Subject: Re: [PATCH v2 07/13] memcg: Slab accounting.
Posted by Glauber Costa on Thu, 15 Mar 2012 11:40:07 GMT
View Forum Message <> Reply to Message

On 03/15/2012 02:04 AM, Suleiman Souhlal wrote:
> On Wed, Mar 14, 2012 at 3:47 AM, Glauber Costa<glommer@parallels.com> wrote:
>> On 03/14/2012 02:50 AM, Suleiman Souhlal wrote:
>>>
>>> On Sun, Mar 11, 2012 at 3:25 AM, Glauber Costa<glommer@parallels.com>
>>> wrote:
>>>>
>>>> On 03/10/2012 12:39 AM, Suleiman Souhlal wrote:
>>>>>
>>>>> +static inline void
>>>>> +mem_cgroup_kmem_cache_prepare_sleep(struct kmem_cache *cachep)
>>>>> +{
>>>>> +   /*
>>>>> +   * Make sure the cache doesn't get freed while we have
>>>>> interrupts
>>>>> +   * enabled.
>>>>> +   */
>>>>> +   kmem_cache_get_ref(cachep);
>>>>> +   rcu_read_unlock();
>>>>> +}
>>>>
>>>>
>>>>
>>>> Is this really needed ? After this function call in slab.c, the slab code
>>>> itself accesses cachep a thousand times. If it could be freed, it would
>>>> already explode today for other reasons?
>>>> Am I missing something here?
>>>
>>>
>>> We need this because once we drop the rcu_read_lock and go to sleep,
>>> the memcg could get deleted, which could lead to the cachep from
>>> getting deleted as well.
>>>
>>> So, we need to grab a reference to the cache, to make sure that the
>>> cache doesn't disappear from under us.
>>
>>
>> Don't we grab a memcg reference when we fire the cache creation?
>> (I did that for slub, can't really recall from the top of my head if
>> you are doing it as well)
>>
>> That would prevent the memcg to go away, while relieving us from the
>> need to take a temporary reference for every page while sleeping.
>

> The problem isn't the memcg going away, but the cache going away.
>
I see the problem.

I still think there are ways to avoid getting a reference at every page,
but it might not be worth the complication...

> Keep in mind that this function is only called in workqueue context.
> (In the earlier revision of the patchset this function was called in
> the process context, but kmem_cache_create() would ignore memory
> limits, because of __GFP_NOACCOUNT.)

ok, fair.

>
> When mem_cgroup_get_kmem_cache() returns a memcg cache, that cache has
> already been created.
 >
> The memcg pointer is not stable between alloc and free: It can become
> NULL when the cgroup gets deleted, at which point the accounting has
> been "moved to root" (uncharged from the cgroup it was charged in).
> When that has happened, we don't want to uncharge it again.
> I think the current code already handles this situation.
>

Okay, convinced.
