## Subject: Re: [PATCH v4 09/14] memcg: kmem accounting lifecycle management Posted by Michal Hocko on Thu, 11 Oct 2012 13:11:43 GMT

View Forum Message <> Reply to Message

```
On Mon 08-10-12 14:06:15, Glauber Costa wrote:
```

- > Because kmem charges can outlive the cgroup, we need to make sure that
- > we won't free the memcg structure while charges are still in flight.
- > For reviewing simplicity, the charge functions will issue
- > mem\_cgroup\_get() at every charge, and mem\_cgroup\_put() at every
- > uncharge.

>

- > This can get expensive, however, and we can do better. mem\_cgroup\_get()
- > only really needs to be issued once: when the first limit is set. In the
- > same spirit, we only need to issue mem\_cgroup\_put() when the last charge
- > is gone.

>

- > We'll need an extra bit in kmem\_accounted for that: KMEM\_ACCOUNTED\_DEAD.
- > it will be set when the cgroup dies, if there are charges in the group.
- > If there aren't, we can proceed right away.

>

- > Our uncharge function will have to test that bit every time the charges
- > drop to 0. Because that is not the likely output of
- > res\_counter\_uncharge, this should not impose a big hit on us: it is
- > certainly much better than a reference count decrease at every
- > operation.

>

> [ v3: merged all lifecycle related patches in one ]

>

- > Signed-off-by: Glauber Costa <glommer@parallels.com>
- > CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
- > CC: Christoph Lameter <cl@linux.com>
- > CC: Pekka Enberg <penberg@cs.helsinki.fi>
- > CC: Michal Hocko <mhocko@suse.cz>
- > CC: Johannes Weiner <hannes@cmpxchg.org>
- > CC: Suleiman Souhlal <suleiman@google.com>

OK, I like the optimization. I have just one comment to the memcg\_kmem\_dead naming but other than that

Acked-by: Michal Hocko <mhocko@suse.cz>

[...]

> +static bool memcg\_kmem\_dead(struct mem\_cgroup \*memcg)

The name is tricky because it doesn't tell you that it clears the flag which made me scratch my head when reading comment in kmem\_cgroup\_destroy

> +{

```
> + return test_and_clear_bit(KMEM_ACCOUNTED_DEAD, &memcg->kmem_accounted);
> +}
> #endif
> /* Stuffs for move charges at task migration. */
> @ @ -4876,6 +4904,20 @ @ static int memcg_init_kmem(struct mem_cgroup *memcg, struct
cgroup_subsys *ss)
> static void kmem_cgroup_destroy(struct mem_cgroup *memcg)
> {
> mem_cgroup_sockets_destroy(memcg);
> + memcg_kmem_mark_dead(memcg);
> + if (res_counter_read_u64(&memcg->kmem, RES_USAGE) != 0)
> + return;
> + /*
> + * Charges already down to 0, undo mem_cgroup_get() done in the charge
> + * path here, being careful not to race with memcg_uncharge_kmem: it is
> + * possible that the charges went down to 0 between mark_dead and the
> + * res counter read, so in that case, we don't need the put
> + */
> + if (memcg_kmem_dead(memcg))
> + mem_cgroup_put(memcg);
Michal Hocko
```

SUSE Labs