Subject: Re: [RFC][PATCH 4/7] RSS accounting hooks over the code
Posted by Nick Piggin on Wed, 14 Mar 2007 06:57:55 GMT
View Forum Message <> Reply to Message

Balbir Singh wrote:
> Nick Piggin wrote:

>> And strangely, this example does not go outside the parameters of
>> what you asked for AFAIKS. In the worst case of one container getting
>> _all_ the shared pages, they will still remain inside their maximum
>> rss limit.
>>
>
> When that does happen and if a container hits it limit, with a LRU
> per-container, if the container is not actually using those pages,
> they'll get thrown out of that container and get mapped into the
> container that is using those pages most frequently.

Exactly. Statistically, first touch will work OK. It may mean some
reclaim inefficiencies in corner cases, but things will tend to
even out.

>> So they might get penalised a bit on reclaim, but maximum rss limits
>> will work fine, and you can (almost) guarantee X amount of memory for
>> a given container, and it will _work_.
>>
>> But I also take back my comments about this being the only design I
>> have seen that gets everything, because the node-per-container idea
>> is a really good one on the surface. And it could mean even less impact
>> on the core VM than this patch. That is also a first-touch scheme.
>>
>
> With the proposed node-per-container, we will need to make massive core
> VM changes to reorganize zones and nodes. We would want to allow
>
> 1. For sharing of nodes
> 2. Resizing nodes
> 3. May be more

But a lot of that is happening anyway for other reasons (eg. memory
plug/unplug). And I don't consider node/zone setup to be part of the
"core VM" as such... it is _good_ if we can move extra work into setup
rather than have it in the mm.

That said, I don't think this patch is terribly intrusive either.

> With the node-per-container idea, it will hard to control page cache

> limits, independent of RSS limits or mlock limits.
>
> NOTE: page cache == unmapped page cache here.

I don't know that it would be particularly harder than any other
first-touch scheme. If one container ends up being charged with too
much pagecache, eventually they'll reclaim a bit of it and the pages
will get charged to more frequent users.

>>> However the messed up accounting that doesn't handle sharing between
>>> groups of processes properly really bugs me. Especially when we have
>>> the infrastructure to do it right.
>>>
>>> Does that make more sense?
>>
>>
>> I think it is simplistic.
>>
>> Sure you could probably use some of the rmap stuff to account shared
>> mapped _user_ pages once for each container that touches them. And
>> this patchset isn't preventing that.
>>
>> But how do you account kernel allocations? How do you account unmapped
>> pagecache?
>>
>> What's the big deal so many accounting people have with just RSS? I'm
>> not a container person, this is an honest question. Because from my
>> POV if you conveniently ignore everything else... you may as well just
>> not do any accounting at all.
>>
>
> We decided to implement accounting and control in phases
>
> 1. RSS control
> 2. unmapped page cache control
> 3. mlock control
> 4. Kernel accounting and limits
>
> This has several advantages
>
> 1. The limits can be individually set and controlled.
> 2. The code is broken down into simpler chunks for review and merging.

But this patch gives the groundwork to handle 1-4, and it is in a small
chunk, and one would be able to apply different limits to different types
of pages with it. Just using rmap to handle 1 does not really seem like a
viable alternative because it fundamentally isn't going to handle 2 or 4.

I'm not saying that you couldn't _later_ add something that uses rmap or
our current RSS accounting to tweak container-RSS semantics. But isn't it
sensible to lay the groundwork first? Get a clear path to something that
is good (not perfect), but *works*?

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