OpenVZ Linux Containers

OpenVZ is a containers-type virtualization solution built on Linux. It creates multiple isolated, secure *containers* (otherwise known as VEs or VPSs) on a single physical server, enabling better utilization and ensuring that applications do not conflict. Each container performs and executes exactly like a stand-alone server; containers can be rebooted independently. With OpenVZ, you can create up to several hundreds containers on a single physical server.

Containers can be thought of as lightweight Virtual Machines. OpenVZ requires both the host and guest OS to be Linux (and Linux distributions can be different in different containers). There is only a 1–3% performance penalty for OpenVZ as compared to using a standalone server.

OpenVZ is free software; everyone can use, redistribute and modify it under the terms of the GNU General Public License.

OpenVZ consists of a modified Linux kernel plus user-level tools. The kernel adds a notion of containers, and provides virtualization, isolation and resource management, as well as checkpointing and live migration.

Virtualization and Isolation

Each container has its own:

Files (system libraries, applications, `/proc` and `/sys`, file locks) ;

Processes (separate process tree start from init with the PID of 1);

Users and groups (including root);

Networking (own virtualized network devices, IP addresses, set of routing and netfilter (iptables) rules);

IPC objects (shared memory, semaphores, messages);

…and much more.

Resource Management

Kernel shares and limits containers' resources, so no single container can abuse system resources. The three main subsystems are:

User Beancounters. A set of per-container resource counters, limits, and guarantees. There is a set of about 20 parameters which is meant to control all the aspects of container operation. This is to prevent a single container from monopolizing system resources. Resources such as kernel memory, network buffers, physical and virtual memory pages etc. are accounted. This can be thought of as an advanced per-container ulimit.

Fair CPU scheduler. Balances the CPU time between containers according to the priorities assigned so no container can abuse the CPU. Can be used to provide hard CPU limits and guarantees.

I/O scheduler. Distributes the available I/O bandwidth between containers according to the priorities assigned. Provides detailed statistics of containers' I/O activity.

Two-level disk quota. First level is per-container disk quota, second level is the standard UNIX per-user and per-group disk quota inside a container.

Live Migration and Checkpointing

OpenVZ kernel is able to freeze and save the complete state of a container into a dump file (the process is known as *checkpointing*). This state can then be restored from the dump file, and then container continues to run. The process is similar to suspend-to-disk on a notebook, the difference is OpenVZ only checkpoints a single container, not the whole system.

The container can also be restored on a different physical server. This enables so-called live migration: there is no need to reboot, so, from a user's perspective, this looks like a delay in response, not a downtime.

User-level Tools

vzctl is a high-level command line tool to control a container. It is used to create/start/stop/delete a container, and to set various container parameters, such as IP addresses, user beancounter limits, CPU shares, disk quotas etc.

Typical vzctl commands:

```
# vzctl create 101 --ostemplate centos-5
# vzctl set 101 --name foo --save
# vzctl set foo --ipadd 10.1.2.2 --save
# vzctl set foo --diskspace 2G --save
# vzctl set foo --numproc 200 --save
# vzctl start foo
# vzctl exec foo ps ax
# vzctl enter foo
# vzctl stop foo
# vzctl destroy foo
```

vzlist is a tool to list containers and their parameters. It has a lot of options and thus can be used from within scripts to automate tasks.

vzsplit is used to create a container configuration suitable to run N containers on a given physical server.

vzmigrate is a script which performs migration (both live and offline).

vzmemcheck shows the total physical server resources utilization, and is used to plan hardware capacity and check for over-commitment.

Templates

Templates are just pre-packaged container images of various Linux distributions used for rapid container deployment. You can use existing pre-created template, modify those, or build your own templates that suits your particular needs.

It is easy to create your own template for OpenVZ – basically, you have to install a consistent set of packages that forms the base of operating system userland. This can be done with th help of utilities such as yum or debootstrap, depending on the distribution.

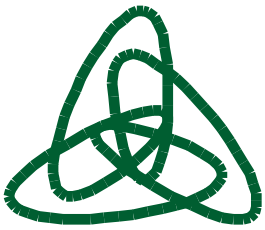
The following precreated templates are available:

- **CentOS** 4 and 5
- **Debian** 3.1, 4 and 5
- **Fedora** 7, 8, 9, 10
- **openSUSE** 10.3 and 11.1
- **Ubuntu** 7.10, 8.04, 8.10 and 9.04

Contributed:

- **ALTLinux**
- **ArchLinux**
- **Gentoo**
- **Slackware**
- etc.

http://openvz.org/



OpenVZ
Linux Containers

Frequently Asked Questions (with answers!)

What is a container (Virtual Environment, Virtual Private Server, VPS, VE)?
A container (CT) is an isolated entity which performs and executes exactly like a stand-alone server. Container can be rebooted independently and have root access, users/groups, IP address(es), memory, processes, files, applications, system libraries and configuration files. OpenVZ allows to have multiple CTs (up to as many as several hundreds) on a single physical server.

What are the highlights of OpenVZ technology?
In short, OpenVZ is highly scalable virtualization technology for Linux with near-zero overhead, strong isolation and rapid customer provisioning that's ready for production use right now. Deployment of OpenVZ improves efficiency, flexibility and quality of service in the enterprise environment.

How is OpenVZ different from other technologies? What are the pitfalls?
Unlike Virtual Machines or paravirtualization, OpenVZ is only using a single operating system kernel – the Linux kernel. That makes it fast and efficient, but incapable of running any other operating systems.

Who needs OpenVZ? How it can be used?
See **Use Cases** at the right.

What applications can run inside an OpenVZ container?
Most of applications can be installed to a container without any modifications. Oracle, DB/2, Weblogic, Websphere and other big applications run just fine inside an OpenVZ container. Applications and services do not have to be aware of OpenVZ. However, direct access to hardware is not available by default.

How scalable is OpenVZ?
OpenVZ technology scales up pretty well – we've tested it on machines with 16 CPUs and 64 GB of RAM. Besides, a single container could be scaled up from taking a little fraction of available resources up to all resources available dynamically – you do not even have to restart the container.

How OpenVZ improves efficiency of services?
For existing hardware, OpenVZ allows to utilize its processing power better by improving average load from 3-5% to at least 30-50%, while still providing ability to handle peak loads. And when its time to buy new servers, you can now use few more powerful servers instead of many little ones – with added benefits of better reliability, better peak performance and typically longer lifespan.

How OpenVZ improves flexibility of services?
Each container is hardware independent and can be moved to another OpenVZ-based system in seconds over the network. This allows for ease of hardware maintenance (move out all containers and do whatever you need with the box) and improved availability (keep a synchronized copy of your container elsewhere and start it up when primary service failed). If your old box is not able to cope with peak load anymore, just live migrate your containers to a new one.

What is a performance overhead?
Near zero. There is no emulation layer, only security isolation, and all checking is done on the kernel level without context switching.

What are performance expectations?
Peak performance is achieved when only one container has active tasks. In this case, it could use 100% of available resources: all CPUs, all physical memory, all disk and network bandwidth. OpenVZ is nt limiting you to a single-CPU virtual machine.

Are there any control panels available for OpenVZ?
See http://wiki.openvz.org/Control_panels

Where do I get (or put) more answers?
OpenVZ wiki is your friend. See http://wiki.openvz.org/

Use cases

Server Consolidation

- Uniform management
- Easy to upgrade
- More scalable
- Fast migration
- Save electricity/rack space

Development and Testing

- Different distros can co-exist
- A container can be created in a minute
- Can have hundreds of containers
- Cloning, snapshots, rollbacks
- A container is a sandbox: work/play, no fear

Security

- Give each app its own isolated container
- Security hole in an app will not affect others
- Plus: dynamic resource management

Hosting

- Isolated users
- A container is like a real server, just cheaper
- Much easier to admin

Educational

- Every student can have root access
- Different distributions
- No need for a lot of hardware

Other Container Technologies

- Parallels Virtuozzo Containers
  (a commercial version of OpenVZ)
- Linux-VServer
- FreeBSD jails
- Solaris 10 Containers
- AIX 6 Workstation Partitions (WPARs)

http://openvz.org/