Containers and Namespaces in the Linux Kernel

Kir Kolyshkin
<kir@openvz.org>
Agenda

• Containers vs Hypervisors

• Kernel components
  – Namespaces
  – Resource management
  – Checkpoint/restart
Hypervisors

- VMware
- Parallels
- QEMU
- Bochs
- Xen
- UML (User Mode Linux)
- KVM

VMM/Hypervisor

VM 1
Guest OS

Virtual Hardware

Virtual Machine Monitor

Hypervisor or Standard Host OS

Hardware
Containers

- OpenVZ / Parallels Containers
- FreeBSD jails
- Linux-VServer
- Solaris Containers/Zones
- IBM AIX6 WPARs (Workload Partitions)
## Comparison

<table>
<thead>
<tr>
<th>Hypervisor (VM)</th>
<th>Containers (CT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• One real HW, many virtual HWs, many OSs</td>
<td>• One real HW (no virtual HW), one kernel, many userspace instances</td>
</tr>
<tr>
<td>• High versatility – can run different OSs</td>
<td>• High density</td>
</tr>
<tr>
<td>• Lower density, performance, scalability</td>
<td>• Dynamic resource allocation</td>
</tr>
<tr>
<td>• «Lowers» are mitigated by new hardware features (such as VT-D)</td>
<td>• Native performance: [almost] no overhead</td>
</tr>
</tbody>
</table>

---

OpenVZ
Linux Containers
## Comparison: a KVM hoster

<table>
<thead>
<tr>
<th>Feature</th>
<th>KVM VPS</th>
<th>OpenVZ / Virtuozzo</th>
<th>Xen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedicated filesystem of your choice</td>
<td>+</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Dedicated RAM with full access and debugging capabilities</td>
<td>+</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Dedicated server like isolation</td>
<td>+</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>VNC connection from the very early boot stage</td>
<td>+</td>
<td>−</td>
<td>limited support</td>
</tr>
<tr>
<td>PPTP VPN</td>
<td>+ limited support</td>
<td>+</td>
<td>+ limited support</td>
</tr>
<tr>
<td>OpenVPN</td>
<td>+ limited support</td>
<td>+</td>
<td>+ limited support</td>
</tr>
<tr>
<td>IPSec VPN</td>
<td>+</td>
<td>−</td>
<td>limited support</td>
</tr>
</tbody>
</table>

- **Firewall Configuration**: + limited support
- **Kernel mode NFS server**: −
- **Independent kernel**: − limited support
- **Independent kernel modules**: − limited support
- **Full control on sockets and processes**: −
- **Full guest OS support (Windows, Linux, BSD, OpenSolaris, etc.)**: + limited support
- **Direct dedicated access to PCI / PCIe cards**: − limited support
- **Fine grained swap configuration per VPS**: − limited support
- **Official integration with the Linux kernel**: −
## Comparison: bike vs car

<table>
<thead>
<tr>
<th>Feature</th>
<th>Bike</th>
<th>Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ecological</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Low price</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Needs parking space</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Periodical maintenance cost</td>
<td>Low</td>
<td>Med</td>
</tr>
<tr>
<td>Needs refuelling</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can drive on a footpath</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Lightweight aluminium frame</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Easy to carry (e.g. take with you on a train)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Fun factor</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

Source: [http://wiki.openvz.org/Bike_vs_car](http://wiki.openvz.org/Bike_vs_car)
## Comparison: car vs bike

<table>
<thead>
<tr>
<th>Feature</th>
<th>Car</th>
<th>Bike</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Needs muscle power</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Passenger and load capacity</td>
<td>Med</td>
<td>Low</td>
</tr>
<tr>
<td>In-vehicle music</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Gearbox</td>
<td>Auto</td>
<td>Man</td>
</tr>
<tr>
<td>Power steering, ABS, ESP, TSC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Ability to have sex inside</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Air conditioning</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Fun factor</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

Source: [http://wiki.openvz.org/Car_vs_Bike](http://wiki.openvz.org/Car_vs_Bike)
OpenVZ vs. Xen from HP labs

• For all the configuration and workloads we have tested, Xen incurs higher virtualization overhead than OpenVZ does.
• For all the cases tested, the virtualization overhead observed in OpenVZ is limited, and can be neglected in many scenarios.
• Xen systems becomes overloaded when hosting four instances of RUBiS, while the OpenVZ system should be able to host at least six without being overloaded.
You can have both!

- Create containers and VMs on the same box
- Best of both worlds
Kernel components

- Namespaces
  - PID
  - Net
  - User
  - IPC
  - etc.

- Resource management (group-based)

- Fancy tricks – checkpoint/restart
Trivial namespace cases

- Filesystem:
  
  ```c
  chroot() syscall
  ```

- Hostname:
  
  ```c
  struct system_utsname per container
  CLONE_NEWUTS flag for clone() syscall
  ```
PID namespace: why?

• Usually a PID is an arbitrary number
• Two special cases:
  – Init (i.e. child reaper) has a PID of 1
  – Can't change PID (process migration)
PID NS: details

- `clone(CLONE_NEWPID)`
- Each task inside pidns has 2 pids
- Child reaper is virtualized
- `/proc/$PID/*` is virtualized
- Multilevel: can create nested pidns
  - slower on fork() where level > 1
- Consequence: PID is no longer unique in kernel
Network namespace: why?

- Various network devices
- IP addresses
- Routing rules
- Netfilter rules
- Sockets
- Timewait buckets, bind buckets
- Routing cache
- Other internal stuff
NET NS: devices

• macvlan
  – same NIC, different MAC
  – NIC is in promisc mode

• veth
  – like a pipe, created in pairs, 2 ends, 2 devices
  – one end goes to NS, other is bridged to real eth

• venet (not in mainstream yet / only in OpenVZ)
  – MACless device
  – IP is ARP announced on the eth
  – host system acts as a router
NET NS: dive into

- Can put a network device into netns
  - `ip link set DEVICE netns PID`

- Can put a process into netns
  - New:
    `clone(CLONE_NEWNET)`
  - Existing:
    `fd = nsfd(NS_NET, pid); setns(fd);`
Other namespaces

- User: UIDs/GIDs
  - Not finished: signal code, VFS inode ownership
- IPC: shmем, semaphores, msg queues
Namespace problems / todo

- Missing namespaces: tty, fuse, binfmt_misc
- Identifying a namespace
  - No namespace ID, just process(es)
- Entering existing namespaces
  - Problem: no way to enter existing NS
  - Proposal: \texttt{fd=nsfd(NS, PID); setns(fd)};
  - Problem: can't enter pidns with current task
  - Proposal: \texttt{clone_at()} with additional PID argument
Resource Management

- Traditional stuff (ulimit etc.) sucks
  - all limits are per-process except for numproc
  - some limits are absent, some are not working
- Answer is CGroups
  - a generic mechanism to group tasks together
  - different resource controllers can be applied
- Resource controllers
  - Memory / disk / CPU … – work in progress
Resource management: OpenVZ

- User Beancounters
  a set of per-CT resource counters, limits, and guarantees
- Fair CPU scheduler
  two-level
  shares, hard limits, VCPU affinity
- Disk quota
  two-level: per-CT and per-UGID inside CT
- Disk I/O priority per CT
Kernel: Checkpointing/Migration

- Complete CT state can be saved in a file
  - running processes
  - opened files
  - network connections, buffers, backlogs, etc.
  - memory segments
- CT state can be restored later
- CT can be restored on a different server
LXC vs OpenVZ

- OpenVZ was off-the-mainline historically
  - developing since 2000
- We are working on merging bits and pieces
- Code in mainline is used by OpenVZ
  - It is also used by LXC (and Linux-VServer)
- OpenVZ is production ready and stable
- LXC is a work-in-progress
  - not a ready replacement for OpenVZ
- We will keep maintaining OpenVZ for a while
Questions / Contacts

kir@openvz.org
containers@linux-foundation.org
http://wiki.openvz.org/
http://lxc.sf.net/
To sum it up

• Platform-independent
  – as long as Linux supports it, we support it
• No problems with scalability or disk I/O
  – lots of memory, lots of CPUs no prob
  – native I/O speed
• Best possible performance
• Plays well with others (Xen, KVM, VMware)
[Backup] Usage Scenarios

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