# Container-based OS Virtualization

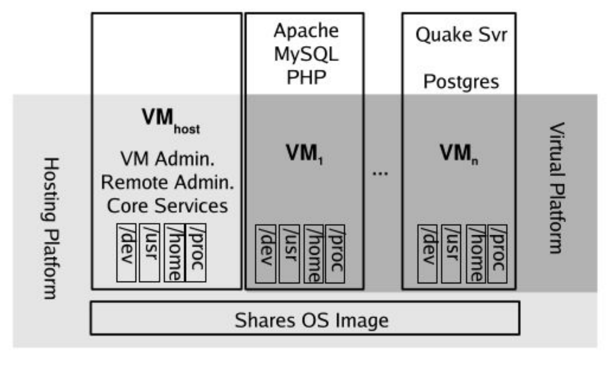
Dustin Wendt

# Outline

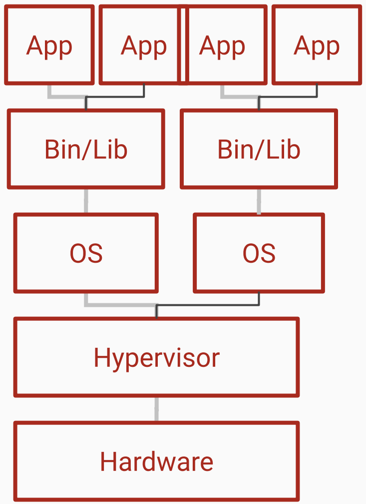
- **Overview**
- Isolation
- Benchmarks

# Containers

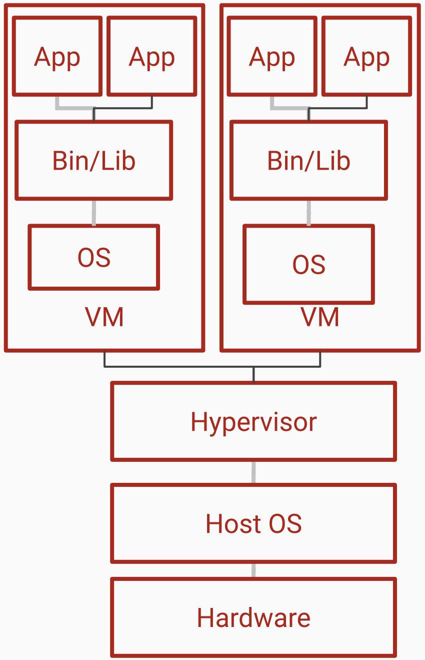
- Forfeit some isolation for efficiency
- Shared Virtualized OS image
- Ran on a single kernel



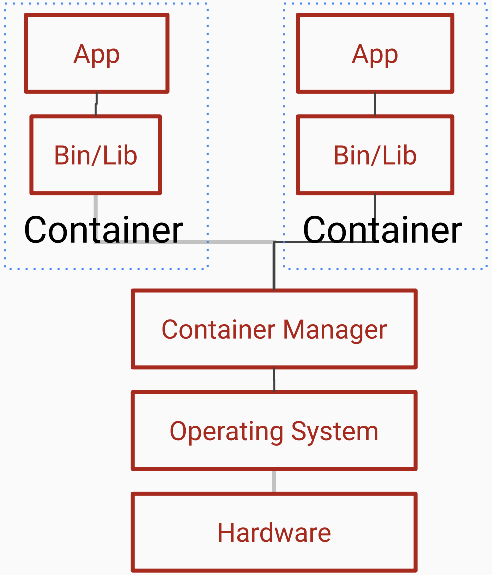

- A (safely shared) set of system executables and libraries
- Examples: Linux-VServer, Virtuozzo, Solaris 11, and Docker




Type 1 Hypervisor




Type 2 Hypervisor




Containers

# Modifications to Operating System

- Xen hypervisor for i32 architecture requires 80K lines of code
- Paravirtualized Linux requires 15K
- VServer adds less than 8700 lines of code to the kernel
- However, VServer creates 50+ new kernel files and touches 300+ others

# Usage Scenarios

- High Performance Computing Clusters
- Database Hosting
- Distributed Hosting
- Reproducing results on different hardware (Docker)

# Hypervisors vs. Containers

| Features | Hypervisors | Containers |
|---|---|---|
| Multiple Kernels | ✅ | ❌ |
| Administrative Power (root) | ✅ | ✅ |
| Checkpoint & Resume | ✅ | ✅ |
| Live Migration | ✅ | ✅ |
| Live System Update | ❌ | ✅ |

# Outline

- Overview
- **Isolation**
- Benchmarks

# Full Isolation

- Fault Isolation
- Resource Isolation
- Security Isolation

# Fault Isolation

- VMs separated from each other via address space in both hypervisor and container
- Only data and code shared is the virtualizing COS or hypervisor
- Any fault in shared code will have the whole system fail

# Resource Isolation

COS handles

- CPU cycles
- I/O Bandwidth
- Memory/Disk Storage

All other physical resources are handled by privileged host VMs

# VServer CPU Scheduling

- Token bucket filter on top of Linux CPU scheduler
- VMs can have a reservation or share of CPU time
- VMs with a share will be scheduled before idle tasks, but after reservations have been fulfilled

# I/O Bandwidth

- Hierarchical token bucket (htb) of the Linux traffic controller
- Each VM is assigned a token bucket with a reserve and/or a share
- Having only a reservation indicates a capped outgoing bandwidth
- Excess bandwidth is assigned proportional to shares

# Memory

- VServer allows setting limits on the amount of memory a VM can acquire
  - Resident Set Size (RSS)
  - # of anonymous memory pages
  - # of pages that can be mlock() and mlockall()
- Watchdog daemon if memory is limited

# Security Isolation

- Separation of name spaces (contexts)
- Access controls (filters)
- Process Filtering
- Network Separation
- Chroot Barrier

# VServer Process Filtering

- VServer reuses PID across all VMs
- Init process has to exist with PID 1. VServer also provides a per VM mapping to a fake init process
- At boot of a VServer system all processes belong to a default host VM
- VServer also has a spectator VM to look at all processes at once

# Network Separation

- OpenVZ fully virtualizes the network subsystem
- VServer shares the network subsystem between all VMs
- VMs are only able to bind sockets to IP addresses set
  - At VM creation
  - Dynamically by the Host VM

# Problems with chroot

- Chroot is volatile
- Chroot does not close file descriptors
- VServer uses a file attribute, Chroot Barrier, that disallows a guest VM from going past its parent directory

# Reducing Resources

- Hard link to shared files that are unlikely to change
- Mark the shared files as CoW
- One Linux server will take up 500MB of disk space and ten unified servers will take up 700MB

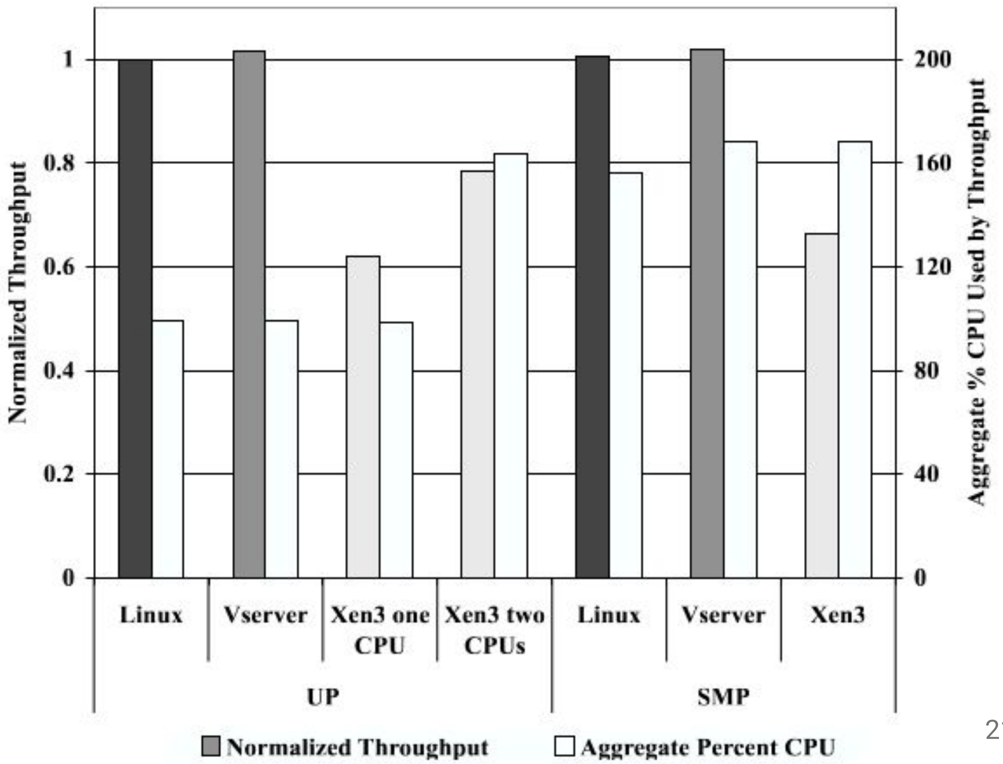
# Outline

- Overview
- Isolation
- **Benchmarks**

# Benchmarks (2007)

| Config | Linux-UP | VServer-UP | Xen3-UP |
|---|---|---|---|
| fork process | 103.2 | 103.4 | 282.7 |
| exec process | 252.9 | 253.6 | 734.4 |
| sh process | 1054.9 | 1047.1 | 1816.3 |
| ctx ( 2p/ 0K) | 2.254 | 2.496 | 3.549 |
| ctx (16p/16K) | 3.008 | 3.310 | 4.133 |
| ctx (16p/64K) | 5.026 | 4.994 | 6.354 |
| page fault | 1.065 | 1.070 | 2.245 |

Table 2: LMbench OS benchmark timings for uniprocessor kernels – times in $\mu s$



# More Recent Benchmarks (2015)

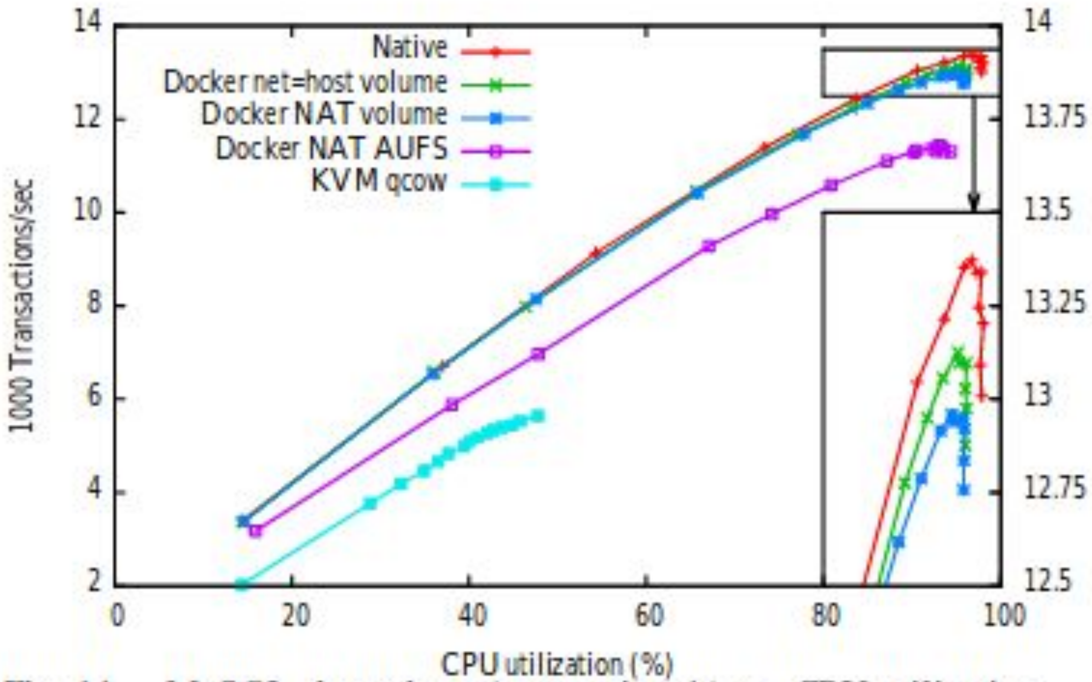



Fig. 11. MySQL throughput (transactions/s) vs. CPU utilization.

# References

http://linux-vserver.org/

Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., & Peterson, L. (2007). Container-based operating system virtualization. *ACM SIGOPS Operating Systems Review, 41*(3), 275. doi:10.1145/1272998.1273025

Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2015). An updated performance comparison of virtual machines and Linux containers. *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. doi:10.1109/ispass.2015.7095802

# Questions?