Subject: [ANNOUNCE] first stable release of OpenVZ kernel virtualization solution
Posted by dev on Mon, 05 Dec 2005 19:13:25 GMT
View Forum Message <> Reply to Message

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.