Subject: [PATCH v2 01/29] slab: dup name string
Posted by Glauber Costa on Fri, 11 May 2012 17:44:03 GMT
View Forum Message <> Reply to Message

The slub allocator creates a copy of the name string, and
frees it later. I would like them both to behave the same,
whether it is the slab starting to create a copy of it itself,
or the slub ceasing to.

This is because when I create memcg copies of it, I have to
kmalloc strings for the new names, and having the allocators to
behave differently here, would make it a lot uglier.

My first submission removed the duplication for the slub. But
the code started to get a bit complicated when dealing with
deletion of chained caches. Also, Christoph voiced his opinion
that patching the slab to keep copies would be better.

So here it is.

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
---
 mm/slab.c | 3 ++-
 1 files changed, 2 insertions(+), 1 deletions(-)

diff --git a/mm/slab.c b/mm/slab.c
index e901a36..91b9c13 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -2118,6 +2118,7 @@ static void __kmem_cache_destroy(struct kmem_cache *cachep)
    kfree(l3);
   }
  }
+ kfree(cachep->name);
  kmem_cache_free(&cache_cache, cachep);
 }

@@ -2526,7 +2527,7 @@ 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 = kstrdup(name, GFP_KERNEL);

  if (setup_cpu_cache(cachep, gfp)) {
   __kmem_cache_destroy(cachep);

--
1.7.7.6