Subject: Re: [PATCH v3 15/16] memcg/sl[au]b: shrink dead caches
Posted by Glauber Costa on Mon, 24 Sep 2012 08:25:00 GMT
View Forum Message <> Reply to Message

On 09/22/2012 12:40 AM, Tejun Heo wrote:
> Hello, Glauber.
>
> On Tue, Sep 18, 2012 at 06:12:09PM +0400, Glauber Costa wrote:
>> @@ -764,10 +777,21 @@ static struct kmem_cache *memcg_create_kmem_cache(struct mem_cgroup *memcg,
>>  goto out;
>>  }
>>
>> + /*
>> +  * Because the cache is expected to duplicate the string,
>> +  * we must make sure it has opportunity to copy its full
>> +  * name. Only now we can remove the dead part from it
>> +  */
>> + name = (char *)new_cachep->name;
>> + if (name)
>> +  name[strlen(name) - 4] = '\0';
>
> This is kinda nasty.  Do we really need to do this?  How long would a
> dead cache stick around?

Without targeted shrinking, until all objects are manually freed, which
may need to wait global reclaim to kick in.

In general, if we agree with duplicating the caches, the problem that
they may stick around for some time will not be avoidable. If you have
any suggestions about alternative ways for it, I'm all ears.

>
>> diff --git a/mm/slab.c b/mm/slab.c
>> index bd9928f..6cb4abf 100644
>> --- a/mm/slab.c
>> +++ b/mm/slab.c
>> @@ -3785,6 +3785,8 @@ static inline void __cache_free(struct kmem_cache *cachep, void *objp,
>>  }
>>
>>  ac_put_obj(cachep, ac, objp);
>> +
>> + kmem_cache_verify_dead(cachep);
>
> Reaping dead caches doesn't exactly sound like a high priority thing
> and adding a branch to hot path for that might not be the best way to
> do it.  Why not schedule an extremely lazy deferrable delayed_work

> which polls for emptiness, say, every miniute or whatever?
>

Because this branch is marked as unlikely, I would expect it not to be a big problem. It will be not taken most of the time, and becomes a very cheap branch. I considered this to be simpler than a deferred work mechanism.

If even then, you guys believe this is still too high, I can resort to that.

> Thanks.
>