Subject: Re: [PATCH v5 06/14] memcg: kmem controller infrastructure
Posted by David Rientjes on Thu, 18 Oct 2012 21:59:44 GMT
View Forum Message <> Reply to Message

On Thu, 18 Oct 2012, Glauber Costa wrote:

> >> @@ -2630,6 +2634,171 @@ static void __mem_cgroup_commit_charge(struct
mem_cgroup *memcg,
> >> memcg_check_events(memcg, page);
> >> }
> >>
> >> +#ifdef CONFIG_MEMCG_KMEM
> >> +static inline bool memcg_can_account_kmem(struct mem_cgroup *memcg)
> >> +{
> >> + return !mem_cgroup_disabled() && !mem_cgroup_is_root(memcg) &&
> >> + (memcg->kmem_accounted & KMEM_ACCOUNTED_MASK);
> >> +}
> >> +
> >> +static int memcg_charge_kmem(struct mem_cgroup *memcg, gfp_t gfp, u64 size)
> >> +{
> >> + struct res_counter *fail_res;
> >> + struct mem_cgroup *_memcg;
> >> + int ret = 0;
> >> + bool may_oom;
> >> +
> >> + ret = res_counter_charge(&memcg->kmem, size, &fail_res);
> >> + if (ret)
> >> + return ret;
> >> +
> >> + /*
> >> + * Conditions under which we can wait for the oom_killer.
> >> + * We have to be able to wait, but also, if we can't retry,
> >> + * we obviously shouldn't go mess with oom.
> >> + */
> >> + may_oom = (gfp & __GFP_WAIT) && !(gfp & __GFP_NORETRY);
> > >
> > What about gfp & __GFP_FS?
> > >
> >
> Do you intend to prevent or allow OOM under that flag? I personally
> think that anything that accepts to be OOM-killed should have GFP_WAIT
> set, so that ought to be enough.
>

The oom killer in the page allocator cannot trigger without __GFP_FS
because direct reclaim has little chance of being very successful and
thus we end up needlessly killing processes, and that tends to happen
quite a bit if we dont check for it. Seems like this would also happen

with memcg if mem_cgroup_reclaim() has a large probability of failing?