Subject: Re: [RFC][PATCH] Make access to taks's nsproxy liter
Posted by ebiederm on Wed, 08 Aug 2007 17:03:20 GMT
View Forum Message <> Reply to Message

Oleg Nesterov <oleg@tv-sign.ru> writes:

> 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.
```

But we should be able to do:

```
write_lock_irq();
rcu_read_lock();
 muck with other tasks nsproxy.
rcu_read_unlock();
write_unlock_irq();
```

Which would make rcu fine.

The real locking we have is that only a task is allowed to modify it's
own nsproxy pointer. Other processes are not.

The practical question is how do we enable other processes to read
a particular tasks nsproxy or something pointed to by it?

Eric

_____
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 17:19:55 GMT
View Forum Message <> Reply to Message

On 08/08, Eric W. Biederman wrote:
>
> Oleg Nesterov <oleg@tv-sign.ru> writes:
>
> > On 08/08, Pavel Emelyanov 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.
>
> But we should be able to do:
>
> write_lock_irq();
> rcu_read_lock();
> muck with other tasks nsproxy.
> rcu_read_unlock();
> write_unlock_irq();
>

> Which would make rcu fine.

Yes sure. I just meant that the patch looks incomplete. But we didn't
hear Paul yet, perhaps I'm just wrong.

> The real locking we have is that only a task is allowed to modify it's
> own nsproxy pointer. Other processes are not.
>
> The practical question is how do we enable other processes to read
> a particular tasks nsproxy or something pointed to by it?

task_lock(). The only problem we can't take it in do_notify_parent(),
but if we add read_lock(tasklist) to sys_unshare, we can safely access
->parent->nsproxy.

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 07:09:55 GMT
View Forum Message <> Reply to Message

Oleg Nesterov wrote:
> On 08/08, Eric W. Biederman wrote:
>> Oleg Nesterov <oleg@tv-sign.ru> writes:
>>
>>> On 08/08, Pavel Emelyanov 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.

```
void __lockfunc _write_lock(rwlock_t *lock)
{
        preempt_disable();
        rwlock_acquire(&lock->dep_map, 0, 0, _RET_IP_);
        LOCK_CONTENDED(lock, _raw_write_trylock, _raw_write_lock);
}
```

preempt_disable == rcu_read_lock() due to

```
#define rcu_read_lock() \
        do { \
                preempt_disable(); \
                __acquire(RCU); \
        } while(0)
```

so currently this is enough to write_lock()

>> But we should be able to do:
>>
>> write_lock_irq();
>> rcu_read_lock();
>> muck with other tasks nsproxy.
>> rcu_read_unlock();
>> write_unlock_irq();
>>
>> Which would make rcu fine.
>
> Yes sure. I just meant that the patch looks incomplete. But we didn't
> hear Paul yet, perhaps I'm just wrong.
>
>> The real locking we have is that only a task is allowed to modify it's
>> own nsproxy pointer. Other processes are not.
>>
>> The practical question is how do we enable other processes to read
>> a particular tasks nsproxy or something pointed to by it?

>
> task_lock(). The only problem we can't take it in do_notify_parent(),
> but if we add read_lock(tasklist) to sys_unshare, we can safely access
> ->parent->nsproxy.

we can safely access parent's nsproxy with this patch like this:

```
rcu_read_lock();
nsproxy = task_nsproxy(p->parent);
BUG_ON(nsproxy == NULL); /* parent should reparent us before exiting nsproxy */
pid_ns = nsproxy->pid_ns;
...
rcu_read_unlock();
```

>
> Oleg.
>
>

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