Subject: Re: [PATCH v2 03/13] memcg: Uncharge all kmem when deleting a cgroup.
Posted by KAMEZAWA Hiroyuki on Tue, 13 Mar 2012 06:27:18 GMT
View Forum Message <> Reply to Message

On Fri, 9 Mar 2012 12:39:06 -0800
Suleiman Souhlal <ssouhlal@FreeBSD.org> wrote:

> Signed-off-by: Suleiman Souhlal <suleiman@google.com>
> ---
> mm/memcontrol.c | 31 +++++++++++++++++++++++++++++-
> 1 files changed, 30 insertions(+), 1 deletions(-)
>
> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> index e6fd558..6fbb438 100644
> --- a/mm/memcontrol.c
> +++ b/mm/memcontrol.c
> @@ -382,6 +382,7 @@ static void mem_cgroup_get(struct mem_cgroup *memcg);
> static void mem_cgroup_put(struct mem_cgroup *memcg);
> static void memcg_kmem_init(struct mem_cgroup *memcg,
> struct mem_cgroup *parent);
> +static void memcg_kmem_move(struct mem_cgroup *memcg);
>
> static inline bool
> mem_cgroup_test_flag(const struct mem_cgroup *memcg, enum memcg_flags flag)
> @@ -3700,6 +3701,7 @@ static int mem_cgroup_force_empty(struct mem_cgroup *memcg,
bool free_all)
> int ret;
> int node, zid, shrink;
> int nr_retries = MEM_CGROUP_RECLAIM_RETRIES;
> + unsigned long usage;
> struct cgroup *cgrp = memcg->css.cgroup;
>
> css_get(&memcg->css);
> @@ -3719,6 +3721,8 @@ move_account:
> /* This is for making all *used* pages to be on LRU. */
> lru_add_drain_all();
> drain_all_stock_sync(memcg);
> + if (!free_all)
> + memcg_kmem_move(memcg);
> ret = 0;
> mem_cgroup_start_move(memcg);
> for_each_node_state(node, N_HIGH_MEMORY) {
> @@ -3740,8 +3744,14 @@ move_account:
> if (ret == -ENOMEM)
> goto try_to_free;
> cond_resched();
> + usage = memcg->res.usage;

> +#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
> + if (free_all && !mem_cgroup_test_flag(memcg,
> +   MEMCG_INDEPENDENT_KMEM_LIMIT))
> +   usage -= memcg->kmem.usage;
> +#endif
>   /* "ret" should also be checked to ensure all lists are empty. */
> - } while (memcg->res.usage > 0 || ret);
> + } while (usage > 0 || ret);
> out:
>   css_put(&memcg->css);
>   return ret;
> @@ -5689,9 +5699,28 @@ memcg_kmem_init(struct mem_cgroup *memcg, struct
mem_cgroup *parent)
>   parent_res = &parent->kmem;
>   res_counter_init(&memcg->kmem, parent_res);
> }
> +
> +static void
> +memcg_kmem_move(struct mem_cgroup *memcg)

the function name says 'move' but the code seems just do 'forget'
or 'leak'...

> +{
> + unsigned long flags;
> + long kmem;
> +
> + spin_lock_irqsave(&memcg->kmem.lock, flags);
> + kmem = memcg->kmem.usage;
> + res_counter_uncharge_locked(&memcg->kmem, kmem);
> + spin_unlock_irqrestore(&memcg->kmem.lock, flags);
> + if (!mem_cgroup_test_flag(memcg, MEMCG_INDEPENDENT_KMEM_LIMIT))
> + res_counter_uncharge(&memcg->res, kmem);
> +}

please update memcg->memsw, too.

Thanks,
-Kame

> #else /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */
> static void
> memcg_kmem_init(struct mem_cgroup *memcg, struct mem_cgroup *parent)
> {
> }

> +
> +static void
> +memcg_kmem_move(struct mem_cgroup *memcg)
> +{
> +}
> #endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */
> --
> 1.7.7.3
>