Subject: Re: [PATCH v3 04/13] kmem accounting basic infrastructure
Posted by Mel Gorman on Thu, 27 Sep 2012 14:28:22 GMT
View Forum Message <> Reply to Message

On Wed, Sep 26, 2012 at 04:08:07PM -0700, 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.

I think the limitations have been fairly clearly explained and any admin
using the interface is going to have *some* familiarity with the limitations.

> 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.

>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.
>
> 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.
>
> > > 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.
>

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.

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.

Due to the cost of accounting, I can see why it would be desirable to enable kmem_accounting for some cgroup trees and not others. It is not unreasonable to expect that an administrator might want to account within one cgroup where processes are accessing millions of files without impairing the performance of another cgroup that is mostly using anonymous memory.

> > > The proposed behavior seems really crazy to me. Do people really
> > > think this is a good idea?
> >

> > It is really sad that you lost the opportunity to say that in a room
> > full of mm developers that could add to this discussion in real time,
> > when after an explanation about this was given, Mel asked if anyone
> > would have any objections to this.
>
> 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 :|

--
Mel Gorman
SUSE Labs
