Subject: [PATCH] cgroup: Remove RCU from task->cgroups
Posted by Colin Cross on Sun, 21 Nov 2010 02:00:24 GMT
View Forum Message <> Reply to Message

The synchronize_rcu call in cgroup_attach_task can be very
expensive. All fastpath accesses to task->cgroups already
use task_lock() or cgroup_lock() to protect against updates,
and only the CGROUP_DEBUG files have RCU read-side critical
sections.

This patch replaces rcu_read_lock() with task_lock(current)
around the debug file acceses to current->cgroups and removes
the synchronize_rcu call in cgroup_attach_task.

Signed-off-by: Colin Cross <ccross@android.com>
---
kernel/cgroup.c | 22 ++++++++----------
1 files changed, 8 insertions(+), 14 deletions(-)

diff --git a/kernel/cgroup.c b/kernel/cgroup.c
index 66a416b..4a40183 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@@ -725,14 +725,11 @@ static struct cgroup *task_cgroup_from_root(struct task_struct *task,
* cgroup_attach_task(), which overwrites one tasks cgroup pointer with
* another. It does so using cgroup_mutex, however there are
* several performance critical places that need to reference
- * task->cgroup without the expense of grabbing a system global
+ * task->cgroups without the expense of grabbing a system global
* mutex. Therefore except as noted below, when dereferencing or, as
- * in cgroup_attach_task(), modifying a task'ss cgroup pointer we use
+ * in cgroup_attach_task(), modifying a task's cgroups pointer we use
* task_lock(), which acts on a spinlock (task->alloc_lock) already in
* the task_struct routinely used for such matters.
- *
- * P.S. One more locking exception. RCU is used to guard the
- * update of a tasks cgroup pointer by cgroup_attach_task()
*/

/**
@@ -1786,7 +1783,7 @@ int cgroup_attach_task(struct cgroup *cgrp, struct task_struct *tsk)
retval = -ESRCH;
goto out;
}
- rcu_assign_pointer(tsk->cgroups, newcg);
+ tsk->cgroups = newcg;
task_unlock(tsk);

```
  /* Update the css_set linked lists if we're using them */
@@ -1802,7 +1799,6 @@ int cgroup_attach_task(struct cgroup *cgrp, struct task_struct *tsk)
     ss->attach(ss, cgrp, oldcgrp, tsk, false);
   }
   set_bit(CGRP_RELEASABLE, &oldcgrp->flags);
- synchronize_rcu();
   put_css_set(cg);

   /*
@@ -4827,9 +4823,9 @@ static u64 current_css_set_refcount_read(struct cgroup *cont,
 {
   u64 count;

- rcu_read_lock();
+ task_lock(current);
   count = atomic_read(&current->cgroups->refcount);
- rcu_read_unlock();
+ task_unlock(current);
   return count;
 }

@@ -4838,12 +4834,10 @@ static int current_css_set_cg_links_read(struct cgroup *cont,
       struct seq_file *seq)
 {
   struct cg_cgroup_link *link;
- struct css_set *cg;

   read_lock(&css_set_lock);
- rcu_read_lock();
- cg = rcu_dereference(current->cgroups);
- list_for_each_entry(link, &cg->cg_links, cg_link_list) {
+ task_lock(current);
+ list_for_each_entry(link, &current->cgroups->cg_links, cg_link_list) {
     struct cgroup *c = link->cgrp;
     const char *name;

@@ -4854,7 +4848,7 @@ static int current_css_set_cg_links_read(struct cgroup *cont,
     seq_printf(seq, "Root %d group %s\n",
       c->root->hierarchy_id, name);
   }
- rcu_read_unlock();
+ task_unlock(current);
   read_unlock(&css_set_lock);
   return 0;
 }
--
1.7.3.1
```

_____

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