Subject: Re: [PATCH v2 07/11] mm: Allocate kernel pages to the right memcg
Posted by Glauber Costa on Wed, 15 Aug 2012 13:39:58 GMT
View Forum Message <> Reply to Message

>>
>> As for the type, do you think using struct mem_cgroup would be less
>> confusing?
>>
>
> Yes and returning the mem_cgroup or NULL instead of bool.

Ok. struct mem_cgroup it is.

>
>> The placeholder is there, but it is later patched
>> to the final thing.
>> With that explained, if you want me to change it to something else, I
>> can do it. Should I ?
>>
>
> Not in this patch anyway. I would have preferred a pattern like this but
> that's about it.
>
> #ifdef CONFIG_MEMCG_KMEM
> extern struct static_key memcg_kmem_enabled_key;
> static inline int memcg_kmem_enabled(void)
> {
> 	return static_key_false(&memcg_kmem_enabled_key);
> }
> #else
>
> static inline bool memcg_kmem_enabled(void)
> {
> 	return false;
> }
> #endif
>

humm, I'll have to think about this name.
"memcg_kmem_enabled" means it is enabled in this cgroup. It is actually
used inside memcontrol.c to denote precisely that.

Now the static branch, of course, means it is globally enabled. Or as I
called here, "on".

> Two reasons. One, it does not use the terms "on" and "enabled"
> interchangeably. The other reason is down to taste as I'm copying the

> pattern I used myself for sk_memalloc_socks(). Of course I am biased.
>
> Also, why is the key exported?
>

Same reason. The slab will now have inline functions that will test
against that. The alloc functions themselves, are inside the page
allocator, and the exports can go away.

But the static branch will still be tested inside inlined functions in
the slab.

That said, for the sake of simplicity, I can make it go away here, and
add that to the right place later.

>>> I also find it *very* strange to have a function named as if it is an
>>> allocation-style function when it in fact it's looking up a mem_cgroup
>>> and charging it (and uncharging it in the error path if necessary). If
>>> it was called memcg_kmem_newpage_charge I might have found it a little
>>> better.
>>
>> I don't feel strongly about names in general. I can change it.
>> Will update to memcg_kmem_newpage_charge() and memcg_kmem_page_uncharge().
>>
>
> I would prefer that anyway. Names have meaning and people make assumptions on
> the implementation depending on the name. We should try to be as consistent
> as possible or maintenance becomes harder. I know there are areas where
> we are not consistent at all but we should not compound the problem.

memcg_kmem_page_charge() is even better I believe, and that is what I
changed this to in my tree.

>>> As this thing is called from within the allocator, it's not clear why
>>> __memcg_kmem_new_page is exported. I can't imagine why a module would call
>>> it directly although maybe you cover that somewhere else in the series.
>>
>> Okay, more people commented on this, so let me clarify: They shouldn't
>> be. They were initially exported when this was about the slab only,
>> because they could be called from inlined functions from the allocators.
>> Now that the charge/uncharge was moved to the page allocator - which
>> already allowed me the big benefit of separating this in two pieces,
>> none of this needs to be exported.
>>
>> Sorry for not noticing this myself, but thanks for the eyes =)
>>
>
> You're welcome. I expect to see all the exports disappear so. If there

> are any exports left I think it would be important to document why they
> have to be exported. This is particularly true because they are
> EXPORT_SYMBOL not EXPORT_SYMBOL_GPL. I think it would be good to know in
> advance why a module (particularly an out-of-tree one) would be
> interested.
>

I will remove them all for now.
