Subject: Re: [PATCH v5 06/14] memcg: kmem controller infrastructure
Posted by Michal Hocko on Mon, 22 Oct 2012 12:51:24 GMT
View Forum Message <> Reply to Message

[Sorry for the late reply]

On Mon 22-10-12 16:34:15, Glauber Costa wrote:
> On 10/20/2012 12:34 AM, David Rientjes wrote:
> > On Fri, 19 Oct 2012, Glauber Costa wrote:
> >
> >>>>> 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?
> >>>
> >>
> >> I can indeed see tests for GFP_FS in some key locations in mm/ before
> >> calling the OOM Killer.
> >>
> >> Should I test for GFP_IO as well?
> >
> > It's not really necessary, if __GFP_IO isn't set then it wouldn't make
> > sense for __GFP_FS to be set.
> >
> >> If the idea is preventing OOM to
> >> trigger for allocations that can write their pages back, how would you
> >> feel about the following test:
> >>
> >> may_oom = (gfp & GFP_KERNEL) && !(gfp & __GFP_NORETRY) ?
> >>
> >
> > I would simply copy the logic from the page allocator and only trigger oom
> > for __GFP_FS and !__GFP_NORETRY.
> >
>
> That seems reasonable to me. Michal ?

Yes it makes sense to be consistent with the global case. While we are
at it, do we need to consider PF_DUMPCORE resp. !__GFP_NOFAIL?

--
Michal Hocko
SUSE Labs