Subject: Re: [PATCH v7 04/10] tcp memory pressure controls
Posted by Glauber Costa on Fri, 02 Dec 2011 17:57:28 GMT
View Forum Message <> Reply to Message

On 11/29/2011 11:49 PM, KAMEZAWA Hiroyuki wrote:
>
>> -static struct mem_cgroup *mem_cgroup_from_cont(struct cgroup *cont)
>> +struct mem_cgroup *mem_cgroup_from_cont(struct cgroup *cont)
>> {
>>   return container_of(cgroup_subsys_state(cont,
>>   mem_cgroup_subsys_id), struct mem_cgroup,
>> @@ -4717,14 +4732,27 @@ static int register_kmem_files(struct cgroup *cont, struct
cgroup_subsys *ss)
>>
>>   ret = cgroup_add_files(cont, ss, kmem_cgroup_files,
>>   ARRAY_SIZE(kmem_cgroup_files));
>> +
>> + if (!ret)
>> + ret = mem_cgroup_sockets_init(cont, ss);
>>   return ret;
>> };
>
> You does initizalication here. The reason what I think is
> 1. 'proto_list' is not available at createion of root cgroup and
>   you need to delay set up until mounting.
>
> If so, please add comment or find another way.
> This seems not very clean to me.

Yes, we do can run into some ordering issues. A part of the
initialization can be done earlier. But I preferred to move it all later
instead of creating two functions for it. But I can change that if you
want, no big deal.

>
>
>
>
>> +static DEFINE_RWLOCK(proto_list_lock);
>> +static LIST_HEAD(proto_list);
>> +
>> +#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
>> +int mem_cgroup_sockets_init(struct cgroup *cgrp, struct cgroup_subsys *ss)
>> +{
>> + struct proto *proto;
>> + int ret = 0;
>> +
>> + read_lock(&proto_list_lock);

>> + list_for_each_entry(proto,&proto_list, node) {
>> + if (proto->init_cgroup)
>> +   ret = proto->init_cgroup(cgrp, ss);
>> +   if (ret)
>> +   goto out;
>> + }
>
> seems indent is bad or {} is missing.
>
Thanks. I'll rewrite it, since I did miss {} around the first if. But no
test could possibly catch it, since what I wanted to write, and what I
wrote by mistake end up being equivalent.

>> +EXPORT_SYMBOL(memcg_tcp_enter_memory_pressure);
>> +
>> +int tcp_init_cgroup(struct cgroup *cgrp, struct cgroup_subsys *ss)
>> +{
>> + /*
>> + * The root cgroup does not use res_counters, but rather,
>> + * rely on the data already collected by the network
>> + * subsystem
>> + */
>> + struct res_counter *res_parent = NULL;
>> + struct cg_proto *cg_proto;
>> + struct tcp_memcontrol *tcp;
>> + struct mem_cgroup *memcg = mem_cgroup_from_cont(cgrp);
>> + struct mem_cgroup *parent = parent_mem_cgroup(memcg);
>> +
>> + cg_proto = tcp_prot.proto_cgroup(memcg);
>> + if (!cg_proto)
>> + return 0;
>> +
>> + tcp = tcp_from_cgproto(cg_proto);
>> + cg_proto->parent = tcp_prot.proto_cgroup(parent);
>> +
>> + tcp->tcp_prot_mem[0] = sysctl_tcp_mem[0];
>> + tcp->tcp_prot_mem[1] = sysctl_tcp_mem[1];
>> + tcp->tcp_prot_mem[2] = sysctl_tcp_mem[2];
>> + tcp->tcp_memory_pressure = 0;
>
> Question:
>
> Is this value will be updated when an admin chages sysctl ?

yes.

> I guess, this value is set at system init script or some which may
> happen later than mounting cgroup.

> I don't like to write a guideline 'please set sysctl val before
> mounting cgroup'

Agreed.

This code is in patch 6 (together with the limiting):

```
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+       rcu_read_lock();
+       memcg = mem_cgroup_from_task(current);
+
+       tcp_prot_mem(memcg, vec[0], 0);
+       tcp_prot_mem(memcg, vec[1], 1);
+       tcp_prot_mem(memcg, vec[2], 2);
+       rcu_read_unlock();
+#endif
```

tcp_prot_mem is just a wrapper around the assignment so we can access
memcg's inner fields.
