Subject: Re: [PATCH v3 04/13] kmem accounting basic infrastructure
Posted by Glauber Costa on Wed, 26 Sep 2012 23:20:27 GMT
View Forum Message <> Reply to Message

On 09/27/2012 03:08 AM, Tejun Heo wrote:
> Hello, Glauber.
>
> On Thu, Sep 27, 2012 at 02:54:11AM +0400, Glauber Costa wrote:
>> I don't. Much has been said in the past about the problem of sharing. A
>> lot of the kernel objects are shared by nature, this is pretty much
>> unavoidable. The answer we have been giving to this inquiry, is that the
>> workloads (us) interested in kmem accounted tend to be quite local in
>> their file accesses (and other kernel objects as well).
>>
>> It should be obvious that not all workloads are like this, and some of
>> them would actually prefer to have their umem limited only.
>>
>> There is nothing unreasonable in tracking user memory only.
>>
>> If we have a global switch for "tracking all kernel memory", who would
>> you account the objects that are heavily shared to? I solve this by not
>> tracking kernel memory for cgroups in such workloads. What do you propose?
>
> One of the things wrong with that is that it exposes the limitation of
> the current implementation as interface to userland, which is never a
> good idea. In addition, how is userland supposed to know which
> workload is shared kmem heavy or not? Details like that are not even
> inherent to workloads. It's highly dependent on kernel implementation
> which may change any day. If we hit workloads like that the right
> thing to do is improving kmemcg so that such problems don't occur, not
> exposing another switch.
>

Sorry, there is nothing implementation dependent in here. One of the
biggest consumers of all this, are dentries. Dentries are related to
the paths you touch. If you touch files in a self-contained directory,
where you don't expect anyone else to touch, this can safely be
considered local. If you touch files all around, this can safely be
considered not local. Where is the implementation dependent part?

> If we can't make that work in reasonable (doesn't have to be perfect)
> way, we might as well just give up on kmem controller. If userland
> has to second-guess kernel implementation details to make use of it,
> it's useless.
>
As I said above, it shouldn't.

>>> Well, that's really playing with words. Limit is per cgroup and

>>> before the limit is set for the first time, everything is accounted to
>>> something else. How is that keeping track?
>>>
>>
>> Even after the limit is set, it is set only by workloads that want kmem
>> to be tracked. If you want to track it during the whole lifetime of the
>> cgroup, you switch it before you put tasks to it. What is so crazy about it?
>
> The fact that the numbers don't really mean what they apparently
> should mean.
>

This is vague. The usage file in the cgroup means how much kernel memory
was used by that cgroup. If it really bothers you that this may not be
set through the whole group's lifetime, it is also easily solvable.

> So, this seems properly crazy to me at the similar level of
> use_hierarchy fiasco. I'm gonna NACK on this.
>

As I said: all use cases I particularly care about are covered by a
global switch.

I am laying down my views because I really believe they make more sense.
But at some point, of course, I'll shut up if I believe I am a lone voice.

I believe it should still be good to hear from mhocko and kame, but from
your point of view, would all the rest, plus the introduction of a
global switch make it acceptable to you?