OpenVZ Linux Containers

OpenVZ is a lightweight virtualization solution built on Linux. It creates multiple isolated, secure *containers* (an improved chroot providing a complete virtual environment) on a single physical server. Each container acts as a separate virtual machine, with its own process IDs, devices, network addresses and routing, and adjustable resource limits. OpenVZ can create hundreds of containers on a single physical server, each of which may be rebooted independently.

Because OpenVZ uses a chroot-based mechanism to provide lightweight virtual machines, both the host and guest OS must be Linux (although each container may run a different Linux distribution). Using containers imposes only a 1-3% performance penalty compared to running the same processes on the host system.

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, provides virtualization, isolation, resource management, checkpointing, and live migration.

Virtualization and Isolation

Each container has its own independent:

- **Files** - system libraries, applications, `/proc` and `/sys`, file locks
- **Processes** - each container has its own PID 1 init
- **Users and groups** - including `root with its own UID 0`
- **Networking** - virtualized network devices, IP addresses, per-container routing and iptables rules
- **IPC objects** - shared memory, semaphores, messages

... and more – everything that makes it feel like a dedicated system.

Resource Management

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

User Beancounters. An advanced per-container ulimit. A set of per-container resource counters, limits, and guarantees (about 20 parameters) meant to prevent a single container from monopolizing system resources and guarantee some minimal share. Resources such as kernel memory, network buffers, physical and virtual memory pages etc. are accounted.

Fair CPU scheduler. Balances 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 available I/O bandwidth between containers according to assigned priorities, with detailed statistics of 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 can freeze/save the complete state of a container into a dump file (a process known as *checkpointing*), then create a new container from this dump file. This 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, allowing *live migration* which doesn't interrupt existing user sessions.

User-level Tools

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

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 that can be used from within scripts to automate tasks.

vzsplit creates a container configuration suitable to run *N* containers on a given physical server.

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

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

Templates

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

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

Precreated templates are available for:

- **CentOS**
- **Debian**
- **Fedora**
- **Ubuntu**
- **openSUSE**
- **ALTLinux**
- **ArchLinux**
- **Gentoo**
- **Slackware**
- **Owl**
- etc.

http://openvz.org/

OpenVZ
Linux Containers

Frequently Asked Questions (with answers!)

What is a container (Virtual Environment, Virtual Private Server)?
A container (CT) is an isolated entity which performs and executes exactly like a stand-alone server. Containers can be rebooted independently and have root access, users/groups, IP address(es), memory, processes, files, applications, system libraries and configuration files.

What are the highlights of OpenVZ technology?
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?
Virtual Machines boot separate kernels on emulated hardware instances. OpenVZ runs all containers under a single Linux kernel. OpenVZ offers much higher density, hosting thousands of containers on a single physical server, but can only run Linux in those containers. Virtual machine solutions usually top out at a few dozen instances, but can run different operating systems in each.

What is the relationship between OpenVZ and LXC?
OpenVZ develops new container technology that then goes upstream into the vanilla Linux kernel. OpenVZ has about a 5 year headstart on LXC, but is actively feeding technology upstream into vanilla containers. Several internal details currently differ (OpenVZ adds new system calls, vanilla uses the cgroups filesystem, new clone flags, and other mechanisms).

What applications can run inside an OpenVZ container?
Applications and services do not have to be aware of OpenVZ, and most install without any modifications: Java, Oracle, DB/2, Weblogic, Websphere and many other big applications run just fine inside OpenVZ containers. However, direct access to hardware is not available by default; if required it must be provided by the system administrator.

How scalable is OpenVZ?
OpenVZ scales as well as Linux: we've tested 64 CPUs with 128 GB of RAM. It scales down to embedded devices like smart phones or plug computers. A single container can dynamically scale from taking a tiny fraction to all available resources, and may be adjusted without restarting it.

How does OpenVZ improve efficiency?
OpenVZ improves utilization of existing hardware by increasing average load while still providing the ability to handle peak loads. When buying new servers, using a few powerful boxes instead of many little ones allows better reliability, better peak performance and typically longer lifespan.

How does OpenVZ improve flexibility of services?
Each container is hardware independent, and can be moved to another OpenVZ-based system over the network in seconds. This eases hardware maintenance (move out all containers and do whatever you need with the box) and improves availability (keep a synchronized copy of your container elsewhere and start it up if primary service fails). When your old box can no longer cope with peak load, live migrate your containers to a new one.

What is the performance overhead?
Near zero. There is no emulation layer, only security isolation and resource accounting. All checking is done in the kernel without context switching.

Is there a GUI or web based control panel available?
Yes, 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
- Dynamic resource management controls runaway processes

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

Recently added features

- VSwap: two limits (RAM/swap) instead of setting 20 parameters
- Containers CPU binding (cpumask)
- PCI device delegation
- NFS mount migration
- Journaled per-container quota
- ext4 safe writeback mode
- Rebased on RHEL 6 kernel (better scalability to high-end SMP)

http://openvz.org/