Subject: Re: [PATCH] Containers: Avoid lockdep warning
Posted by Peter Zijlstra on Thu, 23 Aug 2007 09:46:06 GMT
View Forum Message <> Reply to Message

On Wed, 2007-08-22 at 16:17 -0700, Paul Menage wrote:
> I think this is the right way to handle the lockdep false-positive in
> the current containers patches, but I'm not that familiar with lockdep
> so any suggestions for a better approach are welcomed.
>
>
> In order to avoid a false-positive lockdep warning, we lock the root
> inode of a new filesystem mount prior to taking container_mutex, to
> preserve the invariant that container_mutex nests inside
> inode->i_mutex. In order to prevent a lockdep false positive when
> locking i_mutex on a newly-created container directory inode we use
> mutex_lock_nested(), with a nesting level of I_MUTEX_CHILD since the
> new inode will ultimately be a child directory of the parent whose
> i_mutex is nested outside of container_mutex.

So the normal order is:

```
  inode->i_mutex
    container_mutex (I_MUTEX_NORMAL)
```

and there is a one off reversal where we take container_mutex before
inode->i_mutex, which is safe because the inode is fresh, so nobody
could have possibly taken it to form the deadlock.

mutex_lock_nested() is indeed the proper annotation.

Using I_MUTEX_CHILD forms the chain:

```
  container_mutex
    inode->i_mutex (I_MUTEX_CHILD)
```

In order for this to become a problem there would need to be a:

```
 inode->i_mutex (I_MUTEX_CHILD)
   inode->i_mutex (I_MUTEX_NORMAL)
```

chain, which I suppose is precluded to exists on basis of common
sense :-)

> Signed-off-by: Paul Menage <menage@google.com>

Acked-by: Peter Zijlstra <a.p.zijlstra@chello.nl>

>

```
> ---
>  kernel/container.c |   17 +++++++----------
>  1 file changed, 7 insertions(+), 10 deletions(-)
>
> Index: container-2.6.23-rc3-mm1/kernel/container.c
> ===================================================================
> --- container-2.6.23-rc3-mm1.orig/kernel/container.c
> +++ container-2.6.23-rc3-mm1/kernel/container.c
> @@ -966,13 +966,16 @@ static int container_get_sb(struct file_
>   } else {
>    /* New superblock */
>    struct container *cont = &root->top_container;
> +  struct inode *inode;
>
>    BUG_ON(sb->s_root != NULL);
>
>    ret = container_get_rootdir(sb);
>    if (ret)
>     goto drop_new_super;
> +  inode = sb->s_root->d_inode;
>
> +  mutex_lock(&inode->i_mutex);
>    mutex_lock(&container_mutex);
>
>    /*
> @@ -985,12 +988,14 @@ static int container_get_sb(struct file_
>    ret = allocate_cg_links(css_group_count, &tmp_cg_links);
>    if (ret) {
>     mutex_unlock(&container_mutex);
> +   mutex_unlock(&inode->i_mutex);
>     goto drop_new_super;
>    }
>
>    ret = rebind_subsystems(root, root->subsys_bits);
>    if (ret == -EBUSY) {
>     mutex_unlock(&container_mutex);
> +   mutex_unlock(&inode->i_mutex);
>     goto drop_new_super;
>    }
>
> @@ -1030,16 +1035,8 @@ static int container_get_sb(struct file_
>    BUG_ON(!list_empty(&cont->children));
>    BUG_ON(root->number_of_containers != 1);
>
> -  /*
> -   * I believe that it's safe to nest i_mutex inside
> -   * container_mutex in this case, since no-one else can
> -   * be accessing this directory yet. But we still need
```

> -   * to teach lockdep that this is the case - currently
> -   * a containerfs remount triggers a lockdep warning
> -   */
> - mutex_lock(&cont->dentry->d_inode->i_mutex);
>   container_populate_dir(cont);
> - mutex_unlock(&cont->dentry->d_inode->i_mutex);
> + mutex_unlock(&inode->i_mutex);
>   mutex_unlock(&container_mutex);
>   }
>
> @@ -1529,7 +1526,7 @@ static int container_create_file(struct
>
>   /* start with the directory inode held, so that we can
>   * populate it without racing with another mkdir */
> - mutex_lock(&inode->i_mutex);
> + mutex_lock_nested(&inode->i_mutex, I_MUTEX_CHILD);
>   } else if (S_ISREG(mode)) {
>   inode->i_size = 0;
>   inode->i_fop = &container_file_operations;

_____

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