Subject: Re: [PATCH 2/3] don't take cgroup_mutex in destroy()
Posted by Li Zefan on Sat, 21 Apr 2012 06:47:16 GMT
View Forum Message <> Reply to Message

Glauber Costa wrote:

> On 04/19/2012 07:57 PM, Tejun Heo wrote:
>> On Thu, Apr 19, 2012 at 07:49:17PM -0300, Glauber Costa wrote:
>>> Most of the destroy functions are only doing very simple things
>>> like freeing memory.
>>>
>>> The ones who goes through lists and such, already use its own
>>> locking for those.
>>>
>>> * The cgroup itself won't go away until we free it, (after destroy)
>>> * The parent won't go away because we hold a reference count
>>> * There are no more tasks in the cgroup, and the cgroup is declared
>>>   dead (cgroup_is_removed() == true)
>>>
>>> For the blk-cgroup and the cpusets, I got the impression that the mutex
>>> is still necessary.
>>>
>>> For those, I grabbed it from within the destroy function itself.
>>>
>>> If the maintainer for those subsystems consider it safe to remove
>>> it, we can discuss it separately.
>>
>> I really don't like cgroup_lock() usage spreading more. It's
>> something which should be contained in cgroup.c proper. I looked at
>> the existing users a while ago and they seemed to be compensating
>> deficencies in API, so, if at all possible, let's not spread the
>> disease.
>
> Well, I can dig deeper and see if they are really needed. I don't know cpusets and blkcg *that* well, that's why I took them there, hoping that someone could enlighten me, maybe they aren't really needed even now.
>
> I agree with the compensating: As I mentioned, most of them are already taking other kinds of lock to protect their structures, which is the right thing to do.
>
> There were only two or three spots in cpusets and blkcg where I wasn't that sure that we could drop the lock... What do you say about that ?
> .

We can drop cgroup_mutex for cpusets with changes like this:

(Note: as I'm not able to get the latest code at this momment, this patch is based on 3.0.)

There are several places reading number_of_cpusets, but no one holds cgroup_mutex, except the one in generate_sched_domains(). With this patch, both cpuset_create() and generate_sched_domains() are still holding cgroup_mutex, so it's safe.

```
--- linux-kernel/kernel/cpuset.c.orig 2012-04-21 01:55:57.000000000 -0400
+++ linux-kernel/kernel/cpuset.c 2012-04-21 02:30:53.000000000 -0400
@@ -1876,7 +1876,9 @@ static struct cgroup_subsys_state *cpuse
  cs->relax_domain_level = -1;

  cs->parent = parent;
+ mutex_lock(&callback_mutex);
  number_of_cpusets++;
+ mutex_unlock(&callback_mutex);
  return &cs->css ;
 }

@@ -1890,10 +1892,18 @@ static void cpuset_destroy(struct cgroup
 {
  struct cpuset *cs = cgroup_cs(cont);

- if (is_sched_load_balance(cs))
+ if (is_sched_load_balance(cs)) {
+  /*
+   * This cpuset is under destruction, so no one else can
+   * modify it, so it's safe to call update_flag() without
+   * cgroup_lock.
+   */
   update_flag(CS_SCHED_LOAD_BALANCE, cs, 0);
+ }

+ mutex_lock(&callback_mutex);
  number_of_cpusets--;
+ mutex_lock(&callback_mutex);
  free_cpumask_var(cs->cpus_allowed);
  kfree(cs);
 }
```