Subject: [Patch 1/3] Miscellaneous container fixes
Posted by Srivatsa Vaddagiri on Fri, 01 Dec 2006 16:46:32 GMT
View Forum Message <> Reply to Message

This patches fixes various bugs I hit in the recently posted container
patches.

1. If a subsystem registers with fork/exit hook during bootup (much
   before rcu is initialized), then the resulting synchronize_rcu() in
   container_register_subsys() hangs. Avoid this by not calling
   synchronize_rcu() if we arent fully booted yet.

2. If cpuset_create fails() for some reason, then the resulting
   call to cpuset_destroy can trip. Avoid this by initializing
   container->...->cpuset pointer to NULL in cpuset_create().

3. container_rmdir->cpuset_destroy->update_flag can deadlock on
   container_lock(). Avoid this by introducing __update_flag, which
   doesnt take container_lock().

(I have also hit some lockdep warnings. Will post them after some
review, to make sure that they are not introduced by my patches).

Signed-off-by : Srivatsa Vaddagiri <vatsa@in.ibm.com>

---

 linux-2.6.19-rc6-vatsa/kernel/container.c | 4 ++-
 linux-2.6.19-rc6-vatsa/kernel/cpuset.c | 35 ++++++++++++++++++++++++------
 2 files changed, 31 insertions(+), 8 deletions(-)

```
diff -puN kernel/container.c~container_fixes kernel/container.c
--- linux-2.6.19-rc6/kernel/container.c~container_fixes 2006-12-01 16:19:41.000000000 +0530
+++ linux-2.6.19-rc6-vatsa/kernel/container.c 2006-12-01 16:20:20.000000000 +0530
@@ -1344,6 +1344,7 @@ static long container_create(struct cont
  cont->parent = parent;
  cont->root = parent->root;
  cont->hierarchy = parent->hierarchy;
+ cont->top_container = parent->top_container;

  for_each_subsys(cont->hierarchy, ss) {
   err = ss->create(ss, cont);
@@ -1580,7 +1581,8 @@ int container_register_subsys(struct con
   if (!need_forkexit_callback &&
       (new_subsys->fork || new_subsys->exit)) {
    need_forkexit_callback = 1;
-   synchronize_rcu();
```

```
+  if (system_state == SYSTEM_RUNNING)
+    synchronize_rcu();
   }

   /* If this subsystem requested that it be notified with fork
diff -puN kernel/cpuset.c~container_fixes kernel/cpuset.c
--- linux-2.6.19-rc6/kernel/cpuset.c~container_fixes 2006-12-01 19:02:35.000000000 +0530
+++ linux-2.6.19-rc6-vatsa/kernel/cpuset.c 2006-12-01 20:43:52.000000000 +0530
@@ -97,14 +97,22 @@ struct cpuset {
 /* Update the cpuset for a container */
 static inline void set_container_cs(struct container *cont, struct cpuset *cs)
 {
- cont->subsys[cpuset_subsys.subsys_id] = &cs->css;
+ if (cs)
+  cont->subsys[cpuset_subsys.subsys_id] = &cs->css;
+ else
+  cont->subsys[cpuset_subsys.subsys_id] = NULL;
 }

 /* Retrieve the cpuset for a container */
 static inline struct cpuset *container_cs(struct container *cont)
 {
- return container_of(container_subsys_state(cont, &cpuset_subsys),
-       struct cpuset, css);
+ struct container_subsys_state *css;
+
+ css = container_subsys_state(cont, &cpuset_subsys);
+ if (css)
+  return container_of(css, struct cpuset, css);
+ else
+  return NULL;
 }

 /* Retrieve the cpuset for a task */
@@ -698,7 +706,7 @@ static int update_memory_pressure_enable
  * Call with manage_mutex held.
  */

-static int update_flag(cpuset_flagbits_t bit, struct cpuset *cs, char *buf)
+static int __update_flag(cpuset_flagbits_t bit, struct cpuset *cs, char *buf)
 {
  int turning_on;
  struct cpuset trialcs;
@@ -717,18 +725,27 @@ static int update_flag(cpuset_flagbits_t
   return err;
  cpu_exclusive_changed =
   (is_cpu_exclusive(cs) != is_cpu_exclusive(&trialcs));
- container_lock();
```

```
  if (turning_on)
   set_bit(bit, &cs->flags);
  else
   clear_bit(bit, &cs->flags);
- container_unlock();

  if (cpu_exclusive_changed)
   update_cpu_domains(cs);
  return 0;
 }

+static int update_flag(cpuset_flagbits_t bit, struct cpuset *cs, char *buf)
+{
+ int err;
+
+ container_lock();
+ err = __update_flag(bit, cs, buf);
+ container_unlock();
+
+ return err;
+}
+
 /*
  * Frequency meter - How fast is some event occurring?
  *
@@ -1165,6 +1182,7 @@ int cpuset_create(struct container_subsy
   return 0;
  }
  parent = container_cs(cont->parent);
+ set_container_cs(cont, NULL);
  cs = kmalloc(sizeof(*cs), GFP_KERNEL);
  if (!cs)
   return -ENOMEM;
@@ -1202,9 +1220,12 @@ void cpuset_destroy(struct container_sub
 {
  struct cpuset *cs = container_cs(cont);

+ if (!cs)
+  return;
+
  cpuset_update_task_memory_state();
  if (is_cpu_exclusive(cs)) {
-  int retval = update_flag(CS_CPU_EXCLUSIVE, cs, "0");
+  int retval = __update_flag(CS_CPU_EXCLUSIVE, cs, "0");
   BUG_ON(retval);
  }
  number_of_cpusets--;
_
```

--
Regards,
vatsa