Subject: Linux-VServer example results for sharing vs. separate mappings ...
Posted by Herbert Poetzl on Fri, 23 Mar 2007 19:30:00 GMT
View Forum Message <> Reply to Message

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)

separate guests:

```
GUEST  TIME    ACTIVE BUFFERS CACHE    ANON  MAPPED  SLAB  RECLAIM   URECL
-------------------------------------------------------------------------
001       0 16364  2600 20716  4748  3460  8164   2456    5708
002       7 30700  3816 42112  9052  8200 11056   3884    7172
003      13 44640  4872 62112 13364 12872 13248   5268    7980
004      20 58504  5972 82028 17684 17504 15348   6616    8732
005      28 72352  7056  102052 21948 22172 17640   8020    9620
....
196    1567   2072172  156404 2409368  841168  915484  414056 246952  167104
197    1576   2080836  154680 2402344  845544  920268  414432 246784  167648
```

198 1585 2093424 153400 2399560 849696 924760 414892 246572 168320
199 1593 2103368 151540 2394048 854020 929660 415300 246324 168976
200 1599 2113004 149272 2382964 858344 934336 415528 245896 169632

unified guests:

GUEST TIME ACTIVE BUFFERS CACHE ANON MAPPED SLAB RECLAIM URECL
---
001 0 16576 2620 20948 4760 3444 8232 2520 5712
002 10 31368 4672 74956 9068 8140 12976 5760 7216
003 14 38888 5364 110508 13368 9696 16516 8360 8156
004 18 44068 6104 146044 17696 11236 19868 10972 8896
005 22 49324 6824 181540 21964 12764 23264 13580 9684
....
196 1289 1159780 88856 2503448 841864 304544 383196 232944 150252
197 1294 1166528 88524 2500616 846168 306068 384056 233096 150960
198 1304 1172124 88468 2492268 850452 307596 384560 232988 151572
199 1313 1178876 88896 2488476 854840 309092 385384 233064 152320
200 1322 1184368 88568 2483208 858988 310640 386256 233388 152868

the second test was quite interesting too, as it showed
nicely what the effect on the overall performance can be:

in this test, all guests are started at the same time, and
the script waits until the last guest has successfully
started ...

the 200 separate guests (as you probably can imagine) caused
quite a load when started at once (there are a number of
userspace tools preparing the guest on startup and setting
up the context) and obviously they also pushed the memory
limits somewhat ...

the startup for 200 separate guests (at once) did take this
system 1h 11m 27s (compared to the 26m 39s in sequence)

the startup for 200 unified guests (at once) OTOH, did take
45s (yes, below a minute! compared to 22m 2s in sequential
order)

HTH,
Herbert

PS: if you need details for the setup, and/or want to
recreate that on your system, just let me know, I can
provide all the required data (including the guests)

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by akpm on Sat, 24 Mar 2007 05:42:35 GMT
View Forum Message <> Reply to Message

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?

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.

Because nobody (afaik) is proposing that pagecache be duplicated across instances in this fashion.

We obviously must share pagecache across instances - if we didn't want to do that then we could do something completely dumb such as use xen/kvm/vmware/etc ;)

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 releatedly 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. And there are perhaps things which we can do about it.

_____

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

---

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by Herbert Poetzl on Sat, 24 Mar 2007 18:38:06 GMT
View Forum Message <> Reply to Message

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)

so although all guests use a separate namespace (i.e.
will have separate dentries) they share most of the files
(those which are not modified) via inodes (and the inode
cache of course)

> 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, 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'
I thought I provide some data how much resource sharing
can help (the overall performance)

> Because nobody (afaik) is proposing that pagecache be
> duplicated across instances in this fashion.
>
> We obviously must share pagecache across instances -
> if we didn't want to do that then we could do something
> completely dumb such as use xen/kvm/vmware/etc ;)

exactly my words ...

> 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 releatedly 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 ...

note that the 'frowned upon' accounting Linux-VServer
does seems 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
```

> And there are perhaps things which we can do about it.

best,
Herbert

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

---

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by akpm on Sat, 24 Mar 2007 20:19:06 GMT
View Forum Message <> Reply to Message

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.

> 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 recaim 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.

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.

   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.

Let me repeat: this all has a huge upside in that it reuses the existing page reclaimation 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 releatedly 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*? 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). Or a manual nuke-all-the-pages knob which system administration tools can use. All doable, if we indeed have a demonstrable problem which needs to be addressed.

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.

> note that the 'frowned upon' accounting Linux-VServer
> does seems 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. There's not a lot of
point in every reader having to duplicate the analysis work which the
sender has performed.

_____

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 recaim 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 reclaimation 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 releatedly 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 seems 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: accouting 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

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by akpm on Sun, 25 Mar 2007 04:29:51 GMT
View Forum Message <> Reply to Message

On Sun, 25 Mar 2007 04:21:56 +0200 Herbert Poetzl <herbert@13thfloor.at> wrote:

> > 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?)

Yes, you're missing something very significant. I'm talking about resource management (ie: partitioning) and you're talking about virtual servers. They're different applications, with quite a lot in common.

For resource management, a few fives or tens of containers is probably an upper bound.

An impementation needs to address both requirements.

> > 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

On the contrary - if one container exceeds its allotted resource, we want
the processes in that container to bear the majority of the cost of that.
Ideally, all of the cost.

>
> > 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 ...

The VM presently cannot satisfy resource management requirements, because
piggy activity from one job will impact the performance of all other jobs.

> > > note that the 'frowned upon' accounting Linux-VServer
> > > does seems 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: accouting works even
> for shared/unified guests, all three guests show
> reasonably similar values ...

I don't see "accounting" as being useful for resource managment. I mean,
so we have a bunch of numbers - so what?

The problem is: what do we do when the jobs in a container exceed their
allotment?

With zone-based physical containers we already have pretty much all the
accounting we need, in the existing per-zone accounting.

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

---

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by Balbir Singh on Sun, 25 Mar 2007 09:50:35 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
<snip>
> 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 recaim 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.
>

The current per-container reclaim scheme does reuse a lot of code. As far as code maintenance is concerned, I think it should be easy to merge some of the common functionality by abstracting them out as different functions. The container smartness comes in only in the container_isolate_pages(). This is an easy to understand function.

> 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.
>
>   This is what google are testing, and it works.

Don't we break the global LRU with this scheme?

>
> 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.
>

I think the problem would be figuring out where to allocate memory from? What happens if a software zone spans across many hardware zones?

> c) Something else, similar to the above. Various schemes can be
>   envisaged, it isn't terribly important for this discussion.
>
>
> Let me repeat: this all has a huge upside in that it reuses the existing
> page reclaimation 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:

I think binding the resource controller and the allocator might be a bad idea, I tried experimenting with it and soon ran into some hard to answer questions

1. How do we control the length of the zonelists that we need to
   allocate memory from (in a container)

2. Like you said, how do we share pages across zones (containers)
3. What happens to the global LRU behaviour
4. Do we need a per_cpu_pageset assoicated with containers
5. What do we do with unused memory in a zone, is it shared with
   other zones
6. Changing zones or creating an abstraction out of it is likely
   to impact the entire vm setup core, that is high risk, so
   do we really need to do it this way.

> But how much of a problem will it be *in practice*? 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). Or a
> manual nuke-all-the-pages knob which system administration tools can use.
> All doable, if we indeed have a demonstrable problem which needs to be
> addressed.
>
> 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.
>

The reclaim mechanism proposed *does not impact the non-container users*.
The only impact is container driven reclaim, like every other new feature
this can benefit from good testing in -mm. I believe we have something
simple and understandable to get us started. I would request you to consider
merging the RSS controller and containers patches in -mm. If too many people
complain or we see the problems that you foresee and our testing,
enhancements and maintenance is unable to sort those problems, we know
we'll have another approach to fall back upon :-) It'll also teach us
to listen to the maintainers when they talk of design ;)

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

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

On Sat, Mar 24, 2007 at 08:29:51PM -0800, Andrew Morton wrote:
> On Sun, 25 Mar 2007 04:21:56 +0200 Herbert Poetzl <herbert@13thfloor.at> wrote:
>
> > > 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?)
>
> Yes, you're missing something very significant. I'm talking about
> resource management (ie: partitioning) and you're talking about
> virtual servers. They're different applications, with quite a lot
> in common.

okay, but please explain how that what you call
'resource management' is different from the resource
accounting and resource limits used in Linux-VServer
(and please ignore the implementation details :)

> For resource management, a few fives or tens of containers is probably
> an upper bound.
>
> An impementation needs to address both requirements.
>
> > >   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
>
> On the contrary - if one container exceeds its allotted resource, we
> want the processes in that container to bear the majority of the cost
> of that. Ideally, all of the cost.

well, I totally agree here, the _container_ should bear
the cost, but not the entire system!

I'm fine (and I mentioned that several times) with
penalizing Guests which are over limit either in general
or on the actual resource access, but IMHO it is a very
bad idea to hurt the entire system like a partitioning
system would do ...

> > > 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 ...
>
> The VM presently cannot satisfy resource management requirements,
> because piggy activity from one job will impact the performance
> of all other jobs.

that is correct (at least to some extend)

> > > > note that the 'frowned upon' accounting Linux-VServer
> > > > does seems 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: accouting works even
> > for shared/unified guests, all three guests show
> > reasonably similar values ...
>
> I don't see "accounting" as being useful for resource managment.
> I mean, so we have a bunch of numbers - so what?

IMHO it is the basis for resource management, if you cannot
account the used resources, you will not be able to limit
them in a proper way, no?

> The problem is: what do we do when the jobs in a container exceed
> their allotment?

there are two different kinds of resource, the ones you
basically 'give' on request and the ones which are 'taken'
when there is need. the former ones can easily be checked
with some limit and simply be denied when crossing over.
the latter ones should IMHO penalize the Guest in such way
that the 'good' Guests are not affected or even benefit
from the 'bad' ones ...

This, IMHO is the real challenge in OS-level isolation
and virtualization ... YMMV

> With zone-based physical containers we already have pretty much all
> the accounting we need, in the existing per-zone accounting.

maybe, but for example, saying that we can only have
128 limited Guests, because all you ever will have is like
a dozen 'partitions' makes the entire approach quite useless
for the typical scenarios Linux-VServer and OpenVZ is used
for ... note: we have users with 300+ limited Guests on
larger machines ....

HTC,
Herbert

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

---

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by akpm on Sun, 25 Mar 2007 18:51:09 GMT
View Forum Message <> Reply to Message

On Sun, 25 Mar 2007 15:20:35 +0530 Balbir Singh <balbir@in.ibm.com> wrote:

> Andrew Morton wrote:
> <snip>
> > 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 recaim 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.
> >
>
> The current per-container reclaim scheme does reuse a lot of code. As far
> as code maintenance is concerned, I think it should be easy to merge
> some of the common functionality by abstracting them out as different
> functions. The container smartness comes in only in the
> container_isolate_pages(). This is an easy to understand function.

err, I think I'd forgotten about container_isolate_pages(). Yes, that
addresses my main concern.

> > 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.
> >
> >   This is what google are testing, and it works.
>
> Don't we break the global LRU with this scheme?

Sure, but that's deliberate!

(And we don't have a global LRU - the LRUs are per-zone).

> >
> > 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.
> >
>
> I think the problem would be figuring out where to allocate memory from?
> What happens if a software zone spans across many hardware zones?

Yes, that would be the tricky part. But we generally don't care what
physical zone user pages come from, apart from NUMA optimisation.

> The reclaim mechanism proposed *does not impact the non-container users*.

Yup. Let's keep plugging away with Pavel's approach, see where it gets us.

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by Balbir Singh on Mon, 26 Mar 2007 02:36:07 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
>> Don't we break the global LRU with this scheme?
>
> Sure, but that's deliberate!

>
> (And we don't have a global LRU - the LRUs are per-zone).
>

Yes, true. But if we use zones for containers and say we have 400
of them, with all of them under limit. When the system wants
to reclaim memory, we might not end up reclaiming the best pages.
Am I missing something?

>>> 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.
>>>
>> I think the problem would be figuring out where to allocate memory from?
>> What happens if a software zone spans across many hardware zones?
>
> Yes, that would be the tricky part. But we generally don't care what
> physical zone user pages come from, apart from NUMA optimisation.
>
>> The reclaim mechanism proposed *does not impact the non-container users*.
>
> Yup. Let's keep plugging away with Pavel's approach, see where it gets us.
>

Yes, we have some changes that we've made to the reclaim logic, we hope
to integrate a page cache controller soon. We are also testing the
patches. Hopefully soon enough, they'll be in a good state and we can
request you to merge the containers and the rss limit (plus page cache)
controller soon.

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

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

---

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by akpm on Mon, 26 Mar 2007 05:26:48 GMT
View Forum Message <> Reply to Message

On Mon, 26 Mar 2007 08:06:07 +0530 Balbir Singh <balbir@in.ibm.com> wrote:

> Andrew Morton wrote:
> >> Don't we break the global LRU with this scheme?
> >
> > Sure, but that's deliberate!
> >
> > (And we don't have a global LRU - the LRUs are per-zone).
> >
>
> Yes, true. But if we use zones for containers and say we have 400
> of them, with all of them under limit. When the system wants
> to reclaim memory, we might not end up reclaiming the best pages.
> Am I missing something?

If a zone is under its min_pages limit, it needs reclaim. Who/when/why
that reclaim is run doesn't really matter.

Yeah, we might run into some scaling problems with that many zones.
They're unlikely to be unfixable.

> >>> 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.
> >>>
> >> I think the problem would be figuring out where to allocate memory from?
> >> What happens if a software zone spans across many hardware zones?
> >
> > Yes, that would be the tricky part. But we generally don't care what
> > physical zone user pages come from, apart from NUMA optimisation.
> >
> >> The reclaim mechanism proposed *does not impact the non-container users*.
> >
> > Yup. Let's keep plugging away with Pavel's approach, see where it gets us.
> >
>
> Yes, we have some changes that we've made to the reclaim logic, we hope
> to integrate a page cache controller soon. We are also testing the
> patches. Hopefully soon enough, they'll be in a good state and we can
> request you to merge the containers and the rss limit (plus page cache)
> controller soon.

Now I'm worried again. This separation between "rss controller" and
"pagecache" is largely alien to memory reclaim. With physical containers
these new concepts (and their implementations) don't need to exist - it is

already all implemented.

Designing brand-new memory reclaim machinery in mid-2007 sounds like a very
bad idea. But let us see what it looks like.

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

---

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by Balbir Singh on Mon, 26 Mar 2007 06:05:54 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Mon, 26 Mar 2007 08:06:07 +0530 Balbir Singh <balbir@in.ibm.com> wrote:
>
>> Andrew Morton wrote:
>>>> Don't we break the global LRU with this scheme?
>>> Sure, but that's deliberate!
>>>
>>> (And we don't have a global LRU - the LRUs are per-zone).
>>>
>> Yes, true. But if we use zones for containers and say we have 400
>> of them, with all of them under limit. When the system wants
>> to reclaim memory, we might not end up reclaiming the best pages.
>> Am I missing something?
>
> If a zone is under its min_pages limit, it needs reclaim. Who/when/why
> that reclaim is run doesn't really matter.
>
> Yeah, we might run into some scaling problems with that many zones.
> They're unlikely to be unfixable.
>

ok.

>
>>>>> 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.
>>>>>
>>>> I think the problem would be figuring out where to allocate memory from?
>>>> What happens if a software zone spans across many hardware zones?
>>> Yes, that would be the tricky part. But we generally don't care what

>>> physical zone user pages come from, apart from NUMA optimisation.
>>>
>>>> The reclaim mechanism proposed *does not impact the non-container users*.
>>> Yup. Let's keep plugging away with Pavel's approach, see where it gets us.
>>>
>> Yes, we have some changes that we've made to the reclaim logic, we hope
>> to integrate a page cache controller soon. We are also testing the
>> patches. Hopefully soon enough, they'll be in a good state and we can
>> request you to merge the containers and the rss limit (plus page cache)
>> controller soon.
>
> Now I'm worried again. This separation between "rss controller" and
> "pagecache" is largely alien to memory reclaim. With physical containers
> these new concepts (and their implementations) don't need to exist - it is
> already all implemented.
>
> Designing brand-new memory reclaim machinery in mid-2007 sounds like a very
> bad idea. But let us see what it looks like.
>

I did not mean to worry you again :-) We do not plan to implement brand
new memory reclaim, we intend to modify some bits and pieces for per
container reclaim. We believe at this point that all the necessary
infrastructure is largely present in container_isolate_pages(). Adding
a page cache controller should not require core-mm surgery, just the
accounting bits.

We basically agree that designing a brand new reclaim machinery is a bad
idea, non-container users will not be impacted. Only container driver
reclaim (caused by a container being at it's limit), will see some change
in reclaim behaviour and we shall try and restrict the changes to as
small as possible.

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

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

---

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by Ethan Solomita on Wed, 28 Mar 2007 01:22:02 GMT
View Forum Message <> Reply to Message

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

It might lead to that, but I don't think it would become pathological "easily". If a system has been up for a long time, it's easy to image pagecache pages lying everywhere just because someone somewhere is still using them.

I suggest a variant on what Andrew says: don't change reclaim. Instead, when referencing a page, don't mark the page as referenced if the current task is not permitted to allocate from the page's node. I'm thinking in terms of cpusets, with each task having a nodemask of mems_allowed. This may result in a page being thrown out unnecessarily and brought back in from disk, but when memory is tight that is what happens. An optimization might be to keep track of who is referencing the page and migrate it to their memory instead of reclaiming it, but that would require reclaim to know the task/cpuset/container of the referencing task.
-- Ethan

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

---

Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by Bodo Eggert on Thu, 29 Mar 2007 07:24:43 GMT
View Forum Message <> Reply to Message

Ethan Solomita <solo@google.com> wrote:

> I suggest a variant on what Andrew says: don't change reclaim.
> Instead, when referencing a page, don't mark the page as referenced if
> the current task is not permitted to allocate from the page's node. I'm
> thinking in terms of cpusets, with each task having a nodemask of
> mems_allowed. This may result in a page being thrown out unnecessarily
> and brought back in from disk, but when memory is tight that is what

> happens. An optimization might be to keep track of who is referencing
> the page and migrate it to their memory instead of reclaiming it, but
> that would require reclaim to know the task/cpuset/container of the
> referencing task.

If you are neaar reclaim and the page from that other node hasn't been
touched recently, maybe you could steal it. Off cause this is only
applicable if the checks are cheap enough.
--
Funny quotes:
40. Isn't making a smoking section in a restaurant like making a peeing
    section in a swimming pool?

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