Subject: Re: controlling mmap()'d vs read/write() pages
Posted by Herbert Poetzl on Fri, 23 Mar 2007 00:51:36 GMT
View Forum Message <> Reply to Message

On Tue, Mar 20, 2007 at 03:19:16PM -0600, Eric W. Biederman wrote:
> Dave Hansen <hansendc@us.ibm.com> writes:
>
> >
> > So, I think we have a difference of opinion. I think it's _all_
> > about memory pressure, and you think it is _not_ about accounting
> > for memory pressure. :) Perhaps we mean different things, but we
> > appear to disagree greatly on the surface.
>
> I think it is about preventing a badly behaved container from having a
> significant effect on the rest of the system, and in particular other
> containers on the system.
>
> See below. I think to reach agreement we should start by discussing
> the algorithm that we see being used to keep the system function well
> and the theory behind that algorithm. Simply limiting memory is not
> enough to understand why it works....
>
> > Can we agree that there must be _some_ way to control the amounts of
> > unmapped page cache? Whether that's related somehow to the same way
> > we control RSS or done somehow at the I/O level, there must be some
> > way to control it. Agree?
>
> At lot depends on what we measure and what we try and control.
> Currently what we have been measuring are amounts of RAM, and thus
> what we are trying to control is the amount of RAM. If we want to
> control memory pressure we need a definition and a way to measure it.
> I think there may be potential if we did that but we would still need
> a memory limit to keep things like mlock in check.
>
> So starting with a some definitions and theory.
> RSS is short for resident set size. The resident set being how many
> of pages are current in memory and not on disk and used by the
> application. This includes the memory in page tables, but can
> reasonably be extended to include any memory a process can be shown to
> be using.
>
> In theory there is some minimal RSS that you can give an application
> at which it will get productive work done. Below the minimal RSS
> the application will spend the majority of real time waiting for
> pages to come in from disk, so it can execute the next instruction.
> The ultimate worst case here is a read instruction appearing on one
> page and it's datum on another. You have to have both pages in memory
> at the same time for the read to complete. If you set the RSS hard

> limit to one page the problem will be continually restarting either
> because the page it is on is not in memory or the page it is reading
> from is not in memory.
>
> What we want to accomplish is to have a system that runs multiple
> containers without problems. As a general memory management policy
> we can accomplish this by ensuring each container has at least
> it's minimal RSS quota of pages. By watching the paging activity
> of a container it is possible to detect when that container has
> to few pages and is spend all of it's time I/O bound, and thus
> has slipped below it's minimal RSS.
>
> As such it is possible for the memory management system if we have
> container RSS accounting to dynamically figure out how much memory
> each container needs and to keep everyone above their minimal RSS
> most of the time when that is possible. Basically to do this the
> memory manage code would need to keep dynamic RSS limits, and
> adjust them based upon need.
>
> There is still the case when not all containers can have their
> minimal RSS, there is simply not enough memory in the system.
>
> That is where having a hard settable RSS limit comes in. With this
> we communicate to the application and the users beyond which point we
> consider their application to be abusing the system.
>
> There is a lot of history with RSS limits showing their limitations
> and how they work. It is fundamentally a dynamic policy instead of
> a static set of guarantees which allows for applications with a
> diverse set of memory requirements to work in harmony.
>
> One of the very neat things about a hard RSS limit is that if there
> are extra resources on the system you can improve overall system
> performance by cache pages in the page cache instead writing them
> to disk.

that is exactly what we (Linux-VServer) want ...
(sounds good to me, please keep up the good work in
this direction)

there is nothing wrong with hard limits if somebody
really wants them, even if they hurt the sysstem as
whole, but those limits shouldn't be the default ..

best,
Herbert

> > http://linux-mm.org/SoftwareZones

>
> I will try and take a look in a bit.
>
>
> 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