Subject: Re: [RFC][PATCH] Make access to taks's nsproxy liter
Posted by Pavel Emelianov on Thu, 09 Aug 2007 07:46:57 GMT
View Forum Message <> Reply to Message

Oleg Nesterov wrote:
> On 08/09, Pavel Emelyanov wrote:
>> Paul E. McKenney wrote:
>>> On Wed, Aug 08, 2007 at 08:41:07PM +0400, Oleg Nesterov wrote:
>>>>> +void switch_task_namespaces(struct task_struct *p, struct nsproxy *new)
>>>>> +{
>>>>> + struct nsproxy *ns;
>>>>> +
>>>>> + might_sleep();
>>>>> +
>>>>> + ns = p->nsproxy;
>>>>> + if (ns == new)
>>>>> + return;
>>>>> +
>>>>> + if (new)
>>>>> + get_nsproxy(new);
>>>>> + rcu_assign_pointer(p->nsproxy, new);
>>>>> +
>>>>> + if (ns && atomic_dec_and_test(&ns->count)) {
>>>>> + /*
>>>>> +   * wait for others to get what they want from this
>>>>> +   * nsproxy. cannot release this nsproxy via the
>>>>> +   * call_rcu() since put_mnt_ns will want to sleep
>>>>> +   */
>>>>> + synchronize_rcu();
>>>>> + free_nsproxy(ns);
>>>>> + }
>>>>> +}
>>>> (I may be wrong, Paul cc'ed)
>>>>
>>>> This is correct with the current implementation of RCU, but strictly
>>>> speaking,
>>>> we can't use synchronize_rcu() here, because write_lock_irq() doesn't
>>>> imply
>>>> rcu_read_lock() in theory.
>>> Can you use synchronize_sched() instead? The synchronize_sched()
>> #define synchronize_sched() synchronize_rcu()
>> they are the same? what's the point?
>
> There are the same with the current implementation. RT kernel for example,
> has another, when preempt_disable() doesn't imply rcu_read_lock().

Ok, thanks.

>>> primitive will wait until all preempt/irq-disable code sequences complete.
>>> Therefore, it would wait for all write_lock_irq() code sequences to
>>> complete.
>> But we don't need this. Iff we get the nsproxy under rcu_read_lock() all
>> we need is to wait for RCU sections to complete.
>
> Yes. But this patch complicates the code and slows down group_exit. We don't

Nope - it slows done the code only if the task exiting is the last
one using the nsproxy. In other words - we slowdown the virtual server
stop, not task exit. This is OK.

> access non-current ->nsproxy so often afaics, and task_lock is cheap.
>
> Note also that switch_task_namespaces() might_sleep(), but sys_unshare()
> calls it under task_lock().

I've moved this lower :)

> Oleg.
>
>

_____