Subject: Re: [RFC][PATCH] another swap controller for cgroup
Posted by Paul Menage on Thu, 15 May 2008 07:19:39 GMT
View Forum Message <> Reply to Message

On Wed, May 14, 2008 at 11:23 PM, YAMAMOTO Takashi
<yamamoto@valinux.co.jp> wrote:
> > >
> >
> > You need to be able to sleep in order to take mmap_sem, right?
>
> yes.

OK, well in the thread on balbir's vm limit controller, we discussed
the idea of having mmap_sem held across calls to
mm_update_next_owner(), and hence the cgroup callbacks. Would that
help if mmap_sem were already held when you get the mm_owner_changed()
callback.

>
> > A few comments on the patch:
> >
> > - you're not really limiting swap usage, you're limiting swapped-out
> > address space. So it looks as though if a process has swapped out most
> > of its address space, and forks a child, the total "swap" charge for
> > the cgroup will double. Is that correct?
>
> yes.
>
>
> > If so, why is this better
> > than charging for actual swap usage?
>
> its behaviour is more determinstic and it uses less memory.
> (than nishimura-san's one, which charges for actual swap usage.)
>

Using less memory is good, but maybe not worth it if the result isn't so useful.

I'd say that it's less deterministic than nishimura-san's controller -
with his you just need to know how much swap is in use (which you can
tell by observing the app on a real system) but with yours you also
have to know whether there are any processes sharing anon pages (but
not mms).

Now it's true that if all the apps you need to run do an execve()
after forking, then the number of swap ptes really does track the
amount of swap space in use pretty accurately, since there's not going
to be any sharing of anon memory between mms. And it might be that

people decide that the reduced memory overhead justifies this
limitation. But I think it should be made explicit in the patch
description and documentation that this controller achieves its
reduced overhead at the cost of giving (IMO) bogus results on a rather
ancient but still perfectly legitimate class of Unix application. (The
apache httpd server used to work this way, for instance. It may still
but I've not looked at it in a while).

>
> > - what will happen if someone creates non-NPTL threads, which share an
> > mm but not a thread group (so each of them is a thread group leader)?
>
> a thread which is most recently assigned to a cgroup will "win".
>

Doesn't that risk triggering the BUG_ON(mm->swap_cgroup != oldscg) in
swap_cgroup_attach() ?

>
> > - if you were to store a pointer in the page rather than the
>
> "a pointer"? a pointer to what?

Oops, sorry - I meant to say "a pointer to the mm". So from there you
can get to mm->owner, and hence to mm->owner->cgroups[swap]

Paul