第 4 讲：Optimization of Virtual Machine Monitor

第一节：Introduction

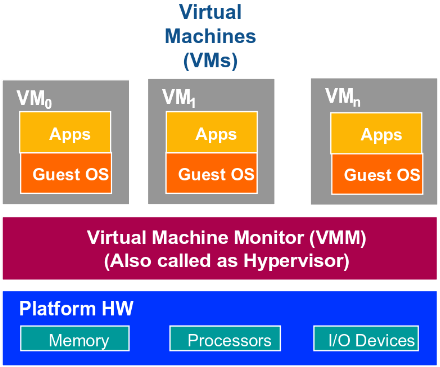
陈渝

清华大学计算机系

yuchen@tsinghua.edu.cn

2020 年 3 月 8 日

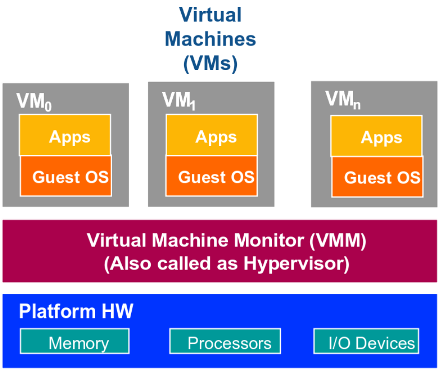
Introduction



Before optimization of VMM ...

- Where is the bottlenecks of VMM?
- How to find these bottlenecks?

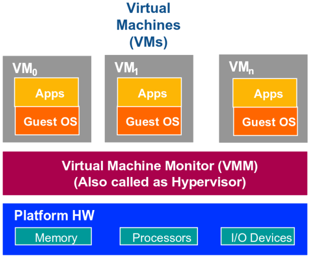
Introduction – Where is the bottlenecks of VMM?

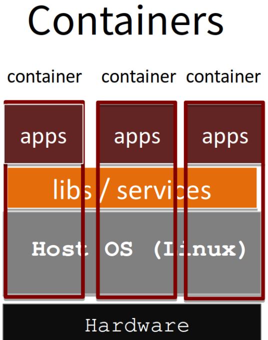


bottlenecks of VMM

- VCPU
- VMEM
- VIO
- etc...

Introduction – bottlenecks of VMM



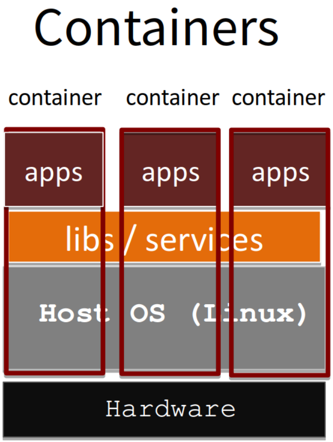


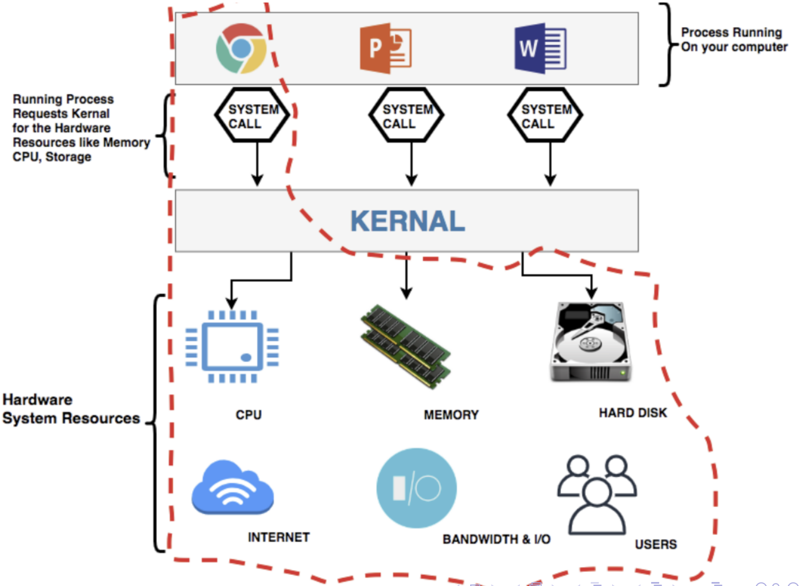
Comparison between VMM and Container

- Target: Docker v.s. KVM/XEN
- Benchmark: Environment and Workload
- Testing & Evaluation

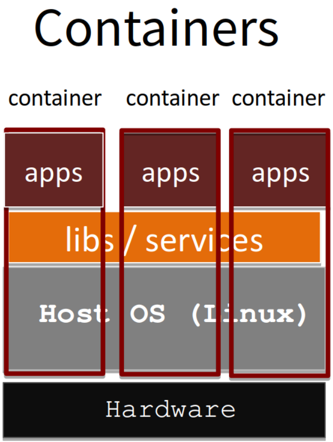
from: An Updated Performance Comparison of Virtual Machines and Linux Containers,TR of IBM, 2014
from: My VM is Lighter (and Safer) than your Container and Linux Containers,SOSP'17, 2017

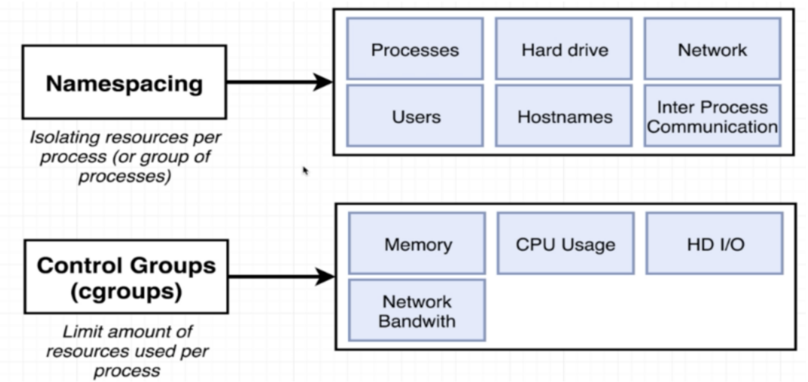
Introduction – Container overview



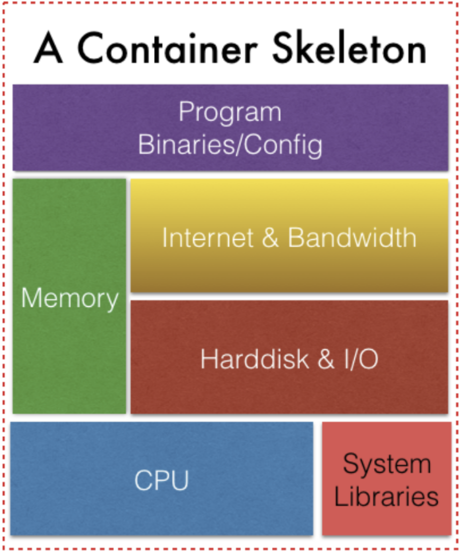


Introduction – Container overview





Introduction – Container overview



What is inside a Container

- Program Binaries/configuration
- Runtime libraries
- Dependency Products/tools
- A Piece of Kernel
- System Resources: CPU/MEM/IO/Net/Storage

We are isolating the program and delicately providing its own system resources and runtime libraries.

Introduction – bottlenecks of VMM – CPU/MEM

Workload		Native	Docker	KVM-untuned	KVM-tuned
PXZ (MB/s)		76.2 [±0.93]	73.5 (-4%) [±0.64]	59.2 (-22%) [±1.88]	62.2 (-18%) [±1.33]
Linpack (GFLOPS)		290.8 [±1.13]	290.9 (-0%) [±0.98]	241.3 (-17%) [±1.18]	284.2 (-2%) [±1.45]
RandomAccess (GUPS)		0.0126 [±0.00029]	0.0124 (-2%) [±0.00044]	0.0125 (-1%) [±0.00032]	Tuned run not warranted
Stream (GB/s)	Add	45.8 [±0.21]	45.6 (-0%) [±0.55]	45.0 (-2%) [±0.19]	
	Copy	41.3 [±0.06]	41.2 (-0%) [±0.08]	40.1 (-3%) [±0.21]	
	Scale	41.2 [±0.08]	41.2 (-0%) [±0.06]	40.0 (-3%) [±0.15]	
	Triad	45.6 [±0.12]	45.6 (-0%) [±0.49]	45.0 (-1%) [±0.20]	

Environment

- two E5-2665 (16 cores with HT), 256 GB of RAM
- Direct 10 Gbps Ethernet link between two Mellanox ConnectX-2 EN NICs
- Ubuntu 13.10, kernel 3.11, Docker 1.0, QEMU 1.5.0
- 32vCPU (Power management was disabled)

Introduction – bottlenecks of VMM – CPU/MEM

Workload		Native	Docker	KVM-untuned	KVM-tuned
PXZ (MB/s)		76.2 [±0.93]	73.5 (-4%) [±0.64]	59.2 (-22%) [±1.88]	62.2 (-18%) [±1.33]
Linpack (GFLOPS)		290.8 [±1.13]	290.9 (-0%) [±0.98]	241.3 (-17%) [±1.18]	284.2 (-2%) [±1.45]
RandomAccess (GUPS)		0.0126 [±0.00029]	0.0124 (-2%) [±0.00044]	0.0125 (-1%) [±0.00032]	Tuned run not warranted
Stream (GB/s)	Add	45.8 [±0.21]	45.6 (-0%) [±0.55]	45.0 (-2%) [±0.19]	
	Copy	41.3 [±0.06]	41.2 (-0%) [±0.08]	40.1 (-3%) [±0.21]	
	Scale	41.2 [±0.08]	41.2 (-0%) [±0.06]	40.0 (-3%) [±0.15]	
	Triad	45.6 [±0.12]	45.6 (-0%) [±0.49]	45.0 (-1%) [±0.20]	

Results

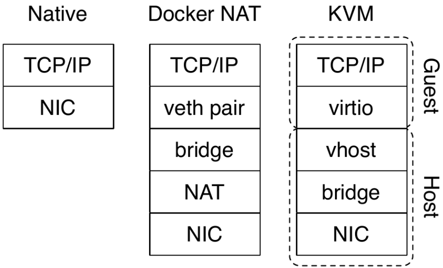
- PXZ: extra TLB pressure of nested paging.
- Linpack: pin vCPUs–> pCPU, expose the underlying cache topology
- STREAM: bandwidth of mem, cost of handling TLB misses, uses large pages

TABLE II. STREAM COMPONENTS

Name	Kernel	Bytes per iteration	FLOPS per iteration
COPY	$a[i] = b[i]$	16	0
SCALE	$a[i] = q * b[i]$	16	1
ADD	$a[i] = b[i] + c[i]$	24	1
TRIAD	$a[i] = b[i] + q * c[i]$	24	2

Introduction – bottlenecks of VMM – Net

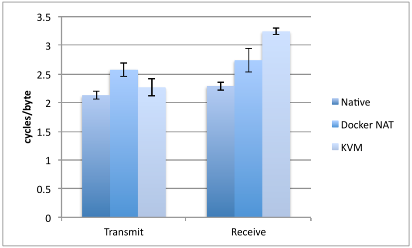
Network configurations



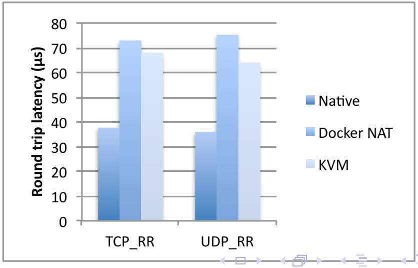
nuttcp to measure network bandwidth

- All three configurations reach 9.3 Gbps
- NAT noticeably increases overhead
- vhost reduces overhead
- NIC is the bottleneck

TCP bulk transfer efficiency (CPU cycles/byte)

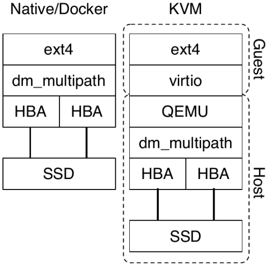


Network round-trip latency (µs)



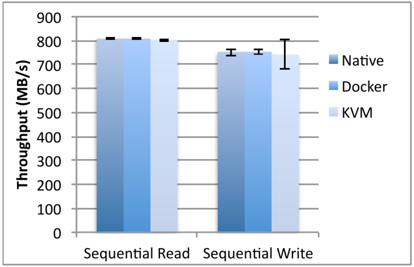
Introduction – bottlenecks of VMM – Storage

Storage configurations

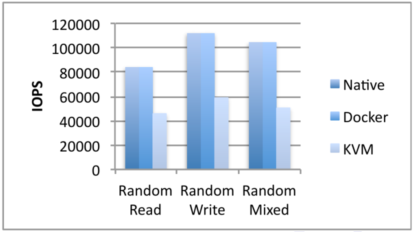


- 20 TB SSD, two 8 Gbps Fibre Channel links
- fio with the libaio backend in O DIRECT mode
- Fibre Channel HBA is the bottleneck

Sequential I/O throughput (MB/s)



Random I/O throughput (IOPS)

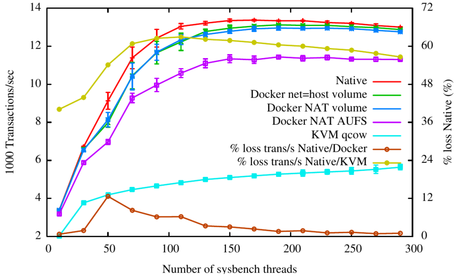


Introduction – bottlenecks of VMM – Real Apps

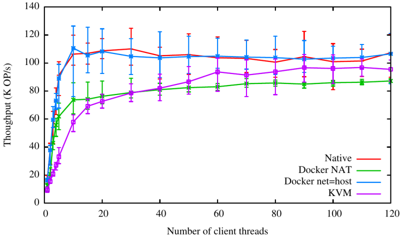
MySQL configurations

Configuration	Network (Figure 1)	Storage
Native	Native	Native
Docker net=host Volume	Native	Native
Docker NAT Volume	NAT	Native
Docker NAT AUFS	NAT	AUFS
KVM	vhost-net	virtio + qcow

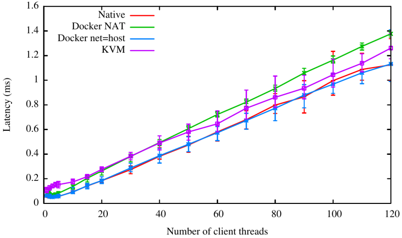
MySQL throughput (transactions/s)

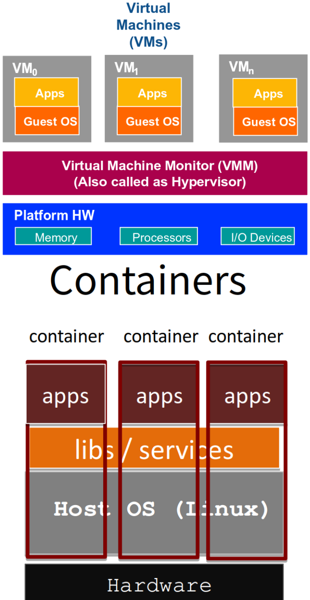


Redis performance (requests/s)



Average latency (in ms) of operations on Redis





Summary

- Containers and VMs impose almost no overhead on CPU and memory usage; they only impact I/O and OS interaction.
- This overhead comes in the form of extra cycles for each I/O operation, so small I/Os suffer much more than large ones.
- Several additional topics worthy of investigation: performance isolation when multiple workloads run on the same server, live resizing of containers and VMs, tradeoffs between scale-up and scale- out, and tradeoffs between live migration and restarting.
- https://github.com/thewmf/kvm-docker-comparison