Subject: Re: [PATCH v3 09/16] sl[au]b: always get the cache from its page in kfree
Posted by Glauber Costa on Wed, 19 Sep 2012 07:42:54 GMT
View Forum Message <> Reply to Message

On 09/18/2012 07:28 PM, Christoph Lameter wrote:
> On Tue, 18 Sep 2012, Glauber Costa wrote:
>
>> index f2d760c..18de3f6 100644
>> --- a/mm/slab.c
>> +++ b/mm/slab.c
>> @@ -3938,9 +3938,12 @@ EXPORT_SYMBOL(__kmalloc);
>> * Free an object which was previously allocated from this
>> * cache.
>> */
>> -void kmem_cache_free(struct kmem_cache *cachep, void *objp)
>> +void kmem_cache_free(struct kmem_cache *s, void *objp)
>> {
>> unsigned long flags;
>> + struct kmem_cache *cachep = virt_to_cache(objp);
>> +
>> + VM_BUG_ON(!slab_equal_or_parent(cachep, s));
>>
>
> This is an extremely hot path of the kernel and you are adding significant
> processing. Check how the benchmarks are influenced by this change.
> virt_to_cache can be a bit expensive.
>

Would it be enough for you to have a separate code path for
!CONFIG_MEMCG_KMEM?

I don't really see another way to do it, aside from deriving the cache
from the object in our case. I am open to suggestions if you do.

>> diff --git a/mm/slub.c b/mm/slub.c
>> index 4778548..a045dfc 100644
>> --- a/mm/slub.c
>> +++ b/mm/slub.c
>> @@ -2604,7 +2604,9 @@ void kmem_cache_free(struct kmem_cache *s, void *x)
>>
>> page = virt_to_head_page(x);
>>
>> - slab_free(s, page, x, _RET_IP_);
>> + VM_BUG_ON(!slab_equal_or_parent(page->slab, s));
>> +
>> + slab_free(page->slab, page, x, _RET_IP_);
>>
>
>

> Less of a problem here but you are eroding one advantage that slab has had
> in the past over slub in terms of freeing objects.
>
likewise.