
Subject: [PATCH 08/33] task containersv11 shared container subsystem group arrays avoid lockdep warning

Posted by [Paul Menage](#) on Mon, 17 Sep 2007 21:03:15 GMT

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

I think this is the right way to handle the lockdep false-positive in the current cgroups 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 cgroup_mutex, to preserve the invariant that cgroup_mutex nests inside inode->i_mutex. In order to prevent a lockdep false positive when locking i_mutex on a newly-created cgroup 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 cgroup_mutex.

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

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

kernel/cgroup.c | 17 ++++++-----
1 files changed, 7 insertions(+), 10 deletions(-)

```
diff -puN  
kernel/cgroup.c~task-cgroupsv11-shared-cgroup-subsystem-group-arrays-avoid-lockdep-warning  
kernel/cgroup.c  
---  
a/kernel/cgroup.c~task-cgroupsv11-shared-cgroup-subsystem-group-arrays-avoid-lockdep-warning  
+++ a/kernel/cgroup.c  
@@ -867,13 +867,16 @@ static int cgroup_get_sb(struct file_  
 } else {  
 /* New superblock */  
 struct cgroup *cont = &root->top_cgroup;  
+ struct inode *inode;  
  
 BUG_ON(sb->s_root != NULL);  
  
 ret = cgroup_get_rootdir(sb);  
 if (ret)  
 goto drop_new_super;  
+ inode = sb->s_root->d_inode;  
  
+ mutex_lock(&inode->i_mutex);  
 mutex_lock(&cgroup_mutex);  
  
/*
```

```

@@ -886,12 +889,14 @@ static int cgroup_get_sb(struct file_
    ret = allocate_cg_links(css_set_count, &tmp_cg_links);
    if (ret) {
        mutex_unlock(&cgroup_mutex);
+       mutex_unlock(&inode->i_mutex);
        goto drop_new_super;
    }

    ret = rebind_subsystems(root, root->subsys_bits);
    if (ret == -EBUSY) {
        mutex_unlock(&cgroup_mutex);
+       mutex_unlock(&inode->i_mutex);
        goto drop_new_super;
    }

@@ -931,16 +936,8 @@ static int cgroup_get_sb(struct file_
    BUG_ON(!list_empty(&cont->children));
    BUG_ON(root->number_of_cgroups != 1);

-   /*
-   * I believe that it's safe to nest i_mutex inside
-   * cgroup_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 cgroupfs remount triggers a lockdep warning
-   */
-   mutex_lock(&cont->dentry->d_inode->i_mutex);
    cgroup_populate_dir(cont);
-   mutex_unlock(&cont->dentry->d_inode->i_mutex);
+   mutex_unlock(&inode->i_mutex);
    mutex_unlock(&cgroup_mutex);
}

@@ -1358,7 +1355,7 @@ static int cgroup_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 = &cgroup_file_operations;

```

--

--

Containers mailing list
 Containers@lists.linux-foundation.org

