OpenVZ

Server Virtualization Open Source Project

Introduction

OpenVZ is an operating system-level server virtualization solution, built on Linux. It creates many isolated, secure virtual environments (VEs, also known as virtual private servers or VPSs) on a single physical server enabling better server utilization and ensuring that applications do not conflict. Each VE performs and executes exactly like a stand-alone server; VEs can be rebooted independently and have root access, users, IP addresses, memory, processes, files, applications, system libraries and configuration files.

With OpenVZ, you can create up to several hundreds VEs on a single box (see chart at right).

Technical Background

OpenVZ consists of a modified Linux kernel plus user-level tools. OpenVZ kernel adds a notion of virtual environments. It provides virtualization, isolation and resource management.

Virtualization and Isolation

Each VE has its own:
- Files (system libraries, applications, virtualized /proc and /sys, file locks)
- Process tree (featuring virtualized PIDs)
- Network (own virtualized network device, IP addresses, set of routing and netfilter (iptables) rules)
- Devices (if needed, a VE can be granted an access to the real devices like network interfaces, disk partitions, serial ports etc.)
- IPC (inter-process comminucation) objects (shared memory, semaphores, messages)

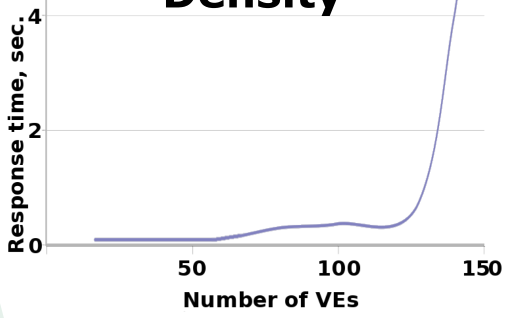
Resource Management

Kernel shares and limits VE resources, so no VE can abuse system resources. The three main subsystems are:
- User Beancounters. A set of per-VE resource counters, limits, and guarantees. Resources such as kernel memory, network buffers, physical and virtual memory pages etc. are accounted. This can be thought of as a very advanced ulimit.
- Fair CPU scheduler. It balances the CPU time between VEs so no VE can abuse the CPU, and can be used to provide CPU share limits and guarantees.
- Two-level disk quota. First level is per-VE disk quota, second level is a standard UNIX per-user and per-group disk quota inside a VE.

Zero Downtime (Live) Migration and Checkpointing

Kernel freezes and saves the complete state (checkpoints) of a VE on one physical server, then moves and restores it on the other, preserving everything, including open network connections. So, from the user's point of view it looks like a delay in processing, not downtime.

Density



The above graph shows the dependency of response time on the number of VEs running. Response time of of Apache web servers on a box with 768 megabytes (3/4 GB) of RAM was measured. In addition to Apache, each VE was running init, syslogd, sendmail, sshd and cron. Response time below one second is considered to be normal. With more than 120 VEs response time becomes unacceptable because of excessive swapping. Estimation is that on the same box

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 VE can be created in a minute
- Can have hundreds of VEs
- Cloning, snapshots, rollbacks
- A VE is a sandbox: work/play, no fear

Security

- Give each app own isolated VE
- Security hole in an application will not affect others
- Dynamic resource management

Hosting

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

Educational

- Every student can have root access
- Different distributions

http://openvz.org/