

---

Subject: Re: [PATCH v8 3/9] socket: initial cgroup code.  
Posted by [KAMEZAWA Hiroyuki](#) on Fri, 09 Dec 2011 02:05:50 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Mon, 5 Dec 2011 19:34:57 -0200  
Glauber Costa <glommer@parallels.com> wrote:

> The goal of this work is to move the memory pressure tcp  
> controls to a cgroup, instead of just relying on global  
> conditions.  
>  
> To avoid excessive overhead in the network fast paths,  
> the code that accounts allocated memory to a cgroup is  
> hidden inside a static\_branch(). This branch is patched out  
> until the first non-root cgroup is created. So when nobody  
> is using cgroups, even if it is mounted, no significant performance  
> penalty should be seen.  
>  
> This patch handles the generic part of the code, and has nothing  
> tcp-specific.  
>  
> Signed-off-by: Glauber Costa <glommer@parallels.com>  
> CC: Kirill A. Shutemov <kirill@shutemov.name>  
> CC: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>  
> CC: David S. Miller <davem@davemloft.net>  
> CC: Eric W. Biederman <ebiederm@xmission.com>  
> CC: Eric Dumazet <eric.dumazet@gmail.com>

I already replied Reviewed-by: but...

```
> /* Writing them here to avoid exposing memcg's inner layout */  
> #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM  
> #ifdef CONFIG_INET  
> #include <net/sock.h>  
> +  
> +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();
> +
> +}
```

Here, you do mem\_cgroup\_get() if !mem\_cgroup\_is\_root().

```
> +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);
> +
> +}
> +
>
```

You don't check !mem\_cgroup\_is\_root(). Hm, root memcg will not be freed by this ?

Thanks,  
-Kame

---