Subject: Re: [RFC][PATCH 4/7] RSS accounting hooks over the code
Posted by ebiederm on Tue, 13 Mar 2007 16:01:54 GMT
View Forum Message <> Reply to Message

Nick Piggin <nickpiggin@yahoo.com.au> writes:

> Eric W. Biederman wrote:
>>
>> First touch page ownership does not guarantee give me anything useful
>> for knowing if I can run my application or not. Because of page
>> sharing my application might run inside the rss limit only because
>> I got lucky and happened to share a lot of pages with another running
>> application. If the next I run and it isn't running my application
>> will fail. That is ridiculous.
>
> Let's be practical here, what you're asking is basically impossible.
>
> Unless by deterministic you mean that it never enters the a non
> trivial syscall, in which case, you just want to know about maximum
> RSS of the process, which we already account).

Not per process I want this on a group of processes, and yes that
is all I want just. I just want accounting of the maximum RSS of
a group of processes and then the mechanism to limit that maximum rss.

>> I don't want sharing between vservers/VE/containers to affect how many
>> pages I can have mapped into my processes at once.
>
> You seem to want total isolation. You could use virtualization?

No. I don't want the meaning of my rss limit to be affected by what
other processes are doing. We have constraints of how many resources
the box actually has. But I don't want accounting so sloppy that
processes outside my group of processes can artificially
lower my rss value, which magically raises my rss limit.

>> Now sharing is sufficiently rare that I'm pretty certain that problems
>> come up rarely. So maybe these problems have not shown up in testing
>> yet. But until I see the proof that actually doing the accounting for
>> sharing properly has intolerable overhead. I want proper accounting
>> not this hand waving that is only accurate on the third Tuesday of the
>> month.
>
> It is basically handwaving anyway. The only approach I've seen with
> a sane (not perfect, but good) way of accounting memory use is this
> one. If you care to define "proper", then we could discuss that.

I will agree that this patchset is probably in the right general ballpark.

But the fact that pages are assigned exactly one owner is pure non-sense.
We can do better. That is all I am asking for someone to at least attempt
to actually account for the rss of a group of processes and get the numbers
right when we have shared pages, between different groups of
processes. We have the data structures to support this with rmap.

Let me describe the situation where I think the accounting in the
patchset goes totally wonky.

Gcc as I recall maps the pages it is compiling with mmap.
If in a single kernel tree I do:
make -jN O=../compile1 &
make -jN O=../compile2 &

But set it up so that the two compiles are in different rss groups.
If I run the concurrently they will use the same files at the same
time and most likely because of the first touch rss limit rule even
if I have a draconian rss limit the compiles will both be able to
complete and finish. However if I run either of them alone if I
use the most draconian rss limit I can that allows both compiles to
finish I won't be able to compile a single kernel tree.

The reason for the failure with a single tree (in my thought
experiment) is that the rss limit was set below the what is actually
needed for the code to work. When we were compiling two kernels and
they were mapping the same pages at the same time we could put the rss
limit below the minimum rss needed for the compile to execute and
still have it complete because of with first touch only one group
accounted for the pages and the other just leached of the first, as
long as both compiles grabbed some of the pages they could complete.

No I know in practice most draconian limits will simply result in the
page staying in the page cache but not mapped into processes in the
group with the draconian limit, or they will result in pages of the
group with the draconian limit being pushed out into the swap cache.
So the chances of actual application failure even with a draconian
rss limit are quite unlikely. (I actually really appreciate this
fact).

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?

Eric

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers