Subject: Re: [PATCH v4 00/25] kmem limitation for memcg
Posted by Glauber Costa on Mon, 18 Jun 2012 12:14:35 GMT
View Forum Message <> Reply to Message

On 06/18/2012 04:10 PM, Kamezawa Hiroyuki wrote:
> (2012/06/18 19:27), Glauber Costa wrote:
>> Hello All,
>>
>> This is my new take for the memcg kmem accounting. This should merge
>> all of the previous comments from you guys, specially concerning the big churn
>> inside the allocators themselves.
>>
>> My focus in this new round was to keep the changes in the cache internals to
>> a minimum. To do that, I relied upon two main pillars:
>>
>> * Cristoph's unification series, that allowed me to put must of the changes
>> in a common file. Even then, the changes are not too many, since the overal
>> level of invasiveness was decreased.
>> * Accounting is done directly from the page allocator. This means some pages
>> can fail to be accounted, but that can only happen when the task calling
>> kmem_cache_alloc or kmalloc is not the same task allocating a new page.
>> This never happens in steady state operation if the tasks are kept in the
>> same memcg. Naturally, if the page ends up being accounted to a memcg that
>> is not limited (such as root memcg), that particular page will simply not
>> be accounted.
>>
>> The dispatcher code stays (mem_cgroup_get_kmem_cache), being the mechanism who
>> guarantees that, during steady state operation, all objects allocated in a page
>> will belong to the same memcg. I consider this a good compromise point between
>> strict and loose accounting here.
>>
>
> 2 questions.
>
> - Do you have performance numbers ?

Not extensive. I've run some microbenchmarks trying to determine the
effect of my code on kmem_cache_alloc, and found it to be in the order
of 2 to 3 %. I would expect that to vanish in a workload benchmark.

>
> - Do you think user-memory memcg should be switched to page-allocator level accounting ?
> (it will require some study for modifying current bached-freeing and per-cpu-stock
> logics...)

I don't see a reason for that. My main goal by doing that was to reduce
the churn in the cache internal structures, but specially because there
is at least two of them, obeying a stable interface. The way I

understand it, memcg for user pages is already pretty well integrated to the page allocator, so the benefit of it is questionable.