Subject: [PATCH v3 0/2] fix problem with static_branch() for sock memcg
Posted by Glauber Costa on Thu, 26 Apr 2012 21:24:21 GMT
View Forum Message <> Reply to Message

Hi,

While trying to fulfill's Christoph's request for using static_branches
to do part of the role of number_of_cpusets in the cpuset cgroup, I took
a much more extensive look at the cpuset code (Thanks Christoph).

I started to feel that removing the cgroup_lock() from cpuset's
destroy is not as safe as I first imagined. At the very best, is not safe
enough to be bundled in a bugfix and deserves its own analysis.

I started then to consider another approach. While I voiced many times
that I would not like to do deferred updates for the static_branches, doing
that during destroy time would be perfectly acceptable IMHO (creation is
another story). In a summary, we are effectively calling the static_branch
updates only when the last reference to the memcg is gone. And that is
already asynchronous by nature, and we cope well with that.

In memcg, it turns out that we already do deferred freeing of the memcg
structure depending on the size of struct mem_cgroup.

My proposal is to always do that, and then we get a worker more or less
for free. Patch 2 is basically the same I had posted before, with minor
adaptations, plus the addition of a commentary explaining a race as
requested by Kame.

Let me know if this is acceptable.

Thanks

Glauber Costa (2):
  Always free struct memcg through schedule_work()
  decrement static keys on real destroy time

 include/net/sock.h | 9 ++++++
 mm/memcontrol.c | 54 ++++++++++++++++++++++++++++++--------
 net/ipv4/tcp_memcontrol.c | 70 ++-
 3 files changed, 113 insertions(+), 20 deletions(-)

--
1.7.7.6