Subject: Re: [PATCH v3 04/13] kmem accounting basic infrastructure
Posted by Glauber Costa on Thu, 27 Sep 2012 18:45:01 GMT
View Forum Message <> Reply to Message

On 09/27/2012 09:46 PM, Tejun Heo wrote:
> Hello,
>
> On Thu, Sep 27, 2012 at 06:57:59PM +0400, Glauber Costa wrote:
>>> Because we're not even trying to actually solve the problem but just
>>> dumping it to userland. If dentry/inode usage is the only case we're
>>> being worried about, there can be better ways to solve it or at least
>>> we should strive for that.
>>
>> Not only it is not the only case we care about, this is not even touched
>> in this series. (It is only touched in the next one). This one, for
>> instance, cares about the stack. The reason everything is being dumped
>> into "kmem", is precisely to make things simpler. I argue that at some
>> point it makes sense to draw a line, and "kmem" is a much better line
>> than any fine grained control - precisely because it is conceptually
>> easier to grasp.
>
> Can you please give other examples of cases where this type of issue
> exists (plenty of shared kernel data structure which is inherent to
> the workload at hand)? Until now, this has been the only example for
> this type of issues.
>

Yes. the namespace related caches (*), all kinds of sockets and network
structures, other file system structures like file struct, vm areas, and
pretty much everything a full container does.

(*) we run full userspace, so we have namespaces + cgroups combination.

>>> I think the cost isn't too prohibitive considering it's already using
>>> memcg. Charging / uncharging happens only as pages enter and leave
>>> slab caches and the hot path overhead is essentially single
>>> indirection. Glauber's benchmark seemed pretty reasonable to me and I
>>> don't yet think that warrants exposing this subtle tree of
>>> configuration.
>>
>> Only so we can get some numbers: the cost is really minor if this is all
>> disabled. It this is fully enable, it can get to some 2 or 3 %, which
>> may or may not be acceptable to an application. But for me this is not
>> even about cost, and that's why I haven't brought it up so far
>
> It seems like Mel's concern is mostly based on performance overhead
> concerns tho.
>

>>> The part I nacked is enabling kmemcg on a populated cgroup and then
>>> starting accounting from then without any apparent indication that any
>>> past allocation hasn't been considered. You end up with numbers which
>>> nobody can't tell what they really mean and there's no mechanism to
>>> guarantee any kind of ordering between populating the cgroup and
>>> configuring it and there's *no* way to find out what happened
>>> afterwards neither. This is properly crazy and definitely deserves a
>>> nack.
>>>
>>
>> Mel suggestion of not allowing this to happen once the cgroup has tasks
>> takes care of this, and is something I thought of myself.
>
> You mean Michal's? It should also disallow switching if there are
> children cgroups, right?
>

No, I meant Mel, quoting this:

"Further I would expect that an administrator would be aware of these
limitations and set kmem_accounting at cgroup creation time before any
processes start. Maybe that should be enforced but it's not a
fundamental problem."

But I guess it is pretty much the same thing Michal proposes, in essence.

Or IOW, if your concern is with the fact that charges may have happened
in the past before this is enabled, we can make sure this cannot happen
by disallowing the limit to be set if currently unset (value changes are
obviously fine) if you have children or any tasks already in the group.