

Hello,

We are happy to announce the release of a stable version of the OpenVZ software, located at <http://openvz.org/>.

OpenVZ is a kernel virtualization solution which can be considered as a natural step in the OS kernel evolution: after multiuser and multitasking functionality there comes an OpenVZ feature of having multiple environments.

Virtualization lets you divide a system into separate isolated execution environments (called VPSs - Virtual Private Servers). From the point of view of the VPS owner (root), it looks like a stand-alone server. Each VPS has its own filesystem tree, process tree (starting from init as in a real system) and so on. The single-kernel approach makes it possible to virtualize with very little overhead, if any.

OpenVZ in-kernel modifications can be divided into several components:

1. Virtualization and isolation.

Many Linux kernel subsystems are virtualized, so each VPS has its own:

- process tree (featuring virtualized pids, so that the init pid is 1);
- filesystems (including virtualized /proc and /sys);
- network (virtual network device, its own ip addresses, set of netfilter and routing rules);
- devices (if needed, any VPS can be granted access to real devices like network interfaces, serial ports, disk partitions, etc);
- IPC objects.

2. Resource Management.

This subsystem enables multiple VPSs to coexist, providing managed resource sharing and limiting.

- User Beancounters is a set of per-VPS resource counters, limits, and guarantees (kernel memory, network buffers, phys pages, etc.).
- Fair CPU scheduler (SFQ with shares and hard limits).
- Two-level disk quota (first-level: per-VPS quota; second-level: ordinary user/group quota inside a VPS)

Resource management is what makes OpenVZ different from other solutions of this kind (like Linux VServer or FreeBSD jails). There are a few resources that can be abused from inside a VPS (such as files, IPC objects, ...) leading to a DoS attack. User Beancounters prevent such abuses.

As virtualization solution OpenVZ makes it possible to do the same things for which people use UML, Xen, QEmu or VMware, but there are differences:

- (a) there is no ability to run other operating systems
(although different Linux distros can happily coexist);
- (b) performance loss is negligible due to absence of any kind of emulation;
- (c) resource utilization is much better.

The last point needs to be elaborated on. OpenVZ allows to utilize system resources such as memory and disk space very efficiently, and because of that has better performance on memory-critical workloads. OpenVZ does not run separate kernels in each VPS and saves memory on kernel internal data. However, even bigger efficiency of OpenVZ comes from dynamic resource allocation.

With other virtualization solutions, you need to specify in advance the amount of memory for each virtual machine and create a disk device and filesystem for it, and the possibilities to change settings later on the fly are very limited.

The dynamic assignment of resources in OpenVZ can significantly improve their utilization. For example, a x86_64 box (2.8 GHz Celeron D, 1GB RAM) is capable to run 100 VPSs with a fairly high performance (VPSs were serving http requests for 4.2Kb static pages at an overall rate of more than 80,000 req/min). Each VPS (running CentOS 4 x86_64) had the following set of processes:

```
[root@ovz-x64 ~]# vzctl exec 1043 ps axf
PID TTY STAT TIME COMMAND
  1 ? Ss 0:00 init
11830 ? Ss 0:00 syslogd -m 0
11897 ? Ss 0:00 /usr/sbin/sshd
11943 ? Ss 0:00 xinetd -stayalive -pidfile ...
12218 ? Ss 0:00 sendmail: accepting connections
12265 ? Ss 0:00 sendmail: Queue runner@01:00:00
13362 ? Ss 0:00 /usr/sbin/httpd
13363 ? S 0:00 \_ /usr/sbin/httpd
13364 ? S 0:00 \_ /usr/sbin/httpd
13365 ? S 0:00 \_ /usr/sbin/httpd
13366 ? S 0:00 \_ /usr/sbin/httpd
13370 ? S 0:00 \_ /usr/sbin/httpd
13371 ? S 0:00 \_ /usr/sbin/httpd
13372 ? S 0:00 \_ /usr/sbin/httpd
13373 ? S 0:00 \_ /usr/sbin/httpd
6416 ? Rs 0:00 ps axf
```

And the list of running VPSs:

```
[root@ovz-x64 ~]# vzlist
VPSID   NPROC STATUS IP_ADDR   HOSTNAME
1001    15 running 10.1.1.1   vps1001
1002    15 running 10.1.1.2   vps1002
[....skipped....]
1099    15 running 10.1.1.99   vps1099
1100    15 running 10.1.1.100  vps1100
```

On the box with 4Gb of RAM one can expect 400 of such VPSs to run without much troubles.

More information is available at <http://openvz.org/>

Thanks,
OpenVZ team.

Subject: Re: [ANNOUNCE] first stable release of OpenVZ kernel virtualization solution

Posted by [Andrew Morton](#) on Tue, 06 Dec 2005 03:20:53 GMT

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

Kirill Korotaev <dev@sw.ru> wrote:

>
> Hello,
>
> We are happy to announce the release of a stable version of the OpenVZ
> software, located at <http://openvz.org/>.
>
> OpenVZ is a kernel virtualization solution which can be considered as a
> natural step in the OS kernel evolution: after multiuser and
> multitasking functionality there comes an OpenVZ feature of having
> multiple environments.

Are you able to give us a high-level overview of how it actually is implemented? IOW: what does the patch do?

> ...
>
> As virtualization solution OpenVZ makes it possible to do the same
> things for which people use UML, Xen, QEmu or VMware, but there are
> differences:
> (a) there is no ability to run other operating systems
> (although different Linux distros can happily coexist);
> (b) performance loss is negligible due to absence of any kind of
> emulation;
> (c) resource utilization is much better.

What are OpenVZ's disadvantages wrt the above?

> The dynamic assignment of resources in OpenVZ can significantly improve
> their utilization. For example, a x86_64 box (2.8 GHz Celeron D, 1GB
> RAM) is capable to run 100 VPSs with a fairly high performance (VPSs
> were serving http requests for 4.2Kb static pages at an overall rate of
> more than 80,000 req/min). Each VPS (running CentOS 4 x86_64) had the
> following set of processes:

```
>
> [root@ovz-x64 ~]# vzctl exec 1043 ps axf
>  PID TTY          STAT TIME COMMAND
>   1 ?        Ss   0:00 init
> 11830 ?        Ss   0:00 syslogd -m 0
> 11897 ?        Ss   0:00 /usr/sbin/sshd
> 11943 ?        Ss   0:00 xinetd -stayalive -pidfile ...
> 12218 ?        Ss   0:00 sendmail: accepting connections
> 12265 ?        Ss   0:00 sendmail: Queue runner@01:00:00
> 13362 ?        Ss   0:00 /usr/sbin/httpd
> 13363 ?        S    0:00 \_ /usr/sbin/httpd
> 13364 ?        S    0:00 \_ /usr/sbin/httpd
> 13365 ?        S    0:00 \_ /usr/sbin/httpd
> 13366 ?        S    0:00 \_ /usr/sbin/httpd
> 13370 ?        S    0:00 \_ /usr/sbin/httpd
> 13371 ?        S    0:00 \_ /usr/sbin/httpd
> 13372 ?        S    0:00 \_ /usr/sbin/httpd
> 13373 ?        S    0:00 \_ /usr/sbin/httpd
> 6416 ?        Rs   0:00 ps axf
```

Do the various kernel instances share httpd text pages?

Subject: Re: [ANNOUNCE] first stable release of OpenVZ kernel virtualization solution

Posted by [dev](#) on Tue, 06 Dec 2005 11:48:38 GMT

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

Ingo, I added Andrey note, added our maillist to CC.

> in general, i think the biggest resource-isolation issue for big servers
> is lowmem. Do you have any ideas how to handle that within OVZ? The
> scenario is: one (relatively low-prio) instance starves another
> (high-prio) OVZ instance of lowmem, getting the system into a kswapd
> storm.
> Ingo

We have per VPS UBC (user beancounters) parameters called "kmemsize"
(almost all kernel structures are accounted into this - page tables,

vmas, etc.), tcprcvbuf, tcpsndbuf and others which allows to control VPS usage of low-mem.

i.e. if OpenVZ is configured appropriately such situation should not happen (provided host system is not highly overcommitted). Situation with overcommit happened on i386 >4GB RAM in 2.4 kernels. But as you remember 4GB split helped a lot in this case. In 2.6 kernels situation is really much better and out tests with high number of VPSs work correctly even without 4GB split.

Additional note from Andrey Savochkin:
some resources such as disk space can be highly overcommitted. Low-mem is special resource and correctly configured OpenVZ server should not have much overcommit for lowmem. This eliminates starvation on lowmem and kswapd storms.

Kirill

Subject: Re: [ANNOUNCE] first stable release of OpenVZ kernel virtualization solution

Posted by [dev](#) on Tue, 06 Dec 2005 11:55:24 GMT

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

Ingo Molnar wrote:

> * Kirill Korotaev <dev@sw.ru> wrote:

>

>

>>We have per VPS UBC (user beancounters) parameters called "kmemsize"

>>(almost all kernel structures are accounted into this - page tables,

>>vmas, etc.), tcprcvbuf, tcpsndbuf and others which allows to control

>>VPS usage of low-mem.

>>

>>i.e. if OpenVZ is configured appropriately such situation should not

>>happen (provided host system is not highly overcommitted). Situation

>>with overcommit happened on i386 >4GB RAM in 2.4 kernels. But as you

>>remember 4GB split helped a lot in this case. In 2.6 kernels situation

>>is really much better and out tests with high number of VPSs work

>>correctly even without 4GB split.

>

>

> interesting. Have you tested the corner case of: 'one lowprio VPS is

> swapping like mad', how it affects highprio VPSs?

Both VPSs have a single page cache. So if one VPS is swapping like hell,

it's neighbour is swapping as well. This naturally means that node is out of memory since you created overcommitted configuration.

It is up to you whether:

- to limit the offender

- kill the offender
- migrate the high-prio or low-prio VPS to another node
- add RAM :)

There are many possible actions here...

You can't travel with all your friends/relatives to the sea side in a car, yeah? You need either to have many cars or to take a bus. Up to you.

Kirill

Subject: Re: [ANNOUNCE] first stable release of OpenVZ kernel virtualization solution

Posted by [dev](#) on Tue, 06 Dec 2005 11:59:25 GMT

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

Ingo Molnar wrote:

> * Kirill Korotaev <dev@sw.ru> wrote:

>

>

>>Yes, we have a patch queue, though it is not available to outside yet :(
>>You can find patch-022stab0XX-core in SRC RPM, which includes the
>>following parts without driver updates (which are included in -combined
>>patch):

>>- mainstream fixes

>>- mainstream security fixes

>>- 4GB split (yours :) , patched by me)

>>- User beancounters (kernel/ub/*, include/ub/*). This includes

>>accounting and limiting of VPSs.

>>- Virtualization itself (ve_struct, kernel/vecalls.c - main code for VPS

>>start/stop, net/ipv4 - virtualization of TPC/IP and netfilters,

>>drivers/net/venet* - virtual network device for VPS, virtual pids, etc.)

>>- fs/simfs - simple filesystem to fake VPS and return correct values on
>>`df` and statfs() output.

>>- fs/vzdq* - 2-level disk quota.

>>- kernel/fairsched.c and kernel/sched.c - fair CPU scheduler.

>>

>>If you wish I can prepare these 8 patches for you a bit later.

>

>

> well ... in general for LKML review it's easier to have split up
> patches.

ok. we'll prepare these 8 patches. Not a problem.

>>Actually I think we'll start doing our developement in git after some
>>time.

>

> the -rt tree has a similar size:

>
> 799 files changed, 28782 insertions(+), 9714 deletions(-)
>
> and GIT isnt the best way for me, it's actually having a quilt
> repository of 110+ patches that works best for me. That way i can keep
> pushing stuff upstream, and have the queue ordered by 'likelihood of
> upstream merging' (putting the least likely items last). Quilt is also
> extremely fast. (faster than GIT doing equivalent stuff)
do you sync your tree and resolve the conflicts for each new kernel version?

> GIT is best if you are an upstream maintainer and want to sync stuff to
> Linus periodically. But it's not the best for separate trees.
We don't use quilt, but the developement way is the same actually:
we have a patch list with patches grouped by subsystem (mainstream,
virtualization, resource management), some of them go upstream, some
not, some are merged to avoid a lot of conflicts later.

Kirill

Subject: Re: [ANNOUNCE] first stable release of OpenVZ kernel virtualization
solution

Posted by [Ingo Molnar](#) on Tue, 06 Dec 2005 12:01:11 GMT

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

* Kirill Korotaev <dev@sw.ru> wrote:

> >interesting. Have you tested the corner case of: 'one lowprio VPS is
> >swapping like mad', how it affects highprio VPSs?

> Both VPSs have a single page cache. So if one VPS is swapping like hell,
> it's neighbour is swapping as well. This naturally means that node is
> out of memory since you created overcommitted configuration.
> It is up to you whether:
> - to limit the offender
> - kill the offender
> - migrate the high-prio or low-prio VPS to another node
> - add RAM :)

well, the other solution is to let certain instances overcommit
userspace RAM, and to just use a swap device for that scenario. An admin
can fix or shut down the offender, but other, more important instances
would still be up and running. Hard limits have other problems: they are
hard failures, instead of graceful failures.

by 'swapping madly' i dont mean lowmem pressure, but plain userspace VM
pressure. I fear it's not flexible enough to not allow for that. I.e. it
would be nice to extend the beancounters to let userspace to `_swap_`

instead of exposing it to a hard limit.

Ingo

Subject: Re: [ANNOUNCE] first stable release of OpenVZ kernel virtualization solution

Posted by [Ingo Molnar](#) on Tue, 06 Dec 2005 13:11:00 GMT

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

* Ingo Molnar <mingo@elte.hu> wrote:

> by 'swapping madly' i dont mean lowmem pressure, but plain userspace
> VM pressure. I fear it's not flexible enough to not allow for that.
> I.e. it would be nice to extend the beancounters to let userspace to
> _swap_ instead of exposing it to a hard limit.

maybe i'm banging on open doors, but the same would be the case not only
for userspace-VM overcommit, but also for dirty data. I.e. there should
be (already is?) a per-instance 'dirty data threshold', to not force
other instances into waiting for writeout/swapout to happen.

or does the OVZ method assume a totally swapless system?

Ingo

Subject: Re: first stable release of OpenVZ kernel virtualization solution

Posted by [Andrey Savochkin](#) on Tue, 06 Dec 2005 14:01:15 GMT

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

On Tue, Dec 06, 2005 at 02:11:20PM +0100, Ingo Molnar wrote:

>
> * Ingo Molnar <mingo@elte.hu> wrote:
>
> > by 'swapping madly' i dont mean lowmem pressure, but plain userspace
> > VM pressure. I fear it's not flexible enough to not allow for that.
> > I.e. it would be nice to extend the beancounters to let userspace to
> > _swap_ instead of exposing it to a hard limit.
>
> maybe i'm banging on open doors, but the same would be the case not only
> for userspace-VM overcommit, but also for dirty data. I.e. there should
> be (already is?) a per-instance 'dirty data threshold', to not force
> other instances into waiting for writeout/swapout to happen.

OVZ certainly has room for improvements with respect to swap.
What I want to point out is that swapout management is a complex task.

When a low-priority VPS exceeds its limits, it is not always beneficial for others to make it swap out: swapout wastes disk bandwidth, and to some extent CPU power. 'Dirty data threshold' could have helped, but it reduces the overall performance of the system, especially if the number of VPSs is small. Imagine only one VPS running: artificial 'dirty data threshold' would certainly be counter-productive.

Subject: Re: [ANNOUNCE] first stable release of OpenVZ kernel virtualization solution

Posted by [dev](#) on Tue, 06 Dec 2005 15:39:16 GMT

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

Hello Andrew,

>>We are happy to announce the release of a stable version of the OpenVZ
>>software, located at <http://openvz.org/>.

>>

>>OpenVZ is a kernel virtualization solution which can be considered as a
>>natural step in the OS kernel evolution: after multiuser and
>>multitasking functionality there comes an OpenVZ feature of having
>>multiple environments.

>

>

> Are you able to give us a high-level overview of how it actually is
> implemented? IOW: what does the patch do?

Will be glad to do so.

On the high-level the system looks like patched Linux Kernel with a number of user space tools. The kernel itself boots on a usual Linux distribution like RHEL4 and works as usual. But there are extensions which allow to create a new VPS context.

User space OpenVZ tools use these extensions to do the following, e.g. on VPS start:

- turn on and configure quota on VPS file system subtree.
- chroot to this filesystem tree.
- create a UBC context with configured resource limits/guarantees.
- create a VPS context and exec init in this newly created environment.
- newly spawned init executes VPS initscripts as if it was a usual Linux box which has switched power on.

There are two patches that can be found on OpenVZ site.

patch-022stabXXX-combined which is a single consolidated OpenVZ patch including driver and mainstream updates. And the more interesting one is patch-022stabXXX-core (available from the SRC RPM), which itself consists of the following parts:

- mainstream fixes and security fixes
- 4GB split

- User beancounters (kernel/ub/*, include/ub/*). This includes accounting and limiting/guarantees of resources.
- Virtualization itself (ve_struct, kernel/vecalls.c - main code for VPS start/stop, net/ipv4 - virtualization of TCP/IP and netfilters, drivers/net/venet* - virtual network device for VPS, virtual pids, etc.)
- fs/simfs - simple filesystem to fake VPS and return correct values on `df` and statfs() output.
- fs/vzdq* - 2-level disk quota.
- kernel/fairsched.c and kernel/sched.c - fair CPU scheduler (SFQ-like scheduler with shares and hardlimits)

We strongly believe that Linux kernel can benefit from Virtualization and Resource Management very much, so are very interested in your and Linus comments on this. Though virtualization and resource management is not a full feature set provided by OpenVZ, they can be used for creation of more secure environments where untrusted users are involved. For example, with virtualization it is possible to isolate set of processes on the node, web server or other application, thus vulnerabilities in this application won't allow to destroy the whole system (e.g. mail server running on the same node), install some backdoors/trojans (in kernel modules) etc. Resource control can be used for the same purposes when deliberate DoS or bugs in applications should not crash the whole system down. Virtualization also allows to do simple backups/restore/migration of VPSs between nodes, thus making maintenance much easier. So the I would summary up:

- Virtualization helps to isolate service, improves manageability
- Resource Management allows to control resources and prevent DoS from untrusted users in multi-user environments. User beancounters can be used for usual users instead of VPSs.

Main user space tools for OpenVZ are:

- vzctl, which is used for most of high-level VPS operations like VPS creation, start/stop, destroy, configuring, setting UBC and other parameters etc.

```
# vzctl create VPSID
```

is used to create VPS.

After VPS creation it can be started via issuing:

```
# vzctl start VPSID
```

To see set of processes inside VPS one can execute:

```
# vzctl exec VPSID ps axf
```

And the most interesting command is 'enter' which allows to get to VPS (to 'enter') from host system via changing context to VPS one:

```
# vzctl enter VPSID
```

```
bash#
```

- vzquota, a tool used for 2level quota support. Allows to turn quota on/off, recalculate it etc.

- vzpkg, a set tool of tools allowing to easily manage VPS templates (redhat, centos, fedora etc.).

>>As virtualization solution OpenVZ makes it possible to do the same
>>things for which people use UML, Xen, QEmu or VMware, but there are
>>differences:

>>(a) there is no ability to run other operating systems

>> (although different Linux distros can happily coexist);

>>(b) performance loss is negligible due to absense of any kind of

>> emulation;

>>(c) resource utilization is much better.

>

> What are OpenVZ's disadvantages wrt the above?

disadvantages:

- unable to run Windows or xBSD on OpenVZ Linux.

- VPS owner can't load/use custom kernel modules

- theoretically stability of such solution has one single point of failure - the kernel. This is mitigated by lots of auto (stress) tests done by us to be sure the kernel is stable (we have more than >200 mainstream patches due to this). The stability is the main goal here, since servers running multiple VPSs work under much higher load and the cost of kernel oops is much higher.

- in some respects dealing with files and processes is harder than with VM which is bounded to it's memory, CPU and disk state, e.g. when doing backups. However, there are many cases when this is rather an advantage, e.g. when you are able to access VPS files and processes from the host system and do some management actions when VPS itself is stuck or performs poorly and no remote access is available.

>>The dynamic assignment of resources in OpenVZ can significantly improve
>>their utilization. For example, a x86_64 box (2.8 GHz Celeron D, 1GB
>>RAM) is capable to run 100 VPSs with a fairly high performance (VPSs
>>were serving http requests for 4.2Kb static pages at an overall rate of
>>more than 80,000 req/min). Each VPS (running CentOS 4 x86_64) had the
>>following set of processes:

[skipped]

> Do the various kernel instances share httpd text pages?

Please note, there is a single kernel instance for all VPSs! This means common page cache etc.

But in this particular example each VPS has it's own file tree and it's own httpd instance in memory with full set of pages (.text, .data, ...), i.e. there were running 100 VPSs without any page sharing involved.

"User Beancounters" accounting is implemented with page sharing in mind and will be correct in this case (i.e. only page fraction will be charged). With page sharing this example scales well up to 200+ VPSs. And sharing itself can be achieved in multiple ways: common part of file system tree (e.g. /lib, /usr, etc.) or via a special filesystem.

Kirill

Subject: Re: [ANNOUNCE] first stable release of OpenVZ kernel virtualization solution

Posted by [Philippe Pegon](#) on Sat, 10 Dec 2005 11:22:59 GMT

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

Hello,

By curiosity, what is the status for IPv6 in OpenVZ (I saw that it was in the roadmap on the website, but maybe you have more informations) ?

thanks

--

Philippe Pegon

Kirill Korotaev wrote:

> Hello,

>

> We are happy to announce the release of a stable version of the OpenVZ software, located at <http://openvz.org/>.

>

> OpenVZ is a kernel virtualization solution which can be considered as a natural step in the OS kernel evolution: after multiuser and multitasking functionality there comes an OpenVZ feature of having multiple environments.

>

> Virtualization lets you divide a system into separate isolated execution environments (called VPSs - Virtual Private Servers). From the point of view of the VPS owner (root), it looks like a stand-alone server. Each VPS has its own filesystem tree, process tree (starting from init as in a real system) and so on. The single-kernel approach makes it possible to virtualize with very little overhead, if any.

>

> OpenVZ in-kernel modifications can be divided into several components:

>

> 1. Virtualization and isolation.

> Many Linux kernel subsystems are virtualized, so each VPS has its own:

> - process tree (featuring virtualized pids, so that the init pid is 1);

> - filesystems (including virtualized /proc and /sys);

- > - network (virtual network device, its own ip addresses,
- > set of netfilter and routing rules);
- > - devices (if needed, any VPS can be granted access to real devices
- > like network interfaces, serial ports, disk partitions, etc);
- > - IPC objects.
- >
- > 2. Resource Management.
- > This subsystem enables multiple VPSs to coexist, providing managed
- > resource sharing and limiting.
- > - User Beancounters is a set of per-VPS resource counters, limits,
- > and guarantees (kernel memory, network buffers, phys pages, etc.).
- > - Fair CPU scheduler (SFQ with shares and hard limits).
- > - Two-level disk quota (first-level: per-VPS quota;
- > second-level: ordinary user/group quota inside a VPS)
- >
- > Resource management is what makes OpenVZ different from other solutions
- > of this kind (like Linux VServer or FreeBSD jails). There are a few
- > resources that can be abused from inside a VPS (such as files, IPC
- > objects, ...) leading to a DoS attack. User Beancounters prevent such
- > abuses.
- >
- > As virtualization solution OpenVZ makes it possible to do the same
- > things for which people use UML, Xen, QEmu or VMware, but there are
- > differences:
- > (a) there is no ability to run other operating systems
- > (although different Linux distros can happily coexist);
- > (b) performance loss is negligible due to absense of any kind of
- > emulation;
- > (c) resource utilization is much better.
- >
- > The last point needs to be elaborated on. OpenVZ allows to utilize
- > system resources such as memory and disk space very efficiently, and
- > because of that has better performance on memory-critical workloads.
- > OpenVZ does not run separate kernels in each VPS and saves memory on
- > kernel internal data. However, even bigger efficiency of OpenVZ comes
- > from dynamic resource allocation.
- >
- > With other virtualization solutions, you need to specify in advance the
- > amount of memory for each virtual machine and create a disk device and
- > filesystem for it, and the possibilities to change settings later on the
- > fly are very limited.
- >
- > The dynamic assignment of resources in OpenVZ can significantly improve
- > their utilization. For example, a x86_64 box (2.8 GHz Celeron D, 1GB
- > RAM) is capable to run 100 VPSs with a fairly high performance (VPSs
- > were serving http requests for 4.2Kb static pages at an overall rate of
- > more than 80,000 req/min). Each VPS (running CentOS 4 x86_64) had the
- > following set of processes:

```

>
> [root@ovz-x64 ~]# vzctl exec 1043 ps axf
> PID TTY      STAT   TIME COMMAND
>  1 ?        Ss     0:00 init
> 11830 ?      Ss     0:00 syslogd -m 0
> 11897 ?      Ss     0:00 /usr/sbin/sshd
> 11943 ?      Ss     0:00 xinetd -stayalive -pidfile ...
> 12218 ?      Ss     0:00 sendmail: accepting connections
> 12265 ?      Ss     0:00 sendmail: Queue runner@01:00:00
> 13362 ?      Ss     0:00 /usr/sbin/httpd
> 13363 ?      S      0:00 \_ /usr/sbin/httpd
> 13364 ?      S      0:00 \_ /usr/sbin/httpd
> 13365 ?      S      0:00 \_ /usr/sbin/httpd
> 13366 ?      S      0:00 \_ /usr/sbin/httpd
> 13370 ?      S      0:00 \_ /usr/sbin/httpd
> 13371 ?      S      0:00 \_ /usr/sbin/httpd
> 13372 ?      S      0:00 \_ /usr/sbin/httpd
> 13373 ?      S      0:00 \_ /usr/sbin/httpd
> 6416 ?      Rs     0:00 ps axf
>
> And the list of running VPSs:
>
> [root@ovz-x64 ~]# vzlist
>  VPSID   NPROC STATUS IP_ADDR   HOSTNAME
>   1001    15 running 10.1.1.1   vps1001
>   1002    15 running 10.1.1.2   vps1002
>   [...skipped....]
>   1099    15 running 10.1.1.99   vps1099
>   1100    15 running 10.1.1.100  vps1100
>
> On the box with 4Gb of RAM one can expect 400 of such VPSs to run
> without much troubles.
>
> More information is available at http://openvz.org/
>
> Thanks,
> OpenVZ team.
>
>
> -
> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html
> Please read the FAQ at http://www.tux.org/lkml/

```

Subject: Re: [ANNOUNCE] first stable release of OpenVZ kernel virtualization

solution

Posted by [dim](#) on Sat, 10 Dec 2005 12:18:48 GMT

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

Hello,

Current status is 'not virtualized'. There were not so many requests to implement this, but probably we'll change this task priority.

> Hello,

>

> By curiosity, what is the status for IPv6 in OpenVZ (I saw that it was
> in the roadmap on the website, but maybe you have more informations) ?

>

> thanks

> --

> Philippe Pegon

--

Thanks,
Dmitry.

Subject: broken-out patches

Posted by [dev](#) on Mon, 12 Dec 2005 13:21:40 GMT

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

Dave, Ingo, we splitted patches for you as promised before:

- mainstream fixes and security fixes
diff-openvz-ms-20051212

- 4GB split (yours :) , patched by me)
diff-openvz-4gb-20051212

- User beancounters (kernel/ub/*, include/ub/*).
This includes accounting and limiting of VPSs.
diff-openvz-ubc-20051212

- Fair CPU scheduler, kernel/fairsched.c and kernel/sched.c
diff-openvz-fairsched-20051212

- Virtualization itself (ve_struct, kernel/vecalls.c -
main code for VPS start/stop, net/ipv4 - virtualization of TPC/IP and
netfilters, drivers/net/venet* - virtual network device for VPS,
virtual pids, etc.)
diff-openvz-ve-20051212-core

- fs/simfs - simple filesystem to fake VPS and return correct values on `df` and statfs() output.

diff-openvz-simfs-20051212

- fs/vzdq* - 2-level disk quota.

diff-openvz-vzdq-20051212

All this patches can be taken from:

<http://download.openvz.org/kernel/broken-out/022stab053.1/>

Kirill
