Subject: Re: [RFC][PATCH 4/7] RSS accounting hooks over the code
Posted by Balbir Singh on Wed, 14 Mar 2007 06:42:06 GMT
View Forum Message <> Reply to Message

Nick Piggin wrote:
> Eric W. Biederman wrote:
>> 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.
>
> Well don't you just sum up the maximum for each process?
>
> Or do you want to only count shared pages inside a container once,
> or something difficult like that?
>
>
>>>> 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.
>
> So what are you going to do about all the shared caches and slabs

> inside the kernel?
>
>
>>> 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.
>
> Well rmap only supports mapped, userspace pages.
>
>
>> 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.
>
> Yeah it is not perfect. Fortunately, there is no perfect solution,
> so we don't have to be too upset about that.
>
> And strangely, this example does not go outside the parameters of
> what you asked for AFAIKS. In the worst case of one container getting
> _all_ the shared pages, they will still remain inside their maximum
> rss limit.
>

When that does happen and if a container hits it limit, with a LRU
per-container, if the container is not actually using those pages,

they'll get thrown out of that container and get mapped into the
container that is using those pages most frequently.

> So they might get penalised a bit on reclaim, but maximum rss limits
> will work fine, and you can (almost) guarantee X amount of memory for
> a given container, and it will _work_.
>
> But I also take back my comments about this being the only design I
> have seen that gets everything, because the node-per-container idea
> is a really good one on the surface. And it could mean even less impact
> on the core VM than this patch. That is also a first-touch scheme.
>

With the proposed node-per-container, we will need to make massive core
VM changes to reorganize zones and nodes. We would want to allow

1. For sharing of nodes
2. Resizing nodes
3. May be more

With the node-per-container idea, it will hard to control page cache
limits, independent of RSS limits or mlock limits.

NOTE: page cache == unmapped page cache here.

>
>> 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?
>
> I think it is simplistic.
>
> Sure you could probably use some of the rmap stuff to account shared
> mapped _user_ pages once for each container that touches them. And
> this patchset isn't preventing that.
>
> But how do you account kernel allocations? How do you account unmapped
> pagecache?
>
> What's the big deal so many accounting people have with just RSS? I'm
> not a container person, this is an honest question. Because from my
> POV if you conveniently ignore everything else... you may as well just
> not do any accounting at all.
>

We decided to implement accounting and control in phases

1. RSS control
2. unmapped page cache control
3. mlock control
4. Kernel accounting and limits

This has several advantages

1. The limits can be individually set and controlled.
2. The code is broken down into simpler chunks for review and merging.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

_____