
Subject: Re: [PATCH v7 1/3] cgroups: read-write lock CLONE_THREAD forking per threadgroup

Posted by [Ben Blum](#) on Fri, 04 Feb 2011 21:25:15 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Mon, Jan 24, 2011 at 01:05:29PM -0800, Andrew Morton wrote:

> On Sun, 26 Dec 2010 07:09:51 -0500

> Ben Blum <bblum@andrew.cmu.edu> wrote:

>
> > Adds functionality to read/write lock CLONE_THREAD fork()ing per-threadgroup
> >
> > From: Ben Blum <bblum@andrew.cmu.edu>
> >
> > This patch adds an rwsem that lives in a threadgroup's signal_struct that's
> > taken for reading in the fork path, under CONFIG_CGROUPS. If another part of
> > the kernel later wants to use such a locking mechanism, the CONFIG_CGROUPS
> > ifdefs should be changed to a higher-up flag that CGROUPS and the other system
> > would both depend on.

> >
> > This is a pre-patch for cgroup-procs-write.patch.

> >

> > ...

> >

> > +/* See the declaration of threadgroup_fork_lock in signal_struct. */
> > +#ifdef CONFIG_CGROUPS
> > +static inline void threadgroup_fork_read_lock(struct task_struct *tsk)
> > +{
> > + down_read(&tsk->signal->threadgroup_fork_lock);
> > +}
> > +static inline void threadgroup_fork_read_unlock(struct task_struct *tsk)
> > +{
> > + up_read(&tsk->signal->threadgroup_fork_lock);
> > +}
> > +static inline void threadgroup_fork_write_lock(struct task_struct *tsk)
> > +{
> > + down_write(&tsk->signal->threadgroup_fork_lock);
> > +}
> > +static inline void threadgroup_fork_write_unlock(struct task_struct *tsk)
> > +{
> > + up_write(&tsk->signal->threadgroup_fork_lock);
> > +}
> > +#else

>

> Risky. sched.h doesn't include rwsem.h.

>

> We could make it do so, but almost every compilation unit in the kernel
> includes sched.h. It would be nicer to make the kernel build
> finer-grained, rather than blunter-grained. Don't be afraid to add new

> header files if that is one way of doing this!

Hmm, good point. But there's also:

```
+#ifdef CONFIG_CGROUPS
+ struct rw_semaphore threadgroup_fork_lock;
+#endif
```

in the signal_struct, also in sched.h, which needs to be there. Or I could change it to a struct pointer with a forward incomplete declaration above, and kcalloc/kfree it? I don't like adding more alloc/free calls but don't know if it's more or less important than header granularity.

-- Ben

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

Subject: Re: [PATCH v7 1/3] cgroups: read-write lock CLONE_THREAD forking per threadgroup

Posted by [akpm](#) on Fri, 04 Feb 2011 21:36:57 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Fri, 4 Feb 2011 16:25:15 -0500

Ben Blum <bblum@andrew.cmu.edu> wrote:

> On Mon, Jan 24, 2011 at 01:05:29PM -0800, Andrew Morton wrote:

> > On Sun, 26 Dec 2010 07:09:51 -0500

> > Ben Blum <bblum@andrew.cmu.edu> wrote:

> >

> > > Adds functionality to read/write lock CLONE_THREAD fork()ing per-threadgroup

> > >

> > > From: Ben Blum <bblum@andrew.cmu.edu>

> > >

> > > This patch adds an rwsem that lives in a threadgroup's signal_struct that's

> > > taken for reading in the fork path, under CONFIG_CGROUPS. If another part of

> > > the kernel later wants to use such a locking mechanism, the CONFIG_CGROUPS

> > > ifdefs should be changed to a higher-up flag that CGROUPS and the other system

> > > would both depend on.

> > >

> > > This is a pre-patch for cgroup-procs-write.patch.

> > >

> > > ...

> > >

> > > +/* See the declaration of threadgroup_fork_lock in signal_struct. */

```

> > > + #ifdef CONFIG_CGROUPS
> > > + static inline void threadgroup_fork_read_lock(struct task_struct *tsk)
> > > + {
> > > + down_read(&tsk->signal->threadgroup_fork_lock);
> > > + }
> > > + static inline void threadgroup_fork_read_unlock(struct task_struct *tsk)
> > > + {
> > > + up_read(&tsk->signal->threadgroup_fork_lock);
> > > + }
> > > + static inline void threadgroup_fork_write_lock(struct task_struct *tsk)
> > > + {
> > > + down_write(&tsk->signal->threadgroup_fork_lock);
> > > + }
> > > + static inline void threadgroup_fork_write_unlock(struct task_struct *tsk)
> > > + {
> > > + up_write(&tsk->signal->threadgroup_fork_lock);
> > > + }
> > > + #else
> >
> > Risky. sched.h doesn't include rwsem.h.
> >
> > We could make it do so, but almost every compilation unit in the kernel
> > includes sched.h. It would be nicer to make the kernel build
> > finer-grained, rather than blunter-grained. Don't be afraid to add new
> > header files if that is one way of doing this!
>
> Hmm, good point. But there's also:
>
> + #ifdef CONFIG_CGROUPS
> +     struct rw_semaphore threadgroup_fork_lock;
> + #endif
>
> in the signal_struct, also in sched.h, which needs to be there. Or I
> could change it to a struct pointer with a forward incomplete
> declaration above, and kmalloc/kfree it? I don't like adding more
> alloc/free calls but don't know if it's more or less important than
> header granularity.

```

What about adding a new header file which includes rwsem.h and sched.h and then defines the new interfaces?

Containers mailing list

Containers@lists.linux-foundation.org

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

Subject: Re: [PATCH v7 1/3] cgroups: read-write lock CLONE_THREAD forking per

threadgroup

Posted by [Ben Blum](#) on Fri, 04 Feb 2011 21:43:54 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Fri, Feb 04, 2011 at 01:36:57PM -0800, Andrew Morton wrote:

> On Fri, 4 Feb 2011 16:25:15 -0500

> Ben Blum <bblum@andrew.cmu.edu> wrote:

>

> > On Mon, Jan 24, 2011 at 01:05:29PM -0800, Andrew Morton wrote:

> > > On Sun, 26 Dec 2010 07:09:51 -0500

> > > Ben Blum <bblum@andrew.cmu.edu> wrote:

> > >

> > > > Adds functionality to read/write lock CLONE_THREAD fork()ing per-threadgroup

> > > >

> > > > From: Ben Blum <bblum@andrew.cmu.edu>

> > > >

> > > > This patch adds an rwsem that lives in a threadgroup's signal_struct that's

> > > > taken for reading in the fork path, under CONFIG_CGROUPS. If another part of

> > > > the kernel later wants to use such a locking mechanism, the CONFIG_CGROUPS

> > > > ifdefs should be changed to a higher-up flag that CGROUPS and the other system

> > > > would both depend on.

> > > >

> > > > This is a pre-patch for cgroup-procs-write.patch.

> > > >

> > > > ...

> > > >

> > > > +/* See the declaration of threadgroup_fork_lock in signal_struct. */

> > > > +**#ifdef** CONFIG_CGROUPS

> > > > +static inline void threadgroup_fork_read_lock(struct task_struct *tsk)

> > > > +{

> > > > + down_read(&tsk->signal->threadgroup_fork_lock);

> > > > +}

> > > > +static inline void threadgroup_fork_read_unlock(struct task_struct *tsk)

> > > > +{

> > > > + up_read(&tsk->signal->threadgroup_fork_lock);

> > > > +}

> > > > +static inline void threadgroup_fork_write_lock(struct task_struct *tsk)

> > > > +{

> > > > + down_write(&tsk->signal->threadgroup_fork_lock);

> > > > +}

> > > > +static inline void threadgroup_fork_write_unlock(struct task_struct *tsk)

> > > > +{

> > > > + up_write(&tsk->signal->threadgroup_fork_lock);

> > > > +}

> > > > +**#else**

> > >

> > > Risky. sched.h doesn't include rwsem.h.

> > >

> > > We could make it do so, but almost every compilation unit in the kernel

> > > includes sched.h. It would be nicer to make the kernel build
> > > finer-grained, rather than blunter-grained. Don't be afraid to add new
> > > header files if that is one way of doing this!
> >
> > Hmm, good point. But there's also:
> >
> > +#ifdef CONFIG_CGROUPS
> > + struct rw_semaphore threadgroup_fork_lock;
> > +#endif
> >
> > in the signal_struct, also in sched.h, which needs to be there. Or I
> > could change it to a struct pointer with a forward incomplete
> > declaration above, and kcalloc/kfree it? I don't like adding more
> > alloc/free calls but don't know if it's more or less important than
> > header granularity.
>
> What about adding a new header file which includes rwsem.h and sched.h
> and then defines the new interfaces?

Er, I mean the definition of signal_struct needs rwsem.h as well, not
just the threadgroup_fork_* functions. (And I suspect moving
signal_struct somewhere else would give bigger problems...)

Containers mailing list

Containers@lists.linux-foundation.org

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