Subject: Re: [RFD][PATCH] memcg: Move Usage at Task Move
Posted by Daisuke Nishimura on Wed, 11 Jun 2008 03:03:45 GMT
View Forum Message <> Reply to Message

On Tue, 10 Jun 2008 17:26:37 +0900, KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

```
> > > This is a trial to move "usage" from old cg to new cg at task move.
> > > Finally, you'll see the problems we have to handle are failure and rollback.
> > >
> > > This one's Basic algorithm is
> > >
> > >      0. can_attach() is called.
> > >      1. count movable pages by scanning page table. isolate all pages from LRU.
> > >      2. try to create enough room in new memory cgroup
> > >      3. start moving page accouing
> > >      4. putback pages to LRU.
> > >      5. can_attach() for other cgroups are called.
> > >
> > You isolate pages and move charges of them by can_attach(),
> > but it means that pages that are allocated between page isolation
> > and moving tsk->cgroups remains charged to old group, right?
> yes.
>
> >
> > I think it would be better if possible to move charges by attach()
> > as cpuset migrates pages by cpuset_attach().
> > But one of the problem of it is that attch() does not return
> > any value, so there is no way to notify failure...
> >
> yes, here again. it makes roll-back more difficult.
>
```
I think so too. That's why I said "one of the problem".

```
> > > A case study.
> > >
> > >   group_A -> limit=1G, task_X's usage= 800M.
> > >   group_B -> limit=1G, usage=500M.
> > >
> > > For moving task_X from group_A to group_B.
> > >   - group_B  should be reclaimed or have enough room.
> > >
> > > While moving task_X from group_A to group_B.
> > >   - group_B's memory usage can be changed
> > >   - group_A's memory usage can be changed
> > >
> > >   We accounts the resouce based on pages. Then, we can't move all resource
> > >   usage at once.
> > >
```

> > >   If group_B has no more room when we've moved 700M of task_X to group_B,
> > >   we have to move 700M of task_X back to group_A. So I implemented roll-back.
> > >   But other process may use up group_A's available resource at that point.
> > >
> > >   For avoiding that, preserve 800M in group_B before moving task_X means that
> > >   task_X can occupy 1600M of resource at moving. (So I don't do in this patch.)
> > >
> > >   This patch uses Best-Effort rollback. Failure in rollback is ignored and
> > >   the usage is just leaked.
> > >
> > If implement rollback in kernel, I think it must not fail to prevent
> > leak of usage.
> > How about using "charge_force" for rollbak?
> >
> means allowing to exceed limit ?
>
Yes.
I agree that exceeding limit is not good, but I
just feel that it's better than leaking usage.
Of cource, I think usage should be decreased later
by some methods.

> > Or, instead of implementing rollback in kernel,
> > how about making user(or middle ware?) re-echo pid to rollbak
> > on failure?
> >
>
> "If the users does well, the system works in better way" is O.K.
> "If the users doesn't well, the system works in broken way" is very bad.
>
Hum...

I think users must know what they are doing.

They must know that moving a process to another group
that doesn't have enough room for it may fail with half state,
if it is the behavior of kernel.
And they should handle the error by themselves, IMHO.

Thanks,
Daisuke Nishimura.