Subject: Re: [PATCH 3/7] per-cgroup slab caches
Posted by Glauber Costa on Wed, 22 Feb 2012 14:08:06 GMT
View Forum Message <> Reply to Message

On 02/22/2012 03:50 AM, Suleiman Souhlal wrote:

```
> On Tue, Feb 21, 2012 at 3:34 AM, Glauber Costa<glommer@parallels.com>  wrote:
>> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
>> index 26fda11..2aa35b0 100644
>> --- a/mm/memcontrol.c
>> +++ b/mm/memcontrol.c
>> +struct kmem_cache *
>> +kmem_cache_dup(struct mem_cgroup *memcg, struct kmem_cache *base)
>> +{
>> +	struct kmem_cache *s;
>> +	unsigned long pages;
>> +	struct res_counter *fail;
>> +	/*
>> +	 * TODO: We should use an ida-like index here, instead
>> +	 * of the kernel address
>> +	 */
>> +	char *kname = kasprintf(GFP_KERNEL, "%s-%p", base->name, memcg);
>
> Would it make more sense to use the memcg name instead of the pointer?
```

Well, yes. But at this point in creation time, we still don't have this
all setup. The css pointer is NULL, so I could not derive the name from
it. Instead of keep fighting what seemed to be a minor issue, I opted to
kick the patches out and be clear with a comment that this is not what I
intend in the way.

Do you know about any good way to grab the cgroup name at create() time ?

```
>> +
>> +	WARN_ON(mem_cgroup_is_root(memcg));
>> +
>> +	if (!kname)
>> +		return NULL;
>> +
>> +	s = kmem_cache_create_cg(memcg, kname, base->size,
>> +				 base->align, base->flags, base->ctor);
>> +	if (WARN_ON(!s))
>> +		goto out;
>> +
>> +
>> +	pages = slab_nr_pages(s);
>> +
>> +	if (res_counter_charge(memcg_kmem(memcg), pages<<  PAGE_SHIFT,&fail)) {
>> +		kmem_cache_destroy(s);
>> +                s = NULL;
>> +       }
>
```
> What are we charging here? Does it ever get uncharged?

We're charging the slab initial pages, that comes from allocations
outside allocate_slab(). But in this sense, it is not very different
than tin foil hats to protect against mind reading. Probably works, but
I am not sure the threat is real (also remembering we can probably want
to port it to the original slab allocator later - and let me be honest -
I know 0 about how that works).

So if the slab starts with 0 pages, this is a nop. If in any case it
does not, it gets uncharged when the cgroup is destroyed.

In all my tests, this played no role. If we can be sure that this won't
be an issue, I'll be happy to remove it.