Subject: [PATCH v6 8/8] Disable task moving when using kernel memory accounting
Posted by Glauber Costa on Mon, 10 Oct 2011 10:24:28 GMT
View Forum Message <> Reply to Message

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 |   31 ++++++++++++++++++-------------
 1 files changed, 18 insertions(+), 13 deletions(-)

diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index ca08be2..8d1beeb 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -408,23 +408,11 @@ void sock_update_memcg(struct sock *sk)

  rcu_read_lock();
  sk->sk_cgrp = mem_cgroup_from_task(current);
-
- /*
-  * We don't need to protect against anything task-related, because
-  * we are basically stuck with the sock pointer that won't change,
-  * even if the task that originated the socket changes cgroups.
-  *
-  * What we do have to guarantee, is that the chain leading us to
-  * the top level won't change under our noses. Incrementing the
-  * reference count via cgroup_exclude_rmdir guarantees that.
-  */
- cgroup_exclude_rmdir(mem_cgroup_css(sk->sk_cgrp));
  rcu_read_unlock();
 }

 void sock_release_memcg(struct sock *sk)
 {
- cgroup_release_and_wakeup_rmdir(mem_cgroup_css(sk->sk_cgrp));
 }

 void memcg_sockets_allocated_dec(struct mem_cgroup *memcg, struct proto *prot)
@@ -5636,10 +5624,17 @@ 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 (from != mem &&
+     res_counter_read_u64(&mem->tcp.tcp_memory_allocated, RES_USAGE)) {
+  printk(KERN_WARNING "Can't move tasks between cgroups: "
+   "Kernel memory held. task: %s\n", p->comm);
+  return 1;
+ }

  if (mem->move_charge_at_immigrate) {
   struct mm_struct *mm;
-  struct mem_cgroup *from = mem_cgroup_from_task(p);

   VM_BUG_ON(from == mem);

@@ -5807,6 +5802,16 @@ static int mem_cgroup_can_attach(struct cgroup_subsys *ss,
     struct cgroup *cgroup,
     struct task_struct *p)
 {
+ struct mem_cgroup *mem = mem_cgroup_from_cont(cgroup);
+ struct mem_cgroup *from = mem_cgroup_from_task(p);
+
+ if (from != mem &&
+     res_counter_read_u64(&mem->tcp.tcp_memory_allocated, RES_USAGE)) {
+  printk(KERN_WARNING "Can't move tasks between cgroups: "
+   "Kernel memory held. task: %s\n", p->comm);
+  return 1;
+ }
+
  return 0;
 }
 static void mem_cgroup_cancel_attach(struct cgroup_subsys *ss,
--
1.7.6.4
```