Subject: Re: Fork bomb limitation in memcg WAS: Re: [PATCH 00/11] kmem controller for memcg: stripped down ve
Posted by Glauber Costa on Thu, 28 Jun 2012 09:01:23 GMT
View Forum Message <> Reply to Message

On 06/27/2012 11:38 PM, David Rientjes wrote:
> On Wed, 27 Jun 2012, Glauber Costa wrote:
>
>> fork bombs are a way bad behaved processes interfere with the rest of
>> the system. In here, I propose fork bomb stopping as a natural
>> consequence of the fact that the amount of kernel memory can be limited,
>> and each process uses 1 or 2 pages for the stack, that are freed when the
>> process goes away.
>>
>
> The obvious disadvantage is that if you use the full-featured kmem
> controller that builds upon this patchset, then you're limiting the about
> of all kmem, not just the stack that this particular set limits. I hope
> you're not proposing it to go upstream before full support for the kmem
> controller is added so that users who use it only to protect again
> forkbombs soon realize that's no longer possible if your applications do
> any substantial slab allocations, particularly anything that does a lot of
> I/O.

Point by point:

1) This is not a disadvantage. The whole point of implementing it as
kmem, not as a "fork controller" or anything in the like, was our
understanding that people should not be "protecting against a x number
of processes". Because this is unnatural. All you have is total of
kernel memory, because that's what the interface gives you. If you can
overuse memory, you can't fork bomb. But that's a consequence.

I'll grant that maybe it was a mistake of mine to try to sell it this
way here, because it may give the wrong idea. In this case I welcome
your comment because it will allow me to be more careful in future
communications. But this was just me trying to show off the feature.

2) No admin should never, ever tune the system to a particular kmem
limit value. And again, this is the whole point of not limiting "number
of processes". I agree that adding slab tracking will raise kernel
memory consumption by a reasonable amount. But reality is that even if
this controller is totally stable, that not only can, but will happen.

Unlike user memory, where whoever writes the application control its
behavior (forget about the libraries for a moment), users never really
control the internals of the kernel. If you ever rely on the fact that
you are currently using X Gb of kmem, and that should be enough, your

setup will break when the data structure grows - as they do - when the
kernel memory consumption rises naturally by algorithms - as it does,
and so on.

3) Agreeing that of course, preventing disruption if we can help it is
good, this feature is not only marked experimental, but default of. This
was done precisely not to disrupt the amount of *user* memory used,
where I actually think it makes a lot of sense ("My application
allocates 4G, that's what I'll give it!"). Even if support is compiled
in, it won't be until you start limiting it that anything will happen.
Kernel Memory won't even be tracked until then. And I also understand
that our use case here may be quite unique: We want the kernel memory
limit to be way lower than the user limit (like 20 % - but that's still
a percentage, not a tune!). When I discussed this around with other
people, the vast majority of them wanted to set kmem = umem. Which
basically means "user memory is the limit, but we want the kernel memory
to be accounted as well".

So yes, although I understand your point - but not fully agree, I would
like to get it merged as is. I don't believe adding slab memory later
will be disruptive, in the same way I didn't believe adding stack later
- in my original slab tracking patch - would be. As I originally stated,
this would allow me to greatly simplify the slab tracking patches, since
we'll be able to focus on that only, instead of a gigantic patch that
does a lot.

Other people have interest in this, so this would allow them to start
building on it as well. That said, I won't oppose adding more code if
you suggest so to make sure people rely less on the accidental fact that
we're only tracking a part of what we will, even if they shouldn't.

- It's already experimental
- It's already default off.
- I could add a warn message first time it is set.
- I could add a boot option.

Or anything like this.

Last, but not least, note that it is totally within my interests to
merge the slab tracking as fast as we can. it'll be a matter of going
back to it, and agreeing in the final form.

>
> In other words, if I want to run netperf in a memcg with the full-featured
> kmem controller enabled, then its kmem limit must be high enough so that
> it doesn't degrade performance that any limitation on stack allocations
> would be too high to effectively stop forkbombs.
>

That is a user setup problem. As I explained before, That's exactly what
we want to discourage by exposing "kernel memory" instead of a
particular tunable.

As it is, I agree, it will stop fork bombs but will take a lot more time
for it than it should. But this only makes it analogous to the
evolutionary precursors to the human eye: It's not perfect, but will
achieve something that may be already of great value for a class of
users. Improvements are certain to follow.

Thank you David!