Subject: Re: [PATCH 1/4] Add notification about some major slab events
Posted by Pavel Emelianov on Tue, 18 Sep 2007 08:03:43 GMT
View Forum Message <> Reply to Message

Christoph Lameter wrote:
> On Mon, 17 Sep 2007, Pavel Emelyanov wrote:
>
>> @@ -1036,7 +1121,10 @@ static struct page *allocate_slab(struct
>>   page = alloc_pages_node(node, flags, s->order);
>>
>>   if (!page)
>> - return NULL;
>> + goto out;
>> +
>> + if (slub_newpage_notify(s, page, flags) < 0)
>> + goto out_free;
>>
>>   mod_zone_page_state(page_zone(page),
>>   (s->flags & SLAB_RECLAIM_ACCOUNT) ?
>> @@ -1044,6 +1132,11 @@ static struct page *allocate_slab(struct
>>   pages);
>>
>>   return page;
>> +
>> +out_free:
>> + __free_pages(page, s->order);
>> +out:
>> + return NULL;
>> }
>
> Ok that looks sane.
>
>> static void setup_object(struct kmem_cache *s, struct page *page,
>> @@ -1136,6 +1229,8 @@ static void rcu_free_slab(struct rcu_hea
>>
>> static void free_slab(struct kmem_cache *s, struct page *page)
>> {
>> + slub_freepage_notify(s, page);
>> +
>>   if (unlikely(s->flags & SLAB_DESTROY_BY_RCU)) {
>>   /*
>>   * RCU free overloads the RCU head over the LRU
>
> Ditto.
>
>> @@ -1555,6 +1650,11 @@ static void __always_inline *slab_alloc(
>>   }
>>   local_irq_restore(flags);

>>
>> + if (object && slub_alloc_notify(s, object, gfpflags) < 0) {
>> + kmem_cache_free(s, object);
>> + return NULL;
>> + }
>> +
>>   if (unlikely((gfpflags & __GFP_ZERO) && object))
>>   memset(object, 0, c->objsize);
>>
>
> Please stay completely out of the fast path. No modifications to
> slab_alloc or slab_free please. It is possible to force all allocations of
> a particular slab of interest to use the slow path in __slab_alloc (maybe
> as a result of the slab page allocation hook returning a certain result
> code). See how the SLAB_DEBUG handling does it. You can adapt that and then do the
> object checks in __slab_alloc.

That's true, but:
1. we perform only a flag check on a fast path
2. currently we cannot force the freeing of an object to go _always_
   through the slow __slab_free(), and thus the following situation is
   possible:
     a. container A allocates an object and this object is
        accounted to it
     b. the object is freed and gets into lockless freelist (but
        stays accounted to A)
     c. container C allocates this object from the freelist
        and thus get unaccounted amount of memory
  this discrepancy can grow up infinitely. Sure, we can mark some caches to
  go through the slow path even on freeing the objects, but isn't it the
  same as checking for SLAB_NOTIFY on fast paths?

Maybe it's worth having the notifiers under config option, so that those
who don't need this won't suffer at all?

Thanks,
Pavel