Subject: Re: [RFC][PATCH 2/7] RSS controller core
Posted by ebiederm on Sun, 18 Mar 2007 16:58:06 GMT
View Forum Message <> Reply to Message

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

> On Mon, 2007-03-12 at 23:41 +0100, Herbert Poetzl wrote:
>>
>> let me give a real world example here:
>>
>> - typical guest with 600MB disk space
>> - about 100MB guest specific data (not shared)
>> - assumed that 80% of the libs/tools are used
>
> I get the general idea here, but I just don't think those numbers are
> very accurate. My laptop has a bunch of gunk open (xterm, evolution,
> firefox, xchat, etc...). I ran this command:
>
> lsof | egrep '/(usr/|lib.*\.so)' | awk '{print $9}' | sort | uniq | xargs du
> -Dcs
>
> and got:
>
> 113840 total
>
> On a web/database server that I have (ps aux | wc -l == 128), I just ran
> the same:
>
> 39168   total
>
> That's assuming that all of the libraries are fully read in and
> populated, just by their on-disk sizes. Is that not a reasonable measure
> of the kinds of things that we can expect to be shared in a vserver? If
> so, it's a long way from 400MB.
>
> Could you try a similar measurement on some of your machines? Perhaps
> mine are just weird.

Think shell scripts and the like. From what I have seen I would agree
that is typical for application code not to dominate application memory usage.
However on the flip side it is non uncommon for application code to dominate
disk usage. Some of us have giant music, video or code databases that consume
a lot of disk space but in many instances servers don't have enormous chunks
of private files, and even when they do they share the files from the distribution.

The result of this is that there are a lot of unmapped pages cached in the page
cache for rarely run executables, that are cached just in case we need them.

So while Herbert's numbers may be a little off the general principle of the entire system doing better if you can share the page cache is very real.

That the page cache isn't accounted for here isn't terribly important we still get the global benefit.

> I don't doubt this, but doing this two-level page-out thing for
> containers/vservers over their limits is surely something that we should
> consider farther down the road, right?

It is what the current VM of linux does. There is removing a page from processes and then there is writing it out to disk. I think the normal term is second chance replacement. The idea is that once you remove a page from being mapped you let it age a little before it is paged back in. This allows pages in high demand to avoid being written to disk, all they incur are minor not major fault costs.

> It's important to you, but you're obviously not doing any of the
> mainline coding, right?

Tread carefully here. Herbert may not be doing a lot of mainline coding or extremely careful review of potential patches but he does seem to have a decent grasp of the basic issues. In addition to a reasonable amount of experience so it is worth listening to what he says.

In addition Herbert does seem to be doing some testing of the mainline code as we get it going. So he is contributing.

>> > What are the consequences if this isn't done? Doesn't
>> > a loaded system eventually have all of its pages used
>> > anyway, so won't this always be a temporary situation?
>>
>> let's consider a quite limited guest (or several
>> of them) which have a 'RAM' limit of 64MB and
>> additional 64MB of 'virtual swap' assigned ...
>>
>> if they use roughly 96MB (memory footprint) then
>> having this 'fluffy' optimization will keep them
>> running without any effect on the host side, but
>> without, they will continously swap in and out
>> which will affect not only the host, but also the
>> other guests ...

Ugh. You really want swap > RAM here. Because there are real cases when you are swapping when all of your pages in RAM can be cached in the page cache. 96MB with 64MB RSS and 64MB swap is almost a sure way to hit your swap page limit and die.

> All workloads that use $limit+1 pages of memory will always pay the
> price, right? :)

They should. When you remove an anonymous page from the pages tables it needs to be allocated and placed in the swap cache. Once you do that it can sit in the page cache like any file backed page. So the container that hits $limit+1 should get the paging pressure and a lot more minor faults. However we still want to globally write thing to disk and optimize that as we do right now.

Eric

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers