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.