Subject: Re: [PATCH v9 3/9] socket: initial cgroup code.
Posted by Jason Baron on Thu, 22 Dec 2011 21:10:29 GMT
View Forum Message <> Reply to Message

On Mon, Dec 12, 2011 at 11:47:03AM +0400, Glauber Costa wrote:
> +
> +static bool mem_cgroup_is_root(struct mem_cgroup *memcg);
> +void sock_update_memcg(struct sock *sk)
> +{
> + /* 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)) {
> + mem_cgroup_get(memcg);
> + sk->sk_cgrp = sk->sk_prot->proto_cgroup(memcg);
> + }
> + rcu_read_unlock();
> + }
> +}
> +EXPORT_SYMBOL(sock_update_memcg);
> +
> +void sock_release_memcg(struct sock *sk)
> +{
> + if (static_branch(&memcg_socket_limit_enabled) && sk->sk_cgrp) {
> + struct mem_cgroup *memcg;
> + WARN_ON(!sk->sk_cgrp->memcg);
> + memcg = sk->sk_cgrp->memcg;
> + mem_cgroup_put(memcg);
> + }
> +}

Hi Glauber,

I think for 'sock_release_memcg()', you want:

static inline sock_release_memcg(sk)
{
 if (static_branch())
  __sock_release_memcg();

}

And then re-define the current sock_release_memcg -> __sock_release_memcg().
In that way the straight line path is a single no-op. As currently
written, there is function call and then an immediate return.

Thanks,

-Jason
