Subject: Re: [PATCH v7 10/10] Disable task moving when using kernel memory accounting
Posted by KAMEZAWA Hiroyuki on Mon, 05 Dec 2011 02:18:35 GMT
View Forum Message <> Reply to Message

On Fri, 2 Dec 2011 16:11:56 -0200
Glauber Costa <glommer@parallels.com> wrote:

> On 11/30/2011 12:22 AM, KAMEZAWA Hiroyuki wrote:
> > On Tue, 29 Nov 2011 21:57:01 -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 a31a278..dd9a6d9 100644
> >> --- a/mm/memcontrol.c
> >> +++ b/mm/memcontrol.c
> >> @@ -5453,10 +5453,19 @@ static int mem_cgroup_can_attach(struct cgroup_subsys *ss,
> >>   {
> >>    int ret = 0;
> >>    struct mem_cgroup *memcg = 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 != memcg&&  !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.\n");
> >> +  return 1;
> >> + }
> >> +#endif
```

> >
> > I wonder....reading all codes again, this is incorrect check.
> >
> > Hm, let me cralify. IIUC, in old code, "prevent moving" is because you hold
> > reference count of cgroup, which can cause trouble at rmdir() as leaking refcnt.

> right.
>
> > BTW, because socket is a shared resource between cgroup, changes in mm->owner
> > may cause task cgroup moving implicitly. So, if you allow leak of resource
> > here, I guess... you can take mem_cgroup_get() refcnt which is memcg-local and
> > allow rmdir(). Then, this limitation may disappear.
>
> Sorry, I didn't fully understand. Can you clarify further?
> If the task is implicitly moved, it will end up calling can_attach as
> well, right?
>
I'm sorry that my explanation is bad.

You can take memory cgroup itself's reference count by mem_cgroup_put/get.
By getting this, memory cgroup object will continue to exist even after
its struct cgroup* is freed by rmdir().

So, assume you do mem_cgroup_get()/put at socket attaching/detatching.

0) A task has a tcp socekts in memcg0.

```
task(memcg0)
 +- socket0 --> memcg0,usage=4096
```

1) move this task to memcg1

```
task(memcg1)
 +- socket0 --> memcg0,usage=4096
```

2) The task create a new socket.

```
task(memcg1)
 +- socekt0 --> memcg0,usage=4096
 +- socket1 --> memcg1,usage=xxxx
```

Here, the task will hold 4096bytes of usage in memcg0 implicitly.

3) an admin removes memcg0
```
task(memcg1)
 +- socket0 -->memcg0, usage=4096 <-----(*)
 +- socket1 -->memcg1, usage=xxxx
```

(*) is invisible to users....but this will not be very big problem.

Thanks,
-Kame

Thanks,
-Kame