Subject: Re: [PATCH v4 07/25] memcg: Reclaim when more than one page needed.
Posted by Glauber Costa on Mon, 25 Jun 2012 13:13:53 GMT
View Forum Message <> Reply to Message

>>>> +
>>>> 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.
>
> Yes, but the test was original to prevent oom when we managed to reclaim
> something. And something might be enough for a single page but now you
> have high order allocations so we can retry without any success.
>

So,

Most of the kmem allocations are likely to be quite small as well. For
the slab, we're dealing with the order of 2-3 pages, and for other
allocations that may happen, like stack, they will be in the order of 2
pages as well.

So one thing I could do here, is define a threshold, say, 3, and only
retry for that very low threshold, instead of following COSTLY_ORDER.
I don't expect two or three pages to be much less likely to be freed
than a single page.

I am fine with ripping of the cond_resched as well.

Let me know if you would be okay with that.