Subject: Re: [PATCH v3 04/13] kmem accounting basic infrastructure
Posted by Glauber Costa on Thu, 27 Sep 2012 14:57:59 GMT
View Forum Message <> Reply to Message

On 09/27/2012 06:49 PM, Tejun Heo wrote:
> Hello, Mel.
>
> On Thu, Sep 27, 2012 at 03:28:22PM +0100, Mel Gorman wrote:
>>> In addition, how is userland supposed to know which
>>> workload is shared kmem heavy or not?
>>
>> By using a bit of common sense.
>>
>> An application may not be able to figure this out but the administrator
>> is going to be able to make a very educated guess. If processes running
>> within two containers are not sharing a filesystem hierarchy for example
>> then it'll be clear they are not sharing dentries.
>>
>> If there was a suspicion they were then it could be analysed with
>> something like SystemTap probing when files are opened and see if files
>> are being opened that are shared between containers.
>>
>> It's not super-easy but it's not impossible either and I fail to see why
>> it's such a big deal for you.
>
> 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.

> Also, the problem is not that it is impossible if you know and
> carefully plan for things beforehand (that would be one extremely
> competent admin) but that the problem is undiscoverable. With kmemcg
> accounting disabled, there's no way to tell a looking cgroup the admin
> thinks running something which doesn'ft tax kmem much could be
> generating a ton without the admin ever noticing.
>
>>> The fact that the numbers don't really mean what they apparently

>>> should mean.
>>
>> I think it is a reasonable limitation that only some kernel allocations are
>> accounted for although I'll freely admit I'm not a cgroup or memcg user
>> either.
>>
>> My understanding is that this comes down to cost -- accounting for the
>> kernel memory usage is expensive so it is limited only to the allocations
>> that are easy to abuse by an unprivileged process. Hence this is
>> initially concerned with stack pages with dentries and TCP usage to
>> follow in later patches.
>
> 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

>>> Sure, conferences are useful for building consensus but that's the
>>> extent of it. Sorry that I didn't realize the implications then but
>>> conferences don't really add any finality to decisions.
>>>
>>> So, this seems properly crazy to me at the similar level of
>>> use_hierarchy fiasco. I'm gonna NACK on this.
>>
>> I think you're over-reacting to say the very least :|
>
> 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.

This would remove this particular piece of objection, and maintain the

per-subtree control.