Subject: Re: [PATCH 04/10] memcg: Introduce __GFP_NOACCOUNT.
Posted by Suleiman Souhlal on Sat, 03 Mar 2012 16:38:06 GMT
View Forum Message <> Reply to Message

On Sat, Mar 3, 2012 at 6:22 AM, Glauber Costa <glommer@parallels.com> wrote:
> On 03/01/2012 03:05 AM, KAMEZAWA Hiroyuki wrote:
>>
>> On Wed, 29 Feb 2012 21:24:11 -0300
>> Glauber Costa<glommer@parallels.com> wrote:
>>
>>> On 02/29/2012 09:10 PM, KAMEZAWA Hiroyuki wrote:
>>>>
>>>> On Wed, 29 Feb 2012 11:09:50 -0800
>>>> Suleiman Souhlal<suleiman@google.com>   wrote:
>>>>
>>>>> On Tue, Feb 28, 2012 at 10:00 PM, KAMEZAWA Hiroyuki
>>>>> <kamezawa.hiroyu@jp.fujitsu.com>   wrote:
>>>>>>
>>>>>> On Mon, 27 Feb 2012 14:58:47 -0800
>>>>>> Suleiman Souhlal<ssouhlal@FreeBSD.org>   wrote:
>>>>>>
>>>>>>> This is used to indicate that we don't want an allocation to be
>>>>>>> accounted
>>>>>>> to the current cgroup.
>>>>>>>
>>>>>>> Signed-off-by: Suleiman Souhlal<suleiman@google.com>
>>>>>>
>>>>>>
>>>>>> I don't like this.
>>>>>>
>>>>>> Please add
>>>>>>
>>>>>> ___GFP_ACCOUNT "account this allocation to memcg"
>>>>>>
>>>>>> Or make this as slab's flag if this work is for slab allocation.
>>>>>
>>>>>
>>>>> We would like to account for all the slab allocations that happen in
>>>>> process context.
>>>>>
>>>>> Manually marking every single allocation or kmem_cache with a GFP flag
>>>>> really doesn't seem like the right thing to do..
>>>>>
>>>>> Can you explain why you don't like this flag?
>>>>>
>>>>
>>>> For example, tcp buffer limiting has another logic for buffer size
>>>> controling.

>>>> _AND_, most of kernel pages are not reclaimable at all.
>>>> I think you should start from reclaimable caches as dcache, icache etc.
>>>>
>>>> If you want to use this wider, you can discuss
>>>>
>>>> + #define GFP_KERNEL   (.....| ___GFP_ACCOUNT)
>>>>
>>>> in future. I'd like to see small start because memory allocation failure
>>>> is always terrible and make the system unstable. Even if you notify
>>>> "Ah, kernel memory allocation failed because of memory.limit? and
>>>> many unreclaimable memory usage. Please tweak the limitation or kill
>>>> tasks!!"
>>>>
>>>> The user can't do anything because he can't create any new task because
>>>> of OOM.
>>>>
>>>> The system will be being unstable until an admin, who is not under any
>>>> limit,
>>>> tweaks something or reboot the system.
>>>>
>>>> Please do small start until you provide Eco-System to avoid a case that
>>>> the admin cannot login and what he can do was only reboot.
>>>>
>>> Having the root cgroup to be always unlimited should already take care
>>> of the most extreme cases, right?
>>>
>> If an admin can login into root cgroup ;)
>> Anyway, if someone have a container under cgroup via hosting service,
>> he can do noting if oom killer cannot recover his container. It can be
>> caused by kernel memory limit. And I'm not sure he can do shutdown because
>> he can't login.
>>
>
> To be fair, I think this may be unavoidable. Even if we are only dealing
> with reclaimable slabs, having reclaimable slabs doesn't mean they are
> always reclaimable. Unlike user memory, that we can swap at will (unless
> mlock'd, but that is a different issue), we can have so many objects locked,
> that reclaim is effectively impossible. And with the right pattern, that may
> not even need to be that many: all one needs to do, is figure out a way to
> pin one object per slab page, and that's it: you'll never get rid of them.
>
> So although obviously being nice making sure we did everything we could to
> recover from oom scenarios, once we start tracking kernel memory, this may
> not be possible. So the whole point for me, is guaranteeing that one
> container cannot destroy the others - which is the reality if one of them
> can go an grab all kmem =p
>
> That said, I gave this an extra thought. GFP flags are in theory targeted at

> a single allocation. So I think this is wrong. We either track or not a
> cache, not an allocation. Once we decided that a cache should be tracked, it
> should be tracked and end of story.
>
> So how about using a SLAB flag instead?

The reason I had to make it a GFP flag in the first place is that
there are some allocations that we really do not want to track that
are in slabs we generally want accounted: We have to do some slab
allocations while we are in the slab accounting code (for the cache
name or when enqueuing a memcg kmem_cache to be created, both of which
are just regular kmallocs, I think).
Another possible example might be the skb data, which are just kmalloc
and are already accounted by your TCP accounting changes, so we might
not want to account them a second time.

-- Suleiman