Subject: Re: [PATCH v2 02/13] memcg: Kernel memory accounting infrastructure.
Posted by Glauber Costa on Wed, 14 Mar 2012 12:29:25 GMT
View Forum Message <> Reply to Message

>> This has been discussed before, I can probably find it in the archives
>> if you want to go back and see it.
>>
>
> Yes. IIUC, we agreed to have independet kmem limit. I just want to think it
> again because there are too many proposals and it seems I'm in confusion.
>

Sure thing. The discussion turned out good, so I'm glad you asked =)

>
>> But in a nutshell:
>>
>> 1) Supposing independent knob disappear (I will explain in item 2 why I
>> don't want it to), I don't thing a flag makes sense either. *If* we are
>> planning to enable/disable this, it might make more sense to put some
>> work on it, and allow particular slabs to be enabled/disabled by writing
>> to memory.kmem.slabinfo (-* would disable all, +* enable all, +kmalloc*
>> enable all kmalloc, etc).
>>
> seems interesting.
I'll try to cook a PoC.

>> All that said, while reading your message, thinking a bit, the following
>> crossed my mind:
>>
>> - We can account the slabs to memcg->res normally, and just store the
>> information that this is kernel memory into a percpu counter, as
>> I proposed recently.
>
> Ok, then user can see the amount of kernel memory.
>
>
>> - The knob goes away, and becomes implicit: if you ever write anything
>> to memory.kmem.limit_in_bytes, we transfer that memory to a separate
>> kmem res_counter, and proceed from there. We can keep accounting to
>> memcg->res anyway, just that kernel memory will now have a separate
>> limit.
>
> Okay, then,
>
> kmem_limit< memory.limit< memsw.limit
>
> ...seems reasonable to me.

> This means, user can specify 'ratio' of kmem in memory.limit.
Yes, I believe so. It is a big improvement over the current interface
we have today, IMHO.

>
> More consideration will be interesting.
>
> - We can show the amount of reclaimable kmem by some means ?
That's hard to do. The users of the cache have this information, the
underlying slab/slub/slut code do not. We need to rely on the cache
owner to provide this, and provide correctly. So the chances we'll have
incorrect information here grows by quite a bit.

> - What happens when a new cgroup created ?

mem_cgroup_create() is called =)
Heh, jokes apart, I don't really follow here. What exactly do you mean?
There shouldn't be anything extremely out of the ordinary.

> - Should we have 'ratio' interface in kernel level ?
I personally don't like a ratio interface. I believe specifying "kmem
should never be allowed to go over X bytes" is more than enough.

> - What happens at task moving ?

 From kmem PoV, nothing. It is ultimately impossible to track a slab
page to a task. The page contains objects that were allocated from
multiple tasks. Only when the whole cgroup is destroyed, is that
we take any action.

> - Should we allow per-slab accounting knob in /sys/kernel/slab/xxx ?
>   or somewhere ?

Not really follow.

> - Should we show per-memcg usage in /sys/kernel/slab/xxx ?
I guess so.

> - Should we have force_empty for kmem (as last resort) ?
We do that when the cgroup is going away. From user action, I suspect
the best we can do is call the shrinkers, and see if they get freed.

>
> With any implementation, my concern is
> - overhead/performance.
Yes. For the next round, we need to add some more detailed benchmarks.
> - unreclaimable kmem
That's actually the reason behind all that!

So if you have a 1 Gb mem allowance, and you fill it with unreclaimable kmem, you are in trouble, yes.

Point is, at least all your allocations will stop working, and something will be done soon. Without kmem tracking, this can grow and grow, outside the cgroups border.

> - shared kmem between cgroups.

Right now both proposals ended up doing account to first user. In theory, it leaves a gap under which a smart cgroup can go pinning a lot of kmem without owning it.

But I still believe this to be the best way forward.
It is hard to determine who the object user is without cooperation from the object caches. And even then, it is even harder to do so without penalizing every single object allocation (right now we only penalize a new page allocation, which is way better performance-wise).

>
>
>> - With this scheme, it may not be necessary to ever have a file
>> memory.kmem.soft_limit_in_bytes. Reclaim is always part of the normal
>> memcg reclaim.
>>
> Good.
>
>> The outlined above would work for us, and make the whole scheme simpler,
>> I believe.
>>
>> What do you think ?
>
> It sounds interesting to me.
>
> Thanks,
> -Kame
>
>
>
>
>
>
>

>
>
>
>
> --
> To unsubscribe from this list: send the line "unsubscribe cgroups" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html