Subject: Re: [PATCH, v4 2/2] cgroups: introduce timer slack subsystem
Posted by Kirill A. Shutsemov on Wed, 09 Feb 2011 13:23:40 GMT
View Forum Message <> Reply to Message

On Tue, Feb 08, 2011 at 03:08:35PM -0800, Andrew Morton wrote:
> On Thu, 3 Feb 2011 16:34:14 +0200
> "Kirill A. Shutsemov" <kirill@shutemov.name> wrote:
>
> > From: Kirill A. Shutemov <kirill@shutemov.name>
> >
> > Provides a way of tasks grouping by timer slack value. Introduces per
> > cgroup max and min timer slack value. When a task attaches to a cgroup,
> > its timer slack value adjusts (if needed) to fit min-max range.
> >
> > It also provides a way to set timer slack value for all tasks in the
> > cgroup at once.
> >
> > This functionality is useful in mobile devices where certain background
> > apps are attached to a cgroup and minimum wakeups are desired.
>
> This description is quite scanty. It doesn't explain what the benefit
> is, when one might want to use it, how to use it, etc. A nice
> description under Documentation/cgroups/ would be appropriate.
>
> I read this and come away with no sense of how useful or desirable this
> code is.

Ok, I'll add file to Documentation/cgroups/ and extend description.

> >
> > ...
> >
> > --- a/include/linux/init_task.h
> > +++ b/include/linux/init_task.h
> > @@ -124,6 +124,8 @@ extern struct cred init_cred;
> > # define INIT_PERF_EVENTS(tsk)
> > #endif
> >
> > +#define TIMER_SLACK_NS_DEFAULT 50000
>
> Seems to be an unrelated cleanup. And given that this #define is only
> used in a single place, its cleanliness is questionable.

I used it in previous patch revision. I'll remove it.

> > /*
> >   * INIT_TASK is used to set up the first task table, touch at
> >   * your own risk!. Base=0, limit=0x1fffff (=2MB)

> > @@ -177,7 +179,7 @@ extern struct cred init_cred;
> > .cpu_timers = INIT_CPU_TIMERS(tsk.cpu_timers), \
> > .fs_excl = ATOMIC_INIT(0), \
> > .pi_lock = __RAW_SPIN_LOCK_UNLOCKED(tsk.pi_lock), \
> > - .timer_slack_ns = 50000, /* 50 usec default slack */ \
> > + .timer_slack_ns = TIMER_SLACK_NS_DEFAULT, \
> > .pids = { \
> > [PIDTYPE_PID] = INIT_PID_LINK(PIDTYPE_PID), \
> > [PIDTYPE_PGID] = INIT_PID_LINK(PIDTYPE_PGID), \
> >
> > ...
> >
> > +extern int (*timer_slack_check)(struct task_struct *task,
> > + unsigned long slack_ns);
>
> Nope. Please put this in a header file so we can be sure that callers
> see the same prototype as does the definition.

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;
> > +}
> > +
> > +static void tslack_cgroup_attach(struct cgroup_subsys *subsys,
> > + struct cgroup *cgroup, struct cgroup *prev,
> > + struct task_struct *tsk, bool threadgroup)
> > +{
> > + tslack_adjust_task(cgroup_to_tslack_cgroup(cgroup), tsk);
> > +}
> > +

> > +static int tslack_write_set_slack_ns(struct cgroup *cgroup, struct cftype *cft,
> > + u64 val)
> > +{
> > + struct timer_slack_cgroup *tslack_cgroup;
> > + struct cgroup_iter it;
> > + struct task_struct *task;
> > +
> > + tslack_cgroup = cgroup_to_tslack_cgroup(cgroup);
> > + if (!is_timer_slack_allowed(cgroup_to_tslack_cgroup(cgroup), val))
> > + return -EPERM;
> > +
> > + /* Change timer slack value for all tasks in the cgroup */
> > + cgroup_iter_start(cgroup, &it);
> > + while ((task = cgroup_iter_next(cgroup, &it)))
> > + task->timer_slack_ns = val;
> > + cgroup_iter_end(cgroup, &it);
> > +
> > + return 0;
> > +}
> > +
> > +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();
> > + };
>
> Extraneous ";".
>
> > +}
>
> These proposed new userspace interfaces should be documented somewhere,
> please. They are the most important part of the patch.
>
> >
> > ...
> >
> > +struct cgroup_subsys timer_slack_subsys = {
> > + .name = "timer_slack",
> > + .module = THIS_MODULE,
> > +#ifdef CONFIG_CGROUP_TIMER_SLACK

>
> Confused. This file won't even be compiled if
> CONFIG_CGROUP_TIMER_SLACK=n?

Timer slack cgroup subsystem can be compiled as module. In this case
CONFIG_CGROUP_TIMER_SLACK_MODULE defined, not
CONFIG_CGROUP_TIMER_SLACK.
I'll replace it with

#ifndef CONFIG_CGROUP_TIMER_SLACK_MODULE

to avoid confusion.

> > + .subsys_id = timer_slack_subsys_id,
> > +#endif
> > + .create = tslack_cgroup_create,
> > + .destroy = tslack_cgroup_destroy,
> > + .attach = tslack_cgroup_attach,
> > + .populate = tslack_cgroup_populate,
> > +};
> > +
> >
> > ...
> >
> > @@ -1694,12 +1698,15 @@ SYSCALL_DEFINE5(prctl, int, option, unsigned long, arg2,
unsigned long, arg3,
> >   error = current->timer_slack_ns;
> >   break;
> >   case PR_SET_TIMERSLACK:
> > -   if (arg2 <= 0)
> > -   current->timer_slack_ns =
> > -   current->default_timer_slack_ns;
> > -   else
> > -   current->timer_slack_ns = arg2;
> > -   error = 0;
> > +   if (arg2 <= 0) {
> > +   me->timer_slack_ns = me->default_timer_slack_ns;
> > +   break;
> > +   }
> > +
> > +   error = timer_slack_check ?
> > +   timer_slack_check(me, arg2) : 0;
>
> This is racy against init_cgroup_timer_slack(). And against
> exit_cgroup_timer_slack().
>
> This whole scheme of using a single hook to a single "check" function
> is rather grubby. I guess that using a notifier_call_chain() would fix

> things: support multiple "check" functions and hopefully fix the races.

Ok, I'll rewrite it using blocking_notifier_call_chain().

> > +   if (!error)
> > +   me->timer_slack_ns = arg2;
> >   break;
> >   case PR_MCE_KILL:
> >   if (arg4 | arg5)
>
> I don't know whether PR_SET_TIMERSLACK is presently documented in the
> manpages.

It's in todo list:

http://git.kernel.org/?p=docs/man-pages/man-pages.git;a=blob;f=man2/prctl.2#l42

> Please cc linux-api@vger.kernel.org on future versions of this work and
> ensure that the patch (via changelog, Documentation or code comments)
> has sufficient information for the manpages maintainers to be able to
> easily update the manpages.

Ok.

Thank you for reviewing.

--
 Kirill A. Shutemov

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