
Subject: [PATCH 1/2] cgroup: Set CGRP_RELEASABLE when adding to a cgroup
Posted by [Colin Cross](#) on Wed, 24 Nov 2010 05:37:03 GMT

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

Changes the meaning of CGRP_RELEASABLE to be set on any cgroup that has ever had a task or cgroup in it, or had css_get called on it. The bit is set in cgroup_attach_task, cgroup_create, and __css_get. It is not necessary to set the bit in cgroup_fork, as the task is either in the root cgroup, in which can never be released, or the task it was forked from already set the bit in cgroup_attach_task.

Signed-off-by: Colin Cross <ccross@android.com>

include/linux/cgroup.h | 12 +-----
kernel/cgroup.c | 54 ++++++-----
2 files changed, 25 insertions(+), 41 deletions(-)

diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h

index ed4ba11..9e13078 100644

--- a/include/linux/cgroup.h

+++ b/include/linux/cgroup.h

@@ -84,12 +84,6 @@ enum {
 CSS_REMOVED, /* This CSS is dead */
};

/* Caller must verify that the css is not for root cgroup */

static inline void __css_get(struct cgroup_subsys_state *css, int count)

{
- atomic_add(count, &css->refcnt);
-}

/*
 * Call css_get() to hold a reference on the css; it can be used
 * for a reference obtained via:
@@ -97,6 +91,7 @@ static inline void __css_get(struct cgroup_subsys_state *css, int count)
 * - task->cgroups for a locked task
 */

+extern void __css_get(struct cgroup_subsys_state *css, int count);

static inline void css_get(struct cgroup_subsys_state *css)

{
 /* We don't need to reference count the root state */
@@ -143,10 +138,7 @@ static inline void css_put(struct cgroup_subsys_state *css)
 enum {
 /* Control Group is dead */
 CGRP_REMOVED,
- /*

```

- * Control Group has previously had a child cgroup or a task,
- * but no longer (only if CGRP_NOTIFY_ON_RELEASE is set)
- */
+ /* Control Group has ever had a child cgroup or a task */
  CGRP_RELEASABLE,
  /* Control Group requires release notifications to userspace */
  CGRP_NOTIFY_ON_RELEASE,
diff --git a/kernel/cgroup.c b/kernel/cgroup.c
index 66a416b..34e855e 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@@ -338,7 +338,15 @@ static void free_css_set_rcu(struct rcu_head *obj)
  * compiled into their kernel but not actually in use */
  static int use_task_css_set_links __read_mostly;

-static void __put_css_set(struct css_set *cg, int taskexit)
+/*
+ * refcounted get/put for css_set objects
+ */
+static inline void get_css_set(struct css_set *cg)
+{
+ atomic_inc(&cg->refcount);
+}
+
+static void put_css_set(struct css_set *cg)
+{
+ struct cg_cgroup_link *link;
+ struct cg_cgroup_link *saved_link;
@@ -364,12 +372,8 @@ static void __put_css_set(struct css_set *cg, int taskexit)
  struct cgroup *cgrp = link->cgrp;
  list_del(&link->cg_link_list);
  list_del(&link->cgrp_link_list);
- if (atomic_dec_and_test(&cgrp->count) &&
-     notify_on_release(cgrp)) {
- if (taskexit)
- set_bit(CGRP_RELEASABLE, &cgrp->flags);
+ if (atomic_dec_and_test(&cgrp->count))
+ check_for_release(cgrp);
- }

  kfree(link);
}
@@ -379,24 +383,6 @@ static void __put_css_set(struct css_set *cg, int taskexit)
}

/*
- * refcounted get/put for css_set objects
- */

```

```

-static inline void get_css_set(struct css_set *cg)
-{
- atomic_inc(&cg->refcount);
-}
-
-static inline void put_css_set(struct css_set *cg)
-{
- __put_css_set(cg, 0);
-}
-
-static inline void put_css_set_taskexit(struct css_set *cg)
-{
- __put_css_set(cg, 1);
-}
-
-/*
- * compare_css_sets - helper function for find_existing_css_set().
- * @cg: candidate css_set being tested
- * @old_cg: existing css_set for a task
@@ -1801,7 +1787,7 @@ int cgroup_attach_task(struct cgroup *cgrp, struct task_struct *tsk)
    if (ss->attach)
        ss->attach(ss, cgrp, oldcgrp, tsk, false);
    }
- set_bit(CGRP_RELEASABLE, &oldcgrp->flags);
+ set_bit(CGRP_RELEASABLE, &cgrp->flags);
    synchronize_rcu();
    put_css_set(cg);

@@ -3427,6 +3413,8 @@ static long cgroup_create(struct cgroup *parent, struct dentry *dentry,
    if (err < 0)
        goto err_remove;

+ set_bit(CGRP_RELEASABLE, &parent->flags);
+
    /* The cgroup directory was pre-locked for us */
    BUG_ON(!mutex_is_locked(&cgrp->dentry->d_inode->i_mutex));

@@ -3645,7 +3633,6 @@ again:
    cgroup_d_remove_dir(d);
    dput(d);

- set_bit(CGRP_RELEASABLE, &parent->flags);
    check_for_release(parent);

    /*
@@ -4240,7 +4227,7 @@ void cgroup_exit(struct task_struct *tsk, int run_callbacks)
    tsk->cgroups = &init_css_set;
    task_unlock(tsk);

```

```

    if (cg)
-   put_css_set_taskexit(cg);
+   put_css_set(cg);
}

/**
@@ -4410,6 +4397,14 @@ static void check_for_release(struct cgroup *cgrp)
}

/* Caller must verify that the css is not for root cgroup */
+void __css_get(struct cgroup_subsys_state *css, int count)
+{
+   atomic_add(count, &css->refcnt);
+   set_bit(CGRP_RELEASABLE, &css->cgroup->flags);
+}
+EXPORT_SYMBOL_GPL(__css_get);
+
+/* Caller must verify that the css is not for root cgroup */
void __css_put(struct cgroup_subsys_state *css, int count)
{
    struct cgroup *cgrp = css->cgroup;
@@ -4417,10 +4412,7 @@ void __css_put(struct cgroup_subsys_state *css, int count)
    rcu_read_lock();
    val = atomic_sub_return(count, &css->refcnt);
    if (val == 1) {
-   if (notify_on_release(cgrp)) {
-       set_bit(CGRP_RELEASABLE, &cgrp->flags);
-       check_for_release(cgrp);
-   }
+   check_for_release(cgrp);
        cgroup_wakeup_rmdir_waiter(cgrp);
    }
    rcu_read_unlock();
--
1.7.3.1

```

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