Subject: Re: [PATCH v2 06/11] memcg: kmem controller infrastructure
Posted by KAMEZAWA Hiroyuki on Mon, 20 Aug 2012 13:36:16 GMT
View Forum Message <> Reply to Message

```
(2012/08/16 2:00), Glauber Costa wrote:
> On 08/15/2012 08:38 PM, Greg Thelen wrote:
>> On Wed, Aug 15 2012, Glauber Costa wrote:
>>
>>> On 08/14/2012 10:58 PM, Greg Thelen wrote:
>>>> On Mon, Aug 13 2012, Glauber Costa wrote:
>>>>
>>>>>>>> + WARN_ON(mem_cgroup_is_root(memcg));
>>>>>>>> + size = (1 << order) << PAGE_SHIFT;
>>>>>>>> + memcg_uncharge_kmem(memcg, size);
>>>>>>>> + mem_cgroup_put(memcg);
>>>>>> Why do we need ref-counting here ? kmem res_counter cannot work as
>>>>>> reference ?
>>>>> This is of course the pair of the mem_cgroup_get() you commented on
>>>>> earlier. If we need one, we need the other. If we don't need one, we
>>>>> don't need the other =)
>>>>>
>>>>> The guarantee we're trying to give here is that the memcg structure will
>>>>> stay around while there are dangling charges to kmem, that we decided
>>>>> not to move (remember: moving it for the stack is simple, for the slab
>>>>> is very complicated and ill-defined, and I believe it is better to treat
>>>>> all kmem equally here)
>>>>
>>>> By keeping memcg structures hanging around until the last referring kmem
>>>> page is uncharged do such zombie memcg each consume a css_id and thus
>>>> put pressure on the 64k css_id space?  I imagine in pathological cases
>>>> this would prevent creation of new cgroups until these zombies are
>>>> dereferenced.
>>>
>>> Yes, but although this patch makes it more likely, it doesn't introduce
>>> that. If the tasks, for instance, grab a reference to the cgroup dentry
>>> in the filesystem (like their CWD, etc), they will also keep the cgroup
>>> around.
>>
>> Fair point.  But this doesn't seems like a feature.  It's probably not
>> needed initially, but what do you think about creating a
>> memcg_kernel_context structure which is allocated when memcg is
>> allocated?  Kernel pages charged to a memcg would have
>> page_cgroup->mem_cgroup=memcg_kernel_context rather than memcg.  This
>> would allow the mem_cgroup and its css_id to be deleted when the cgroup
>> is unlinked from cgroupfs while allowing for the active kernel pages to
>> continue pointing to a valid memcg_kernel_context.  This would be a
>> reference counted structure much like you are doing with memcg.  When a
>> memcg is deleted the memcg_kernel_context would be linked into its
```

>> surviving parent memcg. This would avoid needing to visit each kernel
>> page.
>
> You need more, you need at the res_counters to stay around as well. And
> probably other fields.
>
> So my fear here is that as you add fields to that structure, you can
> defeat a bit the goal of reducing memory consumption. Still leaves the
> css space, yes. But by doing this we can introduce some subtle bugs by
> having a field in the wrong structure.
>

Hm, can't we free css_id and delete css structure from the css_id idr tree
when a memcg goes zombie ?

Thanks,
-Kame