

---

Subject: Re: [PATCH v5 13/18] memcg/sl[au]b Track all the memcg children of a kmem\_cache.

Posted by [JoonSoo Kim](#) on Mon, 29 Oct 2012 15:26:43 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

2012/10/19 Glauber Costa <glommer@parallels.com>:

```
> +void kmem_cache_destroy_memcg_children(struct kmem_cache *s)
> +{
> +    struct kmem_cache *c;
> +    int i;
> +
> +    if (!s->memcg_params)
> +        return;
> +    if (!s->memcg_params->is_root_cache)
> +        return;
> +
> +    /*
> +     * If the cache is being destroyed, we trust that there is no one else
> +     * requesting objects from it. Even if there are, the sanity checks in
> +     * kmem_cache_destroy should caught this ill-case.
> +     *
> +     * Still, we don't want anyone else freeing memcg_caches under our
> +     * noses, which can happen if a new memcg comes to life. As usual,
> +     * we'll take the set_limit_mutex to protect ourselves against this.
> +     */
> +    mutex_lock(&set_limit_mutex);
> +    for (i = 0; i < memcg_limited_groups_array_size; i++) {
> +        c = s->memcg_params->memcg_caches[i];
> +        if (c)
> +            kmem_cache_destroy(c);
> +    }
> +    mutex_unlock(&set_limit_mutex);
> +}
```

It may cause NULL deref.

Look at the following scenario.

1. some memcg slab caches has remained object.
2. start to destroy memcg.
3. `schedule_delayed_work(kmem_cache_destroy_work_func, @delay 60hz)`
4. all remained object is freed.
5. start to destroy root cache.
6. `kmem_cache_destroy` makes '`s->memcg_params->memcg_caches[i]`' NULL!!
7. Start delayed work function.
8. `cachep` in `kmem_cache_destroy_work_func()` may be NULL

Thanks.

---