Subject: Re: [PATCH v2 18/29] memcg: kmem controller charge/uncharge infrastructure
Posted by KAMEZAWA Hiroyuki on Wed, 16 May 2012 09:15:37 GMT
View Forum Message <> Reply to Message

(2012/05/16 17:25), Glauber Costa wrote:

> On 05/16/2012 12:18 PM, KAMEZAWA Hiroyuki wrote:
>>> If at this point the memcg hits a NOFAIL allocation worth 2 pages, by
>>>> the method I am using, the memcg will be at 4M + 4k after the
>>>> allocation. Charging it to the root memcg will leave it at 4M - 4k.
>>>>
>>>> This means that to be able to allocate a page again, you need to free
>>>> two other pages, be it the 2 pages used by the GFP allocation or any
>>>> other. In other words: the memcg that originated the charge is held
>>>> accountable for it. If he says it can't fail for whatever reason, fine,
>>>> we respect that, but we punish it later for other allocations.
>>>>
>> I personally think 'we punish it later' is bad thing at resource accounting.
>> We have 'hard limit'. It's not soft limit.
>
> That only makes sense if you will fail the allocation. If you won't, you
> are over your hard limit anyway. You are just masquerading that.
>

'showing usage > limit to user' and 'avoid accounting'
is totally different user experience.

>>>> Without that GFP_NOFAIL becomes just a nice way for people to bypass
>>>> those controls altogether, since after a ton of GFP_NOFAIL allocations,
>>>> normal allocations will still succeed.
>>>>
>> Allowing people to bypass is not bad because they're kernel.
>
> No, they are not. They are in process context, on behalf of a process
> that belongs to a valid memcg. If they happen to be a kernel thread,
> !current->mm test will send the allocation to the root memcg already.
>

Yes, but it's kernel code. There will be some special reason to use __GFP_NOFAIL.

>>
>> But, IIUC, from gfp.h
>> ==

>>   * __GFP_NOFAIL: The VM implementation_must_ retry infinitely: the caller
>>   * cannot handle allocation failures. This modifier is deprecated and no new
>>   * users should be added.
>> ==
>>
>> GFP_NOFAIL will go away and no new user is recommended.
>>
> Yes, I am aware of that. That's actually why I don't plan to insist on
> this too much - although your e-mail didn't really convince me.
>
> It should not matter in practice.
>
>> So, please skip GFP_NOFAIL accounting and avoid to write
>> "usage may go over limit if you're unfortune, sorry" into memcg documentation.
>
> I won't write that, because that's not true. Is more like: "Allocations
> that can fail will fail if you go over limit".
>

>>
>>>> The change you propose is totally doable. I just don't believe it should
>>>> be done.
>>>>
>>>> But let me know where you stand.
>>>>
>> My stand point is keeping "usage<= limit" is the spec. and
>> important in enterprise system. So, please avoid usage> limit.
>>
> As I said, I won't make a case here because those allocations shouldn't
> matter in real life anyway. I can change it.
>

My standing point is that 'usage > limit' is bug. So please avoid it if
__GFP_NOFAIL allocation is not very important.

Thanks,
-Kame