Subject: Re: [PATCH 0/9] Containers (V9): Generic Process Containers
Posted by Srivatsa Vaddagiri on Mon, 30 Apr 2007 17:04:45 GMT
View Forum Message <> Reply to Message

On Sun, Apr 29, 2007 at 02:37:21AM -0700, Paul Jackson wrote:
> It builds and boots and mounts the cpuset file system ok.
> But trying to write the 'mems' file hangs the system hard.

Basically we are attempting a read_lock(&tasklist_lock) in
container_task_count() after taking write_lock_irq(&tasklist_lock) in
update_nodemask()!

This patch seems to fix the prb for me:

Fix write_lock() followed by read_lock() bug by introducing a 2nd
argument to be passed into container_task_count. Other choice is to
introduce a lock and unlocked versions of container_task_count() ..

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

```
diff -puN include/linux/container.h~lock_fix include/linux/container.h
--- linux-2.6.21-rc7/include/linux/container.h~lock_fix 2007-04-30 21:56:10.000000000 +0530
+++ linux-2.6.21-rc7-vatsa/include/linux/container.h 2007-04-30 21:56:32.000000000 +0530
@@ -164,7 +164,7 @@ int container_is_removed(const struct co

 int container_path(const struct container *cont, char *buf, int buflen);

-int container_task_count(const struct container *cont);
+int container_task_count(const struct container *cont, int take_lock);

 /* Return true if the container is a descendant of the current container */
 int container_is_descendant(const struct container *cont);
diff -puN kernel/container.c~lock_fix kernel/container.c
--- linux-2.6.21-rc7/kernel/container.c~lock_fix 2007-04-30 21:56:10.000000000 +0530
+++ linux-2.6.21-rc7-vatsa/kernel/container.c 2007-04-30 22:06:45.000000000 +0530
@@ -1193,21 +1193,25 @@ int container_add_files(struct container
 /* Count the number of tasks in a container. Could be made more
  * time-efficient but less space-efficient with more linked lists
  * running through each container and the css_group structures
- * that referenced it. */
+ * that referenced it. 'take_lock' indicates whether caller has locked
+ * tasklist or not.
+ */
```

```
-int container_task_count(const struct container *cont) {
+int container_task_count(const struct container *cont, int take_lock) {
  int count = 0;
  struct task_struct *g, *p;
  struct container_subsys_state *css;
  int subsys_id;
  get_first_subsys(cont, &css, &subsys_id);

- read_lock(&tasklist_lock);
+ if (take_lock)
+  read_lock(&tasklist_lock);
  do_each_thread(g, p) {
   if (task_subsys_state(p, subsys_id) == css)
    count ++;
  } while_each_thread(g, p);
- read_unlock(&tasklist_lock);
+ if (take_lock)
+  read_unlock(&tasklist_lock);
  return count;
 }

@@ -1311,7 +1315,7 @@ static int container_tasks_open(struct i
   * caller from the case that the additional container users didn't
   * show up until sometime later on.
   */
- npids = container_task_count(cont);
+ npids = container_task_count(cont, 1);
  pidarray = kmalloc(npids * sizeof(pid_t), GFP_KERNEL);
  if (!pidarray)
   goto err1;
diff -puN kernel/cpuset.c~lock_fix kernel/cpuset.c
--- linux-2.6.21-rc7/kernel/cpuset.c~lock_fix 2007-04-30 21:56:10.000000000 +0530
+++ linux-2.6.21-rc7-vatsa/kernel/cpuset.c 2007-04-30 21:59:25.000000000 +0530
@@ -631,13 +631,13 @@ static int update_nodemask(struct cpuset
   * enough mmarray[] w/o using GFP_ATOMIC.
   */
  while (1) {
-  ntasks = container_task_count(cs->css.container);  /* guess */
+  ntasks = container_task_count(cs->css.container, 1); /* guess */
   ntasks += fudge;
   mmarray = kmalloc(ntasks * sizeof(*mmarray), GFP_KERNEL);
   if (!mmarray)
    goto done;
   write_lock_irq(&tasklist_lock);  /* block fork */
-  if (container_task_count(cs->css.container) <= ntasks)
+  if (container_task_count(cs->css.container, 0) <= ntasks)
    break;    /* got enough */
```

```
   write_unlock_irq(&tasklist_lock); /* try again */
   kfree(mmarray);
diff -puN kernel/container_debug.c~lock_fix kernel/container_debug.c
--- linux-2.6.21-rc7/kernel/container_debug.c~lock_fix 2007-04-30 21:58:54.000000000 +0530
+++ linux-2.6.21-rc7-vatsa/kernel/container_debug.c 2007-04-30 21:59:04.000000000 +0530
@@ -34,7 +34,7 @@ static u64 taskcount_read(struct contain
 {
  u64 count;
  container_lock();
- count = container_task_count(cont);
+ count = container_task_count(cont, 1);
  container_unlock();
  return count;
 }
_


--
Regards,
vatsa
```