Subject: Re: [PATCH v3 00/28] kmem limitation for memcg
Posted by Christoph Lameter on Tue, 29 May 2012 15:07:39 GMT
View Forum Message <> Reply to Message

On Mon, 28 May 2012, Glauber Costa wrote:

> > It would be best to merge these with my patchset to extract common code
> > from the allocators. The modifications of individual slab allocators would
> > then be not necessary anymore and it would save us a lot of work.
> >
> Some of them would not, some of them would still be. But also please note that
> the patches here that deal with differences between allocators are usually the
> low hanging fruits compared to the rest.
>
> I agree that long term it not only better, but inevitable, if we are going to
> merge both.
>
> But right now, I think we should agree with the implementation itself - so if
> you have any comments on how I am handling these, I'd be happy to hear. Then
> we can probably set up a tree that does both, or get your patches merged and
> I'll rebase, etc.

Just looked over the patchset and its quite intrusive. I have never been fond of cgroups (IMHO hardware needs to be partitioned at physical boundaries) so I have not too much insight into what is going on in that area.

The idea to just duplicate the caches leads to some weird stuff like the refcounting and the recovery of the arguments used during slab creation.

I think it may be simplest to only account for the pages used by a slab in a memcg. That code could be added to the functions in the slab allocators that interface with the page allocators. Those are not that performance critical and would do not much harm.

If you need per object accounting then the cleanest solution would be to duplicate the per node arrays per memcg (or only the statistics) and have the kmem_cache structure only once in memory.

Its best if information is only in one place for design and for performance.