breakage.

```
> > > > +static u64 tslack_read_range(struct cgroup *cgroup, struct cftype *cft)
> > > > +{
> > > > + struct timer_slack_cgroup *tslack_cgroup;
> > > > +
> > > > + tslack_cgroup = cgroup_to_tslack_cgroup(cgroup);
> > > > + switch (cft->private) {
```

> > > > + case TIMER_SLACK_MIN:
> > > > + return tslack_cgroup->min_slack_ns;
> > > > + case TIMER_SLACK_MAX:
> > > > + return tslack_cgroup->max_slack_ns;
> > > > + default:
> > > > + BUG();
> > >
> > >   BUG() for soemthing which can be dealt with sensible ?
> >
> > tslack_read_range() and tslack_write_range() have written to handle
> > defined cftypes. If it used for other cftype it's a bug().
>
> The only caller is initiated from here, right? So we really don't need
> another bug just because you might fatfinger your own code.

People make mistakes. I think BUG() is useful here.

> > > > + list_for_each_entry(cur, &cgroup->children, sibling) {
> > > > + child = cgroup_to_tslack_cgroup(cur);
> > > > + if (type == TIMER_SLACK_MIN && val > child->min_slack_ns)
> > > > +   return -EBUSY;
> > >
> > >   I thought the whole point is to propagate values through the group.
> >
> > I think silent change here is wrong. cpuset returns -EBUSY in similar
> > case.
>
> And how is cpuset relevant for this ? Not at all. This is about
> timer_slack and we better have a well defined scheme for all of this
> and not some cobbled together thing with tons of exceptions and corner
> cases. Of course undocumented as far the code goes.

I don't like silent cascade changes. Userspace can implement it if
needed. -EBUSY is appropriate.

--
 Kirill A. Shutemov
____
Containers mailing list
Containers@lists.linux-foundation.org
 https://lists.linux-foundation.org/mailman/listinfo/containe rs