Subject: Re: [PATCH v2 04/11] kmem accounting basic infrastructure
Posted by Glauber Costa on Wed, 15 Aug 2012 09:33:55 GMT
View Forum Message <> Reply to Message

>> We always account to both user and kernel resource_counters. This
>> effectively means that an independent kernel limit is in place when the
>> limit is set to a lower value than the user memory. A equal or higher
>> value means that the user limit will always hit first, meaning that kmem
>> is effectively unlimited.
>
> Well, it contributes to the user limit so it is not unlimited. It just
> falls under a different limit and it tends to contribute less.

You are right, but this is just wording. I will update it, but what I
really mean here is that an independent limit is no imposed on kmem.

> This can
> be quite confusing. I am still not sure whether we should mix the two
> things together. If somebody wants to limit the kernel memory he has to
> touch the other limit anyway. Do you have a strong reason to mix the
> user and kernel counters?

This is funny, because the first opposition I found to this work was
"Why would anyone want to limit it separately?" =p

It seems that a quite common use case is to have a container with a
unified view of "memory" that it can use the way he likes, be it with
kernel memory, or user memory. I believe those people would be happy to
just silently account kernel memory to user memory, or at the most have
a switch to enable it.

What gets clear from this back and forth, is that there are people
interested in both use cases.

> My impression was that kernel allocation should simply fail while user
> allocations might reclaim as well. Why should we reclaim just because of
> the kernel allocation (which is unreclaimable from hard limit reclaim
> point of view)?

That is not what the kernel does, in general. We assume that if he wants
that memory and we can serve it, we should. Also, not all kernel memory
is unreclaimable. We can shrink the slabs, for instance. Ying Han
claims she has patches for that already...

> I also think that the whole thing would get much simpler if those two
> are split. Anyway if this is really a must then this should be
> documented here.

Well, documentation can't hurt.

>
> This doesn't check for the hierachy so kmem_accounted might not be in
> sync with it's parents. mem_cgroup_create (below) needs to copy
> kmem_accounted down from the parent and the above needs to check if this
> is a similar dance like mem_cgroup_oom_control_write.
>

I don't see why we have to.

I believe in a A/B/C hierarchy, C should be perfectly able to set a
different limit than its parents. Note that this is not a boolean.

Also, right now, C can become completely unlimited (by not setting a
limited) and this is, indeed, not the desired behavior.

A later patch will change kmem_accounted to a bitfield, and we'll use
one of the bits to signal that we should account kmem because our parent
is limited.
