Subject: Re: [PATCH v6 10/10] Disable task moving when using kernel memory accounting
Posted by Glauber Costa on Mon, 28 Nov 2011 11:00:24 GMT
View Forum Message <> Reply to Message

On 11/28/2011 02:32 AM, KAMEZAWA Hiroyuki wrote:
> On Fri, 25 Nov 2011 15:38:16 -0200
> Glauber Costa<glommer@parallels.com> wrote:
>
>> Since this code is still experimental, we are leaving the exact
>> details of how to move tasks between cgroups when kernel memory
>> accounting is used as future work.
>>
>> For now, we simply disallow movement if there are any pending
>> accounted memory.
>>
>> Signed-off-by: Glauber Costa<glommer@parallels.com>
>> CC: Hiroyouki Kamezawa<kamezawa.hiroyu@jp.fujitsu.com>
>> ---
>>   mm/memcontrol.c | 23 ++++++++++++++++++++++-
>>   1 files changed, 22 insertions(+), 1 deletions(-)
>>
>> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
>> index 2df5d3c..ab7e57b 100644
>> --- a/mm/memcontrol.c
>> +++ b/mm/memcontrol.c
>> @@ -5451,10 +5451,19 @@ static int mem_cgroup_can_attach(struct cgroup_subsys *ss,
>>   {
>>   int ret = 0;
>>   struct mem_cgroup *mem = mem_cgroup_from_cont(cgroup);
>> + struct mem_cgroup *from = mem_cgroup_from_task(p);
>> +
>> +#if defined(CONFIG_CGROUP_MEM_RES_CTLR_KMEM)&& defined(CONFIG_INET)
>> + if (from != mem&& !mem_cgroup_is_root(from)&&
>> +   res_counter_read_u64(&from->tcp_mem.tcp_memory_allocated, RES_USAGE)) {
>> + printk(KERN_WARNING "Can't move tasks between cgroups: "
>> +   "Kernel memory held. task: %s\n", p->comm);
>> + return 1;
>> + }
>> +#endif
>
> Hmm, the kernel memory is not guaranteed as being held by the 'task' ?
>
> How about
> "Now, moving task between cgroup is disallowed while the source cgroup
>   containes kmem reference." ?
>
> Hmm.. we need to fix this task-move/rmdir issue before production use.

>
>
> Thanks,
> -Kame
>
Hi Kame,

Let me tell you the direction I am going wrt task movement: The only reasons I haven't included so far, is that I believe it needs more testing, and as you know, I am right now more interested in getting past the initial barriers for inclusion. I am committed to fix anything that needs to be fixed - stylish or non-stylish before we remove the experimental flag.

So what I intend to do, is to basically
* lock the task,
* scan through its file descriptors list,
* identify which of them are sockets,
* cast them to struct sock *,
* see if it has a cgrp associated
* see if cgrp == from

At this point we can decrement sockets allocated by 1 in from, and memory_allocated by sk_forward_alloc (increasing by equal quantities in the destination cgroup)

I belive it will work.
