Subject: Re: [PATCH v3 04/13] kmem accounting basic infrastructure
Posted by Tejun Heo on Wed, 03 Oct 2012 22:43:16 GMT
View Forum Message <> Reply to Message

Hey, Michal.

On Mon, Oct 01, 2012 at 11:27:56AM +0200, Michal Hocko wrote:
> > Yeah but, if we can get it down to, say, around 1% under most
> > workloads for memcg users, it is quite questionable to introduce full
> > hierarchical configuration to allow avoiding that, isn't it?
>
> Remember that the kmem memory is still accounted to u+k if it is enabled
> which could be a no-go because some workloads (I have provided an
> example that those which are trusted are generally safe to ignore kernel
> memory overhead) simply don't want to consider additional memory which
> is mostly invisible for them.

Maybe it's because my exposure to cgroup usage is different from yours
but the argument that not accounting kernel memory is something
inherently beneficial is lost on me. For compatibility, overhead
and/or implementation complexity issues, yeah, sure, we can't always
(well more like usually) have all we want but I don't get how not
accounting kernel memory is something inherently necessary or
beneficial. This is all about provisioning physical memory to
different groups of users and memory is memory (that's why u+k makes
sense, right?). Without kmemcg enabled, we not only lack a way to
control kernel memory usage but also a way to even watch per-group
usages.

> > But you can apply the same "do I need/want it at all" question to the
> > configuration parameter too.
>
> Yes but, as I've said, the global configuration parameter is too
> coarse. You can have a mix of trusted and untrusted workloads at the
> same machine (e.g. web server which is inherently untrusted) and trusted
> (local batch jobs which just needs a special LRU aging).

This too stems from the same difference stated above. You think
there's inherent distinction between trusted and untrusted workloads
and they need different features from the kernel while I think why
trust anyone if you can untrust everyone and consider the knob as a
compatibility thing.

> I think that comparing kmem accounting with use_hierarchy is not fair.
> Glauber tried to explain why already so I will not repeat it here.
> I will just mention one thing. use_hierarchy has been introduces becuase
> hierarchies were expensive at the time. kmem accounting is about should
> we do u or u+k accounting. So there is a crucial difference.

It may be less crazy but I think there are enough commonalities to at
least make a comparison. Mel seems to think it's mostly about
performance overhead. You think that not accounting kmem is something
inherently necessary.

> That is right but I think that the current discussion shows that a mixed
> (kmem disabled and kmem enabled hierarchies) workloads are far from
> being theoretical and a global knob is just too coarse. I am afraid we

I'm not sure there's much evidence in this thread. The strongest upto
this point seems to be performance overhead / difficulty of general
enough implementation. As for "trusted" workload, what are the
inherent benefits of trusting if you don't have to?

> will see "we want that per hierarchy" requests shortly and that would
> just add a new confusion where global knob would complicate it
> considerably (do we really want on/off/per_hierarchy global knob?).

Hmmm? The global knob is just the same per_hierarchy knob at the
root. It's hierarchical after all.

Anyways, as long as the "we silently ignore what happened before being
enabled" is gone, I won't fight this anymore. It isn't broken after
all. But, please think about making things simpler in general, cgroup
is riddled with mis-designed complexities and memcg seems to be
leading the charge at times.

Thanks.

--
tejun