Subject: Re: [PATCH v5 08/18] memcg: infrastructure to match an allocation to the right cache
Posted by Glauber Costa on Thu, 25 Oct 2012 11:05:22 GMT
View Forum Message <> Reply to Message

```
On 10/24/2012 10:10 PM, JoonSoo Kim wrote:
> 2012/10/19 Glauber Costa <glommer@parallels.com>:
>> @@ -2930,9 +2937,188 @@ int memcg_register_cache(struct mem_cgroup *memcg, struct kmem_cache *s)
>>
>>  void memcg_release_cache(struct kmem_cache *s)
>>  {
>> +	struct kmem_cache *root;
>> +	int id = memcg_css_id(s->memcg_params->memcg);
>> +
>> +	if (s->memcg_params->is_root_cache)
>> +		goto out;
>> +
>> +	root = s->memcg_params->root_cache;
>> +	root->memcg_params->memcg_caches[id] = NULL;
>> +	mem_cgroup_put(s->memcg_params->memcg);
>> +out:
>>  	kfree(s->memcg_params);
>>  }
>
> memcg_css_id should be called after checking "s->memcg_params->is_root_cache".
> Because when is_root_cache == true, memcg_params has no memcg object.
>

Good catch.

>> +/*
>> + * This lock protects updaters, not readers. We want readers to be as fast as
>> + * they can, and they will either see NULL or a valid cache value. Our model
>> + * allow them to see NULL, in which case the root memcg will be selected.
>> + *
>> + * We need this lock because multiple allocations to the same cache from a non
>> + * GFP_WAIT area will span more than one worker. Only one of them can create
>> + * the cache.
>> + */
>> +static DEFINE_MUTEX(memcg_cache_mutex);
>> +static struct kmem_cache *memcg_create_kmem_cache(struct mem_cgroup *memcg,
>> +						  struct kmem_cache *cachep)
>> +{
>> +	struct kmem_cache *new_cachep;
>> +	int idx;
>> +
>> +	BUG_ON(!memcg_can_account_kmem(memcg));
>> +
>> +	idx = memcg_css_id(memcg);
>> +
>> +	mutex_lock(&memcg_cache_mutex);
>> +	new_cachep = cachep->memcg_params->memcg_caches[idx];
>> +	if (new_cachep)
>> +		goto out;
>> +
>> +	new_cachep = kmem_cache_dup(memcg, cachep);
>> +
>> +	if (new_cachep == NULL) {
>> +		new_cachep = cachep;
>> +		goto out;
>> +	}
>> +
>> +	mem_cgroup_get(memcg);
>> +	cachep->memcg_params->memcg_caches[idx] = new_cachep;
>> +	wmb(); /* the readers won't lock, make sure everybody sees it */
>
> Is there any rmb() pair?
> As far as I know, without rmb(), wmb() doesn't guarantee anything.
>
```

There should be. But it seems I missed it. Speaking of which, I should
wmb() after the NULL assignment in release cache as well.

Thanks

```
>> +	new_cachep->memcg_params->memcg = memcg;
>> +	new_cachep->memcg_params->root_cache = cachep;
>
> It may be better these assignment before the statement
> "cachep->memcg_params->memcg_caches[idx] = new_cachep".
> Otherwise, it may produce race situation.
>
> And assigning value to memcg_params->memcg and root_cache is redundant,
> because it is already done in memcg_register_cache().
>
```

Thanks.

As for the redundancy, for memcg you are right. For root cache,
unfortunately not. Up to this patch, this is the only reference to it.
This reference will be moved to a different location in a further patch.
But then, IIRC, I delete it from here.

```
>> +/*
>> + * Return the kmem_cache we're supposed to use for a slab allocation.
>> + * We try to use the current memcg's version of the cache.
>> + *
>> + * If the cache does not exist yet, if we are the first user of it,
>> + * we either create it immediately, if possible, or create it asynchronously
>> + * in a workqueue.
>> + * In the latter case, we will let the current allocation go through with
>> + * the original cache.
>> + *
>> + * Can't be called in interrupt context or from kernel threads.
>> + * This function needs to be called with rcu_read_lock() held.
>> + */
>> +struct kmem_cache *__memcg_kmem_get_cache(struct kmem_cache *cachep,
>> +					  gfp_t gfp)
>> +{
>> +	struct mem_cgroup *memcg;
>> +	int idx;
>> +
>> +	if (cachep->memcg_params && cachep->memcg_params->memcg)
>> +		return cachep;
>
> In __memcg_kmem_get_cache, cachep may be always root cache.
> So checking "cachep->memcg_params->memcg" is somewhat strange.
> Is it right?
>
>
```

Yes, this is somewhat paranoid, and a bit historical. We were
anticipating that we could call the allocation already with the right
cache set, and in this case we would just return it.

I think I'll just VM_BUG_ON this.

Thanks for you review here.