Subject: Re: [PATCH v3 00/28] kmem limitation for memcg
Posted by Glauber Costa on Tue, 29 May 2012 15:44:54 GMT
View Forum Message <> Reply to Message

On 05/29/2012 07:07 PM, Christoph Lameter wrote:
> On Mon, 28 May 2012, Glauber Costa wrote:
>
>>> It would be best to merge these with my patchset to extract common code
>>> from the allocators. The modifications of individual slab allocators would
>>> then be not necessary anymore and it would save us a lot of work.
>>>
>> Some of them would not, some of them would still be. But also please note that
>> the patches here that deal with differences between allocators are usually the
>> low hanging fruits compared to the rest.
>>
>> I agree that long term it not only better, but inevitable, if we are going to
>> merge both.
>>
>> But right now, I think we should agree with the implementation itself - so if
>> you have any comments on how I am handling these, I'd be happy to hear. Then
>> we can probably set up a tree that does both, or get your patches merged and
>> I'll rebase, etc.
>
> Just looked over the patchset and its quite intrusive.

Thank you very much, Christoph, appreciate it.

> I have never been
> fond of cgroups (IMHO hardware needs to be partitioned at physical
> boundaries) so I have not too much insight into what is going on in that
> area.

There is certainly a big market for that, and certainly a big market for
what we're doing as well. So there are users interested in Containers
technology, and I don't really see it as "partitioning it here" vs
"partitioning there". It's just different.

Moreover, not everyone doing cgroups are doing containers. Some people
are isolating a service, or a paticular job.

I agree it is an intrusive change, but it used to be even more. I did my
best to diminish its large spread.

> The idea to just duplicate the caches leads to some weird stuff like the
> refcounting and the recovery of the arguments used during slab creation.

The refcounting is only needed so we are sure the parent cache won't go
away without the child caches going away. I can try to find a better way

to do that, specifically.

>
> I think it may be simplest to only account for the pages used by a slab in
> a memcg. That code could be added to the functions in the slab allocators
> that interface with the page allocators. Those are not that performance
> critical and would do not much harm.

No, I don't think so. Well, accounting the page is easy, but when we do
a new allocation, we need to match a process to its correspondent page.
This will likely lead to flushing the internal cpu caches of the slub,
for instance, hurting performance. That is because once we allocate a
page, all objects on that page need to belong to the same cgroup.

Also, you talk about intrusiveness, accounting pages is a lot more
intrusive, since then you need to know a lot about the internal
structure of each cache. Having the cache replicated has exactly the
effect of isolating it better.

I of course agree this is no walk in the park, but accounting something
that is internal to the cache, and that each cache will use and organize
in its own private way, doesn't make it any better.

> If you need per object accounting then the cleanest solution would be to
> duplicate the per node arrays per memcg (or only the statistics) and have
> the kmem_cache structure only once in memory.

No, it's all per-page. Nothing here is per-object, maybe you
misunderstood something?