Subject: Re: [PATCH v3 09/16] sl[au]b: always get the cache from its page in kfree

Posted by Glauber Costa on Fri, 21 Sep 2012 09:30:59 GMT

View Forum Message <> Reply to Message

```
On 09/21/2012 01:33 PM, Pekka Enberg wrote:
> On Wed, Sep 19, 2012 at 10:42 AM, Glauber Costa <glommer@parallels.com> 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.
>
> We should assume that most distributions enable CONFIG_MEMCG_KMEM,
> right? Therfore, any performance impact should be dependent on whether
> or not kmem memcg is *enabled* at runtime or not.
>
> Can we use the "static key" thingy introduced by tracing folks for this?
>
Yes.

I am already using static keys extensively in this patchset, and that is
how I intend to handle this particular case.
```