Subject: Re: [PATCH v3 04/13] kmem accounting basic infrastructure
Posted by Michal Hocko on Mon, 01 Oct 2012 09:27:56 GMT
View Forum Message <> Reply to Message

Hi,

On Sun 30-09-12 17:47:50, Tejun Heo wrote:
[...]
> > > The hot path overhead is quite minimal - it doesn't do much more than
> > > indirecting one more time. In terms of memory usage, it sure could
> > > lead to a bit more fragmentation but even if it gets to several megs
> > > per cgroup, I don't think that's something excessive. So, there is
> > > overhead but I don't believe it to be prohibitive.
> >
> > Remember that users do not want to pay even "something minimal" when the
> > feature is not needed.
>
> 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.

> > > The distinction between "trusted" and "untrusted" is something
> > > artificially created due to the assumed deficiency of kmemcg
> > > implementation.
> >
> > Not really. It doesn't have to do anything with the overhead (be it
> > memory or runtime). It really boils down to "do I need/want it at all".
> > Why would I want to think about how much kernel memory is in use in the
> > first place? Or do you think that user memory accounting should be
> > deprecated?
>
> 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).

> I can see your point but the decision seems muddy to me, and if muddy,
> I prefer to err on the side of being too conservative.
>

> > > This is userland visible API.
> >
> > I am not sure which API visible part you have in mind but
> > kmem.limit_in_bytes will be there whether we go with global knob or "no
> > limit no accounting" approach.
>
> I mean full hierarchical configuration of it. It becomes something
> which each memcg user cares about instead of something which the base
> system / admin flips on system boot.
>
> > > We can always expand the flexibility. Let's do the simple thing
> > > first. As an added bonus, it would enable using static_keys for
> > > accounting branches too.
> >
> > While I do agree with you in general and being careful is at place in
> > this area as time shown several times, this seems to be too restrictive
> > in this particular case.
> > We won't save almost no code with the global knob so I am not sure
> > what we are actually saving here. Global knob will just give us all or
> > nothing semantic without making the whole thing simpler. You will stick
> > with static branches and checkes whether the group accountable anyway,
> > right?
>
> The thing is about the same argument can be made about .use_hierarchy
> too. It doesn't necessarily make the code much harier. Especially
> because the code is structured with that feature on mind, removing
> .use_hierarchy might not remove whole lot of code; however, the wider
> range of behavior which got exposed through that poses a much larger
> problem when we try to make modifications on related behaviors. We
> get a lot more locked down by seemingly not too much code and our long
> term maintainability / sustainability suffers as a result.

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.

> I'm not trying to say this is as bad as .use_hierarchy but want to
> point out that memcg and cgroup in general have had pretty strong
> tendency to choose overly flexible and complex designs and interfaces
> and it's probably about time we become more careful especially about
> stuff which is visible to userland.

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
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?).
--
Michal Hocko
SUSE Labs