
Subject: Re: [PATCH, v6 3/3] cgroups: introduce timer slack controller
Posted by [Thomas Gleixner](#) on Mon, 14 Feb 2011 14:00:03 GMT
[View Forum Message](#) <> [Reply to Message](#)

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?

```
> +struct cgroup_subsys timer_slack_subsys;  
> +struct timer_slack_cgroup {  
> + struct cgroup_subsys_state css;  
> + unsigned long min_slack_ns;  
> + unsigned long max_slack_ns;  
> +};  
> +  
> +enum {  
> + TIMER_SLACK_MIN,  
> + TIMER_SLACK_MAX,  
> +};  
> +  
> +static struct timer_slack_cgroup *cgroup_to_tslack_cgroup(struct cgroup *cgroup)  
> +{  
> + struct cgroup_subsys_state *css;  
> +  
> + css = cgroup_subsys_state(cgroup, timer_slack_subsys.subsys_id);  
> + return container_of(css, struct timer_slack_cgroup, css);  
> +}  
> +  
> +static int is_timer_slack_allowed(struct timer_slack_cgroup *tslack_cgroup,
```

bool perhaps ?

```
> + unsigned long slack_ns)  
> +{  
> + if (slack_ns < tslack_cgroup->min_slack_ns ||  
> + slack_ns > tslack_cgroup->max_slack_ns)  
> + return false;
```

```

> + return true;
> +}
> +
> +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,

```

> + 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 access 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 ?

```

> +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 ?

```

> + }
> +}
> +
> +static int validate_change(struct cgroup *cgroup, u64 val, int type)
> +{
> + struct timer_slack_cgroup *tslack_cgroup, *child;
> + struct cgroup *cur;
> +
> + BUG_ON(type != TIMER_SLACK_MIN && type != TIMER_SLACK_MAX);

```

Ditto. That should be -EINVAL or such.

```

> + if (val > ULONG_MAX)
> + return -EINVAL;
> +
> + if (cgroup->parent) {
> + struct timer_slack_cgroup *parent;
> + parent = cgroup_to_tslack_cgroup(cgroup->parent);
> + if (!is_timer_slack_allowed(parent, val))
> + return -EPERM;
> + }
> +
> + tslack_cgroup = cgroup_to_tslack_cgroup(cgroup);
> + if (type == TIMER_SLACK_MIN && val > tslack_cgroup->max_slack_ns)
> + return -EINVAL;
> + if (type == TIMER_SLACK_MAX && val < tslack_cgroup->min_slack_ns)
> + return -EINVAL;
> +
> + 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.

```
> + if (type == TIMER_SLACK_MAX && val < child->max_slack_ns)
> + return -EBUSY;
```

This is completely confusing w/o any line of comment.

Thanks

tglx

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
