

---

Subject: Re: Re: [RFC][PATCH 2/7] RSS controller core  
Posted by [Herbert Poetzl](#) on Tue, 13 Mar 2007 14:59:38 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Tue, Mar 13, 2007 at 03:48:34AM -0800, Andrew Morton wrote:

> > On Tue, 13 Mar 2007 13:19:53 +0300 Kirill Korotaev <dev@sw.ru> wrote:

> > Andrew Morton wrote:

> > >>> - shared mappings of 'shared' files (binaries

> > >>> and libraries) to allow for reduced memory

> > >>> footprint when N identical guests are running

> > >>>

> > >>> So, it sounds like this can be phrased as a requirement like:

> > >>>

> > >>> "Guests must be able to share pages."

> > >>>

> > >>> Can you give us an idea why this is so?

> > >>>

> > >>> sure, one reason for this is that guests tend to

> > >>> be similar (or almost identical) which results

> > >>> in quite a lot of 'shared' libraries and executables

> > >>> which would otherwise get cached for each guest and

> > >>> would also be mapped for each guest separately

> > >>>

> > > noooooooo. What you're saying there amounts to text replication.

> > > There is no proposal here to create duplicated copies of pagecache

> > > pages: the VM just doesn't support that (Nick has some protopatches

> > > which do this as a possible NUMA optimisation).

> > >

> > > So these mmapped pages will continue to be shared across all

> > > guests. The problem boils down to "which guest(s) get charged for

> > > each shared page".

> > >

> > > A simple and obvious and easy-to-implement answer is "the guest

> > > which paged it in". I think we should firstly explain why that is

> > > insufficient.

> > I guess by "paged it in" you essentially mean

> > "mapped the page into address space for the \*first\* time"?

>

> Not really - I mean "first allocated the page". ie: major fault(),

> read(), write(), etc.

>

> > i.e. how many times the same page mapped into 2 address spaces

> > in the same container should be accounted for?

> >

> > We believe ONE. It is better due to:

> > - it allows better estimate how much RAM container uses.

> > - if one container mapped a single page 10,000 times,

> > it doesn't mean it is worse than a container which mapped only 200  
> > pages and that it should be killed in case of OOM.  
>  
> I'm not sure that we need to account for pages at all, nor care about  
> rss.  
>  
> If we use a physical zone-based containment scheme: fake-numa,  
> variable-sized zones, etc then it all becomes moot.

sounds good to me, just not sure it provides what we  
need, but I'm sure I'll figure that with your help ...

> You set up a container which has 1.5GB of physial memory then toss  
> processes into it. As that process set increases in size it will  
> toss out stray pages which shouldn't be there, then it will start  
> reclaiming and swapping out its own pages and eventually it'll get an  
> oom-killing.

okay, let me ask a few naive questions about this scheme:

how does this work for a `_file_` which is shared between  
two guests (e.g. an executable like bash, hardlinked  
between guests) when both guests are in a different  
zone-based container?

- + assumed that the file is read in the first time,  
will it be accounted to the first guest doing so?

- + assumed it is accessed in the second guest, will  
it cause any additional cache/mapping besides the  
dentry stuff?

- + will container A be able to 'toss out' pages  
'shared' with container B (assumed sharing is  
possible :)

- + when the container A tosses out the pages for this  
executable, will guest B still be able to use them?

- + when the pages are tossed out, will they require  
the system to read them in again, or will they  
stay available ala swap cache?

> No RSS accounting or page accounting in sight, because we already \*have\*  
> that stuff, at the physical level, in the zone.

I'm fine with that ...

- > Overcommitment can be performed by allowing different containers to
- > share the same zone set, or by dynamically increasing or decreasing
- > the size of a physical container.

here the question is, can a guest have several of those 'virtual zones' assigned, so that there is a container specific and a shared zone for example?

- > This all works today with fake-numa and cpusets, no kernel changes
- > needed.

sounds good!

- > It could be made to work fairly simply with a multi-zone approach, or
- > with resizeable zones.
- >
- > I'd be interested in knowing what you think the shortcomings of
- > this are likely to be,.

will do so once I have a better understanding how this approach will work ...

TIA,  
Herbert

---