Subject: Re: [PATCH 23/29] memory controller memory accounting v7
Posted by Peter Zijlstra on Thu, 13 Sep 2007 10:18:01 GMT
View Forum Message <> Reply to Message

On Thu, 2007-09-13 at 15:19 +0530, Balbir Singh wrote:

> VM_BUG_ON(pc && !locked)

Even better :-)

> >> +/*
> >> + * Charge the memory controller for page usage.
> >> + * Return
> >> + * 0 if the charge was successful
> >> + * < 0 if the cgroup is over its limit
> >> + */
> >> +int mem_cgroup_charge(struct page *page, struct mm_struct *mm)
> >> +{
> >> + struct mem_cgroup *mem;
> >> + struct page_cgroup *pc, *race_pc;
> >> +
> >> + /*
> >> + * Should page_cgroup's go to their own slab?
> >> + * One could optimize the performance of the charging routine
> >> + * by saving a bit in the page_flags and using it as a lock
> >> + * to see if the cgroup page already has a page_cgroup associated
> >> + * with it
> >> + */
> >> + lock_page_cgroup(page);
> >> + pc = page_get_page_cgroup(page);
> >> + /*
> >> + * The page_cgroup exists and the page has already been accounted
> >> + */
> >> + if (pc) {
> >> + atomic_inc(&pc->ref_cnt);
> >> + goto done;
> >> + }
> >> +
> >> + unlock_page_cgroup(page);
> >> +
> >> + pc = kzalloc(sizeof(struct page_cgroup), GFP_KERNEL);
> >> + if (pc == NULL)
> >> + goto err;
> >> +
> >> + rcu_read_lock();
> >> + /*
> >> + * We always charge the cgroup the mm_struct belongs to
> >> + * the mm_struct's mem_cgroup changes on task migration if the

> >> + * thread group leader migrates. It's possible that mm is not
> >> + * set, if so charge the init_mm (happens for pagecache usage).
> >> + */
> >> + if (!mm)
> >> + mm = &init_mm;
> >> +
> >> + mem = rcu_dereference(mm->mem_cgroup);
> >> + /*
> >> + * For every charge from the cgroup, increment reference
> >> + * count
> >> + */
> >> + css_get(&mem->css);
> >> + rcu_read_unlock();
> >> +
> >> + /*
> >> + * If we created the page_cgroup, we should free it on exceeding
> >> + * the cgroup limit.
> >> + */
> >> + if (res_counter_charge(&mem->res, 1)) {
> >> + css_put(&mem->css);
> >> + goto free_pc;
> >> + }
> >> +
> >> + lock_page_cgroup(page);
> >> + /*
> >> + * Check if somebody else beat us to allocating the page_cgroup
> >> + */
> >> + race_pc = page_get_page_cgroup(page);
> >> + if (race_pc) {
> >> + kfree(pc);
> >> + pc = race_pc;
> >> + atomic_inc(&pc->ref_cnt);
> >
> > This inc
> >
> >> + res_counter_uncharge(&mem->res, 1);
> >> + css_put(&mem->css);
> >> + goto done;
> >> + }
> >> +
> >> + atomic_set(&pc->ref_cnt, 1);
> >
> > combined with this set make me wonder...
> >
>
> I am not sure I understand this comment.

Is that inc needed? the pc is already associated with the page and

should thus already have a reference, so this inc would do 1->2, but we then set it to 1 again. seems like a superfluous operation.

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers