# Containers and Namespaces
## in the Linux Kernel

*Kir Kolyshkin*
*<kir@openvz.org>*

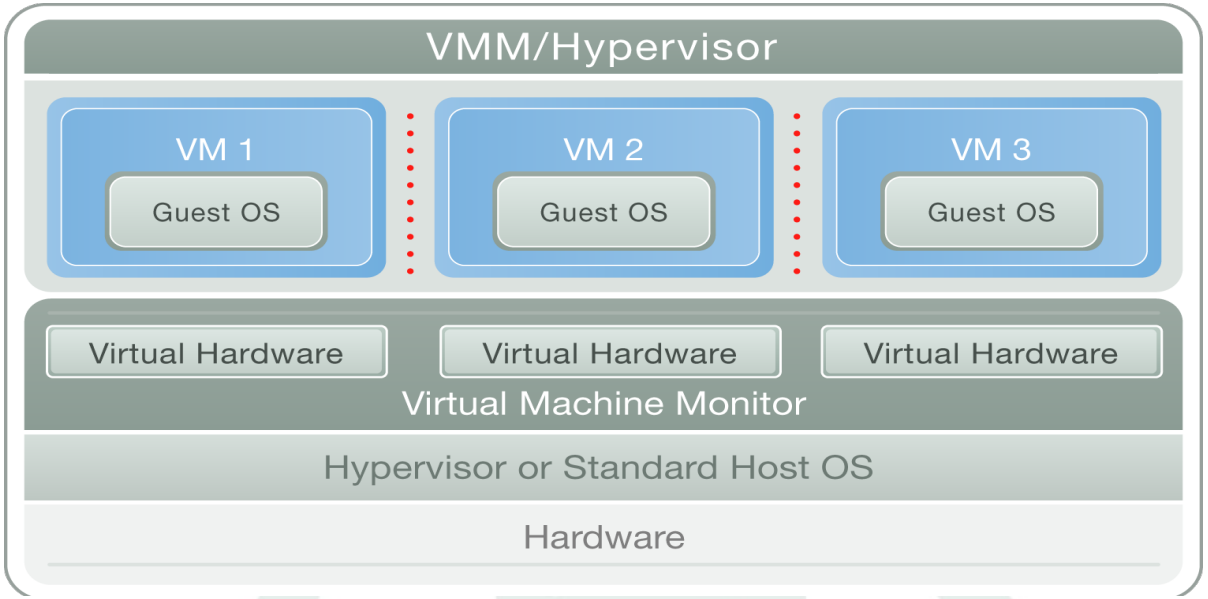
Parallels™

OpenVZ
Linux Containers

# Agenda

- Containers vs Hypervisors
- Kernel components
  - Namespaces
  - Resource management
  - Checkpoint/restart

# Hypervisors

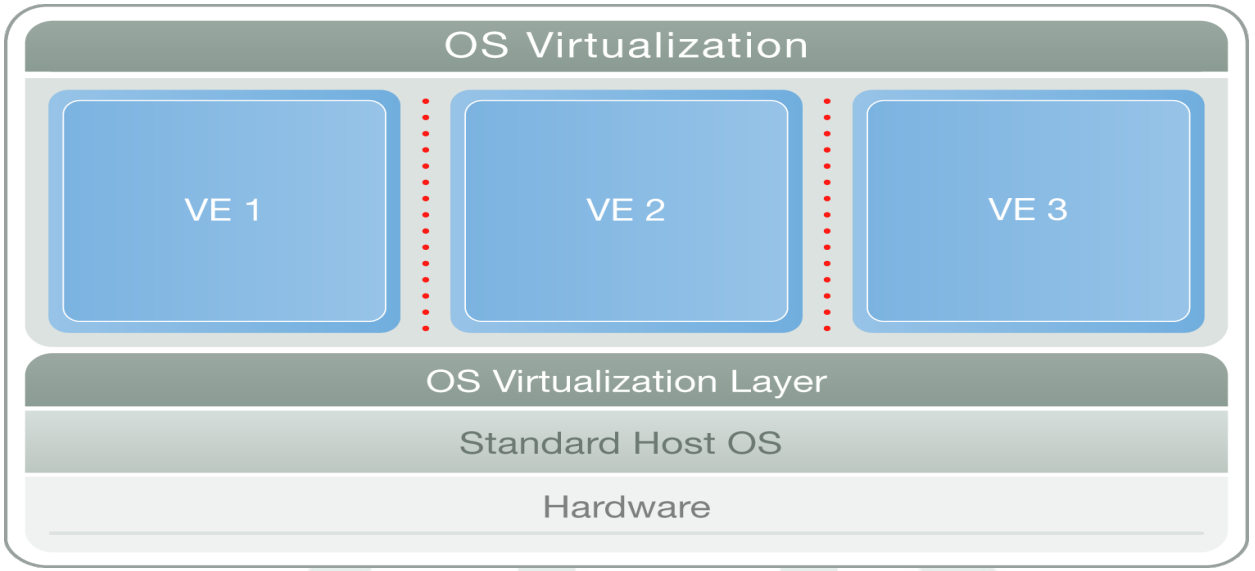



- VMware
- Parallels
- QEmu
- Bochs

- Xen
- UML
  (User Mode Linux)
- KVM

OpenVZ
Linux Containers

# Containers




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

OpenVZ

Parallels™

Linux VServer

SOLARIS™

IBM

OpenVZ Linux Containers

# Comparison

**Hypervisor (VM)**

- One real HW, many virtual HWs, many OSs
- High versatility – can run different OSs
- Lower density, performance, scalability
- «Lowers» are mitigated by new hardware features (such as VT-D)

**Containers (CT)**

- One real HW (no virtual HW), one kernel, many userspace instances
- High density
- Dynamic resource allocation
- Native performance: [almost] no overhead

# Comparison: a KVM hoster

## KVM VPS vs OpenVZ/Virtuozzo vs Xen

| | KVM VPS | OpenVZ / Virtuozzo | Xen |
|---|---|---|---|
| Dedicated filesystem of your choice (with direct block level access) | ✓ | – | ✓ |
| Dedicated RAM with full access and debugging capabilities | ✓ | – | ✓ |
| Dedicated server like isolation | ✓ | – | ✓ |
| VNC connection from the very early boot stage | ✓ | – | limited support |
| PPTP VPN | ✓ | limited support | ✓ |
| OpenVPN | ✓ | limited support | ✓ |
| IPSec VPN | ✓ | – | limited support |
| 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 | ✓ | – | ✓ |

OpenVZ Linux Containers

# Comparison: bike vs car

| Feature | Bike | Car |
|---|---|---|
| Ecological | Yes | No |
| Low price | Low | High |
| Needs parking space | No | Yes |
| Periodical maintenance cost | Low | Med |
| Needs refuelling | No | Yes |
| Can drive on a footpath | Yes | No |
| Lightweight aluminium frame | Yes | No |
| Easy to carry (e.g. take with you on a train) | Yes | No |
| Fun factor | High | Low |

Source: http://wiki.openvz.org/Bike_vs_car

# Comparison: car vs bike

| Feature | Car | Bike |
|---|---|---|
| Speed | High | Low |
| Needs muscle power | No | Yes |
| Passenger and load capacity | Med | Low |
| In-vehicle music | Yes | No |
| Gearbox | Auto | Man |
| Power steering, ABS, ESP, TSC | Yes | No |
| Ability to have sex inside | Yes | No |
| Air conditioning | Yes | No |
| Fun factor | High | Low |

Source: 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

OpenVZ
Linux Containers

# Kernel components

- Namespaces
  - PID
  - Net
  - User
  - IPC
  - etc.
- Resource management (group-based)
- Fancy tricks – checkpoint/restart

# Trivial namespace cases

- Filesystem:

  `chroot()` syscall

- Hostname:

  `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: shmem, 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: `fd=nsfd(NS, PID); setns(fd);`
  - problem: can't enter pidns with current task
  - proposal: 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