Subject: [PATCH v2 0/5] Fix problem with static_key decrement
Posted by Glauber Costa on Mon, 23 Apr 2012 19:37:42 GMT
View Forum Message <> Reply to Message

This is my proposed fix for the sock memcg static_key
problem raised by Kamezawa. In a summary, the problem is
as follows:

We are decrementing the jump label when the socket limit is set back to
unlimited. The problem is that the sockets outlive the memcg, so we can
only do that when the last reference count is dropped. It is worth
mentioning that kmem controller for memcg will have the exact same problem.

If, however, there are no sockets in flight, mem_cgroup_put() during
->destroy() will be the last one, and the decrementing will happen there.

But static_key updates cannot happen with the cgroup_mutex held. This is
because cpusets hold it from within the cpu_hotplug.lock - that static_keys
take through get_online_cpus() in its cpu hotplug handler.

Removing the cgroup_lock() dependency from cpusets is a lot harder, since the
code for generate_sched_domain() rely on that lock to be held, and it interact
with the cgroup core code by quite a bit.

The aim of this series is to make ->destroy() a stable point for jump label
updating, by calling it without the cgroup_mutex held. I believe it to be a
good thing in itself, since it removes a bit the reach of the almighty
cgroup_mutex.

I am ready to make any further modifications on this that you guys
deem necessary.

Thanks

Glauber Costa (5):
  don't attach a task to a dead cgroup
  blkcg: protect blkcg->policy_list
  change number_of_cpusets to an atomic
  don't take cgroup_mutex in destroy()
  decrement static keys on real destroy time

 block/blk-cgroup.c | 2 +
 include/linux/cpuset.h | 6 ++--
 include/net/sock.h | 9 ++++++++
 kernel/cgroup.c | 12 ++++++++----
 kernel/cpuset.c | 10 ++++++----
 mm/memcontrol.c | 20 ++++++++++++++++++-

net/ipv4/tcp_memcontrol.c | 50 ++------
7 files changed, 87 insertions(+), 22 deletions(-)

--
1.7.7.6