

---

Subject: Re: [RFC][PATCH 2/7] RSS controller core  
Posted by [Herbert Poetzl](#) on Mon, 19 Mar 2007 15:48:08 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Sun, Mar 18, 2007 at 11:42:15AM -0600, Eric W. Biederman wrote:

> Dave Hansen <hansendc@us.ibm.com> writes:

>

> > On Fri, 2007-03-16 at 12:54 -0600, Eric W. Biederman wrote:

> >> Dave Hansen <hansendc@us.ibm.com> writes:

> >

> >> - Why do limits have to apply to the unmapped page cache?

> >

> > To me, it is just because it consumes memory. Unmapped cache is, of  
> > course, much more easily reclaimed than mapped files, but it still  
> > fundamentally causes pressure on the VM.

> >

> > To me, a process sitting there doing constant reads of 10 pages has the  
> > same overhead to the VM as a process sitting there with a 10 page file  
> > mmaped, and reading that.

>

> I can see temporarily accounting for pages in use for such a  
> read/write and possibly during things such as read ahead.

>

> However I doubt it is enough memory to be significant, and as  
> such is probably a waste of time accounting for it.

>

> A memory limit is not about accounting for memory pressure, so I think  
> the reasoning for wanting to account for unmapped pages as a hard  
> requirement is still suspect.

> A memory limit is to prevent one container from hogging all of the  
> memory in the system, and denying it to other containers.

exactly!

nevertheless, you might want to extend that to swapping  
and to the very expensive page in/out operations too

> The page cache by definition is a global resource that facilitates  
> global kernel optimizations. If we kill those optimizations we  
> are on the wrong track. By requiring limits there I think we are  
> very likely to kill our very important global optimizations, and bring  
> the performance of the entire system down.

that is my major concern for most of the 'straight forward'  
virtualizations proposed (see Xen comment)

> >> - Could you mention proper multi process RSS limits.

> >> (I.e. we count the number of pages each group of processes have mapped  
 > >> and limit that).  
 > >> It is the same basic idea as partial page ownership, but instead of  
 > >> page ownership you just count how many pages each group is using and  
 > >> strictly limit that. There is no page owner ship or partial charges.  
 > >> The overhead is just walking the rmap list at map and unmap time to  
 > >> see if this is the first users in the container. No additional kernel  
 > >> data structures are needed.  
 > >  
 > > I've tried to capture this. Let me know what else you think it  
 > > needs.  
 >  
 > Requirements:  
 > - The current kernel global optimizations are preserved and useful.  
 >  
 > This does mean one container can affect another when the  
 > optimizations go awry but on average it means much better  
 > performance. For many the global optimizations are what make  
 > the in-kernel approach attractive over paravirtualization.

total agreement here

> Very nice to have:  
 > - Limits should be on things user space have control of.  
 >  
 > Saying you can only have X bytes of kernel memory for file  
 > descriptors and the like is very hard to work with. Saying you  
 > can have only N file descriptors open is much easier to deal with.

yep, and IMHO more natural ...

> - SMP Scalability.  
 >  
 > The final implementation should have per cpu counters or per task  
 > reservations so in most instances we don't need to bounce a global  
 > cache line around to perform the accounting.

agreed, we want to optimize for small systems  
 as well as for large ones, and SMP/NUMA is quite  
 common in the server area (even for small servers)

> Nice to have:  
 >  
 > - Perfect precision.  
 >  
 > Having every last byte always accounted for is nice but a  
 > little bit of bounded fuzziness in the accounting is acceptable  
 > if it that make the accounting problem more tractable.

as long as the accounting is consistent, i.e.  
you do not lose resources by repetitive operations  
inside the guest (or through guest-guest interaction)  
as this could be used for DoS and intentional unfairness

> We need several more limits in this discussion to get a full picture,  
> otherwise we may try and build the all singing all dancing limit.

> - A limit on the number of anonymous pages.  
> (Pages that are or may be in the swap cache).

> - Filesystem per container quotas.  
> (Only applicable in some contexts but you get the idea).

with shared files, otherwise an lvm partition does  
a good job for that already ...

> - Inode, file descriptor, and similar limits.

> - I/O limits.

I/O and CPU limits are special, as they have the temporal  
component, i.e. you are not interested in 10s CPU time,  
instead you want 0.5s/s CPU (same for I/O)

note: this is probably also true for page in/out

- sockets
- locks
- dentries

HTH,  
Herbert

> Eric

>

> \_\_\_\_\_

> Containers mailing list

> Containers@lists.linux-foundation.org

> <https://lists.linux-foundation.org/mailman/listinfo/containers>

\_\_\_\_\_  
Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

---