Subject: Re: [PATCH 17/23] kmem controller charge/uncharge infrastructure
Posted by David Rientjes on Tue, 24 Apr 2012 20:21:43 GMT
View Forum Message <> Reply to Message

On Tue, 24 Apr 2012, Frederic Weisbecker wrote:

```
> > This seems horribly inconsistent with memcg charging of user memory since
> > it charges to p->mm->owner and you're charging to p.  So a thread attached
> > to a memcg can charge user memory to one memcg while charging slab to
> > another memcg?
>
> Charging to the thread rather than the process seem to me the right behaviour:
> you can have two threads of a same process attached to different cgroups.
>
> Perhaps it is the user memory memcg that needs to be fixed?
>
```

No, because memory is represented by mm_struct, not task_struct, so you
must charge to p->mm->owner to allow for moving threads amongst memcgs
later for memory.move_charge_at_immigrate. You shouldn't be able to
charge two different memcgs for memory represented by a single mm.

```
> > > +
> > > + if (!mem_cgroup_kmem_enabled(memcg))
> > > +  goto out;
> > > +
> > > + mem_cgroup_get(memcg);
> > > + ret = memcg_charge_kmem(memcg, gfp, size) == 0;
> > > + if (ret)
> > > +  mem_cgroup_put(memcg);
> > > +out:
> > > + rcu_read_unlock();
> > > + return ret;
> > > +}
> > > +EXPORT_SYMBOL(__mem_cgroup_charge_kmem);
> > > +
> > > +void __mem_cgroup_uncharge_kmem(size_t size)
> > > +{
> > > + struct mem_cgroup *memcg;
> > > +
> > > + rcu_read_lock();
> > > + memcg = mem_cgroup_from_task(current);
> > > +
> > > + if (!mem_cgroup_kmem_enabled(memcg))
> > > +  goto out;
> > > +
> > > + mem_cgroup_put(memcg);
> > > + memcg_uncharge_kmem(memcg, size);
```

> > > +out:
> > > + rcu_read_unlock();
> > > +}
> > > +EXPORT_SYMBOL(__mem_cgroup_uncharge_kmem);