Subject: Re: [PATCH v3 13/28] slub: create duplicate cache
Posted by Glauber Costa on Wed, 30 May 2012 07:54:16 GMT
View Forum Message <> Reply to Message

On 05/30/2012 05:29 AM, Tejun Heo wrote:
> 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.

Well, my code in the slab is totally wrapped in static branches. They
only come active when the first group is *limited* (not even created:
you can have a thousand memcg, if none of them are kmem limited, nothing
will happen).

After that, the cost paid is to find out at which cgroup the process is
at. I believe that if we had a faster way for this (like for instance:
if we had a single hierarchy, the scheduler could put this in a percpu
variable after context switch - or any other method), then the cost of
it could be really low, even when this is enabled.

> 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?

It is a matter of degree. There *is* a lot of stuff in common code, and
I tried to do it as much as I could. Christoph gave me a nice idea about
hinting to the page allocator that this page is a slab page before the
allocation, and then we could account from the page allocator directly -
without touching the cache code at all. This could be done, but some
stuff would still be there, basically because of differences in how the
allocator behaves. I think long term Christoph's effort to produce
common code among them will help a lot, if they stabilize their behavior
in certain areas.

I will rework this series to try work more towards this goal, but at
least for now I'll keep duplicating the caches. I still don't believe
that a loose accounting to the extent Christoph proposed will achieve
what we need this to achieve.