
Subject: Re: [PATCH v3 12/28] slab: pass memcg parameter to
kmem_cache_create

Posted by [Glauber Costa](#) on Tue, 29 May 2012 16:36:06 GMT

[View Forum Message](#) <> [Reply to Message](#)

On 05/29/2012 08:33 PM, Christoph Lameter wrote:

> On Tue, 29 May 2012, Glauber Costa wrote:

>

>>> Ok this only duplicates the kmalloc arrays. Why not the others?

>>

>> It does duplicate the others.

>>

>> First it does a while loop on the kmalloc caches, then a list_for_each_entry

>> in the rest. You probably missed it.

>

> There is no need to separately duplicate the kmalloc_caches. Those are

> included on the cache_chain.

>

>>>> @@ -2543,7 +2564,12 @@ kmem_cache_create (const char *name, size_t size,

>>>> size_t align,

>>>> cachep->ctor = ctor;

>>>> cachep->name = name;

>>>>

>>>> + if (g_cpucache_up >= FULL)

>>>> + mem_cgroup_register_cache(memcg, cachep);

>>>

>>> What happens if a cgroup was active during creation of slab xxy but

>>> then a process running in a different cgroup uses that slab to allocate

>>> memory? Is it charged to the first cgroup?

>>

>> I don't see this situation ever happening. kmem_cache_create, when called

>> directly, will always create a global cache. It doesn't matter which cgroups

>> are or aren't active at this time or any other. We create copies per-cgroup,

>> but we create it lazily, when someone will touch it.

>

> How do you detect that someone is touching it?

kmem_alloc_cache will create mem_cgroup_get_kmem_cache.

(protected by static_branches, so won't happen if you don't have at

least non-root memcg using it)

* Then it detects which memcg the calling process belongs to,

* if it is the root memcg, go back to the allocation as quickly as we
 can

* otherwise, in the creation process, you will notice that each cache
 has an index. memcg will store pointers to the copies and find them by
 the index.

From this point on, all the code of the caches is reused (except for accounting the page)
