Subject: Re: [RFD][PATCH] memcg: Move Usage at Task Move
Posted by yamamoto on Tue, 10 Jun 2008 12:57:03 GMT
View Forum Message <> Reply to Message

> On Tue, 10 Jun 2008 14:50:32 +0900 (JST)
> yamamoto@valinux.co.jp (YAMAMOTO Takashi) wrote:
>
> > > 3. Use Lazy Manner
> > > When the task moves, we can mark the pages used by it as
> > > "Wrong Charge, Should be dropped", and add them some penalty in the LRU.
> > > Pros.
> > > - no complicated ones.
> > > - the pages will be gradually moved at memory pressure.
> > > Cons.
> > > - A task's usage can exceed the limit for a while.
> > > - can't handle mlocked() memory in proper way.
> > >
> > > 4. Allow Half-moved state and abandon rollback.
> > > Pros.
> > > - no complicated ones in the code.
> > > Cons.
> > > - the users will be in chaos.
> >
> > how about:
> >
> > 5. try to move charges as your patch does.
> > if the target cgroup's usage is going to exceed the limit,
> > try to shrink it. if it failed, just leave it exceeded.
> > (ie. no rollback)
> > for the memory subsystem, which can use its OOM killer,
> > the failure should be rare.
> >
>
> Hmm, allowing exceed and cause OOM kill ?
>
> One difficult point is that the users cannot know they can move task
> without any risk. How to handle the risk can be a point.
> I don't like that approarch in general because I don't like "exceed"
> status. But implementation will be easy.

regardless of how to handle task moves,
it's important to provide information to help users
to avoid unreasonable cgroup/task placement.
otherwise, they will be surprised by OOM-killer etc anyway.

having said that, if you decide to put too large tasks into
a cgroup with too small limit, i don't think that there are
many choices besides OOM-kill and allowing "exceed".

actually, i think that #3 and #5 are somewhat similar.
a big difference is that, while #5 shrinks the cgroup immediately,
#3 does it later. in case we need to do OOM-kill, i prefer to do it
sooner than later.

> > > After writing this patch, for me, "3" is attractive. now.
> > > (or using Lazy manner and allow moving of usage instead of freeing it.)
> > >
> > > One reasone is that I think a typical usage of memory controller is
> > > fork()->move->exec(). (by libcg ?) and exec() will flush the all usage.
> >
> > i guess that moving long-running applications can be desirable
> > esp. for not so well-designed systems.
> >
>
> hmm, for not so well-designed systems....true.
> But "5" has the same kind of risks for not so well-desgined systems ;)

i don't claim that #5 is a perfect solution for everyone. :)

YAMAMOTO Takashi