Subject: Re: [PATCH 17/23] kmem controller charge/uncharge infrastructure
Posted by David Rientjes on Tue, 24 Apr 2012 22:54:20 GMT
View Forum Message <> Reply to Message

On Tue, 24 Apr 2012, Glauber Costa wrote:

> > Yes, for user memory, I see charging to p->mm->owner as allowing that
> > process to eventually move and be charged to a different memcg and there's
> > no way to do proper accounting if the charge is split amongst different
> > memcgs because of thread membership to a set of memcgs. This is
> > consistent with charges for shared memory being moved when a thread
> > mapping it moves to a new memcg, as well.
>
> But that's the problem.
>
> When we are dealing with kernel memory, we are allocating a whole slab page.
> It is essentially impossible to track, given a page, which task allocated
> which object.
>

Right, so you have to make the distinction that slab charges cannot be migrated by memory.move_charge_at_immigrate (and it's not even specified to do anything beyond user pages in Documentation/cgroups/memory.txt), but it would be consistent to charge the same memcg for a process's slab allocations as the process's user allocations.

My response was why we shouldn't be charging user pages to mem_cgroup_from_task(current) rather than mem_cgroup_from_task(current->mm->owner) which is what is currently implemented.

If that can't be changed so that we can still migrate user memory amongst memcgs for memory.move_charge_at_immigrate, then it seems consistent to have all allocations done by a task to be charged to the same memcg. Hence, I suggested current->mm->owner for slab charging as well.