Subject: Re: [PATCH v3 15/16] memcg/sl[au]b: shrink dead caches
Posted by JoonSoo Kim on Fri, 21 Sep 2012 09:28:02 GMT

View Forum Message <> Reply to Message

Hi, Glauber.

>> 2012/9/18 Glauber Costa <glommer@parallels.com>:
>>> diff --git a/mm/slub.c b/mm/slub.c
>>> index 0b68d15..9d79216 100644
>>> --- a/mm/slub.c
>>> +++ b/mm/slub.c
>>> @@ -2602,6 +2602,7 @@ redo:
>>>   } else
>>>   __slab_free(s, page, x, addr);
>>>
>>> +   kmem_cache_verify_dead(s);
>>> }
>>
>> As far as u know, I am not a expert and don't know anything about memcg.
>> IMHO, this implementation may hurt system performance in some case.
>>
>> In case of memcg is destoried, remained kmem_cache is marked "dead".
>> After it is marked,
>> every free operation to this "dead" kmem_cache call
>> kmem_cache_verify_dead() and finally call kmem_cache_shrink().
>
> As long as it is restricted to that cache, this is a non issue.
> dead caches are exactly what they name imply: dead.
>
> Means that we actively want them to go away, and just don't kill them
> right away because they have some inflight objects - which we expect not
> to be too much.

Hmm.. I don't think so.
We can destroy memcg whenever we want, is it right?
If it is right, there is many inflight objects when we destory memcg.
If there is so many inflight objects, performance of these processes
can be hurt too much.

>> And, I found one case that destroying memcg's kmem_cache don't works properly.
>> If we destroy memcg after all object is freed, current implementation
>> doesn't destroy kmem_cache.
>> kmem_cache_destroy_work_func() check "cachep->memcg_params.nr_pages == 0",
>> but in this case, it return false, because kmem_cache may have
>> cpu_slab, and cpu_partials_slabs.
>> As we already free all objects, kmem_cache_verify_dead() is not invoked forever.
>> I think that we need another kmem_cache_shrink() in
>> kmem_cache_destroy_work_func().

>
> I'll take a look here. What you describe makes sense, and can
> potentially happen. I tried to handle this case with care in
> destroy_all_caches, but I may have always made a mistake...
>
> Did you see this actively happening, or are you just assuming this can
> happen from your read of the code?

Just read of the code.

Thanks.