Subject: Re: [PATCH v2 06/11] memcg: kmem controller infrastructure
Posted by Michal Hocko on Thu, 16 Aug 2012 15:05:18 GMT
View Forum Message <> Reply to Message

On Thu 16-08-12 13:57:07, Glauber Costa wrote:
> On 08/16/2012 01:53 PM, Michal Hocko wrote:
> > On Wed 15-08-12 18:27:45, Glauber Costa wrote:
> >>
> >>>>
> >>>> I see now, you seem to be right.
> >>>
> >>> No I am not because it seems that I am really blind these days...
> >>> We were doing this in mem_cgroup_do_charge for ages:
> >>> if (!(gfp_mask & __GFP_WAIT))
> >>>   return CHARGE_WOULDBLOCK;
> >>>
> >>> /me goes to hide and get with further feedback with a clean head.
> >>>
> >>> Sorry about that.
> >>>
> >> I am as well, since I went to look at mem_cgroup_do_charge() and missed
> >> that.
> >
> > I thought we are not doing atomic allocations in user pages accounting
> > but I was obviously wrong because at least shmem uses atomic
> > allocations for ages.
> >
> >> Do you have any other concerns specific to this patch ?
> >
> > I understood you changed also handle thingy. So the patch should be
> > correct.
> > Do you plan to send an updated version?
> >
> That depends more on you than on me! =)
>
> Do you still have any concerns regarding the u+k charging as it stands
> now? That would be the last big concern I heard during this iteration.

Well, I am still not 100% sure because I still see technical
difficulties that are not addressed by the patchset (memcg-oom, memcg
slab shrinking, possibly others). More importantly this is changing the
current semantic of the limit so we should better be careful about it
and check that we are not making the code tight to specific workloads
without a way out.

On the other hand I do not want to block the progress here without
having _really_ good arguments against that couldn't be handled later
(and it seems that some of my concerns are work in progress already).

I have to admit I like several things about the patchset. Especially the
way how it enables easy-to-setup (aka don't care about kmem details just
make sure you can cap the thing) as well as "I know exactly what I want
to do" usecases.
It is also good nice that only users of the feature are affected by
potential issues.

So I think it is worth a broader attention which could produce other use
cases which could show potential drawbacks from the u+k semantic but I
would be still very careful about merging it to the Linus tree and only
merge it after at least the memcg reclaim path is slab aware. Living in
the -mm tree should help us with the testing converage.

Does it sounds reasonable?
--
Michal Hocko
SUSE Labs