
Subject: [PATCH 12/23] slab: pass memcg parameter to kmem_cache_create
Posted by [Glauber Costa](#) on Sun, 22 Apr 2012 23:53:29 GMT
[View Forum Message](#) <> [Reply to Message](#)

Allow a memcg parameter to be passed during cache creation.

Default function is created as a wrapper, passing NULL to the memcg version. We only merge caches that belong to the same memcg.

This code was mostly written by Suleiman Souhlal and only adapted to my patchset, plus a couple of simplifications

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
CC: Michal Hocko <mhocko@suse.cz>
CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
CC: Johannes Weiner <hannes@cmpxchg.org>
CC: Suleiman Souhlal <suleiman@google.com>

mm/slab.c | 38 ++++++-----
1 files changed, 29 insertions(+), 9 deletions(-)

diff --git a/mm/slab.c b/mm/slab.c
index a0d51dd..362bb6e 100644

--- a/mm/slab.c

+++ b/mm/slab.c

@@ -2287,14 +2287,15 @@ static int __init_refok setup_cpu_cache(struct kmem_cache
*cachep, gfp_t gfp)

* cacheline. This can be beneficial if you're counting cycles as closely

* as davem.

*/

-struct kmem_cache *

-kmem_cache_create (const char *name, size_t size, size_t align,

- unsigned long flags, void (*ctor)(void *))

+static struct kmem_cache *

+__kmem_cache_create(struct mem_cgroup *memcg, const char *name, size_t size,

+ size_t align, unsigned long flags, void (*ctor)(void *))

{

- size_t left_over, slab_size, ralign;

+ size_t left_over, orig_align, ralign, slab_size;

struct kmem_cache *cachep = NULL, *pc;

gfp_t gfp;

+ orig_align = align;

/*

* Sanity checks... these are all serious usage bugs.

```

*/
@@ -2311,7 +2312,6 @@ kmem_cache_create (const char *name, size_t size, size_t align,
*/
if (slab_is_available()) {
    get_online_cpus();
- mutex_lock(&cache_chain_mutex);
}

list_for_each_entry(pc, &cache_chain, next) {
@@ -2331,9 +2331,9 @@ kmem_cache_create (const char *name, size_t size, size_t align,
    continue;
}

- if (!strcmp(pc->name, name)) {
+ if (!strcmp(pc->name, name) && !memcg) {
    printk(KERN_ERR
-        "kmem_cache_create: duplicate cache %s\n", name);
+        "kmem_cache_create: duplicate cache %s\n", name);
    dump_stack();
    goto oops;
}
@@ -2434,6 +2434,9 @@ kmem_cache_create (const char *name, size_t size, size_t align,
    cachep->nodelists = (struct kmem_list3 **)&cachep->array[nr_cpu_ids];

    set_obj_size(cachep, size);
+ #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ cachep->memcg_params.orig_align = orig_align;
+ #endif
+ #if DEBUG

/*
@@ -2541,7 +2544,12 @@ kmem_cache_create (const char *name, size_t size, size_t align,
    BUG_ON(ZERO_OR_NULL_PTR(cachep->slabp_cache));
}
cachep->ctor = ctor;
- cachep->name = name;
+ cachep->name = (char *)name;
+
+ #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ mem_cgroup_register_cache(memcg, cachep);
+ atomic_set(&cachep->memcg_params.refcnt, 1);
+ #endif

if (setup_cpu_cache(cachep, gfp)) {
    __kmem_cache_destroy(cachep);
@@ -2566,11 +2574,23 @@ oops:
    panic("kmem_cache_create(): failed to create slab `%s'\n",
        name);

```

```

    if (slab_is_available()) {
-   mutex_unlock(&cache_chain_mutex);
    put_online_cpus();
    }
    return cachep;
}
+
+struct kmem_cache *
+kmem_cache_create(const char *name, size_t size, size_t align,
+ unsigned long flags, void (*ctor)(void *))
+{
+ struct kmem_cache *cachep;
+
+ mutex_lock(&cache_chain_mutex);
+ cachep = __kmem_cache_create(NULL, name, size, align, flags, ctor);
+ mutex_unlock(&cache_chain_mutex);
+
+ return cachep;
+}
EXPORT_SYMBOL(kmem_cache_create);

```

```

#ifdef DEBUG

```

```

--

```

1.7.7.6
