Subject: Re: [RFC][PATCH 4/7] RSS accounting hooks over the code
Posted by Nick Piggin on Thu, 15 Mar 2007 05:01:03 GMT
View Forum Message <> Reply to Message

Kirill Korotaev wrote:

>>The approaches I have seen that don't have a struct page pointer, do
>>intrusive things like try to put hooks everywhere throughout the kernel
>>where a userspace task can cause an allocation (and of course end up
>>missing many, so they aren't secure anyway)... and basically just
>>nasty stuff that will never get merged.
>
>
> User beancounters patch has got through all these...
> The approach where each charged object has a pointer to the owner container,
> who has charged it - is the most easy/clean way to handle
> all the problems with dynamic context change, races, etc.
> and 1 pointer in page struct is just 0.1% overehad.

The pointer in struct page approach is a decent one, which I have
liked since this whole container effort came up. IIRC Linus and Alan
also thought that was a reasonable way to go.

I haven't reviewed the rest of the beancounters patch since looking
at it quite a few months ago... I probably don't have time for a
good review at the moment, but I should eventually.

>>Struct page overhead really isn't bad. Sure, nobody who doesn't use
>>containers will want to turn it on, but unless you're using a big PAE
>>system you're actually unlikely to notice.
>
>
> big PAE doesn't make any difference IMHO
> (until struct pages are not created for non-present physical memory areas)

The issue is just that struct pages use low memory, which is a really
scarce commodity on PAE. One more pointer in the struct page means
64MB less lowmem.

But PAE is crap anyway. We've already made enough concessions in the
kernel to support it. I agree: struct page overhead is not really
significant. The benefits of simplicity seems to outweigh the downside.

>>But again, I'll say the node-container approach of course does avoid
>>this nicely (because we already can get the node from the page). So
>>definitely that approach needs to be discredited before going with this
>>one.
>

>
> But it lacks some other features:
> 1. page can't be shared easily with another container

I think they could be shared. You allocate _new_ pages from your own
node, but you can definitely use existing pages allocated to other
nodes.

> 2. shared page can't be accounted honestly to containers
>   as fraction=PAGE_SIZE/containers-using-it

Yes there would be some accounting differences. I think it is hard
to say exactly what containers are "using" what page anyway, though.
What do you say about unmapped pages? Kernel allocations? etc.

> 3. It doesn't help accounting of kernel memory structures.
>   e.g. in OpenVZ we use exactly the same pointer on the page
>   to track which container owns it, e.g. pages used for page
>   tables are accounted this way.

?
page_to_nid(page) ~= container that owns it.

> 4. I guess container destroy requires destroy of memory zone,
>   which means write out of dirty data. Which doesn't sound
>   good for me as well.

I haven't looked at any implementation, but I think it is fine for
the zone to stay around.

> 5. memory reclamation in case of global memory shortage
>   becomes a tricky/unfair task.

I don't understand why? You can much more easily target a specific
container for reclaim with this approach than with others (because
you have an lru per container).

> 6. You cannot overcommit. AFAIU, the memory should be granted
>   to node exclusive usage and cannot be used by by another containers,
>   even if it is unused. This is not an option for us.

I'm not sure about that. If you have a larger number of nodes, then
you could assign more free nodes to a container on demand. But I
think there would definitely be less flexibility with nodes...

I don't know... and seeing as I don't really know where the google
guys are going with it, I won't misrepresent their work any further ;)

>>Everyone seems to have a plan ;) I don't read the containers list...
>>does everyone still have *different* plans, or is any sort of consensus
>>being reached?
>
>
> hope we'll have it soon :)

Good luck ;)

--
SUSE Labs, Novell Inc.
Send instant messages to your online friends http://au.messenger.yahoo.com

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers