Subject: Re: [PATCH v4 04/14] kmem accounting basic infrastructure
Posted by Michal Hocko on Fri, 12 Oct 2012 08:27:28 GMT
View Forum Message <> Reply to Message

On Fri 12-10-12 11:36:38, Glauber Costa wrote:
> On 10/11/2012 02:11 PM, Michal Hocko wrote:
> > On Mon 08-10-12 14:06:10, Glauber Costa wrote:
[...]
> >> + if (!memcg->kmem_accounted && val != RESOURCE_MAX) {
> >
> > Just a nit but wouldn't memcg_kmem_is_accounted(memcg) be better than
> > directly checking kmem_accounted?
> > Besides that I am not sure I fully understand RESOURCE_MAX test. Say I
> > want to have kmem accounting for monitoring so I do
> > echo -1 > memory.kmem.limit_in_bytes
> >
> > so you set the value but do not activate it. Isn't this just a reminder
> > from the time when the accounting could be deactivated?
> >
>
> No, not at all.
>
> I see you have talked about that in other e-mails, (I was on sick leave
> yesterday), so let me consolidate it all here:
>
> What we discussed before, regarding to echo -1 > ... was around the
> disable code, something that we no longer allow. So now, if you will
> echo -1 to that file *after* it is limited, you get in track only mode.
>
> But for you to start that, you absolutely have to write something
> different than -1.
>
> Just one example: libcgroup, regardless of how lame we think it is in
> this regard, will write to all cgroup files by default when a file is
> updated. If you haven't written anything, it will still write the same
> value that the file had before.

Ohh, I wasn't aware of that and it sounds pretty lame.

> This means that an already deployed libcg-managed installation will
> suddenly enable kmem for every cgroup. Sure this can be fixed in
> userspace, but:
>
> 1) There is no reason to break it, if we can

You are right

> 2) It is perfectly reasonable to expect that if you write to a file the

> same value that was already there, nothing happens.

Fair enough

--
Michal Hocko
SUSE Labs