Subject: Re: [PATCH v7 00/10] Request for Inclusion: per-cgroup tcp memory pressure
Posted by KAMEZAWA Hiroyuki on Mon, 05 Dec 2011 09:51:08 GMT
View Forum Message <> Reply to Message

On Mon, 5 Dec 2011 07:09:51 -0200
Glauber Costa <glommer@parallels.com> wrote:

> On 12/05/2011 12:06 AM, KAMEZAWA Hiroyuki wrote:
> > On Fri, 2 Dec 2011 16:04:08 -0200
> > Glauber Costa<glommer@parallels.com> wrote:
> >
> >> On 11/30/2011 12:11 AM, KAMEZAWA Hiroyuki wrote:
> >>> On Tue, 29 Nov 2011 21:56:51 -0200
> >>> Glauber Costa<glommer@parallels.com>   wrote:
> >>>
> >>>> Hi,
> >>>>
> >>>> This patchset implements per-cgroup tcp memory pressure controls. It did not change
> >>>> significantly since last submission: rather, it just merges the comments Kame had.
> >>>> Most of them are style-related and/or Documentation, but there are two real bugs he
> >>>> managed to spot (thanks)
> >>>>
> >>>> Please let me know if there is anything else I should address.
> >>>>
> >>>
> >>> After reading all codes again, I feel some strange. Could you clarify ?
> >>>
> >>> Here.
> >>> ==
> >>> +void sock_update_memcg(struct sock *sk)
> >>> +{
> >>> + /* right now a socket spends its whole life in the same cgroup */
> >>> + if (sk->sk_cgrp) {
> >>> + WARN_ON(1);
> >>> + return;
> >>> + }
> >>> + if (static_branch(&memcg_socket_limit_enabled)) {
> >>> + struct mem_cgroup *memcg;
> >>> +
> >>> + BUG_ON(!sk->sk_prot->proto_cgroup);
> >>> +
> >>> + rcu_read_lock();
> >>> + memcg = mem_cgroup_from_task(current);
> >>> + if (!mem_cgroup_is_root(memcg))
> >>> +   sk->sk_cgrp = sk->sk_prot->proto_cgroup(memcg);
> >>> + rcu_read_unlock();
> >>> ==

> >>>
> >>> sk->sk_cgrp is set to a memcg without any reference count.
> >>>
> >>> Then, no check for preventing rmdir() and freeing memcgroup.
> >>>
> >>> Is there some css_get() or mem_cgroup_get() somewhere ?
> >>>
> >>
> >> There were a css_get in the first version of this patchset. It was
> >> removed, however, because it was deemed anti-intuitive to prevent rmdir,
> >> since we can't know which sockets are blocking it, or do anything about
> >> it. Or did I misunderstand something ?
> >>
> >
> > Maybe I misuderstood. Thank you. Ok, there is no css_get/put and
> > rmdir() is allowed. But, hmm....what's guarding threads from stale
> > pointer access ?
> >
> > Does a memory cgroup which is pointed by sk->sk_cgrp always exist ?
> >
> If I am not mistaken, yes, it will. (Ok, right now it won't)
>
> Reason is a cgroup can't be removed if it is empty.
> To make it empty, you need to move the tasks away.
>
> So the sockets will be moved away as well when you do it. So right now
> they are not, so it would then probably be better to increase a
> reference count with a comment saying that it is temporary.
>

I'm sorry if I misunderstand.

At task exit, __fput() will be called against file descriptors, yes.
__fput() calles f_op->release() => inet_release() => tcp_close().

But TCP socket may be alive after task exit until it gets down to
protocol close. For example, until the all message in send buffer
is acked, socket and tcp connection will not be disappear.

In short, socket's lifetime is different from it's task's.
So, there may be sockets which are not belongs to any task.

Thanks,
-Kame