
Subject: Re: [PATCH v4 07/25] memcg: Reclaim when more than one page needed.

Posted by [Glauber Costa](#) on Wed, 20 Jun 2012 19:43:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

On 06/20/2012 05:47 PM, Michal Hocko wrote:

> On Mon 18-06-12 14:28:00, Glauber Costa wrote:

>> From: Suleiman Souhlal <ssouhlal@FreeBSD.org>

>>

>> mem_cgroup_do_charge() was written before slab accounting, and expects
>> three cases: being called for 1 page, being called for a stock of 32 pages,
>> or being called for a hugepage. If we call for 2 or 3 pages (and several
>> slabs used in process creation are such, at least with the debug options I
>> had), it assumed it's being called for stock and just retried without reclaiming.

>>

>> Fix that by passing down a minsize argument in addition to the csize.

>>

>> And what to do about that (csize == PAGE_SIZE && ret) retry? If it's
>> needed at all (and presumably is since it's there, perhaps to handle
>> races), then it should be extended to more than PAGE_SIZE, yet how far?
>> And should there be a retry count limit, of what? For now retry up to
>> COSTLY_ORDER (as page_alloc.c does), stay safe with a cond_resched(),
>> and make sure not to do it if __GFP_NORETRY.

>>

>> [v4: fixed nr pages calculation pointed out by Christoph Lameter]

>>

>> Signed-off-by: Suleiman Souhlal <suleiman@google.com>

>> Signed-off-by: Glauber Costa <glommer@parallels.com>

>> Reviewed-by: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

>

> I think this is not ready to be merged yet.

Fair Enough

> Two comments below.

>

> [...]

>> @@ -2210,18 +2211,18 @@ static int mem_cgroup_do_charge(struct mem_cgroup *memcg,
gfp_t gfp_mask,

>> } else

>> mem_over_limit = mem_cgroup_from_res_counter(fail_res, res);

>> /*

>> - * nr_pages can be either a huge page (HPAGE_PMD_NR), a batch

>> - * of regular pages (CHARGE_BATCH), or a single regular page (1).

>> - *

>> * Never reclaim on behalf of optional batching, retry with a

>> * single page instead.

>> */

>> - if (nr_pages == CHARGE_BATCH)

```
>> + if (nr_pages > min_pages)
>>     return CHARGE_RETRY;
>>
>>     if (!(gfp_mask & __GFP_WAIT))
>>         return CHARGE_WOULDBLOCK;
>>
>> + if (gfp_mask & __GFP_NORETRY)
>> +     return CHARGE_NOMEM;
>
> This is kmem specific and should be prepared out in case this should
> be merged before the rest.
ok.
```

> Btw. I assume that oom==false when called from kmem...

What prevents the oom killer to be called for a reclaimable kmem allocation that can be satisfied ?

```
>> +
>>     ret = mem_cgroup_reclaim(mem_over_limit, gfp_mask, flags);
>>     if (mem_cgroup_margin(mem_over_limit) >= nr_pages)
>>         return CHARGE_RETRY;
>> @@ -2234,8 +2235,10 @@ static int mem_cgroup_do_charge(struct mem_cgroup *memcg,
>>     gfp_t gfp_mask,
>>     * unlikely to succeed so close to the limit, and we fall back
>>     * to regular pages anyway in case of failure.
>>     */
>> - if (nr_pages == 1 && ret)
>> + if (nr_pages <= (1 << PAGE_ALLOC_COSTLY_ORDER) && ret) {
>> +     cond_resched();
>>     return CHARGE_RETRY;
>> + }
>
> What prevents us from looping for unbounded amount of time here?
> Maybe you need to consider the number of reclaimed pages here.
```

Why would we even loop here? It will just return CHARGE_RETRY, it is up to the caller to decide whether or not it will retry.