Subject: Re: [PATCH v3 04/13] kmem accounting basic infrastructure
Posted by Tejun Heo on Sun, 30 Sep 2012 08:23:58 GMT
View Forum Message <> Reply to Message

Hello, Glauber.

On Thu, Sep 27, 2012 at 10:30:36PM +0400, Glauber Costa wrote:
> > But that happens only when pages enter and leave slab and if it still
> > is significant, we can try to further optimize charging. Given that
> > this is only for cases where memcg is already in use and we provide a
> > switch to disable it globally, I really don't think this warrants
> > implementing fully hierarchy configuration.
>
> Not totally true. We still have to match every allocation to the right
> cache, and that is actually our heaviest hit, responsible for the 2, 3 %
> we're seeing when this is enabled. It is the kind of path so hot that
> people frown upon branches being added, so I don't think we'll ever get
> this close to being free.

Sure, depening on workload, any addition to alloc/free could be noticeable. I don't know. I'll write more about it when replying to Michal's message. BTW, __memcg_kmem_get_cache() does seem a bit heavy. I wonder whether indexing from cache side would make it cheaper? e.g. something like the following.

```
kmem_cache *__memcg_kmem_get_cache(cachep, gfp)
{
 struct kmem_cache *c;

 c = cachep->memcg_params->caches[percpu_read(kmemcg_slab_idx)];
 if (likely(c))
  return c;
 /* try to create and then fall back to cachep */
}
```

where kmemcg_slab_idx is updated from sched notifier (or maybe add and use current->kmemcg_slab_idx?). You would still need __GFP_* and in_interrupt() tests but current->mm and PF_KTHREAD tests can be rolled into index selection.

Thanks.

--
tejun