
Subject: Re: Re: [PATCH v3 13/28] slub: create duplicate cache
Posted by [James Bottomley](#) on Wed, 30 May 2012 07:28:26 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Wed, 2012-05-30 at 10:29 +0900, Tejun Heo wrote:

> Hello, Christoph, Glauber.

>

> On Wed, May 30, 2012 at 12:25:36AM +0400, Glauber Costa wrote:

> > >We have never worked container like logic like that in the kernel due to

> > >the complicated logic you would have to put in. The requirement that all

> > >objects in a page come from the same container is not necessary. If you

> > >drop this notion then things become very easy and the patches will become

> > >simple.

> >

> > I promise to look at that in more detail and get back to it. In the

> > meantime, I think it would be enlightening to hear from other

> > parties as well, specially the ones also directly interested in

> > using the technology.

>

> I don't think I'm too interested in using the technology ;) and

> haven't read the code (just glanced through the descriptions and

> discussions), but, in general, I think the approach of duplicating

> memory allocator per-memcg is a sane approach. It isn't the most

> efficient one with the possibility of wasting considerable amount of

> caching area per memcg but it is something which can mostly stay out

> of the way if done right and that's how I want cgroup implementations

> to be.

Exactly: we at parallels initially disliked the cgroup multiplied by slab approach (Our beancounters do count objects) because we feared memory wastage and density is very important to us (which tends to mean efficient use of memory) however, when we ran through the calculations in Prague, you can show that we have ~200 slabs and if each wastes half a page, thats ~4MB memory lost per container. Since most virtual environments are of the order nowadays of 0.5GB, we feel it's an annoying but acceptable price to pay.

James

> The two goals for cgroup controllers that I think are important are

> proper (no, not crazy perfect but good enough) isolation and an

> implementation which doesn't impact !cg path in an intrusive manner -

> if someone who doesn't care about cgroup but knows and wants to work

> on the subsystem should be able to mostly ignore cgroup support. If

> that means overhead for cgroup users, so be it.

>

> Without looking at the actual code, my rainbow-farting unicorn here

> would be having a common slXb interface layer which handles

> interfacing with memory allocator users and cgroup and let slXb
> implement the same backend interface which doesn't care / know about
> cgroup at all (other than using the correct allocation context, that
> is). Glauber, would something like that be possible?
>
> Thanks.
>
