Subject: Re: [RFC] Transactional CGroup task attachment
Posted by Daisuke Nishimura on Mon, 14 Jul 2008 06:28:22 GMT
View Forum Message <> Reply to Message

```
On Fri, 11 Jul 2008 09:20:58 +0900, KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
wrote:
> Thank you for your effort.
>
> On Wed, 9 Jul 2008 23:46:33 -0700
> "Paul Menage" <menage@google.com> wrote:
> > 3) memory
> >
> > Curently the memory cgroup only uses the mm->owner's cgroup at charge
> > time, and keeps a reference to the cgroup on the page. However,
> > patches have been proposed that would move all non-shared (page count
> > == 1) pages to the destination cgroup when the mm->owner moves to a
> > new cgroup. Since it's not possible to prevent page count changes
> > without locking all mms on the system, even this transaction approach
> > can't really give guarantees. However, something like the following
> > would probably be suitable. It's very similar to the memrlimit
> > approach, except for the fact that we have to handle the fact that the
> > number of pages we finally move might not be exactly the same as the
> > number of pages we thought we'd be moving.
> >
> > prepare_attach_sleep() {
> >   down_read(&mm->mmap_sem);
> >   if (mm->owner != state->task) return 0;
> >   count = count_unshared_pages(mm);
> >   // save the count charged to the new cgroup
> >   state->subsys[memcgroup_subsys_id] = (void *)count;
> >   if ((ret = res_counter_charge(state->dest, count)) {
> >     up_read(&mm->mmap_sem);
> >   }
> >   return ret;
> > }
> >
> > commit_attach() {
> >   if (mm->owner == state->task) {
> >     final_count = move_unshared_pages(mm, state->dest);
> >     res_counter_uncharge(state->src, final_count);
> >     count = state->subsys[memcgroup_subsys_id];
> >     res_counter_force_charge(state->dest, final_count - count);
> >   }
> >   up_read(&mm->mmap_sem);
> > }
> >
> > abort_attach_sleep() {
> >   if (mm->owner == state->task) {
```

```
> >     count = state->subsys[memcgroup_subsys_id];
> >     res_counter_uncharge(state->dest, count);
> >   }
> >   up_read(&mm->mmap_sem);
> > }
> >
>
```
> At frist look, maybe works well. we need some special codes (to move resource)
> but that's all.
>
> My small concern is a state change between prepare_attach_sleep() ->
> commit_attach(). Hmm...but as you say, we cannot do down_write(mmap_sem).
> Maybe inserting some check codes to mem_cgroup_charge() to stop charge while
> move is the last thing we can do.
>
I have two comments.

- I think page reclaiming code decreases the memory charge
  without holding mmap_sem(e.g. try_to_unmap(), __remove_mapping()).
  Shouldn't we handle these cases?
- When swap controller is merged, I should implement
  prepare_attach_nosleep() which holds swap_lock.

> Anyway, if unwinding is supported officially, I think we can find a way
> to go.
>
I think so too.

Thanks,
Daisuke Nishimura.