Subject: Re: [PATCH 0/8] RSS controller based on process containers (v3.1)
Posted by Balbir Singh on Fri, 08 Jun 2007 17:07:56 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Fri, Jun 08, 2007 at 04:39:28PM +0400, Pavel Emelianov wrote:
>> Herbert Poetzl wrote:
>>> On Mon, Jun 04, 2007 at 05:25:25PM +0400, Pavel Emelianov wrote:
>>>> Adds RSS accounting and control within a container.
>>>>
>>>> Changes from v3
>>>> - comments across the code
>>>> - git-bisect safe split
>>>> - lost places to move the page between active/inactive lists
>>>>
>>>> Ported above Paul's containers V10 with fixes from Balbir.
>>>>
>>>> RSS container includes the per-container RSS accounting
>>>> and reclamation, and out-of-memory killer.
>>>>
>>>>
>>>> Each mapped page has an owning container and is linked into its
>>>> LRU lists just like in the global LRU ones. The owner of the page
>>>> is the container that touched the page first.
>>>> As long as the page stays mapped it holds the container, is accounted
>>>> into its usage and lives in its LRU list. When page is unmapped for
>>>> the last time it releases the container.
>>>> The RSS usage is exactly the number of pages in its booth LRU lists,
>>>> i.e. the nu,ber of pages used by this container.
>>> so there could be two guests, unified (i.e. sharing
>>> most of the files as hardlinks), where the first one
>>> holds 80% of the resulting pages, and the second one
>>> 20%, and thus shows much lower 'RSS' usage as the
>>> other one, although it is running the very same
>>> processes and providing identical services?

Hi, Herbert,

For page reclaim one page can belong to only container LRU and only
each physical page is accounted for. Consider what happens when we
equally charge each container

1. Lets say you have containers A and B, both showing that they
   are charged 45% of memory usage
2. The sum of these charges is equal to 90%, but the real memory
   used is just 70%, since 20% of the charges are shared.

The system administrator will find this confusing while assigning resources

>> Herbert!!! Where have you been so long?
>
> I was on vacation in april, and it took almost the
> entire may to process the backlog ...
>
>> You must have missed that we've decided not to account pages
>> sharing right now, but start with that model. Later we'll make
>> sharing accountable.
>
> well, there are two ways not to account sharing:
>
> 1) account it to the first user
> 2) account it to every user
>
> while the first one typically causes quite an
> imbalance when applied to shared resources (as
> Linux-VServer uses them), the latter one creates
> a new, but fair, metric for the usage
>
> to make that clear: we definitely prefer the
> latter one over the former, because we do not
> want to bring that imbalance to our shared guests
> (besides that, the second one doesn't add any
> overhead compared to the first one, more than that
> it probably simplifies the entire design)
>

Please see comments by Nick at

http://lkml.org/lkml/2007/3/14/40

The imbalance is only temporary in the long run, the container
that uses a page more aggressively will get charged for it.
If the container that first brought the pages in is actively
using it, then it's only fair to charge it.

>> (There's something bad with my memory. I have a feeling that I
>> have already told that... many times...)
>

>
>>>> When this usage exceeds the limit set some pages are reclaimed
>>>> from the owning container. In case no reclamation possible the OOM
>>>> killer starts thinning out the container.
>>> so the system (physical machine) starts reclaiming
>>> and probably swapping even when there is no need

>>> to do so?
>> Good catch! The system will start reclaiming right when the
>> container hits the limit to expend its IO bandwidth. Not some
>> other's one that hit the global limit due to some bad container
>> was allowed to go above it.
>
> well, from the system PoV, a constantly swapping
> guest (on an otherwise unused host) is definitely
> something you do not really want, not to talk
> about a tightly packed host system, where guests
> start hogging the I/O with _unnecessary_ swapping
>

Why do you call the swapping unnecessary? We defined a limit for
the containers, so that other containers are not impacted by
the memory usage of our container. We go overboard, it's either

1. We get penalized
2. We are not well configured, reconfigure

>>> e.g. a system with a single guest, limited to 10k
>>> pages, with a working set of 15k pages in different
>>> apps would continuously swap (trash?) on an otherwise
>>> unused (100k+ pages) system?
>>>

What to do with the free memory is an open question, we've
even discussed implementing soft limits (may be if required
sometime in the future).

>>>> Thus the container behaves like a standalone machine -
>>>> when it runs out of resources, it tries to reclaim some
>>>> pages, and if it doesn't succeed, kills some task.
>>> is that really what we want?
>> A kind of ;)
>
> okay, to clarify, we (Linux-VServer) do not want
> that behavior ...
>

Hmmm... could you define the behaviour you expect?

>>> I think we can do _better_ than a standalone machine
>>> and in many cases we really should ...
>> That's it! You are right - this is our ultimate goal. And we
>> plan to get there step by step. And we will appreciate your
>> patches fixing BUGS, improving the performance, extending the

>> functionality, etc.
>
> well, I would rip out the entire accounting and
> add a summation accounting as we do it right now,
> no problem with keeping the reclaim mechanisms
> though ... but I doubt that this is what you have
> in mind?
>

Patches are always welcome!

>>> best,
>>> Herbert
>> Thanks for your attention,
>> Pavel
>
> just to make it clear, I don't want any limits
> in mainline which penalize the first started
> guest in a shared guest scenario .. or to rephrase,
> such a limit would be useless for our purpose
>

In summary, I would like to emphasize on the following points

1. Let's not build an ideal system, lets build something use-able, review-able and understand-able
2. Let's add more features when we hit a bottleneck/problem, code development process is always iterative
3. Please try our patches, test them, provide numbers, feedback, new patches to improve upon the existing code (in a reasonable amount of time, to keep the development on track), you never know you might end up liking what you use :-)

> best,
> Herbert
>

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