Subject: [PATCH] slub: don't create a copy of the name string in kmem_cache_create
Posted by Glauber Costa on Fri, 13 Apr 2012 21:06:10 GMT
View Forum Message <> Reply to Message

When creating a cache, slub keeps a copy of the cache name through strdup. The slab however, doesn't do that. This means that everyone registering caches have to keep a copy themselves anyway, since code needs to work on all allocators.

Having slab create a copy of it as well may very well be the right thing to do: but at this point, the callers are already there

My motivation for it comes from the kmem slab cache controller for memcg. Because we create duplicate caches, having a more consistent behavior here really helps.

I am sending the patch, however, more to probe on your opinion about it. If you guys agree, but don't want to merge it - since it is not fixing anything, nor improving any situation etc, I am more than happy to carry it in my series until it gets merged (fingers crossed).

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
---
 mm/slub.c | 14 ++----------
 1 files changed, 2 insertions(+), 12 deletions(-)

diff --git a/mm/slub.c b/mm/slub.c
index ffe13fd..af8cee9 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -3925,7 +3925,6 @@ struct kmem_cache *kmem_cache_create(const char *name, size_t size,
   size_t align, unsigned long flags, void (*ctor)(void *))
 {
  struct kmem_cache *s;
- char *n;

  if (WARN_ON(!name))
   return NULL;
@@ -3949,26 +3948,20 @@ struct kmem_cache *kmem_cache_create(const char *name, size_t size,
   return s;
  }

- n = kstrdup(name, GFP_KERNEL);

```
- if (!n)
-  goto err;
-
  s = kmalloc(kmem_size, GFP_KERNEL);
  if (s) {
-  if (kmem_cache_open(s, n,
+  if (kmem_cache_open(s, name,
      size, align, flags, ctor)) {
    list_add(&s->list, &slab_caches);
    up_write(&slub_lock);
    if (sysfs_slab_add(s)) {
     down_write(&slub_lock);
     list_del(&s->list);
-    kfree(n);
     kfree(s);
     goto err;
    }
    return s;
   }
-  kfree(n);
   kfree(s);
  }
 err:
@@ -5212,7 +5205,6 @@ static void kmem_cache_release(struct kobject *kobj)
 {
  struct kmem_cache *s = to_slab(kobj);

- kfree(s->name);
  kfree(s);
 }

@@ -5318,11 +5310,9 @@ static int sysfs_slab_add(struct kmem_cache *s)
   return err;
  }
  kobject_uevent(&s->kobj, KOBJ_ADD);
- if (!unmergeable) {
+ if (!unmergeable)
   /* Setup first alias */
   sysfs_slab_alias(s, s->name);
-  kfree(name);
- }
  return 0;
 }

--
1.7.7.6
```

Subject: Re: [PATCH] slub: don't create a copy of the name string in kmem_cache_create
Posted by Christoph Lameter on Mon, 16 Apr 2012 14:02:07 GMT
View Forum Message <> Reply to Message

On Fri, 13 Apr 2012, Glauber Costa wrote:

> When creating a cache, slub keeps a copy of the cache name through
> strdup. The slab however, doesn't do that. This means that everyone
> registering caches have to keep a copy themselves anyway, since code
> needs to work on all allocators.
>
> Having slab create a copy of it as well may very well be the right
> thing to do: but at this point, the callers are already there

What would break if we would add that to slab? I think this is more robust because right now slab relies on the caller not freeing the string.

Subject: Re: [PATCH] slub: don't create a copy of the name string in kmem_cache_create
Posted by Glauber Costa on Mon, 16 Apr 2012 14:41:19 GMT
View Forum Message <> Reply to Message

On 04/16/2012 11:02 AM, Christoph Lameter wrote:
> On Fri, 13 Apr 2012, Glauber Costa wrote:
>
>> When creating a cache, slub keeps a copy of the cache name through
>> strdup. The slab however, doesn't do that. This means that everyone
>> registering caches have to keep a copy themselves anyway, since code
>> needs to work on all allocators.
>>
>> Having slab create a copy of it as well may very well be the right
>> thing to do: but at this point, the callers are already there
>
> What would break if we would add that to slab? I think this is more robust
> because right now slab relies on the caller not freeing the string.

Hard to think of anything, since we call kmem_cache_create() outside of interrupt context anyway.

We have one more point in which we can fail - specially now that we are constraining memory usage, but one can argue that if we are short on memory, better not create another cache anyway.

My main reason for taking this out of slub, instead of adding to the slab, is that I don't remember any single bug report about that - and there are certainly people around using slab, and the interface has been

around for so long, that pretty much everyone will assume this anyway.

I am happy, however, to patch it either way.