
Subject: Re: [PATCH v4 06/14] memcg: kmem controller infrastructure
Posted by [Glauber Costa](#) on Fri, 12 Oct 2012 09:13:04 GMT
[View Forum Message](#) <> [Reply to Message](#)

On 10/12/2012 12:57 PM, Michal Hocko wrote:

> On Fri 12-10-12 12:44:57, Glauber Costa wrote:

>> On 10/12/2012 12:39 PM, Michal Hocko wrote:

>>> On Fri 12-10-12 11:45:46, Glauber Costa wrote:

>>>> On 10/11/2012 04:42 PM, Michal Hocko wrote:

>>>>> On Mon 08-10-12 14:06:12, Glauber Costa wrote:

>>> [...]

>>>>> + /*

>>>>> + * Conditions under which we can wait for the oom_killer.

>>>>> + * __GFP_NORETRY should be masked by __mem_cgroup_try_charge,

>>>>> + * but there is no harm in being explicit here

>>>>> + */

>>>>> + may_oom = (gfp & __GFP_WAIT) && !(gfp & __GFP_NORETRY);

>>>>>

>>>>> Well we _have to_ check __GFP_NORETRY here because if we don't then we

>>>>> can end up in OOM. mem_cgroup_do_charge returns CHARGE_NOMEM for

>>>>> __GFP_NORETRY (without doing any reclaim) and of oom==true we decrement

>>>>> oom retries counter and eventually hit OOM killer. So the comment is

>>>>> misleading.

>>>>>

>>>> I will update. What i understood from your last message is that we don't

>>>> really need to, because try_charge will do it.

>>>>

>>> IIRC I just said it couldn't happen before because migration doesn't go

>>> through charge and thp disable oom by default.

>>>>

>>>>

>> I had it changed to:

>>

>> /*

>> * Conditions under which we can wait for the oom_killer.

>> * We have to be able to wait, but also, if we can't retry,

>> * we obviously shouldn't go mess with oom.

>> */

>> may_oom = (gfp & __GFP_WAIT) && !(gfp & __GFP_NORETRY);

>>

> OK

>

>>

>>>>>> +

>>>>>> + _memcg = memcg;

>>>>>> + ret = __mem_cgroup_try_charge(NULL, gfp, size >> PAGE_SHIFT,

>>>>>> + &_memcg, may_oom);

>>>>>> +

```

>>>>> + if (!ret) {
>>>>> + ret = res_counter_charge(&memcg->kmem, size, &fail_res);
>>>>>
>>>>> Now that I'm thinking about the charging ordering we should charge the
>>>>> kmem first because we would like to hit kmem limit before we hit u+k
>>>>> limit, don't we.
>>>>> Say that you have kmem limit 10M and the total limit 50M. Current `u'
>>>>> would be 40M and this charge would cause kmem to hit the `k' limit. I
>>>>> think we should fail to charge kmem before we go to u+k and potentially
>>>>> reclaim/oom.
>>>>> Or has this been already discussed and I just do not remember?
>>>>>
>>>> This has never been discussed as far as I remember. We charged u first
>>>> since day0, and you are so far the first one to raise it...
>>>>
>>>> One of the things in favor of charging 'u' first is that
>>>> mem_cgroup_try_charge is already equipped to make a lot of decisions,
>>>> like when to allow reclaim, when to bypass charges, and it would be good
>>>> if we can reuse all that.
>>>>
>>> Hmm, I think that we should prevent from those decisions if kmem charge
>>> would fail anyway (especially now when we do not have targeted slab
>>> reclaim).
>>>
>>
>> Let's revisit this discussion when we do have targeted reclaim. For now,
>> I'll agree that charging kmem first would be acceptable.
>>
>> This will only make a difference when  $K < U$  anyway.
>
> Yes and it should work as advertised (aka hit the k limit first).
>
Just so we don't ping-pong in another submission:

```

I changed memcontrol.h's memcg_kmem_newpage_charge to include:

```

/* If the test is dying, just let it go. */
if (unlikely(test_thread_flag(TIF_MEMDIE)
    || fatal_signal_pending(current)))
    return true;

```

I'm also attaching the proposed code in memcontrol.c

File Attachments

1) [chch.patch](#), downloaded 469 times
