Linux Virtualization Technology Alternatives

Kir Kolyshkin <kir@openvz.org>  
OpenVZ project manager

Werner Fischer <wfischer@thomas-krenn.com>  
Thomas-Krenn.AG cluster solutions developer
Agenda

- Ways to Virtualize
- OpenVZ Details
- Usage Szenarios
- Current Status
Ways to Virtualize

- 1/4: Hardware emulation
- 2/4: Hardware virtualization
- 3/4: Paravirtualization
- 4/4: Virtualization on the OS level
- Multi-server virtualization
1/4: Hardware emulation

CPU and other components like chipset, I/O cards, ... are emulated

Examples:
- Bochs
- QEMU
- PearPC
- Hercules
- MS Virtual PC for Mac (PPC)
1/4: Hardware emulation

- guest OS does not need to be modified
- eg. emulation of PowerPC, ARM, SPARC, MIPS, ... on a x86 machine
- CPU emulation translates hardware instructions from the emulated to the native CPU
  -> overhead, performance loss
2/4: Hardware virtualization

CPU is NOT emulated, other components like chipset, I/O cards, ... are emulated

Examples:

- VMware Workstation/Server
- Parallels Desktop
- MS Virtual PC/Server
- VirtualBox
- XEN 3 mit Intel-VT/AMD-V CPU
- KVM mit Intel-VT/AMD-V CPU
2/4: Hardware virtualization

- guest OS does not need to be modified
- CPU arch of host = CPU arch of guest
- better performance than hardware emulation
3/4: Paravirtualization

No emulation, Host provides a special API for accessing the hardware

Examples:
- XEN 3
- User Mode Linux

Future:
- VMware Technology Preview (Virtual Machine Interface, VMI)
- Windows Server Virtualization (hypervisor-based)
3/4: Paravirtualization

- guest OS MUST be modified
- ABI (Application Binary Interface) stays unchanged
- CPU arch of host = CPU arch of guest

<table>
<thead>
<tr>
<th>Apps</th>
<th>Apps</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>modified Guest OS</td>
<td>modified Guest OS</td>
<td></td>
</tr>
</tbody>
</table>

Virtual Machine Monitor (VMM), aka Hypervisor

Hardware
4/4: Virtualization on the OS level

No emulation, one single Kernel for both host and guests

Examples:
- Linux-VServer
- OpenVZ/Virtuozzo
- Sun Solaris Container
- FreeBSD Jails
4/4: Virtualization on the OS level

- OS Host = OS guests (e.g. “only” Linux on Linux)
- very low overhead, since there is no emulation, and syscalls only need to go through one and not two Kernels
- guests are booted in seconds!
## Ways to Virtualize: summary

<table>
<thead>
<tr>
<th></th>
<th>Hardware Emulation</th>
<th>Hardware Virtualization</th>
<th>Para-virtualization</th>
<th>Virtualization on the OS level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>operating mode</strong></td>
<td>CPU + chipset, I/O, ... emulated</td>
<td>CPU directly, I/O, ... emulated</td>
<td>no emulation, special API for HW-Access</td>
<td>no emulation, 1 Kernel for host and guests</td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td>Development, Games (C64, ...)</td>
<td>Server, Desktop</td>
<td>Server, Desktop</td>
<td>Server</td>
</tr>
<tr>
<td><strong>Examples</strong></td>
<td>Bochs, QEMU, PearPC, Hercules, Virt. PC für Mac</td>
<td>VMware Server, Parallels Deskt., Virt. PC/Server, VirtualBox, XEN 3 mit VT/V</td>
<td>XEN, User Mode Linux</td>
<td>Linux Vserver, OpenVZ/Virtuozzo, Solaris Container, FreeBSD Jails</td>
</tr>
</tbody>
</table>
Comparison

- Can run different OSs on the same box
- Low density/scalability
- Slow/complex management – OS sprawl problem
- Low/moderate perf.

- Native performance: no overhead
- Dynamic resource allocation, best scalability
- Single OS per box: easier to manage
Agenda

- Ways to Virtualize
- OpenVZ Details
- Usage Szenarios
- Current Status
Evolution of Operating Systems

- **Multitask**
  many processes

- **Multiuser**
  many users

- **Multiple execution environments**
  many virtual environments (VEs, VPSs, containers, guests, partitions...)
OpenVZ: components

• Kernel
  - Virtualization and Isolation
  - Resource Management
  - Checkpointing

• Tools
  - vzctl: Virtual Environment (VE) control utility
  - vzpkg: VE software package management

• Templates
  - precreated VE images for fast VE creation
Kernel: Virtualization & Isolation

Each virtual environment has its own

- **Files**
  System libraries, applications, virtualized `/proc` and `/sys`, virtualized locks etc.

- **Process tree**
  Featuring virtualized PIDs, so that the init PID is 1

- **Network**
  Virtual network device, its own IP addresses, set of netfilter and routing rules

- **Devices**
  Plus if needed, any VE can be granted access to real devices like network interfaces, serial ports, disk partitions, etc.

- **IPC objects**
  shared memory, semaphores, messages

- …
Kernel: Resource Management

Managed resource sharing and limiting.

- **User BeanCounters** is a set of per-VE resource counters, limits, and guarantees (kernel memory, network buffers, phys pages, etc.)

- **Fair CPU scheduler** (with shares and hard limits)

- **Two-level disk quota** (first-level: per-VE quota; second-level: ordinary user/group quota inside a VE)

- **New: I/O scheduler** (two-level, based on CFQ)

Resource management is what makes OpenVZ different from other OS virtualization solutions.
Kernel: Checkpointing/Migration

• Complete VE state can be saved in a file
  - running processes
  - opened files
  - network connections, buffers, backlogs, etc.
  - memory segments
• VE state can be restored later
• VE can be restored on a different server
Tools: VE control

# vzctl create 101 --ostemplate fedora-core-5
# vzctl set 101 --ipadd 192.168.4.45 --save
# vzctl start 101
# vzctl exec 101 ps ax

<table>
<thead>
<tr>
<th>PID</th>
<th>TTY</th>
<th>STAT</th>
<th>TIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>init</td>
</tr>
<tr>
<td>11830</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>syslogd -m 0</td>
</tr>
<tr>
<td>11897</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>/usr/sbin/sshd</td>
</tr>
<tr>
<td>11943</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>xinetd -stayalive -pidfile ...</td>
</tr>
<tr>
<td>12218</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>sendmail: accepting connections</td>
</tr>
<tr>
<td>12265</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>sendmail: Queue runner@01:00:00</td>
</tr>
<tr>
<td>13362</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>/usr/sbin/httpd</td>
</tr>
<tr>
<td>13363</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>_ /usr/sbin/httpd</td>
</tr>
<tr>
<td>13373</td>
<td>?</td>
<td>S</td>
<td>0:00</td>
<td>_ /usr/sbin/httpd</td>
</tr>
<tr>
<td>6416</td>
<td>?</td>
<td>Rs</td>
<td>0:00</td>
<td>ps axf</td>
</tr>
</tbody>
</table>

# vzctl enter 101
bash# logout
# vzctl stop 101
# vzctl destroy 101
Tools: Templates

# vzpkgls
fedora-core-5-i386-default
centos-4-x86_64-minimal

# vzpkgcache
(create templates from metadata/updates existing templates)

# vzyum 101 install gcc
(installs gcc and its deps to VE 101)
Density

Response time, sec.

Number of VEs

768 (¾) MB RAM - up to 120 VEs
2GB RAM - up to 320 VEs
Hello all, just downloaded and installed OpenVZ, and i must say its a big improvement over other VPS systems that i have tested IMHO.

http://forum.openvz.org/index.php?t=msg&goto=646#msg_646

I use virtuozzo in my day job and openvz is very much the same. Just no windows GUI which I hate using anyway! Virtuozzo and openvz are wonderful - I don't know why more people aren't using them. I hear a lot of hype for <...> and but virtuozzo/openvz is so great for many common needs. I'm very happy to be using openvz - very good for my side projects that I can't afford real virtuozzo for.

http://forum.openvz.org/index.php?t=msg&goto=650#msg_650

Last week when we were in limbo about what to do, it was decided to try out Virtualization. From what is written in the press the system has alot of promise, <...> but was far too complicated to get working in our configuration. OpenVZ was the only virtual server system that was simple to install and get working.

http://forum.openvz.org/index.php?t=msg&goto=568#msg_568
Agenda

• Ways to Virtualize
• OpenVZ Details
• Usage Szenarios
• Current Status
Usage Scenarios

- Server Consolidation
- Hosting
- Development and Testing
- Security
- Educational
- HA Clustering
Server Consolidation

A bunch of servers:
- harder to manage
- upgrade is a pain
- eats up rack space
- high electricity bills

A bunch of VEs:
- uniform management
- easily upgradeable and scalable
- fast migration
Hosting

- Web server serving hundreds of virtual hosts
- Users see each other processes etc
- DoS attacks
- Unable to change/upgrade hardware

- Users are isolated from each other
- VE is like a real server, just cheap
- Much easier to admin
Development & Testing

- A lot of hardware
- Zoo: many different Linux distros
- Frequent reinstalls take much time

- Fast provisioning
- Different distros can co-exist on one box
- Cloning, snapshots, rollbacks
- VE is a sandbox – work and play, no fear
Security

- Several network services are running
- One of them has a hole
- Cracker gets through

- Put each service into a separate VE
- OpenVZ creates walls between applications
- Added benefit: dynamic resource management
Educational

- No root access
- Frequent reinstalls
- DoS attacks

- Everybody and his dog can have a root access
- Different Linux distros
- No need for a lot of hardware
HA Clustering

- Apps need to be modified
- Complete VEs are clustered

Classic HA Cluster:
- Node1
  - /bin/
  - /etc/
  - /home/
  - /usr/
  - /var/
  - ...
- Node2
  - /bin/
  - /etc/
  - /home/
  - /usr/
  - /var/
  - ...
- /data

Virtualized HA Cluster:
- Node1
  - /vz/
  - private/101/root/
  - bin/
  - etc/
  - home/
  - var/
  - ...
  - (Base-system)
- Node2
  - /vz/
  - private/101/root/
  - bin/
  - etc/
  - home/
  - var/
  - ...
  - (Base-system)

Local data
- /vz/
  - private/101/root/
  - bin/
  - etc/
  - home/
  - var/
  - ...

Shared data

wiki.openvz.org/HA_cluster_with_DRBD_and_Heartbeat
Agenda

- Ways to Virtualize
- OpenVZ Details
- Usage Szenarios
- Current Status
Recent achievements

- NFS and FUSE in VE
- VE I/O accounting and scheduling
- Checkpointing/live migration for IA64
- Port to RHEL5 kernel
- Port to vanilla 2.6.20
Mainstream kernel integration

• Collaborative community effort:
  – OpenVZ
  – IBM (Metacluster)
  – Linux-VServer
  – Eric Biederman (namespaces)
  – Google (Paul Menage, containers)

• Current progress (as of linux-2.6.20):
  – IPC namespaces/virtualization
  – utksname() virtualization
  – preliminary support for PID namespaces

• More to come soon (networking, beancounters)
How can you help?

• Use OpenVZ
• Contribute to OpenVZ, be a part of community:
  – Programmer
    • fixes
    • enhancements
    • new functionality
  – Non-programmer
    • bug reports
    • work with wiki
    • answer support questions
To sum it up

- Platform-independent
  - as long as Linux support it, we support it
- No problems with scalability or disk IO
  - lots of memory, lots of CPUs no prob
  - native I/O speed
- Best possible performance
- Can play well with others (Xen, KVM, ...)

36
Project Links

- Main site: http://openvz.org/
- Downloads: http://download.openvz.org/
- Wiki: http://wiki.openvz.org/
- Sources: http://git.openvz.org/
- Forum: http://forum.openvz.org/
- Bug Tracking: http://bugzilla.openvz.org/
- Blog: http://blog.openvz.org/
- Mailing lists: users@openvz.org
devel@openvz.org
announce@openvz.org