Subject: Re: [PATCH v2 18/29] memcg: kmem controller charge/uncharge infrastructure
Posted by KAMEZAWA Hiroyuki on Wed, 16 May 2012 08:18:28 GMT
View Forum Message <> Reply to Message

(2012/05/16 15:42), Glauber Costa wrote:

> On 05/15/2012 06:57 AM, KAMEZAWA Hiroyuki wrote:
>>> +#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
>>>> +int memcg_charge_kmem(struct mem_cgroup *memcg, gfp_t gfp, s64 delta)
>>>> +{
>>>> + struct res_counter *fail_res;
>>>> + struct mem_cgroup *_memcg;
>>>> + int may_oom, ret;
>>>> + bool nofail = false;
>>>> +
>>>> + may_oom = (gfp& __GFP_WAIT)&& (gfp& __GFP_FS)&&
>>>> + !(gfp& __GFP_NORETRY);
>>>> +
>>>> + ret = 0;
>>>> +
>>>> + if (!memcg)
>>>> + return ret;
>>>> +
>>>> + _memcg = memcg;
>>>> + ret = __mem_cgroup_try_charge(NULL, gfp, delta / PAGE_SIZE,
>>>> + &_memcg, may_oom);
>>>> +
>>>> + if ((ret == -EINTR) || (ret&& (gfp& __GFP_NOFAIL))) {
>>>> + nofail = true;
>>>> + /*
>>>> + * __mem_cgroup_try_charge() chose to bypass to root due
>>>> + * to OOM kill or fatal signal.
>>>> + * Since our only options are to either fail the
>>>> + * allocation or charge it to this cgroup, force the
>>>> + * change, going above the limit if needed.
>>>> + */
>>>> + res_counter_charge_nofail(&memcg->res, delta,&fail_res);
>>>> + if (do_swap_account)
>>>> + res_counter_charge_nofail(&memcg->memsw, delta,
>>>> + &fail_res);
>>>> + } else if (ret == -ENOMEM)
>>>> + return ret;
>>>> +
>>>> + if (nofail)
>>>> + res_counter_charge_nofail(&memcg->kmem, delta,&fail_res);
>>>> + else
>>>> + ret = res_counter_charge(&memcg->kmem, delta,&fail_res);

>>
>> Ouch, you allow usage> limit ? It's BUG.
>>
>> IMHO, if GFP_NOFAIL, memcg accounting should be skipped. Please
>>
>> if (gfp_mask& __GFP_NOFAIL)
>> return 0;
>>
>> Or avoid calling memcg_charge_kmem() you can do that as you do in patch 19/29,
>> I guess you can use a trick like
>>
>> == in 19/29
>> + if (!current->mm || atomic_read(&current->memcg_kmem_skip_account))
>> + return cachep;
>> +
>>   gfp |= cachep->allocflags;
>> ==
>>
>> == change like this
>>   gfp |= cachep->allocflags;
>>
>>   if (!current->mm || current->memcg_kmem_skip_account || gfp& __GFP_NOFAIL))
>> ==
>>
>> Is this difficult ?
>>
>> Thanks,
>> -Kame
>
> Well, we disagree with that.
> I actually voiced this earlier to Suleiman in the thread, but it is good
> that you brought this up again - this is quite important.
>
> I will repeat my rationale here, and if you still are not convinced,
> tell me and I will be happy to switch over.
>
> I believe that the whole reasoning behind this, is to have allocations
> failing if we go over limit. If the allocation won't fail anyway, it
> doesn't really matter who we charge this to.
>
> However, if the allocation still came from a particular memcg, those
> nofail allocation may prevent it to use more memory when a normal
> allocation takes place.
>
> Consider this:
>
> limit = 4M
> usage = 4M - 4k

>
> If at this point the memcg hits a NOFAIL allocation worth 2 pages, by
> the method I am using, the memcg will be at 4M + 4k after the
> allocation. Charging it to the root memcg will leave it at 4M - 4k.
>
> This means that to be able to allocate a page again, you need to free
> two other pages, be it the 2 pages used by the GFP allocation or any
> other. In other words: the memcg that originated the charge is held
> accountable for it. If he says it can't fail for whatever reason, fine,
> we respect that, but we punish it later for other allocations.
>

I personally think 'we punish it later' is bad thing at resource accounting.
We have 'hard limit'. It's not soft limit.

> Without that GFP_NOFAIL becomes just a nice way for people to bypass
> those controls altogether, since after a ton of GFP_NOFAIL allocations,
> normal allocations will still succeed.
>

Allowing people to bypass is not bad because they're kernel.

But, IIUC, from gfp.h
==
 * __GFP_NOFAIL: The VM implementation _must_ retry infinitely: the caller
 * cannot handle allocation failures. This modifier is deprecated and no new
 * users should be added.
==

GFP_NOFAIL will go away and no new user is recommended.

So, please skip GFP_NOFAIL accounting and avoid to write
"usage may go over limit if you're unfortune, sorry" into memcg documentation.

> The change you propose is totally doable. I just don't believe it should
> be done.
>
> But let me know where you stand.
>

My stand point is keeping "usage <= limit" is the spec. and
important in enterprise system. So, please avoid usage > limit.

Thanks,
-Kame