Subject: Re: [PATCH 02/10] memcg: Uncharge all kmem when deleting a cgroup.
Posted by KAMEZAWA Hiroyuki on Wed, 29 Feb 2012 06:22:27 GMT
View Forum Message <> Reply to Message

On Mon, 27 Feb 2012 14:58:45 -0800
Suleiman Souhlal <ssouhlal@FreeBSD.org> wrote:

> A later patch will also use this to move the accounting to the root
> cgroup.
>
> Signed-off-by: Suleiman Souhlal <suleiman@google.com>
> ---
> mm/memcontrol.c | 30 ++++++++++++++++++++++++++++-
> 1 files changed, 29 insertions(+), 1 deletions(-)
>
> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> index 11e31d6..6f44fcb 100644
> --- a/mm/memcontrol.c
> +++ b/mm/memcontrol.c
> @@ -378,6 +378,7 @@ 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);
>
> /* Writing them here to avoid exposing memcg's inner layout */
> #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
> @@ -3674,6 +3675,7 @@ 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);
> @@ -3693,6 +3695,8 @@ 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) {
> @@ -3714,8 +3718,13 @@ 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 && !memcg->independent_kmem_limit)
> +   usage -= memcg->kmem_bytes.usage;
> +#endif

Why we need this even if memcg_kmem_move() does uncharge ?

Thanks,
-Kame

>   /* "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;
> @@ -5632,9 +5641,28 @@ memcg_kmem_init(struct mem_cgroup *memcg, struct
mem_cgroup *parent)
>   res_counter_init(&memcg->kmem_bytes, parent_res);
>   memcg->independent_kmem_limit = 0;
> }
> +
> +static void
> +memcg_kmem_move(struct mem_cgroup *memcg)
> +{
> + unsigned long flags;
> + long kmem_bytes;
> +
> + spin_lock_irqsave(&memcg->kmem_bytes.lock, flags);
> + kmem_bytes = memcg->kmem_bytes.usage;
> + res_counter_uncharge_locked(&memcg->kmem_bytes, kmem_bytes);
> + spin_unlock_irqrestore(&memcg->kmem_bytes.lock, flags);
> + if (!memcg->independent_kmem_limit)
> + res_counter_uncharge(&memcg->res, kmem_bytes);
> +}
> #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
>
>