A performance benchmarking analysis of Hypervisors, Containers and Unikernels on ARMv8 and x86 CPUs
EuCNC 2018, Ljubljana, Slovenia June 18-21
Authorship and sponsorship

**Virtual Open Systems** is a high-tech software company active in open source virtualization solutions and custom services for complex mixed-criticality automotive systems, NFV networking infrastructures, mobile devices and in general for embedded heterogeneous multicore systems around new generation processor architectures.

This work is done in the context of the H2020 “Next Generation Platform as a Service” project (www.ngpaas.eu).
Agenda

- Objectives
- Evaluated Solutions
  - Virtual Machines
  - Containers
  - Unikernels
- Benchmark configuration
- Benchmark results
- Conclusion
Agenda

➢ Objectives

➢ Evaluated Solutions
  – Virtual Machines
  – Containers
  – Unikernels

➢ Benchmark configuration

➢ Benchmark results

➢ Conclusion
Objectives

- Software Defined Network (SDN) and Network Function Virtualization (NFV) technologies are emerging in the Edge Computing

- Efficient virtualization technologies are becoming crucial

- New lightweight techniques (Containers, Unikernels) have emerged

- This work focuses on comparing the performance of open-source virtualization technologies on X86 and ARMv8
Agenda

- Objectives
- Evaluated Solutions
  - Virtual Machines
  - Containers
  - Unikernels
- Benchmark configuration
- Benchmark results
- Conclusion
Virtual Machines (VMs)

- Virtualization is a technology that allows to create multiple environments or dedicated resources from a single, physical hardware system.
- Software called a hypervisor connects directly to that hardware and allows to split one system into separate environments called Virtual Machines (VMs).
- Hypervisor solution benchmarked is KVM.
Kernel Virtual Machine (KVM)

- KVM is a full virtualization solution
- It makes Linux Kernel act as a Type-1 hypervisor
- KVM relies on user space tools like Quick Emulator (QEMU)
- QEMU is used to emulate and provide device abstractions
- KVM also provides support for paravirtual devices through Virtio, for better performance
**Containers** are a virtualization method for deploying and running distributed applications without launching an entire VM for each application.

- They depend on sharing the same base OS among themselves
- Loosely isolated
- Container engines benchmarked are **Docker** and **rkt**. 
Docker

- Most popularly used container engine
- Easy deployment and management of cloud applications
- Stable support for different architectures and different applications
- Uses libcontainer to take advantages of Linux namespaces and cgroups
rkt (pronounce rocket) has a security-minded approach as its primary distinguishing feature from Docker

- Has support for all "Docker Images"
- Has security features like:
  - Fetching container images as a non-root user
  - Option to use KVM or VM based isolation as stage 1
  - Support for SVirt in addition to a default SELinux policy
Unikernels

- Unikernels are specialized, single-address-space machine images constructed using library operating systems.
- Built by combining only the specialized application image and OS software parts required to support it.
- Size of the traditional VMs is reduced.
- Also use an Hypervisor (such as KVM), there are actually also VMs!
- Unikernel solutions benchmarked are Rumprun and OSv.
The **Rumprun** unikernel is based on the driver components of **rump kernels**

- Rump Kernel is derived by picking the desired components from the **NetBSD anykernel**
- Execute existing **POSIX** applications on KVM or Xen
- Doesn’t support `exec()` and `fork()` system calls
OSv uses the concept of a library OS to provide a Lightweight OS

Application threads and the kernel share the same address space to reduce overhead

Only stable architecture supported is x86, so far
Agenda

- Objectives
- Evaluated Solutions
  - Virtual Machines
  - Containers
  - Unikernels
- Benchmark configuration
- Benchmark results
- Conclusion
Benchmarking Tools

- CPU performance
  - Benchmarked using **SysBench**
- Memory bandwidth
  - Benchmarked using **STREAM**
- Network Bandwidth
  - Benchmarked using **Iperf**
Benchmarking Configuration

- **x86 64 bit platform**
  - Two Intel Xeon Processors E5-2623 v4
  - 8 cores @2.60GHz
  - Intel VT-x hardware virtualization extension
  - 32GB of DDR4 RAM

- **ARMv8 platform**
  - One Cavium ThunderX rev1 processor
  - 48 cores @2GHz
  - Hardware assisted virtualization extension
  - 128GB of DDR4 RAM
Agenda

- Objectives
- Evaluated Solutions
  - Virtual Machines
  - Containers
  - Unikernels
- Benchmark configuration
- Benchmark results
- Conclusion
CPU performance comparison

SysBench on an x86 server

➢ Rumprun provides near native performance

➢ Containers have 0.45% overhead

➢ KVM has 0.7% overhead

➢ OSv has the worst performance with 1.6% overhead
SysBench on an ARMv8 server

- KVM has a overhead of 0.8%
- Containers produce near-native performance
- Containers have very stable performance with negligible standard deviation
Memory Bandwidth comparison

STREAM on an x86 server with 1 thread

- Docker, rkt and Rumprun have negligible overhead
- OSv has a small overhead range of 0.6%-1.3%
- KVM has the maximum overhead range of 0.6%-1.6%
STREAM on an ARMv8 server with 1 thread

- KVM has overhead of about 2% for Copy and about 3% for Scale operations
- Containers induce no overhead
Memory Bandwidth comparison

STREAM on an ARMv8 server with 4 threads

➢ KVM overhead scales to above 3% in all the cases
➢ Containers induce no overhead
Memory Bandwidth comparison

STREAM on an ARMv8 server with 8 threads

- KVM overhead slightly increases further to 4%
- Containers continue to produce near-native performance
Network Bandwidth comparison

Iperf on an x86 server

➢ Docker, rkt and OSv provide the highest performance

➢ KVM comparatively is 80% less efficient

➢ Rumprun has terrible performance issues with a max bandwidth of just 1.37 Gbps
Iperf on an ARMv8 server

- **KVM performs better** than both the container engines
- Docker comparatively has a performance overhead of almost 15.6%
- **rkt** shows an overhead of 7.2% compared to KVM
Agenda

- Objectives
- Evaluated Solutions
  - Virtual Machines
  - Containers
  - Unikernels
- Benchmark configuration
- Benchmark results
- Conclusion
Conclusion

- **Unikernels** are still quite young and not production ready (no ARMv8 stable support), but are very promising

- **Containers** are generally the fastest and the easiest to deploy

- **KVM VMs** provide small CPU and memory overhead with a strong isolation
Future Work

- Extend to benchmarking other metrics like:
  - Security
  - Scalability

- Benchmark Unikernels on ARMv8 once they are fully compatible and stable

- Benchmark performance by launching containers inside VMs
THANK YOU!

contact@virtualopensystems.com
Web: virtualopensystems.com