Subject: Re: [PATCH 06/11] memcg: kmem controller infrastructure
Posted by David Rientjes on Wed, 27 Jun 2012 04:01:32 GMT
View Forum Message <> Reply to Message

On Tue, 26 Jun 2012, Glauber Costa wrote:

> > > @@ -416,6 +423,43 @@ static inline void sock_update_memcg(struct sock *sk)
> > > static inline void sock_release_memcg(struct sock *sk)
> > > {
> > > }
> > > +
> > > +#define mem_cgroup_kmem_on 0
> > > +#define __mem_cgroup_new_kmem_page(a, b, c) false
> > > +#define __mem_cgroup_free_kmem_page(a,b)
> > > +#define __mem_cgroup_commit_kmem_page(a, b, c)
> > > +#define is_kmem_tracked_alloc (false)
> > > #endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */
> > > +
> > > +static __always_inline
> > > +bool mem_cgroup_new_kmem_page(gfp_t gfp, void *handle, int order)
> > > +{
> > > + if (!mem_cgroup_kmem_on)
> > > + return true;
> > > + if (!is_kmem_tracked_alloc)
> > > + return true;
> > > + if (!current->mm)
> > > + return true;
> > > + if (in_interrupt())
> > > + return true;
> >
> > You can't test for current->mm in irq context, so you need to check for
> > in_interrupt() first.
> >
> Right, thanks.
>
> > Also, what prevents __mem_cgroup_new_kmem_page()
> > from being called for a kthread that has called use_mm() before
> > unuse_mm()?
>
> Nothing, but I also don't see how to prevent that.

You can test for current->flags & PF_KTHREAD following the check for
in_interrupt() and return true, it's what you were trying to do with the
check for !current->mm.