Subject: [PATCH -mm 2/3] cgroup: simplify init_subsys() (v2)
Posted by Li Zefan on Mon, 07 Apr 2008 01:45:56 GMT
View Forum Message <> Reply to Message

Changelog v2:
---
1. Add a BUG_ON() to insure no processes haved been forked before
all subsystems have been registered.
2. Update the document about the fork callback.

We are at system boot and there is only 1 cgroup group (i,e,
init_css_set), so we don't need to run through the css_set
linked list. Neither do we need to run through the task list,
since no processes have been created yet.

Also referring to a comment in cgroup.h:

```
struct css_set
{
 ...
 /*
  * Set of subsystem states, one for each subsystem. This array
  * is immutable after creation apart from the init_css_set
  * during subsystem registration (at boot time).
  */
 struct cgroup_subsys_state *subsys[CGROUP_SUBSYS_COUNT];
}
```

Signed-off-by: Li Zefan <lizf@cn.fujitsu.com>
Reviewed-by: Paul Menage <menage@google.com>

I just resent this patch for Andrew. ;)

One of the purposes of this patch is to be a preparation for the
3rd patch. so this should be applied before that patch.

```
---
 Documentation/cgroups.txt |    3 +--
 kernel/cgroup.c           |   35 +++++++++----------------------
 2 files changed, 10 insertions(+), 28 deletions(-)

diff --git a/Documentation/cgroups.txt b/Documentation/cgroups.txt
index 31d12e2..c298a66 100644
--- a/Documentation/cgroups.txt
+++ b/Documentation/cgroups.txt
@@ -500,8 +500,7 @@ post-attachment activity that requires memory allocations or blocking.
```

```
 void fork(struct cgroup_subsy *ss, struct task_struct *task)

-Called when a task is forked into a cgroup. Also called during
-registration for all existing tasks.
+Called when a task is forked into a cgroup.

 void exit(struct cgroup_subsys *ss, struct task_struct *task)

diff --git a/kernel/cgroup.c b/kernel/cgroup.c
index f79e60d..250e28e 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@@ -2471,7 +2471,6 @@ static int cgroup_rmdir(struct inode *unused_dir, struct dentry *dentry)
 static void __init cgroup_init_subsys(struct cgroup_subsys *ss)
 {
  struct cgroup_subsys_state *css;
- struct list_head *l;

  printk(KERN_INFO "Initializing cgroup subsys %s\n", ss->name);

@@ -2482,35 +2481,19 @@ static void __init cgroup_init_subsys(struct cgroup_subsys *ss)
  BUG_ON(IS_ERR(css));
  init_cgroup_css(css, ss, dummytop);

- /* Update all cgroup groups to contain a subsys
+ /* Update the init_css_set to contain a subsys
   * pointer to this state - since the subsystem is
-  * newly registered, all tasks and hence all cgroup
-  * groups are in the subsystem's top cgroup. */
- write_lock(&css_set_lock);
- l = &init_css_set.list;
- do {
-  struct css_set *cg =
-   list_entry(l, struct css_set, list);
-  cg->subsys[ss->subsys_id] = dummytop->subsys[ss->subsys_id];
-  l = l->next;
- } while (l != &init_css_set.list);
- write_unlock(&css_set_lock);
-
-  /* If this subsystem requested that it be notified with fork
-   * events, we should send it one now for every process in the
-   * system */
- if (ss->fork) {
-  struct task_struct *g, *p;
-
-  read_lock(&tasklist_lock);
-  do_each_thread(g, p) {
-   ss->fork(ss, p);
```

```
-  } while_each_thread(g, p);
-  read_unlock(&tasklist_lock);
- }
+  * newly registered, all tasks and hence the
+  * init_css_set is in the subsystem's top cgroup. */
+ init_css_set.subsys[ss->subsys_id] = dummytop->subsys[ss->subsys_id];

  need_forkexit_callback |= ss->fork || ss->exit;

+ /* At system boot, before all subsystems have been
+  * registered, no tasks have been forked, so we don't
+  * need to invoke fork callbacks here. */
+ BUG_ON(!list_empty(&init_task.tasks));
+
  ss->active = 1;
 }

--
1.5.4.rc3
```