Subject: Re: [PATCH v3 04/13] kmem accounting basic infrastructure
Posted by James Bottomley on Sun, 30 Sep 2012 11:25:52 GMT
View Forum Message <> Reply to Message

On Sun, 2012-09-30 at 19:37 +0900, Tejun Heo wrote:
> Hello, James.
>
> On Sun, Sep 30, 2012 at 09:56:28AM +0100, James Bottomley wrote:
> > The beancounter approach originally used by OpenVZ does exactly this.
> > There are two specific problems, though, firstly you can't count
> > references in generic code, so now you have to extend the cgroup
> > tentacles into every object, an invasiveness which people didn't really
> > like.
>
> Yeah, it will need some hooks. For dentry and inode, I think it would
> be pretty well isolated tho. Wasn't it?

But you've got to ask yourself who cares about accurate accounting per
container of dentry and inode objects? They're not objects that any
administrator is used to limiting. What we at parallels care about
isn't accurately accounting them, it's that one container can't DoS
another by exhausting system resources. That's achieved equally well by
first charge slab accounting, so we don't really have an interest in
pushing object accounting code for which there's no use case.

> > Secondly split accounting causes oddities too, like your total
> > kernel memory usage can appear to go down even though you do nothing
> > just because someone else added a share. Worse, if someone drops the
> > reference, your usage can go up, even though you did nothing, and push
> > you over your limit, at which point action gets taken against the
> > container. This leads to nasty system unpredictability (The whole point
> > of cgroup isolation is supposed to be preventing resource usage in one
> > cgroup from affecting that in another).
>
> In a sense, the fluctuating amount is the actual resource burden the
> cgroup is putting on the system, so maybe it just needs to be handled
> better or maybe we should charge fixed amount per refcnt? I don't
> know.

Yes, we considered single charge per reference accounting as well
(although I don't believe anyone went as far as producing an
implementation). The problem here is now that the sum of the container
resources no-longer bears any relation to the host consumption. This
makes it very difficult for virtualisation orchestration systems to make
accurate decisions when doing dynamic resource scheduling (DRS).

Conversely, as ugly as you think it, first use accounting is actually
pretty good at identifying problem containers (at least with regard to

memory usage) for DRS because containers which are stretching the memory
tend to accumulate the greatest number of first charges over the system
lifetime.

> > We discussed this pretty heavily at the Containers Mini Summit in Santa
> > Rosa. The emergent consensus was that no-one really likes first use
> > accounting, but it does solve all the problems and it has the fewest
> > unexpected side effects.
>
> But that's like fitting the problem to the mechanism. Maybe that is
> the best which can be done, but the side effect there is way-off
> accounting under pretty common workload, which sounds pretty nasty to
> me.

All we need kernel memory accounting and limiting for is DoS prevention.
There aren't really any system administrators who care about Kernel
Memory accounting (at least until the system goes oom) because there are
no absolute knobs for it (all there is are a set of weird and wonderful
heuristics, like dirty limit ratio and drop caches). Kernel memory
usage has a whole set of regulatory infrastructure for trying to make it
transparent to the user.

Don't get me wrong: if there were some easy way to get proper memory
accounting for free, we'd be happy but, because it has no practical
application for any of our customers, there's a limited price we're
willing to pay to get it.

James