OpenVZ
Linux Containers

Kir Kolyshkin
<kir@openvz.org>

OpenVZ
Linux Containers

OpenStack Design Summit

16 April 2012, San Francisco

Agenda

- Virtualization approaches
- Containers versus hypervisors:
- OpenVZ internals
- New features

What is virtualization?

Virtualization is a technique for deploying technologies. Virtualization creates a level of indirection or an abstraction layer between a physical object and the managing or using application.

http://www.aarohi.net/info/glossary.html

Virtualization is a framework or methodology of dividing the resources of a computer into multiple execution environments...

http://www.kernelthread.com/publications/virtualization/

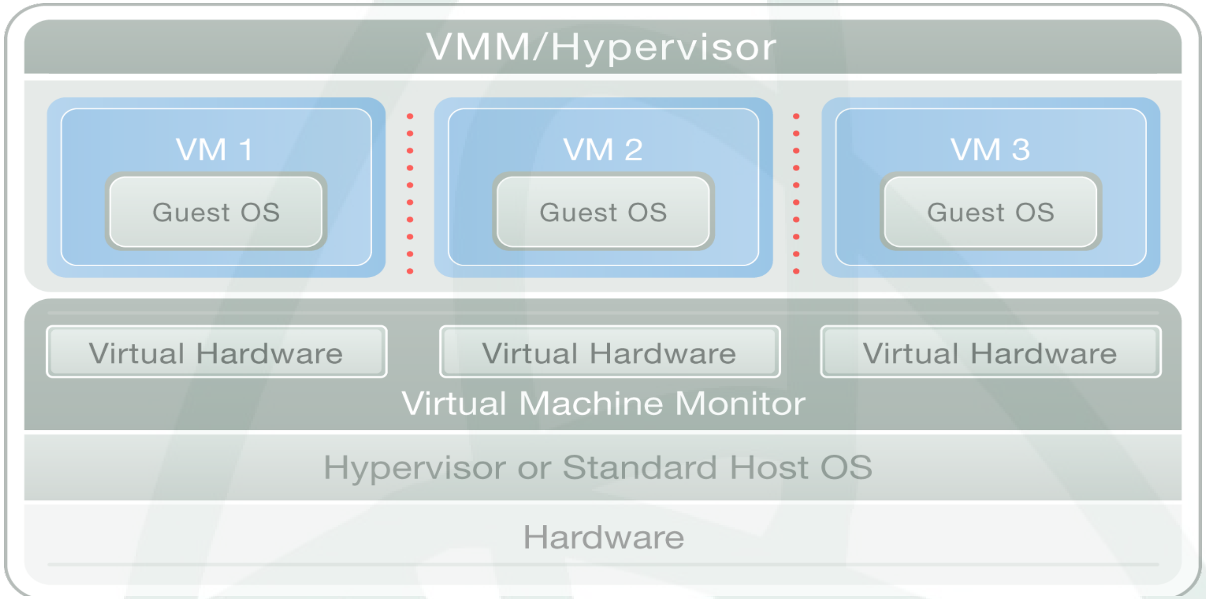
A key benefit of the virtualization is the ability to run multiple operating systems on a single physical server and share the underlying hardware resources – known as **partitioning**.

http://www.vmware.com/pdf/virtualization.pdf

Ways to Virtualize

- Hardware Emulation
- Para-Virtualization
- Containers-type Virtualization (a.k.a. OS-level Virtualization)
- Multi-server virtualization

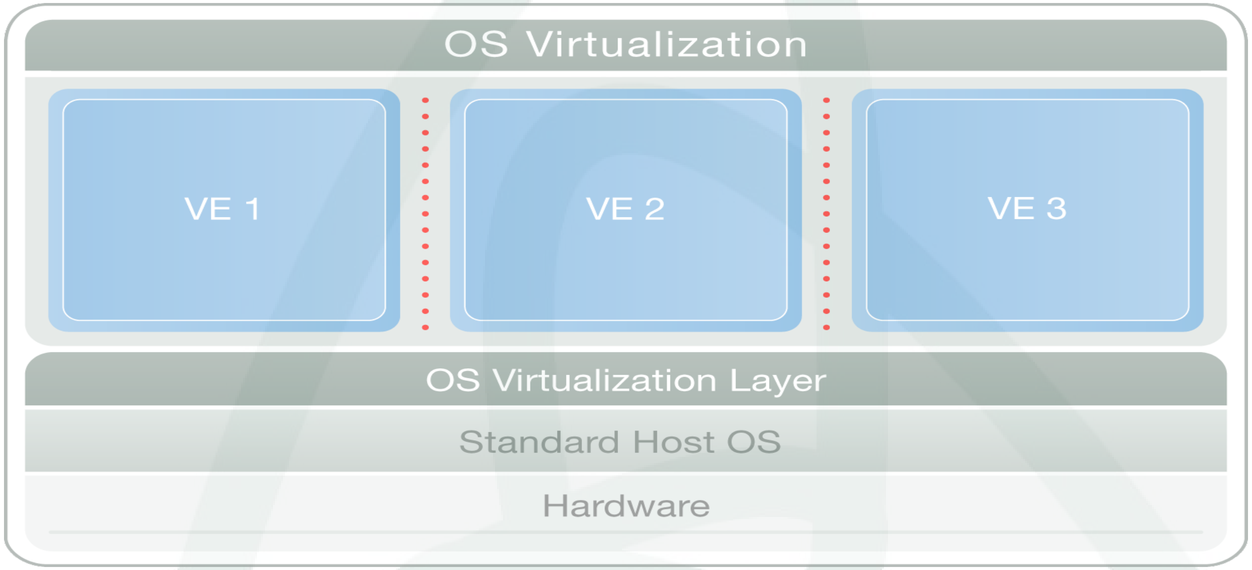
Emulation/Paravirtualization



- VMware
- Parallels
- QEmu
- Bochs

- Xen
- KVM

Containers



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

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)

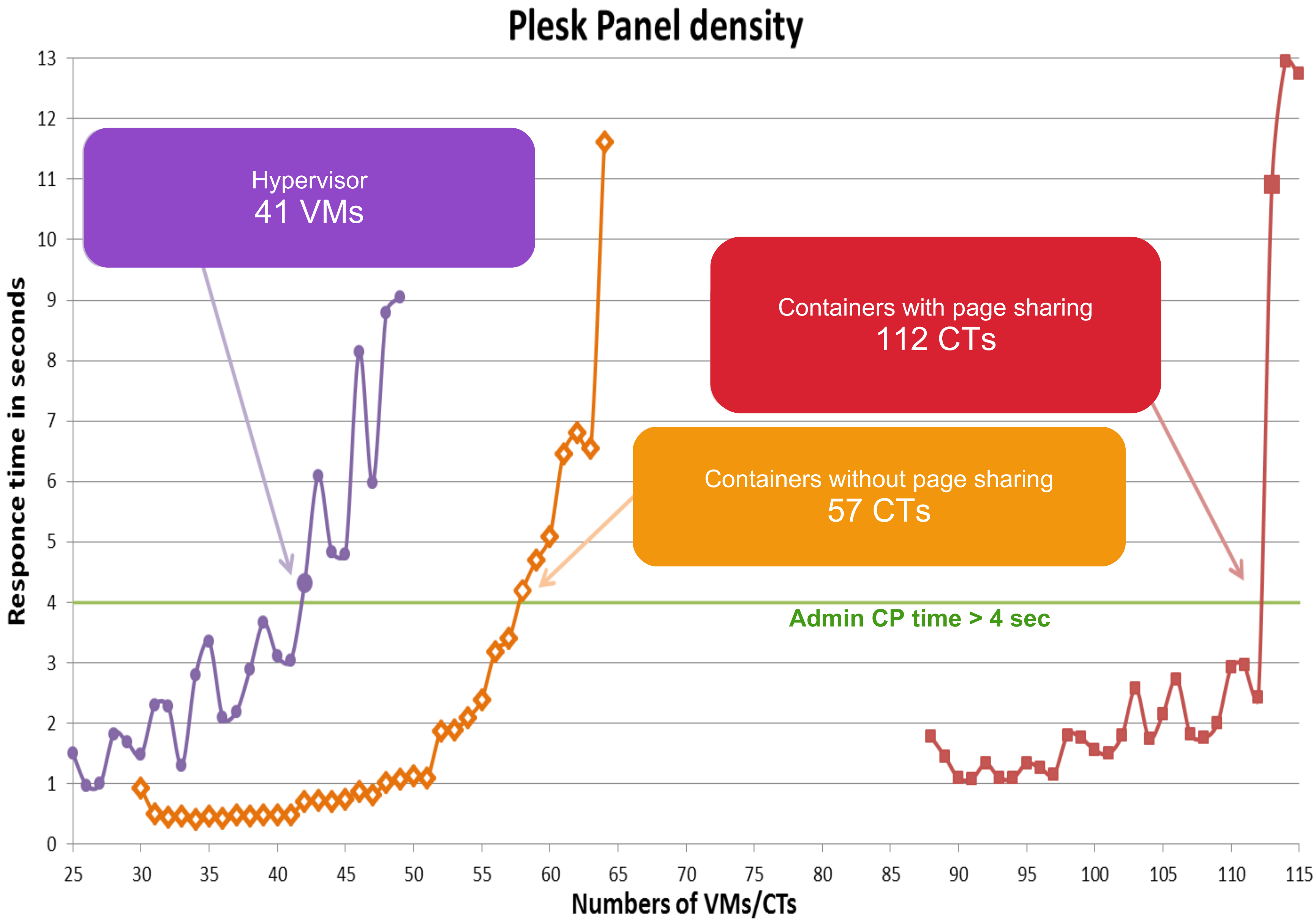
Containers (CT)

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

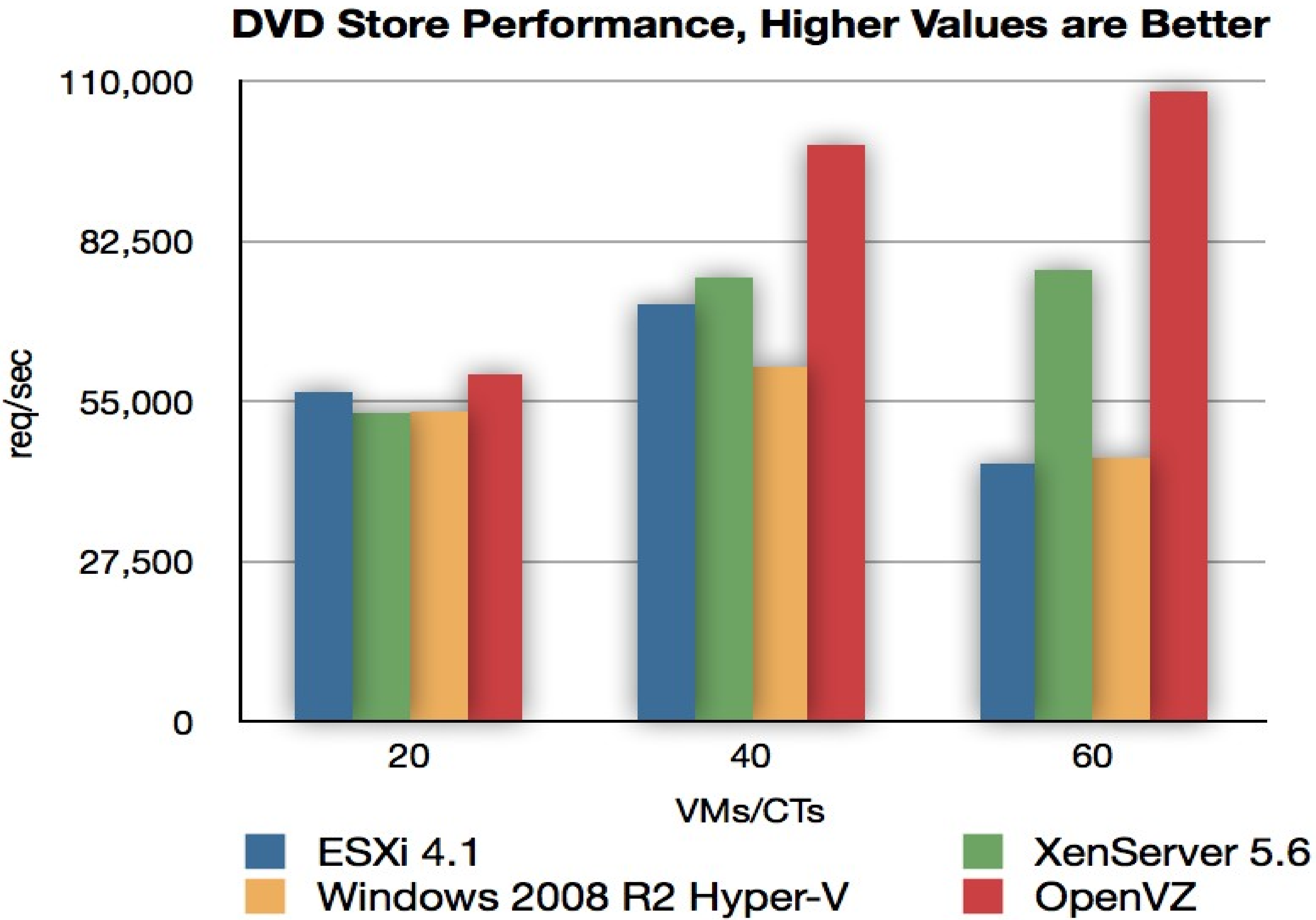
Containers are thinner and more performant than hypervisors

- Containers
  - Share host OS and Drivers
  - Have small virtualization layer
  - Naturally share pages
- Hypervisors
  - Have separate OS plus virtual Hardware
  - Hardware emulation requires VMM state
  - Have trouble sharing Guest OS pages
- Containers are more elastic than hypervisors
- Container slicing of the OS is ideally suited to cloud slicing
- Hypervisors’ only advantage in IaaS is support for different OS families on one server

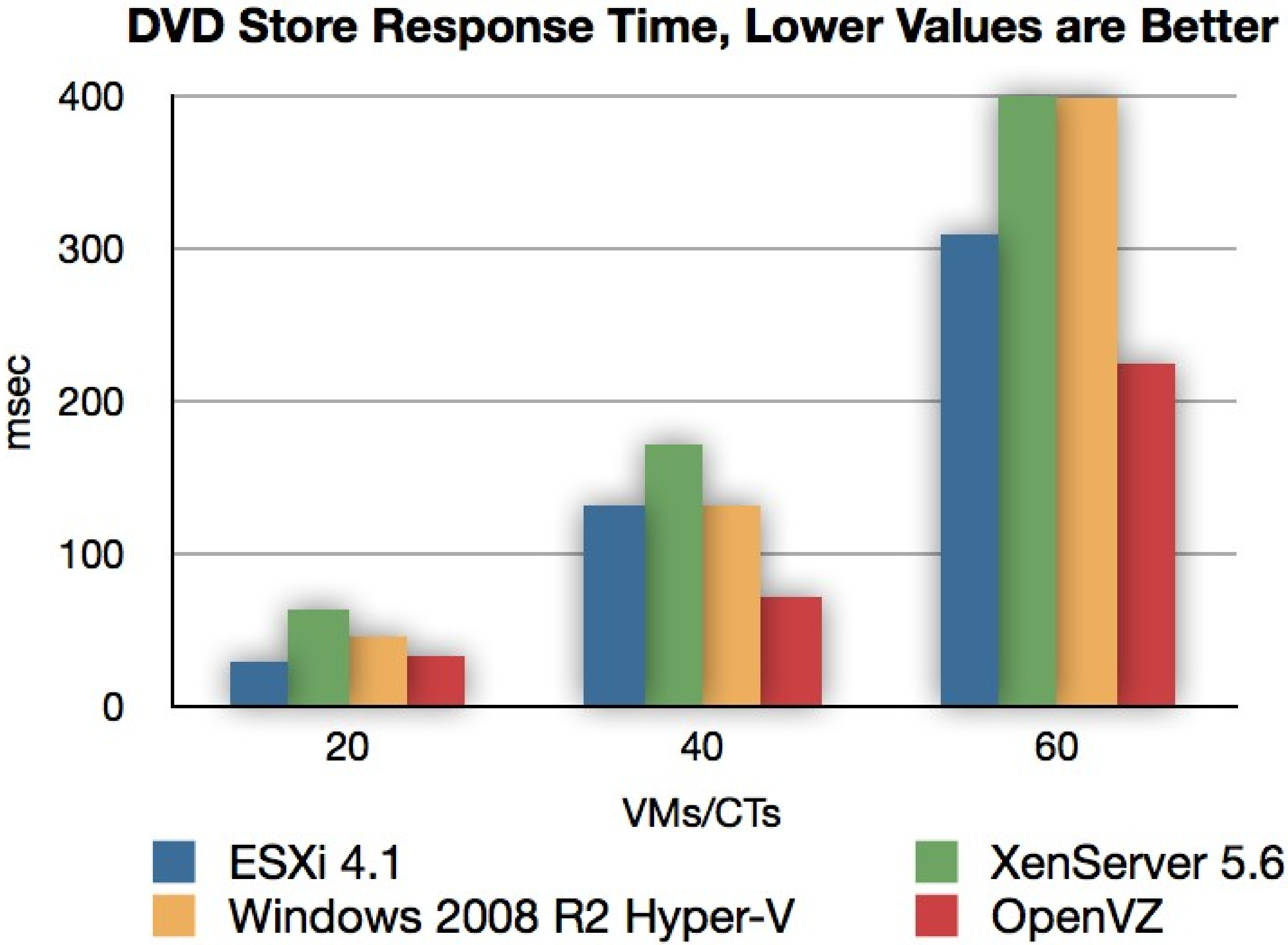
Density


Plesk Panel density
Hypervisor
41 VMs
Containers with page sharing
112 CTs
Containers without page sharing
57 CTs
Admin CP time > 4 sec
Responce time in seconds
Numbers of VMs/CTs
0
1
2
3
4
5
6
7
8
9
10
11
12
13
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115

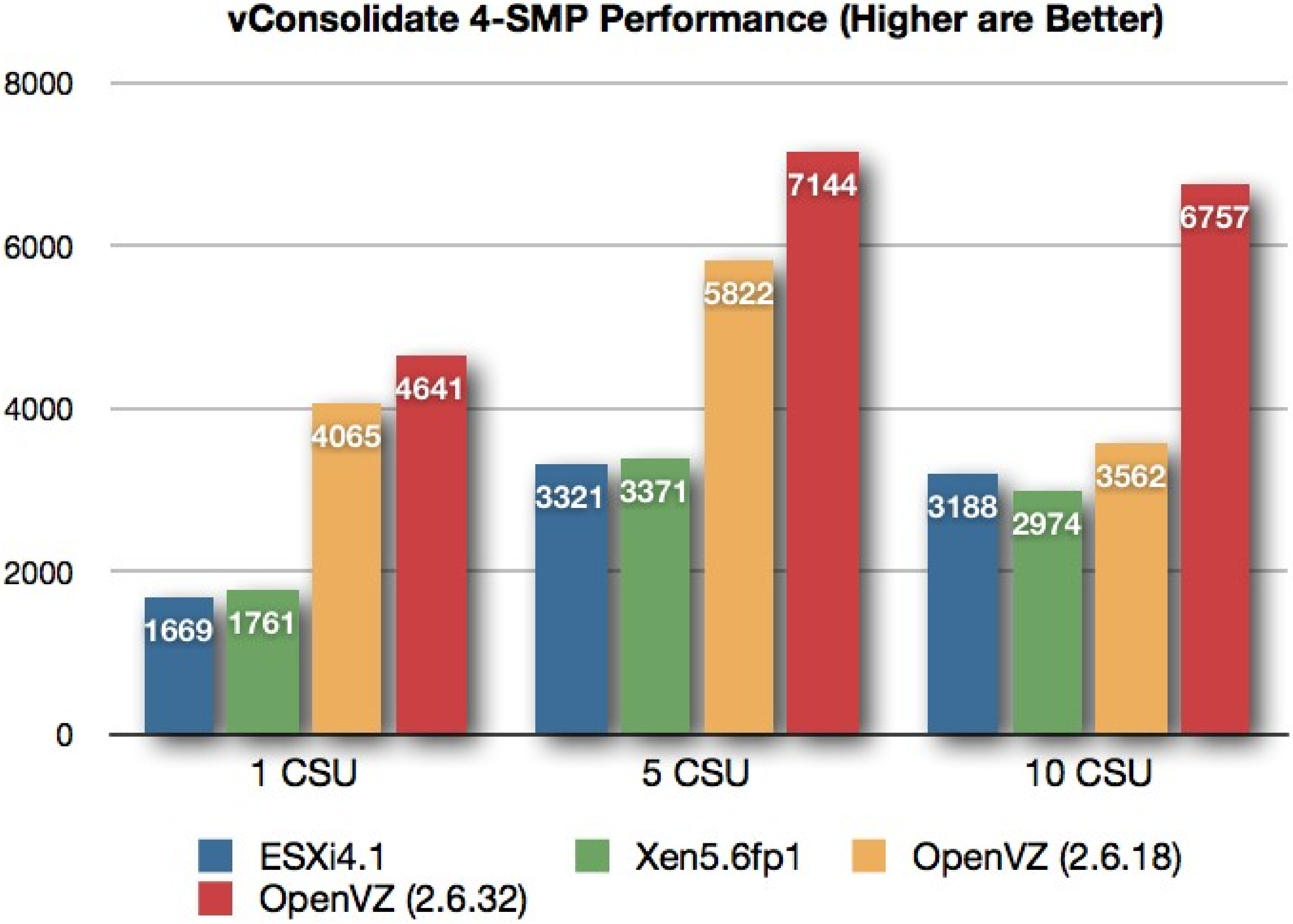
Perf: LAMP throughput



Perf: LAMP response time



Performance: vConsolidate



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

Evolution of Operating Systems

- **Multi**task
  many processes
- **Multi**user

  many users
- **Multi**container

  many containers (CTs, VEs, VPSs, guests, partitions...)

OpenVZ: components

- Kernel
  - Namespaces: virtualization and Isolation
  - CGroups: Resource Management
  - Checkpoint/restart (live migration)
- Tools
  - vzctl: containers control utility
- Templates
  - precreated images for fast container creation

Kernel: Virtualization & Isolation

Each container has its own

- Files: chroot()
  System libraries, applications, virtualized `/proc` and `/sys`, virtualized locks etc.
- Process tree (PID namespace)
  Featuring virtualized PIDs, so that the init PID is 1
- Network (net namespace)
  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 (IPC namespace)
  shared memory, semaphores, messages
- …

Kernel: Resource Management

Managed resource sharing and limiting.

- **User Beancounters** per-CT 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-CT quota; second-level: ordinary user/group quota inside a CT)
- **Disk I/O priority** (also 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

Tools: CT control

```
# vzctl create 101 --ostemplate fedora-15
# vzctl set 101 --ipadd 20.21.22.23/24 --save
# vzctl start 101
# vzctl exec 101 ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 init
11830 ?        Ss     0:00 syslogd -m 0
11897 ?        Ss     0:00 /usr/sbin/sshd
11943 ?        Ss     0:00 xinetd -stayalive -pidfile ...
12218 ?        Ss     0:00 sendmail: accepting connections
12265 ?        Ss     0:00 sendmail: Queue runner@01:00:00
13362 ?        Ss     0:00 /usr/sbin/httpd
13363 ?        S      0:00  \_ /usr/sbin/httpd
..............................................
13373 ?        S      0:00  \_ /usr/sbin/httpd
6416 ?         Rs     0:00 ps axf
# vzctl enter 101
bash# logout
# vzctl stop 101
# vzctl destroy 101
```

Feature: VSwap

- A new approach to memory management, only two parameters to configure: RAM, swap
- Appeared in RHEL6-based OpenVZ kernel
- Swap is virtual, no actual I/O is performed
- Slow down to emulate real swap
- Only when actual global RAM shortage occurs, virtual swap goes into the real swap

Feature: ploop

- Reimplementation of Linux loop device
- Modular architecture
- Support for different file formats (“plain”, QCOW2, etc)
- Network storage is supported (NFS)
- Snapshots and fast provisioning via stacked images
- Write tracker for faster live migration

Feature: CRIU

- **C**heckpoint/**R**estore (mostly) **I**n **U**serspace
- Kernel manage processes, knows everything
- All efforts to merge CPT to Linux kernel failed
- Solution: let's do it in userpace!
- Minimal kernel intervention
- First set of patches was recently accepted

- See http://criu.org/

CRIU merge comment from akpm

Checkpoint/restart feature work.

A note on this: this is a project by various mad Russians to perform c/r mainly from userspace, with various oddball helper code added into the kernel where the need is demonstrated.

So rather than some large central lump of code, what we have is little bits and pieces popping up in various places which either expose something new or which permit something which is normally kernel-private to be modified.

The overall project is an ongoing thing. I've judged that the size and scope of the thing means that we're more likely to be successful with it if we integrate the support into mainline piecemeal rather than allowing it all to develop out-of-tree.

Development

- Most of the magic is in the kernel
- Use RHEL kernels as a base
- Support it for many years
  (RHEL4 kernels from 2006 are still supported)
- Merge bits and pieces upstream, then reuse
- Plans to include most of the kernel stuff till RHEL7

LXC vs OpenVZ

- OpenVZ was off-the-mainline historically
  - developing since 2000
- We are working on merging bits and pieces, with more than 1500 patches in mainline
- Code in mainline is used by OpenVZ
- 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

To mainline we go

- || Parallels is the driving force behind Linux containers
- Collaborating with the Linux community to move OpenVZ upstream into the Linux kernel
- Ensures that Linux delivers a single consistent container technology
- vzctl will support mainline containers
- No rerun of Xen/KVM wars!

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
- Unbeatable density and performance
- Reliable, supported, free
- Plays well with others

Thank you!

http://openvz.org/

http://criu.org/

kir@openvz.org