Subject: Re: [patch 3/4] Container Freezer: Implement freezer cgroup subsystem
Posted by Paul Menage on Tue, 24 Jun 2008 21:27:10 GMT
View Forum Message <> Reply to Message

On Tue, Jun 24, 2008 at 6:58 AM, Matt Helsley <matthltc@us.ibm.com> wrote:
> From: Cedric Le Goater <clg@fr.ibm.com>
> Subject: [patch 3/4] Container Freezer: Implement freezer cgroup subsystem
>
> This patch implements a new freezer subsystem for Paul Menage's
> control groups framework.

You can s/Paul Menage's// now that it's in mainline.
> +static const char *freezer_state_strs[] = {
> +   "RUNNING",
> +   "FREEZING",
> +   "FROZEN",
> +};
> +
> +/* Check and update whenever adding new freezer states. Currently is:
> +   strlen("FREEZING") */
> +#define STATE_MAX_STRLEN 8
> +

That's a bit nasty ...

But hopefully it could go away when the write_string() method is
available in cgroups? (See my patchset from earlier this week).

> +
> +struct cgroup_subsys freezer_subsys;
> +
> +/* Locking and lock ordering:
> + *
> + * can_attach(), cgroup_frozen():
> + * rcu (task->cgroup, freezer->state)
> + *
> + * freezer_fork():
> + * rcu (task->cgroup, freezer->state)
> + * freezer->lock
> + *   task_lock
> + *   sighand->siglock
> + *
> + * freezer_read():
> + * rcu (freezer->state)
> + * freezer->lock (upgrade to write)
> + *   read_lock css_set_lock
> + *
> + * freezer_write()

```
> + * cgroup_lock
> + *  rcu
> + *   freezer->lock
> + *    read_lock css_set_lock
> + *     task_lock
> + *      sighand->siglock
> + *
> + * freezer_create(), freezer_destroy():
> + * cgroup_lock [ by cgroup core ]
> + */


> +static int freezer_can_attach(struct cgroup_subsys *ss,
> +                              struct cgroup *new_cgroup,
> +                              struct task_struct *task)
> +{
> +       struct freezer *freezer;
> +       int retval = 0;
> +
> +       /*
> +        * The call to cgroup_lock() in the freezer.state write method prevents
> +        * a write to that file racing against an attach, and hence the
> +        * can_attach() result will remain valid until the attach completes.
> +        */
> +       rcu_read_lock();
> +       freezer = cgroup_freezer(new_cgroup);
> +       if (freezer->state == STATE_FROZEN)
> +               retval = -EBUSY;
```

Is it meant to be OK to move a task into a cgroup that's currently in
the FREEZING state but not yet fully frozen?

```
> +       struct freezer *freezer;
> +
> +       rcu_read_lock(); /* needed to fetch task's cgroup
> +                           can't use task_lock() here because
> +                           freeze_task() grabs that */
```

I'm not sure that RCU is the right thing for this. All that the RCU
lock will guarantee is that the freezer structure you get a pointer to
doesn't go away. It doesn't guarantee that the task doesn't move
cgroup, or that the cgroup doesn't get a freeze request via a write.
But in this case, the fork callback is called before the task is added
to the task_list/pidhash, or to its cgroups' linked lists. So it
shouldn't be able to change groups. Racing against a concurrent write
to the cgroup's freeze file may be more of an issue.

Can you add a __freeze_task() that has to be called with task_lock(p)
already held?

```
> +	freezer = task_freezer(task);
```

Maybe BUG_ON(freezer->state == STATE_FROZEN) ?

```
> +
> +static ssize_t freezer_read(struct cgroup *cgroup,
> +			    struct cftype *cft,
> +			    struct file *file, char __user *buf,
> +			    size_t nbytes, loff_t *ppos)
> +{
> +	struct freezer *freezer;
> +	enum freezer_state state;
> +
> +	rcu_read_lock();
> +	freezer = cgroup_freezer(cgroup);
> +	state = freezer->state;
> +	if (state == STATE_FREEZING) {
> +		/* We change from FREEZING to FROZEN lazily if the cgroup was
> +		 * only partially frozen when we exitted write. */
> +		spin_lock_irq(&freezer->lock);
> +		if (freezer_check_if_frozen(cgroup)) {
> +			freezer->state = STATE_FROZEN;
> +			state = STATE_FROZEN;
> +		}
> +		spin_unlock_irq(&freezer->lock);
> +	}
> +	rcu_read_unlock();
> +
> +	return simple_read_from_buffer(buf, nbytes, ppos,
> +				       freezer_state_strs[state],
> +				       strlen(freezer_state_strs[state]));
> +}
```

Technically this could return weird results if someone read it
byte-by-byte and the status changed between reads. If you used
read_seq_string rather than read you'd avoid that.

```
> +		return -EIO;
> +
> +	cgroup_lock();
```

If you're taking cgroup_lock() here in freezer_write(), there's no
need for the rcu_read_lock() in freezer_freeze()

Paul

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