Subject: Re: [PATCH v3 04/13] kmem accounting basic infrastructure
Posted by James Bottomley on Sun, 30 Sep 2012 08:56:28 GMT
View Forum Message <> Reply to Message

On Sun, 2012-09-30 at 17:02 +0900, Tejun Heo wrote:
> On Sun, Sep 30, 2012 at 04:57:00PM +0900, Tejun Heo wrote:
> > On Thu, Sep 27, 2012 at 10:45:01PM +0400, Glauber Costa wrote:
> > > > 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.
> >
> > This is probably me being dumb but wouldn't resources used by full
> > namespaces be mostly independent? Which parts get shared? Also, if
> > you do full namespace, isn't it more likely that you would want fuller
> > resource isolation too?
>
> Just a thought about dentry/inode. Would it make sense to count total
> number of references per cgroup and charge the total amount according
> to that? Reference counts are how the shared ownership is represented
> after all. Counting total per cgroup isn't accurate and pathological
> cases could be weird tho.

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. 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).

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.

If you have an alternative that wasn't considered then, I'm sure everyone would be interested, but it isn't split accounting.

James