
Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...

Posted by [Herbert Poetzl](#) on Sun, 25 Mar 2007 02:21:56 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Sat, Mar 24, 2007 at 12:19:06PM -0800, Andrew Morton wrote:

> On Sat, 24 Mar 2007 19:38:06 +0100 Herbert Poetzl <herbert@13thfloor.at> wrote:

>
>> On Fri, Mar 23, 2007 at 09:42:35PM -0800, Andrew Morton wrote:
>>> On Fri, 23 Mar 2007 20:30:00 +0100 Herbert Poetzl <herbert@13thfloor.at> wrote:

>>>>
>>>>
>>>> Hi Eric!
>>>> Hi Folks!

>>>>
>>>> here is a real world example result from one of my tests
>>>> regarding the benefit of sharing over separate memory
>>>>
>>>> the setup is quite simple, a typical machine used by
>>>> providers all over the world, a dual Pentium D 3.2GHz
>>>> with 4GB of memory and a single 160GB SATA disk running
>>>> a Linux-VServer kernel (2.6.19.7-vs2.2.0-rc18)

>>>>
>>>> the Guest systems used are Mandriva 2007 guests with
>>>> syslog, crond, sshd, apache, postfix and postgresql
>>>> installed and running (all in all 17 processes per guest)

>>>>
>>>> the disk space used by one guests is roughly 148MB
>>>>
>>>> in addition to that, a normal host system is running
>>>> with a few daemons (like sshd, httpd, postfix ...)

>>>>
>>>>
>>>> the first test setup is starting 200 of those guests
>>>> one after the other and measuring the memory usage
>>>> before and after the guest did start, as well as
>>>> recording the time used to start them ...

>>>>
>>>> this is done right after the machine was rebooted, in
>>>> one test with 200 separate guests (i.e. 200 x 148MB)
>>>> and in a second run with 200 unified guests (which
>>>> means roughly 138MB of shared files)

>>>>
>>>> Please define your terms.
>>>> What is a "separated guest", what is a "unified guest"
>>>> and how do they differ?

>>
>> separated guests are complete Linux Distributions which
>> do not share (filesystem wise) anything with any other

> > guest ... i.e. all files and executables have to be
> > paged in and get separate mappings (and thus separate
> > memory)
> >
> > unified guests use a mechanism we (Linux-VServer) call
> > 'unification' which can be considered an advanced form
> > of hard linking (i.e. we add special flags to protect
> > those hard links from modification. such a file is
> > copied on demand (CoW Link Breaking) on the first attempt
> > to be modified (attributes or content)
>
> OK.
>
> > > If a "separated" guest is something in which separate
> > > guests will use distinct physical pages to cache the
> > > contents of /etc/passwd (ie: a separate filesystem
> > > per guest) then I don't think that's interesting
> > > information, frankly.
> >
> > well, you didn't bother to answer my questions regarding
> > your suggested approach yet,
>
> Have been a bit distracted lately, and these discussions
> seem to go on an on without ever converging.

well, it's never easy if there are different ideologies
try to find a common denominator, but contrary to you,
I have the feeling that progress is made ...

> > and as I am concerned that
> > some of the suggested approaches sacrifice performance
> > and resource sharing/efficiency for simplicity or (as
> > we recently had) 'ability to explain it to the customer'
>
> The problem is memory reclaim. A number of schemes which
> have been proposed require a per-container page reclaim
> mechanism - basically a separate scanner.
>
> This is a huge, huge, huge problem. The present scanner
> has been under development for over a decade and has had
> tremendous amounts of work and testing put into it.
> And it still has problems. But those problems will be
> gradually addressed.
>
> A per-container reclaim scheme really really really wants
> to reuse all that stuff rather than creating a separate,
> parallel, new scanner which has the same robustness
> requirements, only has a decade less test and development

- > done on it. And which permanently doubles our maintenance
- > costs.

I completely agree here

- > So how do we reuse our existing scanner? With physical containers.
- > One can envisage several schemes:
- >
- > a) slice the machine into 128 fake NUMA nodes, use each node as the
- > basic block of memory allocation, manage the binding between these
- > memory hunks and process groups with cpusets.

128 sounds a little small to me, considering that we already see 300+ Guests on older machines
(or am I missing something here?)

- > This is what google are testing, and it works.
- >
- > b) Create a new memory abstraction, call it the "software zone",
- > which is mostly decoupled from the present "hardware zones". Most of
- > the MM is reworked to use "software zones". The "software zones" are
- > runtime-resizeable, and obtain their pages via some means from the
- > hardware zones. A container uses a software zone.
- >
- > c) Something else, similar to the above. Various schemes can be
- > envisaged, it isn't terribly important for this discussion.

for me, the most natural approach is the one with the least impact and smallest number of changes in the (granted quite complex) system: leave everything as is, from the 'entire system' point of view, and do adjustments and decisions with the additional Guest/Context information in mind ...

e.g. if we decide to reclaim pages, and the 'normal' mechanism would end up with 100 'equal' candidates, the Guest badness can be a good additional criterion to decide which pages get thrown out ...

OTOH, the Guest status should never control the entire system behaviour in a way which harms the overall performance or resource efficiency

- > Let me repeat: this all has a huge upside in that it reuses the
- > existing page reclamation logic. And cpusets. Yes, we do discover
- > glitches, but those glitches (such as Christoph's recent discovery of
- > suboptimal interaction between cpusets and the global dirty ratio)
- > get addressed, and we tend to strengthen the overall MM system as we

> address them.
>
> So what are the downsides? I think mainly the sharing issue:
>
>>> The issue with pagecache (afaik) is that if we use
>>> containers based on physical pages (an approach which
>>> is much preferred by myself) then we can get in a
>>> situation where a pagecache page is physically in
>>> container A, is not actually used by any process in
>>> container A, but is being relatedly referenced by
>>> processes which are in other containers and hence
>>> unjustly consumes resources in container A.
>>
>>> How significant a problem this is likely to be I do
>>> not know.
>>
>> well, with a little imagination, you can extrapolate
>> that from the data you removed from this email, as one
>> example case would be to start two unified guests one
>> after the other, then shutdown almost everything in
>> the first one, you will end up with the first one being
>> accounted all the 'shared' data used by the second one
>> while the second one will have roughly the resources
>> accounted the first one actually uses ...
>
> Right - that sort of thing.
>
> But how much of a problem will it be *in practice*?

that is a good question, and the answer probably depends
very much on what scenarios you are looking at ...

> Probably a lot of people just won't notice or care.
> There will be a few situations where it may be a problem,
> but perhaps we can address those?

> Forced migration of pages from one zone into another
> is possible.

> Or change the reclaim code so that a page which hasn't
> been referenced from a process within its hardware
> container is considered unreferenced (so it gets reclaimed).

that might easily lead to some ping-pong behaviour,
when two similar guest are executing similar binaries
but not at the same time ...

> Or a manual nuke-all-the-pages knob which system

> administration tools can use.

that sounds interesting ...

> All doable, if we indeed have a demonstrable problem
> which needs to be addressed.

all in all I seem to be missing the 'original problem'
which basically forces us to do all those things you
describe instead of letting the Linux Memory System
work as it works right now and just get the accounting
right ...

> And I do think it's worth trying to address these things,
> because the thought of implementing a brand new memory
> reclaim mechanism scares the pants off me.

right you are and I completely agree here too ...

> > note that the 'frowned upon' accounting Linux-VServer
> > does seem to work for those cases quite fine .. here
> > the relevant accounting/limits for three guests, the
> > first two unified and started in strict sequence, the
> > third one completely separate

> >

> > Limit	current	min/max	soft/hard	hits
> > VM:	41739	0/ 64023	-1/ -1	0
> > RSS:	8073	0/ 9222	-1/ -1	0
> > ANON:	3110	0/ 3405	-1/ -1	0
> > RMAP:	4960	0/ 5889	-1/ -1	0
> > SHM:	7138	0/ 7138	-1/ -1	0

> >

> > Limit	current	min/max	soft/hard	hits
> > VM:	41738	0/ 64163	-1/ -1	0
> > RSS:	8058	0/ 9383	-1/ -1	0
> > ANON:	3108	0/ 3505	-1/ -1	0
> > RMAP:	4950	0/ 5912	-1/ -1	0
> > SHM:	7138	0/ 7138	-1/ -1	0

> >

> > Limit	current	min/max	soft/hard	hits
> > VM:	41738	0/ 63912	-1/ -1	0
> > RSS:	8050	0/ 9211	-1/ -1	0
> > ANON:	3104	0/ 3399	-1/ -1	0
> > RMAP:	4946	0/ 5885	-1/ -1	0
> > SHM:	7138	0/ 7138	-1/ -1	0

>

> Sorry, I tend to go to sleep when presented with rows and rows of
> numbers. Sure, it's good to show the data but I much prefer it if the

> sender can tell us what the data means: the executive summary.

sorry, I'm more the technical person and I hate 'executive summaries' and similar stuff, but the message is simple and clear: accounting works even for shared/unified guests, all three guests show reasonably similar values ...

> There's not a lot of point in every reader having to duplicate the
> analysis work which the sender has performed.

I'm confident, a simple comparison can be expected from folks doing complicate patch reviews and such

but I will try to present a better 'conclusion' next time ...

best,
Herbert

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