Subject: Re: [PATCH 00/11] kmem controller for memcg: stripped down version
Posted by akpm on Tue, 26 Jun 2012 21:55:39 GMT
View Forum Message <> Reply to Message

On Tue, 26 Jun 2012 11:17:49 +0400
Glauber Costa <glommer@parallels.com> wrote:

> On 06/26/2012 03:27 AM, Andrew Morton wrote:
> > On Mon, 25 Jun 2012 18:15:17 +0400
> > Glauber Costa <glommer@parallels.com> wrote:
> >
> >> What I am proposing with this series is a stripped down version of the
> >> kmem controller for memcg that would allow us to merge significant parts
> >> of the infrastructure, while leaving out, for now, the polemic bits about
> >> the slab while it is being reworked by Cristoph.
> >>
> >> Me reasoning for that is that after the last change to introduce a gfp
> >> flag to mark kernel allocations, it became clear to me that tracking other
> >> resources like the stack would then follow extremely naturaly. I figured
> >> that at some point we'd have to solve the issue pointed by David, and avoid
> >> testing the Slab flag in the page allocator, since it would soon be made
> >> more generic. I do that by having the callers to explicit mark it.
> >>
> >> So to demonstrate how it would work, I am introducing a stack tracker here,
> >> that is already a functionality per-se: it successfully stops fork bombs to
> >> happen. (Sorry for doing all your work, Frederic =p). Note that after all
> >> memcg infrastructure is deployed, it becomes very easy to track anything.
> >> The last patch of this series is extremely simple.
> >>
> >> The infrastructure is exactly the same we had in memcg, but stripped down
> >> of the slab parts. And because what we have after those patches is a feature
> >> per-se, I think it could be considered for merging.
> >
> > hm. None of this new code makes the kernel smaller, faster, easier to
> > understand or more fun to read!
> Not sure if this is a general comment - in case I agree - or if targeted
> to my statement that this is "stripped down". If so, it is of course
> smaller relative to my previous slab accounting patches.

It's a general comment. The patch adds overhead: runtime costs and
maintenance costs. Do its benefits justify that cost?

> The infrastructure is largely common, but I realized that a future user,
> tracking the stack, would be a lot simpler and could be done first.
>
> > Presumably we're getting some benefit for all the downside. When the
> > time is appropriate, please do put some time into explaining that
> > benefit, so that others can agree that it is a worthwhile tradeoff.

> >
>
> Well, for one thing, we stop fork bombs for processes inside cgroups.

"inside cgroups" is a significant limitation! Is this capability
important enough to justify adding the new code? That's unobvious
(to me).

Are there any other user-facing things which we can do with this
feature? Present, or planned?

> I can't speak for everybody here, but AFAIK, tracking the stack through
> the memory it used, therefore using my proposed kmem controller, was an
> idea that good quite a bit of traction with the memcg/memory people.
> So here you have something that people already asked a lot for, in a
> shape and interface that seem to be acceptable.

mm, maybe. Kernel developers tend to look at code from the point of
view "does it work as designed", "is it clean", "is it efficient", "do
I understand it", etc. We often forget to step back and really
consider whether or not it should be merged at all.

I mean, unless the code is an explicit simplification, we should have
a very strong bias towards "don't merge".