Subject: Re: [RFC][PATCH] Make access to taks's nsproxy liter
Posted by paulmck on Wed, 08 Aug 2007 17:23:09 GMT
View Forum Message <> Reply to Message

On Wed, Aug 08, 2007 at 08:41:07PM +0400, Oleg Nesterov wrote:
> This time Paul E. McKenney actually cc'ed, sorry for the extra
> noise...
>
> On 08/08, Pavel Emelyanov wrote:
> >
> > When someone wants to deal with some other taks's namespaces
> > it has to lock the task and then to get the desired namespace
> > if the one exists. This is slow on read-only paths and may be
> > impossible in some cases.
> >
> > E.g. Oleg recently noticed a race between unshare() and the
> > (just sent for review) pid namespaces - when the task notifies
> > the parent it has to know the parent's namespace, but taking
> > the task_lock() is impossible there - the code is under write
> > locked tasklist lock.
> >
> > On the other hand switching the namespace on task (daemonize)
> > and releasing the namespace (after the last task exit) is rather
> > rare operation and we can sacrifice its speed to solve the
> > issues above.
>
> Still it is a bit sad we slow down process's exit. Perhaps I missed
> some other ->nsproxy access, but can't we make a simpler patch?
>
> --- kernel/fork.c 2007-07-28 16:58:17.000000000 +0400
> +++ /proc/self/fd/0 2007-08-08 20:30:33.325216944 +0400
> @@ -1633,7 +1633,9 @@ asmlinkage long sys_unshare(unsigned lon
>
>   if (new_nsproxy) {
>   old_nsproxy = current->nsproxy;
> +   read_lock(&tasklist_lock);
>   current->nsproxy = new_nsproxy;
> +   read_unlock(&tasklist_lock);
>   new_nsproxy = old_nsproxy;
>   }
>
>
> This way ->nsproxy is stable under task_lock() or write_lock(tasklist).
>
> > +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()
primitive will wait until all preempt/irq-disable code sequences complete.
Therefore, it would wait for all write_lock_irq() code sequences to
complete.

Does this work?

    Thanx, Paul

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

---

Subject: Re: [RFC][PATCH] Make access to taks's nsproxy liter
Posted by Oleg Nesterov on Wed, 08 Aug 2007 17:36:47 GMT
View Forum Message <> Reply to Message

On 08/08, 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()
> primitive will wait until all preempt/irq-disable code sequences complete.
> Therefore, it would wait for all write_lock_irq() code sequences to
> complete.

Thanks Paul!

But we also need to cover the case when ->nsproxy is used under rcu_read_lock(),
so if we go this way, we'd better add rcu_read_lock() to do_notify_parent.*() as
Eric suggested.

Oleg.

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

Subject: Re: [RFC][PATCH] Make access to taks's nsproxy liter
Posted by Pavel Emelianov on Thu, 09 Aug 2007 07:15:49 GMT
View Forum Message <> Reply to Message

Paul E. McKenney wrote:
> On Wed, Aug 08, 2007 at 08:41:07PM +0400, Oleg Nesterov wrote:
>> This time Paul E. McKenney actually cc'ed, sorry for the extra
>> noise...
>>
>> On 08/08, Pavel Emelyanov wrote:
>>> When someone wants to deal with some other taks's namespaces
>>> it has to lock the task and then to get the desired namespace
>>> if the one exists. This is slow on read-only paths and may be
>>> impossible in some cases.
>>>
>>> E.g. Oleg recently noticed a race between unshare() and the
>>> (just sent for review) pid namespaces - when the task notifies
>>> the parent it has to know the parent's namespace, but taking
>>> the task_lock() is impossible there - the code is under write
>>> locked tasklist lock.
>>>
>>> On the other hand switching the namespace on task (daemonize)
>>> and releasing the namespace (after the last task exit) is rather
>>> rare operation and we can sacrifice its speed to solve the
>>> issues above.
>> Still it is a bit sad we slow down process's exit. Perhaps I missed
>> some other ->nsproxy access, but can't we make a simpler patch?
>>
>> --- kernel/fork.c 2007-07-28 16:58:17.000000000 +0400
>> +++ /proc/self/fd/0 2007-08-08 20:30:33.325216944 +0400
>> @@ -1633,7 +1633,9 @@ asmlinkage long sys_unshare(unsigned lon
>>
>>   if (new_nsproxy) {
>>   old_nsproxy = current->nsproxy;
>> +   read_lock(&tasklist_lock);
>>   current->nsproxy = new_nsproxy;
>> +   read_unlock(&tasklist_lock);
>>   new_nsproxy = old_nsproxy;
>>   }
>>
>>
>> This way ->nsproxy is stable under task_lock() or write_lock(tasklist).
>>
>>> +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?

> 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.

> Does this work?
>
>   Thanx, Paul
>

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

Subject: Re: [RFC][PATCH] Make access to taks's nsproxy liter
Posted by Oleg Nesterov on Thu, 09 Aug 2007 07:39:00 GMT
View Forum Message <> Reply to Message

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().

> >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
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().

Oleg.

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

---

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.
>
>

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

Subject: Re: [RFC][PATCH] Make access to taks's nsproxy liter
Posted by Oleg Nesterov on Thu, 09 Aug 2007 07:49:35 GMT
View Forum Message <> Reply to Message

On 08/09, Oleg Nesterov wrote:
>
> Note also that switch_task_namespaces() might_sleep(), but sys_unshare()
> calls it under task_lock().

Ah, sorry, didn't notice your patch moves task_lock() down in sys_unshare().

Oleg.

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

Subject: Re: [RFC][PATCH] Make access to taks's nsproxy liter
Posted by Oleg Nesterov on Thu, 09 Aug 2007 08:06:09 GMT
View Forum Message <> Reply to Message

On 08/09, Pavel Emelyanov wrote:
>
> Oleg Nesterov wrote:
> >
> >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.

Ah yes, you are right. This is sad, because now I have no "hard" argument
against this patch :) Except "complicates" may be...

Oleg.

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