Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Andrew Morton on Thu, 18 May 2006 17:34:30 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> wrote:
>
> This patchset introduces a per-process utsname namespace. These can
> be used by openvz, vserver, and application migration to virtualize and
> isolate utsname info (i.e. hostname). More resources will follow, until
> hopefully most or all vserver and openvz functionality can be implemented
> by controlling resource namespaces from userspace.
>

Generally, I think that the whole approach of virtualising the OS so it can run multiple independent instances of userspace is a good one. It's an extension and a strengthening of things which Linux is already doing and it pushes further along a path we've been taking for many years. If done right, it's even possible that each of these featurettes could improve the kernel in its own right - better layering, separation, etc.

The approach which you appear to be taking is to separate the bits of functionality apart and to present them as separate works each of which is reviewed-by, acceptable-to and will-be-used-by all of the interested projects. That's ideal, and is very much appreciated.

All of which begs the question "now what?".

What we do _not_ want to do is to merge up a pile of infrastructural stuff which never gets used. On the other hand, we don't want to be in a position where nothing is merged into mainline until the entirety of vserver &&/|| openvs is ready to be merged.

I see two ways of justifying a mainline merge of things such as this

a) We make an up-front decision that Linux _will_ have OS-virtualisation capability in the future and just start putting in place the pieces for that, even if some of them are not immediately useful.

   I suspect that'd be acceptable, although I worry that we'd get partway through and some issues would come up which are irreconcilable amongst the various groups.

   It would help set minds at ease if someone could produce a bullet-point list of what features the kernel will need to get it to the stage where "most or all vserver and openvz functionality can be implemented by controlling resource namespaces from userspace." Then we can discuss that list, make sure that everyone's pretty much in

agreement.

It would be good if that list were to identify which features are useful to Linux in their own right, and which ones only make sense within a whole virtualise-the-OS setup.

b) Only merge into mainline those feature which make sense in a standalone fashion. eg, we don't merge this patchset unless the "per-process utsname namespace" feature is useful to and usable by a sufficiently broad group of existing Linux users.

I suspect this will be a difficult approach.

The third way would be to buffer it all up in -mm until everything is sufficiently in place and then slam it all in. That might not be feasible for various reasons - please advise..

A fourth way would be for someone over there to run a git tree - you all happily work away, I redistribute it in -mm for testing and one day it's all ready to merge. I don't really like this approach. It ends up meaning that nobody else reviews the new code, nobody else understands what it's doing, etc. It's generally subversive of the way we do things.

Eric, Kirill, Herbert: let us know your thoughts, please.

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by John Kelly on Thu, 18 May 2006 19:23:20 GMT
View Forum Message <> Reply to Message

On Thu, 18 May 2006 10:34:30 -0700, Andrew Morton <akpm@osdl.org> wrote:

>I see two ways of justifying a mainline merge of things such as this

>a) We make an up-front decision that Linux _will_ have OS-virtualisation
> capability in the future

After using OpenVZ for a short time, I wonder how I ever managed without it. For application development and testing, having a little sandbox with only a few PIDs running makes it easier to debug things.

> and just start putting in place the pieces for that, even if some
> of them are not immediately useful. I suspect that'd be acceptable,
> although I worry that we'd get partway through and some issues would
> come up which are irreconcilable amongst the various groups.

>From a user's POV, I want it ASAP. As for conflicts, why not cross
that bridge when you come to it?

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Sam Vilain on Thu, 18 May 2006 23:28:51 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:

>Generally, I think that the whole approach of virtualising the OS so it can
>run multiple independent instances of userspace is a good one.
>[...]
>All of which begs the question "now what?".
>[...]
> It would help set minds at ease if someone could produce a
> bullet-point list of what features the kernel will need to get it to the
> stage where "most or all vserver and openvz functionality can be
> implemented by controlling resource namespaces from userspace." Then we
> can discuss that list, make sure that everyone's pretty much in
> agreement.
>
>

This is a heartening position to hear from someone such as yourself; we
seem to be at a near consensus of the way forward.

Here's a list based on the one I came up with when I originally started
my line of development, which got shot down so badly it lost a few
priority points on my workqueue scheduler :-).

0. features that don't need namespaces per se

a. Bind Mount Options (mount --bind -o ro, etc)
b. FS - immutable linkage invert (immulink)

1. core vserver patch - no features (this stuff is succeeded by Serge's
set)
a. struct and ps addition; internal API and refcounting
b. syscall, and switch (to be canned)
c. /proc visibility
d. debugging
e. history

2. isolation features

a. IPC, semaphore, and signal restrictions
b. proc/array filtering

c. IPv4 chbind
d. FS chroot() barrier
e. general /proc filtering
f. ptrace
g. process admin: alloc_uid, find_user, sys_setpriority

3. virtualisation features

a. uts information
b. initpid virtualisation
c. uptime
d. time
e. load average
f. ksyslog
g. vshelper (reboot support)
h. vroot (quota, fs IOCTL, etc)
i. general PID virtualisation (eric)
j. ngnet (network stack virtualisation)

4. resource tracking features

a. scheduler tracking hook
b. FS namespace counting
c. FS namespace tagging
d. ulimits
e. RSS usage
f. IO - async tracking

5. resource sharing features

a. scheduling v1 - TBF and vavavoom
b. disk scheduler integration
c. RSS limits
d. FS - mad cow

6. resource limit features

a. scheduler
b. rlimits
c. disklimits

7. super whizzy features

a. Namespace checkpointing
b. Namespace migration
c. HA Cluster Computing (think Tandem)

Can anyone see any that are missed?

As far as how it is tested etc, I have no particular preferences,
whatever people are happy with. I'll continue to track submissions in
the utsl.gen.nz repository:

http://utsl.gen.nz/gitweb/?p=vserver

I'll import Serge's new submission there now.

Sam.

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Paul Jackson on Fri, 19 May 2006 04:24:17 GMT
View Forum Message <> Reply to Message

> Can anyone see any that are missed?

I have no idea if this fits, as I am no virtual kernel wizard,
but how about various NUMA stuff, such as what CPUs and Memory
Nodes are online, and the three ways of controlling task and
memory placement on them:
  sched_setaffinity/sched_getaffinity
  set_mempolicy/get_mempolicy/mbind
  /dev/cpuset

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Fri, 19 May 2006 09:23:03 GMT
View Forum Message <> Reply to Message

Paul Jackson <pj@sgi.com> writes:

>> Can anyone see any that are missed?
>
> I have no idea if this fits, as I am no virtual kernel wizard,
> but how about various NUMA stuff, such as what CPUs and Memory
> Nodes are online, and the three ways of controlling task and
> memory placement on them:
> sched_setaffinity/sched_getaffinity
> set_mempolicy/get_mempolicy/mbind
> /dev/cpuset

I expect especially on very large machines for some of this to be done
in conjunction with setting up the isolated instances of user space.
But anything actually touching the hardware is an independent dimension.

Eric

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Fri, 19 May 2006 11:41:45 GMT
View Forum Message <> Reply to Message

Andrew Morton <akpm@osdl.org> writes:

> All of which begs the question "now what?".

I think we are at the point where it is time to start merging patches
into -mm, and having the discussion on what the merge plans are
for the rest of this code.

> What we do _not_ want to do is to merge up a pile of infrastructural stuff
> which never gets used. On the other hand, we don't want to be in a
> position where nothing is merged into mainline until the entirety of
> vserver &&/|| openvs is ready to be merged.

The namespaces I see needed for a useable result are:
- fs namespace (already merged)
- uts namespace
- sysvipc namespace
- time namespace
- uid/gid (keys?) namespace
- network namespace
- pid namespace

> I see two ways of justifying a mainline merge of things such as this
>
> a) We make an up-front decision that Linux _will_ have OS-virtualisation
>   capability in the future and just start putting in place the pieces for
>   that, even if some of them are not immediately useful.
>
>   I suspect that'd be acceptable, although I worry that we'd get
>   partway through and some issues would come up which are irreconcilable
>   amongst the various groups.

I think I see a third way of justifying a mainline merge. We make an
up-front decision that we will improve the existing chroot jail
functionality in Linux and start making improvements. Even if some of

the improvements are quite small.

Except for partial steps while the code is being refactored, we should
never have steps that are not immediately useful.

This reduces the danger of irreconcilable differences, because being
part way through is still useful.

The only namespace that I see as really contentious is the pid
namespace, and even there I don't think we have read an impasse.
There remains a bunch of patches left to write that replace raw pid_t
values with struct pid references, but once that happens the patches
to implement the pid namespace will be small, and I don't see any
previous problems that we can't resolve when the conversation happens.

> It would help set minds at ease if someone could produce a
> bullet-point list of what features the kernel will need to get it to the
> stage where "most or all vserver and openvz functionality can be
> implemented by controlling resource namespaces from userspace." Then we
> can discuss that list, make sure that everyone's pretty much in
> agreement.

So this is slightly the wrong question. If you look at Sam's list you
will see that there are several independent dimensions to the complete
solution. Most of them dealing with the increase in the number of users
and the amount of work that is happening on a single kernel in this
context.

Basically we need to expect a lot of kernel tuning after we get the
basics working.

The proper question is: What needs to happen before we can run separate
user space instances?

The namespaces I have previously listed. There is also a lot of
cleanup work with sysctl, proc, sysfs, netlink and some other
fundamental interfaces that needs to happen as well. Until each
namespace gets merged we are in a race with other people looking
at enhancing those namespaces. So a complete of what needs to be
fixed is impossible.

> b) Only merge into mainline those feature which make sense in a
> standalone fashion. eg, we don't merge this patchset unless the
> "per-process utsname namespace" feature is useful to and usable by a
> sufficiently broad group of existing Linux users.
>
> I suspect this will be a difficult approach.

I agree if the feature must be useful and usable by a sufficiently
broad group of existing Linux users. Of course I suspect the current
fs namespace fails this test.

I would rather the criteria be, that the functionality that is well
defined and not detrimental to the rest of users.

> The third way would be to buffer it all up in -mm until everything is
> sufficiently in place and then slam it all in. That might not be feasible
> for various reasons - please advise..

Fundamentally I don't think there are problems buffering things up in -mm,
but I worry that we would start having -mm too different from the
stable kernel at some point.

For some of the pieces like the networking stack we need to go through
the respective maintainers, and their development trees to avoid
conflicts. For the sysvipc, utsname, and we have avoided that
because they are absolutely trivial namespaces and they don't have
active maintainers.

> A fourth way would be for someone over there to run a git tree - you all
> happily work away, I redistribute it in -mm for testing and one day it's
> all ready to merge. I don't really like this approach. It ends up meaning
> that nobody else reviews the new code, nobody else understands what it's
> doing, etc. It's generally subversive of the way we do things.

The only part of this picture that might make sense is if we have a
process by which we can decide if patches are good and acceptable to
the various projects independent of deciding if they are good for
the kernel proper, which might take some of the burden off of the
rest of the kernel maintainers.

If we were working in an area of the kernel where we didn't affect
anyone else it would be business as usual and not really subversive.
But since we can't implement things this way I agree that this
code needs to be reviewed as much as possible.

> Eric, Kirill, Herbert: let us know your thoughts, please.

Eric

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Herbert Poetzl on Fri, 19 May 2006 12:42:35 GMT
View Forum Message <> Reply to Message

On Thu, May 18, 2006 at 10:34:30AM -0700, Andrew Morton wrote:
> "Serge E. Hallyn" <serue@us.ibm.com> wrote:
> >
> > This patchset introduces a per-process utsname namespace. These can
> > be used by openvz, vserver, and application migration to virtualize and
> > isolate utsname info (i.e. hostname). More resources will follow, until
> > hopefully most or all vserver and openvz functionality can be implemented
> > by controlling resource namespaces from userspace.
>
> Generally, I think that the whole approach of virtualising the OS
> so it can run multiple independent instances of userspace is a good
> one. It's an extension and a strengthening of things which Linux is
> already doing and it pushes further along a path we've been taking
> for many years.

yes, I too think that Linux has been moving in that
direction for a long time now, maybe even too long, so
that other OSes (like BSD or Solaris) already managed
to get a virtualization layer up and running (although
maybe at a much simpler level than we plan to do)

> If done right, it's even possible that each of these featurettes could
> improve the kernel in its own right - better layering, separation,
> etc.

agreed, most 'features' will require a cleanup of some
otherwise completely untouched areas, which (hopefully)
will improve those areas ...

> The approach which you appear to be taking is to separate the bits
> of functionality apart and to present them as separate works each of
> which is reviewed-by, acceptable-to and will-be-used-by all of the
> interested projects. That's ideal, and is very much appreciated.

IMHO many things will make perfect sense on their own
even without the 'other' virtualizations or isolations.
With Linux-VServer it's an every day occurance, that
folks just 'cherry pick' the isolation features and build
their own level of virtual/isolated environment.

at this point, many thanks to Sam, Eric and Serge
who do a really good job in massaging patches :)

> All of which begs the question "now what?".
>
> What we do _not_ want to do is to merge up a pile of infrastructural
> stuff which never gets used. On the other hand, we don't want to be in
> a position where nothing is merged into mainline until the entirety of

> vserver &&/|| openvs is ready to be merged.

yes, I agree here, and I'm pretty sure that we are
still missing many 'stakeholders' here just because
we do not see all possible areas of use ... let me
give a simple example here:

"pid virtualization"

- Linux-VServer doesn't really need that right now.
  we are perfectly fine with "pid isolation" here, we
  only "virtualize" the init pid to make pstree happy

- Snapshot/Restart and Migration will require "full"
  pid virtualization (that's where Eric and OpenVZ
  are heading towards)

- OpenSSI and *Mosix require system wide pid spaces
  which probably could be implemented with virtual
  pid spaces as well

- many security addons provide something called pid
  randomization, and I think they could probably
  benefit from a virtual pid space, too

now does that mean that e.g. Linux-VServer is against
"pid virtualization"? well, we are mainly against all
unnecessary overhead and strictly against losing the
ability to keep it simple for the user, i.e. somebody
who does not require all that stuff should be able to
pick the features (or spaces) she really needs ...

> I see two ways of justifying a mainline merge of things such as this
>
> a) We make an up-front decision that Linux _will_ have
> OS-virtualisation capability in the future and just start putting
> in place the pieces for that, even if some of them are not
> immediately useful.

as long as this doesn't automatically mean bloat, I'm
more than happy with such a decision ...

> I suspect that'd be acceptable, although I worry that we'd get
> partway through and some issues would come up which are
> irreconcilable amongst the various groups.

I'm pretty sure that we _will_ hit some issues, but
I'm also sure that we will be able to work out those

issues, after all the 'end user' has to decide what
should be in mainline and what not ...

> It would help set minds at ease if someone could produce a
> bullet-point list of what features the kernel will need to get
> it to the stage where "most or all vserver and openvz functionality
> can be implemented by controlling resource namespaces from
> userspace." Then we can discuss that list, make sure that
> everyone's pretty much in agreement.

excellent idea, will start preparing such a list
from our PoV, so that we can merge that with the
other lists ...

> It would be good if that list were to identify which features are
> useful to Linux in their own right, and which ones only make
> sense within a whole virtualise-the-OS setup.

that's probably the hardest part ...

> b) Only merge into mainline those feature which make sense in a
> standalone fashion. eg, we don't merge this patchset unless the
> "per-process utsname namespace" feature is useful to and usable
> by a sufficiently broad group of existing Linux users.

the question here is, who are the users and _how_
will they get the feature? I see the following cases
here, which might overlap ...

- the feature makes perfectly sense in mainline as
  standalone feature (maybe even adds no overhead
  and/or simplifies/generalizes design) but has no
  direct 'user' per se (think private namespaces
  or linux capabilities)

- the feature is used by a number of projects in
  very different ways to improve or even realize
  certain 'other' features (think ext2/3 xattrs)

- the feature (although it adds pretty much overhead
  and/or complicates the design) is really useful
  for everyday use, and most folks who discover it
  do not understand how they could live without it
  (think various attributes on vfs mounts :)

> I suspect this will be a difficult approach.

yes, but should 'we' decide to take this approach, we

can at least guarantee that we (Linux-VServer) will
try to make use of those new features as soon as they
appear in mainline (as we did until now)

> The third way would be to buffer it all up in -mm until everything
> is sufficiently in place and then slam it all in.

for me, that sounds like a pretty bad idea, at least
for the first steps -- though we might consider this
approach for the last 10 or 20 percent, when we just
have to put the pieces together ...

> That might not be feasible for various reasons - please advise..
>
> A fourth way would be for someone over there to run a git tree - you
> all happily work away, I redistribute it in -mm for testing and one
> day it's all ready to merge. I don't really like this approach. It
> ends up meaning that nobody else reviews the new code, nobody else
> understands what it's doing, etc. It's generally subversive of the way
> we do things.

let me say that I'm strictly against such an approach
as it would be very similar to merging any of the
existing projects without further mainline consideration

> Eric, Kirill, Herbert: let us know your thoughts, please.

thanks for your work and time, we appreciate it

best,
Herbert

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Andrey Savochkin on Fri, 19 May 2006 13:47:57 GMT
View Forum Message <> Reply to Message

Hi Andrew,

What you are saying indeed makes a lot of sense.
We want to start merging virtualization code some way or another.
Yet, if we merge code step-by-step, we do not want a pile of unused
infrastructure for the beginning, which may happen to be not entirely useful
in the future, or even create obstacles for development.
And in the course of merging, we would like to overcome differences in
opinions of various group, and live happily ever after.

I have a practical proposal.

We can start with presenting and merging the most interesting part, network containers. We discuss details, possible approaches, and related subsystems, until networking is finished to its utmost detail.
This will create an example of virtualization of a non-trivial subsystem, and we will have to agree on basic principles of virtualization of related subsystems like proc.

Virtualization of networking presents a lot of challenges and decision-making points with respect to user-visible interfaces: proc, sysctl, netlink events (and netlink sockets themselves), and so on. This code will also become immediately useful as an improvement over chroot.
I am sure that when we come to a mutually acceptable solution with respect to networking, virtualization of all other subsystems can be implemented and merged without many questions.

What do people think about this plan?

Best regards,

Andrey

On Thu, May 18, 2006 at 10:34:30AM -0700, Andrew Morton wrote:
[snip]
>
> All of which begs the question "now what?".
>
> What we do _not_ want to do is to merge up a pile of infrastructural stuff
> which never gets used. On the other hand, we don't want to be in a
> position where nothing is merged into mainline until the entirety of
> vserver &&/|| openvs is ready to be merged.
>
> I see two ways of justifying a mainline merge of things such as this
>
> a) We make an up-front decision that Linux _will_ have OS-virtualisation
> capability in the future and just start putting in place the pieces for
> that, even if some of them are not immediately useful.
>
> I suspect that'd be acceptable, although I worry that we'd get
> partway through and some issues would come up which are irreconcilable
> amongst the various groups.
>
> It would help set minds at ease if someone could produce a
> bullet-point list of what features the kernel will need to get it to the
> stage where "most or all vserver and openvz functionality can be
> implemented by controlling resource namespaces from userspace." Then we
> can discuss that list, make sure that everyone's pretty much in
> agreement.
>

> It would be good if that list were to identify which features are
> useful to Linux in their own right, and which ones only make sense within
> a whole virtualise-the-OS setup.
>
> b) Only merge into mainline those feature which make sense in a
> standalone fashion. eg, we don't merge this patchset unless the
> "per-process utsname namespace" feature is useful to and usable by a
> sufficiently broad group of existing Linux users.
>
> I suspect this will be a difficult approach.
>
> The third way would be to buffer it all up in -mm until everything is
> sufficiently in place and then slam it all in. That might not be feasible
> for various reasons - please advise..
>
> A fourth way would be for someone over there to run a git tree - you all
> happily work away, I redistribute it in -mm for testing and one day it's
> all ready to merge. I don't really like this approach. It ends up meaning
> that nobody else reviews the new code, nobody else understands what it's
> doing, etc. It's generally subversive of the way we do things.
>
> Eric, Kirill, Herbert: let us know your thoughts, please.

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Andrew Morton on Fri, 19 May 2006 15:13:34 GMT
View Forum Message <> Reply to Message

Herbert Poetzl <herbert@13thfloor.at> wrote:
>
> let me
> give a simple example here:

Examples are useful.

> "pid virtualization"
>
> - Linux-VServer doesn't really need that right now.
> we are perfectly fine with "pid isolation" here, we
> only "virtualize" the init pid to make pstree happy
>
> - Snapshot/Restart and Migration will require "full"
> pid virtualization (that's where Eric and OpenVZ
> are heading towards)

snapshot/restart/migration worry me. If they require complete
serialisation of complex kernel data structures then we have a problem,
because it means that any time anyone changes such a structure they need to

update (and test) the serialisation.

This may be a show-stopper, in which case maybe we only need to virtualise
pid #1.

> - OpenSSI and *Mosix require system wide pid spaces
> which probably could be implemented with virtual
> pid spaces as well
>
> - many security addons provide something called pid
> randomization, and I think they could probably
> benefit from a virtual pid space, too

ok.

Anyway. Thanks, guys. It sound like most of this work will be nicely
separable so we can think about each bit as it comes along.

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Andrew Morton on Fri, 19 May 2006 15:25:16 GMT
View Forum Message <> Reply to Message

Andrey Savochkin <saw@sw.ru> wrote:
>
> I have a practical proposal.
> We can start with presenting and merging the most interesting part, network
> containers. We discuss details, possible approaches, and related subsystems,
> until networking is finished to its utmost detail.
> This will create an example of virtualization of a non-trivial subsystem,
> and we will have to agree on basic principles of virtualization of related
> subsystems like proc.
>
> Virtualization of networking presents a lot of challenges and decision-making
> points with respect to user-visible interfaces: proc, sysctl, netlink events
> (and netlink sockets themselves), and so on. This code will also become
> immediately useful as an improvement over chroot.
> I am sure that when we come to a mutually acceptable solution with respect to
> networking, virtualization of all other subsystems can be implemented and
> merged without many questions.
>
> What do people think about this plan?

It sounds like that feature might be the
most-likely-to-cause-maintainer-revolt one, in which case yes, it is
absolutely definitely the one to start with.

Because if it ends up that an acceptable approach cannot be found, and if

this feature is compulsory for any sane virtualisation implementation then
that's it - game over. We want to discover such blockers as early in the
process as possible.

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Fri, 19 May 2006 16:27:32 GMT
View Forum Message <> Reply to Message

Andrew Morton <akpm@osdl.org> writes:

> Herbert Poetzl <herbert@13thfloor.at> wrote:
>>
>> let me
>> give a simple example here:
>
> Examples are useful.
>
>>   "pid virtualization"
>>
>>   - Linux-VServer doesn't really need that right now.
>>   we are perfectly fine with "pid isolation" here, we
>>   only "virtualize" the init pid to make pstree happy
>>
>>   - Snapshot/Restart and Migration will require "full"
>>   pid virtualization (that's where Eric and OpenVZ
>>   are heading towards)
>
> snapshot/restart/migration worry me. If they require complete
> serialisation of complex kernel data structures then we have a problem,
> because it means that any time anyone changes such a structure they need to
> update (and test) the serialisation.

There is a strict limit to what is user visible, and if it isn't user visible
we will never need it in a checkpoint. So internal implementation details
should not matter.

> This may be a show-stopper, in which case maybe we only need to virtualise
> pid #1.

Except we do need something for pid isolation, and a pid namespace is
quite possibly the light weight solution. If you can't see the pid it is
clearly isolated from you.

> Anyway. Thanks, guys. It sound like most of this work will be nicely
> separable so we can think about each bit as it comes along.

Yes, and there are enough issues it is significant.

Eric

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Andrew Morton on Fri, 19 May 2006 16:40:47 GMT
View Forum Message <> Reply to Message

ebiederm@xmission.com (Eric W. Biederman) wrote:
>
> > Herbert Poetzl <herbert@13thfloor.at> wrote:
> >>
> >> let me
> >> give a simple example here:
> > >
> > > Examples are useful.
> > >
> >> "pid virtualization"
> >>
> >> - Linux-VServer doesn't really need that right now.
> >> we are perfectly fine with "pid isolation" here, we
> >> only "virtualize" the init pid to make pstree happy
> >>
> >> - Snapshot/Restart and Migration will require "full"
> >> pid virtualization (that's where Eric and OpenVZ
> >> are heading towards)
> > >
> > > snapshot/restart/migration worry me. If they require complete
> > > serialisation of complex kernel data structures then we have a problem,
> > > because it means that any time anyone changes such a structure they need to
> > > update (and test) the serialisation.
> >
> > There is a strict limit to what is user visible, and if it isn't user visible
> > we will never need it in a checkpoint. So internal implementation details
> > should not matter.

Migration of currently-open sockets (for example) would require storing of
a lot of state, wouldn't it?

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Jeff Dike on Fri, 19 May 2006 17:52:08 GMT
View Forum Message <> Reply to Message

On Fri, May 19, 2006 at 05:41:45AM -0600, Eric W. Biederman wrote:
> I think I see a third way of justifying a mainline merge. We make an
> up-front decision that we will improve the existing chroot jail

> functionality in Linux and start making improvements. Even if some of
> the improvements are quite small.

FWIW, UML can use this stuff incrementally. So, from my point of view,
it's not an all-or-nothing thing.

Jeff

Subject: RE: [PATCH 0/9] namespaces: Introduction
Posted by Hua Zhong on Fri, 19 May 2006 18:28:08 GMT
View Forum Message <> Reply to Message

> snapshot/restart/migration worry me. If they require
> complete serialisation of complex kernel data structures then
> we have a problem, because it means that any time anyone
> changes such a structure they need to update (and test) the
> serialisation.

Checkpoint/Restart/Migration could be very complicated if done at OS level (per process/process group/or any subset of an OS). But
it is much simpler if done on virtual machine level (VMWare/Xen) because there is a natural and clear boundary, and doesn't get
affected if the OS kernel internal changes.

It's good to see some progress in supporting virtualization in Linux, but as Andrew put it, some big decisions need to be made
up-front. One big question is actually how many virtualization technologies Linux should support? Particularly, does it need to
support both OS-level virtualization and VM-level virtualization? And why? And to what degree?

My gut feeling is that the VM approach seems much cleaner and modular, without touching too many areas (except some low-level ones)
inside the kernel and in general very well separated. There are two reasons:

1. In a VM system, the architecture is very simple. Hypervisor and guest OS kernel have clear boundaries and interfaces to
communicate, and OS in general pretty much doesn't need to care if it's running on native hardware or inside a VM. So it adds very
little maintanence burden to the kernel developers (and if there is, it's relatively well understood).

2. Hardware support. With more virtualization built into CPU, VM is only going to get simper.

It seems at least the VM approach is much less risky. It might be helpful if someone could explain why we need both.

Hua

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Dave Hansen on Fri, 19 May 2006 20:04:49 GMT
View Forum Message <> Reply to Message

On Fri, 2006-05-19 at 08:13 -0700, Andrew Morton wrote:
> snapshot/restart/migration worry me. If they require complete
> serialisation of complex kernel data structures then we have a problem,
> because it means that any time anyone changes such a structure they need to
> update (and test) the serialisation.

The idea of actually serializing kernel data structures keeps me up at night. This is especially true when we already have some method of exporting these structures to userspace.

Take VMAs, for example. Should we have a set of interfaces for saving and restoring VMAs, or should we just make any program which is doing checkpoint/restart use /proc/<pid>/maps on checkpoint and mmap() on restore?

It, of course, isn't that simple. Any interface focused on VMAs inside the kernel will have the serialization issues you describe. I think this is such an approach:

http://git.openvz.org/?p=linux-2.6-openvz;a=blob;f=kernel/cpt/cpt_mm.c
http://git.openvz.org/?p=linux-2.6-openvz;a=blob;f=kernel/cpt/rst_mm.c

However, the proc-maps/mmap approach would require new interfaces to be implemented. There are plenty of attributes not currently readily visible to userspace like VM_NONLINEAR, or resources which are normally inaccessible to userspace like deleted files. Those would need extended user/kernel interfaces.

I know of at least one in-kernel commercial checkpoint/restart product which was relatively well tested with "a certain large DB that uses remap_file_pages()" that never even noticed that it completely missed VM_NONLINEAR support until vm-savvy people saw the code. Scary.

-- Dave

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Dave Hansen on Fri, 19 May 2006 20:17:12 GMT
View Forum Message <> Reply to Message

On Fri, 2006-05-19 at 09:40 -0700, Andrew Morton wrote:
> Migration of currently-open sockets (for example) would require storing of
> a lot of state, wouldn't it?

In a word, yes. :)

I don't think the networking guys from either the OpenVZ project or IBM were cc'd on this. Alexey, Daniel, can you elaborate, or point us to any existing code?

-- Dave

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Sam Vilain on Sat, 20 May 2006 00:16:04 GMT
View Forum Message <> Reply to Message

On Fri, 2006-05-19 at 17:47 +0400, Andrey Savochkin wrote:
> We can start with presenting and merging the most interesting part, network
> containers. We discuss details, possible approaches, and related subsystems,
> until networking is finished to its utmost detail.
> This will create an example of virtualization of a non-trivial subsystem,
> and we will have to agree on basic principles of virtualization of related
> subsystems like proc.
[...]
> What do people think about this plan?

Network is an interesting one because you have multiple solutions - the very simple approach of network binding (as used by Jacques Gelina's original IP vhost work from December 1997), and network virtualisation. That virtualisation itself can be broken down into providing merely virtual interfaces (so that, for instance, you can have independent lo interfaces in the virtual servers) as in Vserver 2.1.x, or providing a completely virtualised network stack, as in Vserver ngnet (and possibly OpenVZ?).

Each solution performs the virtualisation at a different level, and has incrementally higher orders of inefficiency and maintenance requirements. Yet none of them are essentially better or worse than the others.

So, we might end up with all three eventually - but binding alone is the simplest and still extremely useful.

Sam.

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Sam Vilain on Sat, 20 May 2006 00:16:11 GMT
View Forum Message <> Reply to Message

On Fri, 2006-05-19 at 05:41 -0600, Eric W. Biederman wrote:
> > It would help set minds at ease if someone could produce a
> > bullet-point list of what features the kernel will need to get it to the
> > stage where "most or all vserver and openvz functionality can be
> > implemented by controlling resource namespaces from userspace." Then we
> > can discuss that list, make sure that everyone's pretty much in
> > agreement.
> So this is slightly the wrong question. If you look at Sam's list you

Yes - the wrong question because it's too top down and encourages
hacks :) It's wrong for the purposes of planning an implementation, but
ok for easing minds about what will be covered, I think.

> will see that there are several independent dimensions to the complete
> solution. Most of them dealing with the increase in the number of users
> and the amount of work that is happening on a single kernel in this
> context.
>
> Basically we need to expect a lot of kernel tuning after we get the
> basics working.
>
> The proper question is: What needs to happen before we can run separate
> user space instances?

My guess would be most of the points under "isolation". The others are
really just fine tuning / resource partitioning and fixing various
things that break under virtualisation because of their design (eg,
quota).

Sam.

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Sat, 20 May 2006 03:18:29 GMT
View Forum Message <> Reply to Message

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Herbert Poetzl on Sat, 20 May 2006 21:24:55 GMT
View Forum Message <> Reply to Message

On Fri, May 19, 2006 at 08:25:16AM -0700, Andrew Morton wrote:
> Andrey Savochkin <saw@sw.ru> wrote:
>>
>> I have a practical proposal. We can start with presenting and

>> merging the most interesting part, network containers. We discuss
>> details, possible approaches, and related subsystems, until
>> networking is finished to its utmost detail. This will create an
>> example of virtualization of a non-trivial subsystem, and we will
>> have to agree on basic principles of virtualization of related
>> subsystems like proc.
>>
>> Virtualization of networking presents a lot of challenges and
>> decision-making points with respect to user-visible interfaces:
>> proc, sysctl, netlink events (and netlink sockets themselves),
>> and so on. This code will also become immediately useful as an
>> improvement over chroot. I am sure that when we come to a mutually
>> acceptable solution with respect to networking, virtualization of
>> all other subsystems can be implemented and merged without many
>> questions.
>>
>> What do people think about this plan?

well, I think it is interesting ...

> It sounds like that feature might be the
> most-likely-to-cause-maintainer-revolt one, in which case yes,
> it is absolutely definitely the one to start with.

yes, I absolutely agree here, this will be one
of the tougher nuts to crack, and therefore it
might be an excellent candidate to proove that
the different virtualization camps can find an
acceptable solution .. together.

> Because if it ends up that an acceptable approach cannot be found,
> and if this feature is compulsory for any sane virtualisation
> implementation then that's it - game over.

this, OTOH is something I'm not convinced of,
because looking at BSD jails, I see a very simple
approach (only one IP, limiting binds) which seems
to be sufficient for all the BSD jails out there

this is probably something which does not meet the
requirements of fully blown distro virtualizations
but actually it might be more than sufficient for
'mainline' linux jails

> We want to discover such blockers as early in the process as
> possible.

yes, I would also appreciate if we could get some

support from the network folks, as I think, most
of them are already working into that direction
(think Van Jacobson's net channels, routing tables)

especially as the network virtualization brings up
a number of questions, which are not easily answered
like the following:

- what policy will be applied inside guests?
  + allow arbitrary packets/rules/routes
  + have some generic limits/basic rules
  + put policy into userspace

- how to 'connect' the virtual interfaces to
  the real network?
  + via routing and bridging?
    (means duplicate stack traversal and
    therefore twice the overhead)
  + via split personality interfaces?
    (less overhead, more complicated cases)
  + directly (only by isolation)

- at what level should the virtualization happen?
  + ethernet level (all protocols)
  + ip level (all ip based and control protocols)
  + udp/tcp level

best,
Herbert

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Sun, 21 May 2006 00:48:25 GMT
View Forum Message <> Reply to Message

Dave Hansen <haveblue@us.ibm.com> writes:

> On Fri, 2006-05-19 at 08:13 -0700, Andrew Morton wrote:
>> snapshot/restart/migration worry me. If they require complete
>> serialisation of complex kernel data structures then we have a problem,
>> because it means that any time anyone changes such a structure they need to
>> update (and test) the serialisation.
>
> The idea of actually serializing kernel data structures keeps me up at
> night. This is especially true when we already have some method of
> exporting these structures to userspace.

Serialization of kernel data structures is a thorny issue, that

we are far enough away from that we don't need to tackle just yet.
I do consider it a failure to export/import things properly if you
need to use the same kernel version.

For the short term all that is interesting from a checkpoint/restart/migration
perspective is that you can have multiple instances of global identifiers,
pids, sysvipc ids, etc.

> However, the proc-maps/mmap approach would require new interfaces to be
> implemented. There are plenty of attributes not currently readily
> visible to userspace like VM_NONLINEAR, or resources which are normally
> inaccessible to userspace like deleted files. Those would need extended
> user/kernel interfaces.

Deleted files are accessible through /proc/<pid>/fd.

Eric

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Pavel Machek on Sun, 21 May 2006 22:57:37 GMT
View Forum Message <> Reply to Message

> Herbert Poetzl <herbert@13thfloor.at> wrote:
> >
> > let me
> > give a simple example here:
>
> Examples are useful.
>
> >   "pid virtualization"
> >
> >   - Linux-VServer doesn't really need that right now.
> >   we are perfectly fine with "pid isolation" here, we
> >   only "virtualize" the init pid to make pstree happy
> >
> >   - Snapshot/Restart and Migration will require "full"
> >   pid virtualization (that's where Eric and OpenVZ
> >   are heading towards)
>
> snapshot/restart/migration worry me. If they require complete
> serialisation of complex kernel data structures then we have a problem,
> because it means that any time anyone changes such a structure they need to
> update (and test) the serialisation.
>
> This may be a show-stopper, in which case maybe we only need to virtualise
> pid #1.

Well, if pid #1 virtualization is only needed for pstree, we may want
to fix pstree instead :-).
	Pavel
--
(english) http://www.livejournal.com/~pavelmachek
(cesky, pictures) http://atrey.karlin.mff.cuni.cz/~pavel/picture/horses/blog.h tml

---

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Mon, 22 May 2006 17:23:38 GMT
View Forum Message <> Reply to Message

Andrew Morton <akpm@osdl.org> writes:

> Andrey Savochkin <saw@sw.ru> wrote:
>>
>> I have a practical proposal.
>> We can start with presenting and merging the most interesting part, network
>> containers. We discuss details, possible approaches, and related subsystems,
>> until networking is finished to its utmost detail.
>> This will create an example of virtualization of a non-trivial subsystem,
>> and we will have to agree on basic principles of virtualization of related
>> subsystems like proc.
>>
>> Virtualization of networking presents a lot of challenges and decision-making
>> points with respect to user-visible interfaces: proc, sysctl, netlink events
>> (and netlink sockets themselves), and so on. This code will also become
>> immediately useful as an improvement over chroot.
>> I am sure that when we come to a mutually acceptable solution with respect to
>> networking, virtualization of all other subsystems can be implemented and
>> merged without many questions.
>>
>> What do people think about this plan?
>
> It sounds like that feature might be the
> most-likely-to-cause-maintainer-revolt one, in which case yes, it is
> absolutely definitely the one to start with.

It sounds like a case of: That first step is a doozy.
We should be able to resolve proc and sysctl issues with just
the uts namespace. So I don't think we necessarily have to take
everything at once.

Beyond that the real sticky issue and what leads to most of the peculiar
cases is the one thing not addressed by doing the network namespace.
How do we keep someone inside a namespace from accessing files in /proc
and other places that they should not be able to access.

It occurred to me that most of the permission checking issues trivially go away if you make the permission checks test for equality of the tuple (uid namespace, uid). At which point a lot of the reasons people have previously put forth for completely reorganizing proc and sysfs go away, because users not in their uid namespace will only be able to access world readable and world writable files. Anything else will simply be inaccessible.

So I think we need to have a serious look at the uid/gid namespace.

This is a bit of a pain because this brings us face to face with the uid mapping problem we have avoided for years on network filesystems, and makes it a problem on local filesystems as well.

Getting both the uid/gid namespace and the network namespace should get the bulk of the infrastructure problems solved.

I am even happy to do the network namespace first on the understanding that permission checking problems caused by different users with the same uid should be ignored until we have handled the uid/gid namespace.

Eric
