Subject: Re: [PATCH v3 12/28] slab: pass memcg parameter to kmem_cache_create
Posted by Frederic Weisbecker on Wed, 30 May 2012 11:01:37 GMT
View Forum Message <> Reply to Message

On Tue, May 29, 2012 at 11:52:55AM -0500, Christoph Lameter wrote:
> On Tue, 29 May 2012, Glauber Costa wrote:
>
> > > How do you detect that someone is touching it?
> >
> > kmem_alloc_cache will create mem_cgroup_get_kmem_cache.
> > (protected by static_branches, so won't happen if you don't have at least
> > non-root memcg using it)
> >
> > > * Then it detects which memcg the calling process belongs to,
> > > * if it is the root memcg, go back to the allocation as quickly as we
> > >   can
> > > * otherwise, in the creation process, you will notice that each cache
> > >   has an index. memcg will store pointers to the copies and find them by
> > >   the index.
> > >
> > > From this point on, all the code of the caches is reused (except for
> > > accounting the page)
>
> Well kmem_cache_alloc cache is the performance critical hotpath.
>
> If you are already there and doing all of that then would it not be better
> to simply count the objects allocated and freed per cgroup? Directly
> increment and decrement counters in a cgroup? You do not really need to
> duplicate the kmem_cache structure and do not need to modify allocators if
> you are willing to take that kind of a performance hit. Put a wrapper
> around kmem_cache_alloc/free and count things.

I believe one of the issues is also that a task can migrate to another cgroup
anytime. But an object that has been charged to a cgroup must be later uncharged
to the same, unless you move the charge as you move the task. But then it means
you need to keep track of the allocations per task, and you also need to be able
to do that reverse mapping (object -> allocating task) because your object can
be allocated by task A but later freed by task B. Then when you do the uncharge
it must happen to the cgroup of A, not the one of B.

That all would be much more complicated and performance sensitive than what this
patchset does. Dealing with duplicate caches for accounting seem to me a good tradeoff
between allocation performance hot path and maintaining cgroups semantics.