Virtual Open Systems

KVM on ARM virtualization

An introduction

Bologna University – Faculty of Engineering, Dec. 2nd 2015

Michele Paolino

m.paolino@virtualopensystems.com

Supported by:

TAPPS

Presentation overview

- **Introduction to virtualization**
- KVM on ARM
- Lab session
- Company overview

What is virtualization?

Virtualization is the capability of a system to abstract the hardware and provide virtual instances of:

- CPUs, Memory, peripheral devices
- Operating Systems, applications
- Networks, network devices
- Storage, etc.

To some extent, this abstraction create also isolation

Why virtualization?

- It allows programmer/engineers to tinker and explore problems easier, which results in better and efficient solutions
- The implementation of the virtualization's abstraction requires a deep knowledge of the virtualized environment as well as of the host system
- Many challenges are still unsolved: performance, security, programmability and ease of use are requested!

Because it is a major driving force for modern technology, which to an extent indirectly affects many fields

When is virtualization used?

- Server consolidation
- Software development (e.g., non-x86 programming, continuous integration)
- Legacy systems support (e.g., banks, gaming[1], etc.)
- Networking (e.g., Software Defined Networking)
- Testing environments
- Isolation and Security
- Cloud infrastructure
- Network Function Virtualization (e.g., virtual switch, router, etc)

[1]http://www.xbox.com/en-GB/xbox-one/backward-compatibility

How does it work?

According to the targeted platform and to the type of virtualization, a particular piece of software is responsible for virtualization

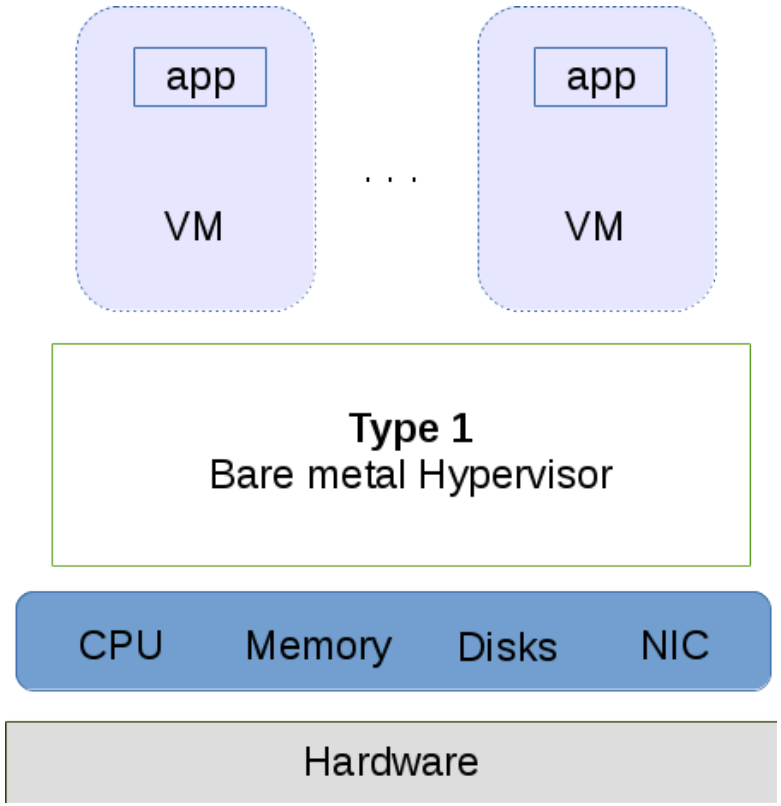
- In the case of Operating Systems virtualization, this software is called “Hypervisor” or Virtual Machine Monitor (VMM)
- It is responsible for:
  - Abstracting resources to hide the underlying hardware
  - Create/manage virtual machines
  - Schedule resources, etc.

Hypervisors

Type I: Bare-metal (or native)

Takes direct control of the hardware

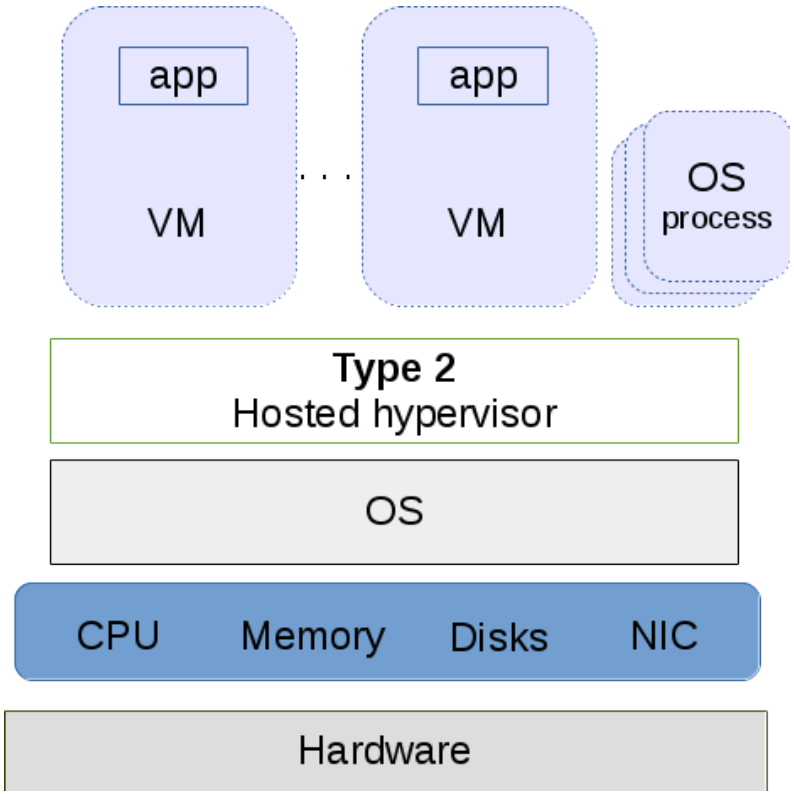
E.g., Xen, Microsoft Hyper-V



Type II: Hosted Hypervisor

Hypervisor is installed on the host OS

E.g., VMware workstation, VirtualBox



Types of virtualization

There are many ways to abstract/virtualize resources:

- Emulation
- Para-virtualization
- Hardware assisted virtualization
- Combination of all the above

Depending on the use case, different approaches end up having various advantages or disadvantages.

Key Features: Speed, Security, ease-of-use, hardware independence

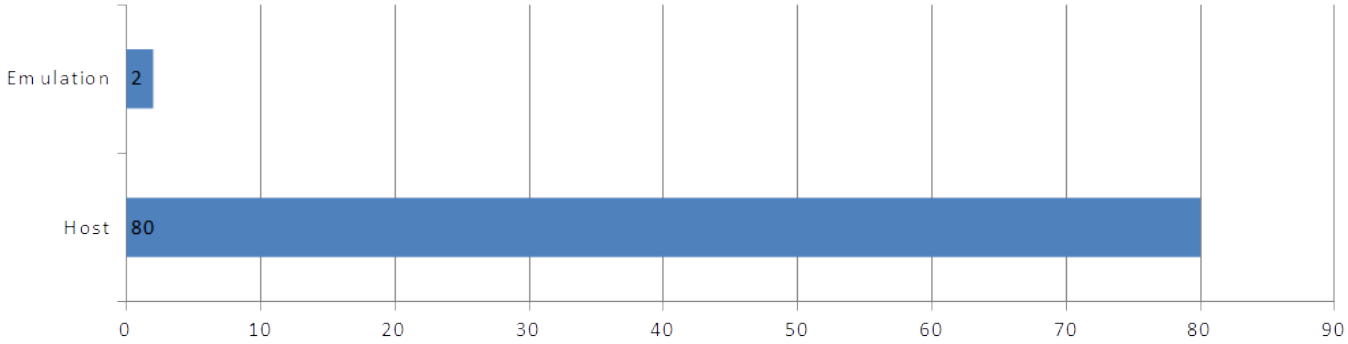
Virtualization challenges

To be compelling, virtualization needs to be near the native performance, while keeping strong abstraction/isolation

- Minimizing the virtualization overhead is *challenging*. I/O performance in particular is one of the most serious concern
- This overhead occurs mainly when the guest system tries to access the hardware, or does things that it shouldn't, e.g., trap and emulate:
  1) Guest tries to communicate with an emulated device
  2) Hypervisor has to pause the guest execution
  3) Emulate the device behavior
  4) Resume the guest
- This cycle is repeating over and over again
- More complex emulation = larger virtualization overhead

Virtualization challenges: an example

The impact that the trap and emulate strategy has on the network controller performance is really significant.



The emulated network interface causes far too many VM exits, stressing the CPU for the actual emulation instead of the I/O requests.

Presentation overview

- Introduction to virtualization
- **KVM on ARM**
- Lab session
- Company overview

KVM/ARM introduction

Initially the ARM architecture didn’t support virtualization

- From ARMv7-A and beyond, hardware virtualization support was introduced
- Later, ARMv8-A moved the architecture to 64-bits (backwards compatible 32-bits) with more feature rich virtualization features
- KVM on ARM is supported in upstream Linux since version 3.9. *Virtual Open Systems developed this support*
- CPU, memory, interrupts and timers virtualization is supported.

ARM architecture

ARM (Acorn/Advanced Risc Machines) was born in the 80' as a fabless RISC processors designer:

- Today ARM empowers smartphones, game consoles, tablet, wearables and smartwatches, drones, cars, connected objects, etc.

- Three main CPU profiles:
  - A application
  - R real-time
  - M microcontroller
- 32/64 bit
- Harward architecture

Image source: http://labs.sogeti.com/the-year-of-the-wearables/#more-55302

ARM virtualization extensions

The first ARM CPU with Virtualization Extensions (VE) presented by ARM is Cortex A15. The ARM VE introduces:

- *CPU*: New *Hyp* execution mode introduced as a trap-and-emulate mechanism
  - It allows instructions to be configured to trap directly into a VM's kernel mode instead of going through Hyp mode
- *Memory*: Extended from 32 to 40bits, with two translation stages (VA → IPA, IPA → PA) for the MMU
- *Interrupts*: The Generic Interrupt Controller (GIC) is virtualized in hardware

KVM introduction

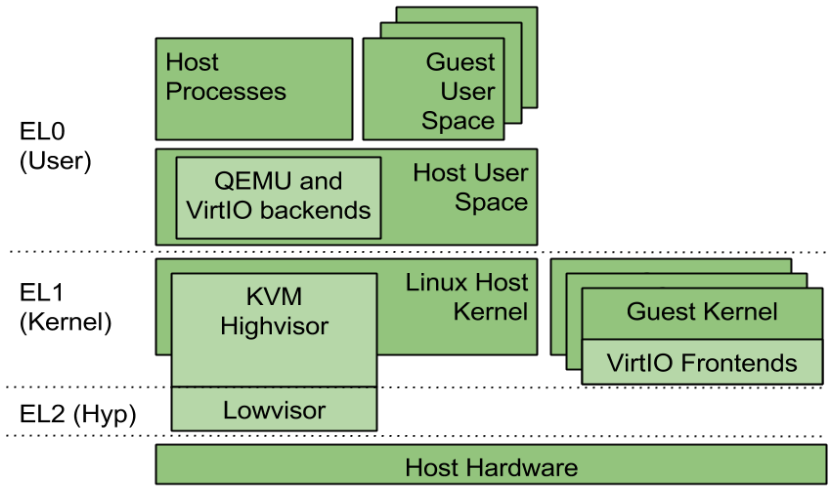
Kernel-based Virtual Machine (KVM) is an open source full virtualization solution

- Incorporated into mainline Linux Kernel from v2.6.20 (x86)
- Ported to ARM in 2012 by Virtual Open Systems (Linux 3.9)
- It is unique in that it turns the Linux kernel into a full-blown hypervisor
- Instead of re-implementing complex scheduling and memory management concepts, KVM relies on the well-tried and tested Linux infrastructure

KVM-QEMU Architecture

KVM is a Linux kernel module, which can be used by any user space application (ioctl interface) to create new virtualized instances of the CPU/memory

- Support for hardware virtualization
- High performance and scalability
- Paravirtualized IO drivers for disks, network, balloon, etc.
- Used in conjunction with userspace tools (**QEMU**, kvmtool)



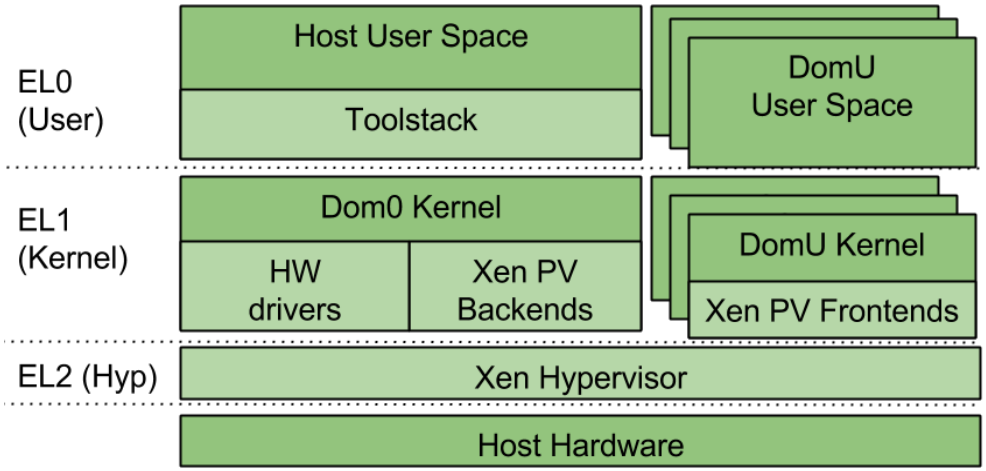
KVM/ARM QEMU

Quick Emulator (QEMU) is not only a full featured emulator solution. It is also used to implement the user-space backed for KVM:

- Emulates guest hardware devices (e.g., USB controller, audio and video, serial port, etc.)
- Provides paravitualization (virtio)
- Instantiate VMs
- Interact with the cloud (e.g., OpenStack)

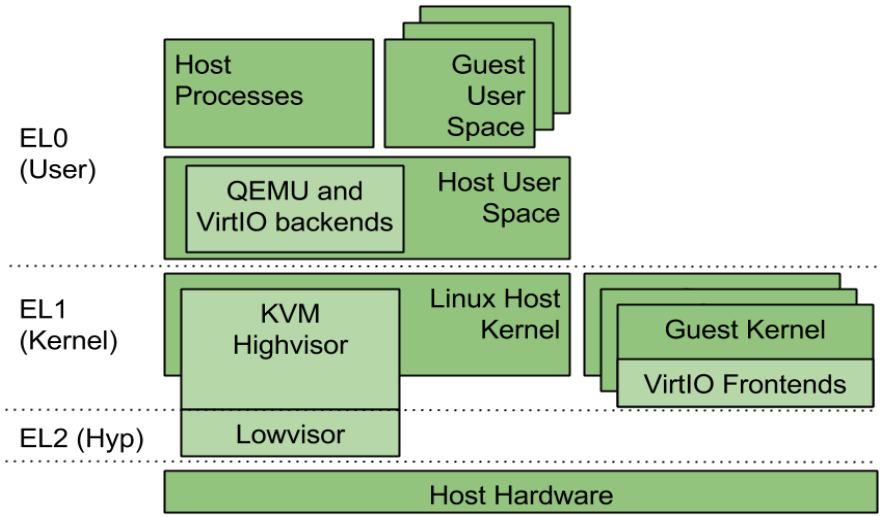
XEN vs KVM Architecture

XEN



Implements from scratch the Hypervisor features (scheduling, memory management, etc.)

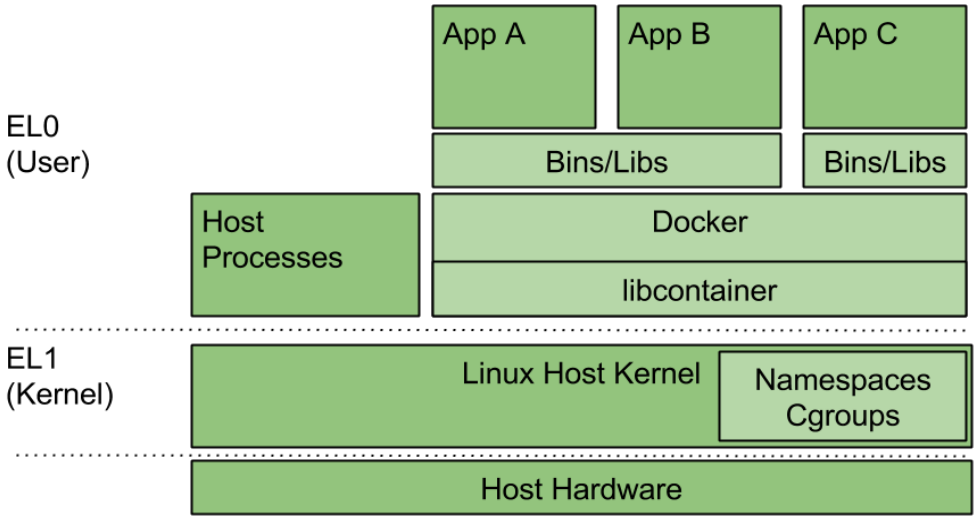
KVM



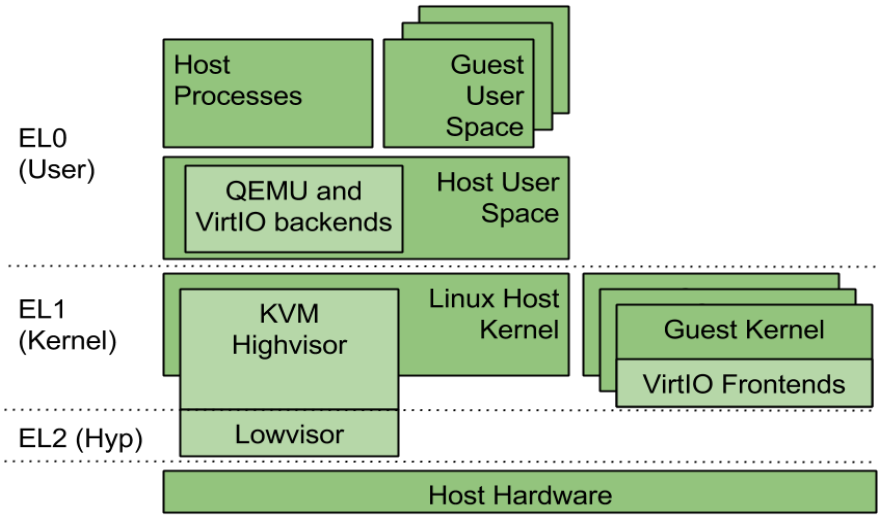
Leverages the Linux kernel for: scheduling, memory management, VM execution and control, etc.

Docker vs KVM Architecture

Docker containers



KVM



Read-only file system (Union) as a basis, a write layer added on top of it, so that containers can commit their changes from the original file system

Leverages the Linux kernel for: scheduling, memory management, VM execution and control, etc.

Presentation overview

- Introduction to virtualization
- KVM on ARM
- **Lab session**
- Company overview

Introduction

During this practical exercise, a KVM/ARM virtual machine will be run on top of an ARMv8 Model/Emulator.

- Installation of the cross compiler tool
- Installation of ARM Foundation Model
- Compilation of Linux kernel for ARM

Cross Compilers

A cross compiler is used when there is a need to compile code for a platform different from the one in which the compiler is running

- gcc (GNU compiler collection) is the most used in open source programming
- It supports many operating systems, CPU architectures (arm, ppc, etc.), and programming languages (C, Java, Ada, Go, etc.)
- It will be installed using Operating System repositories (i.e., apt-get install gcc-aarch64-linux-gnu)

Foundation/FastModels

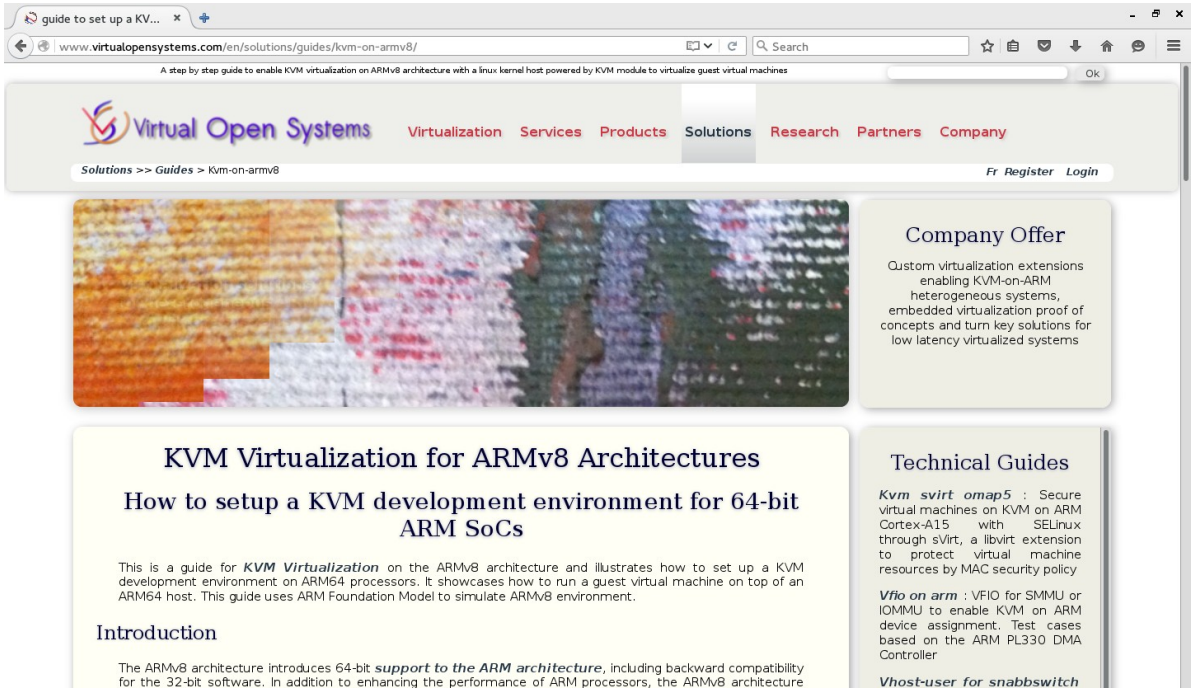
It is mainly used for early prototyping prior to the silicon availability

- Supports ARMv7 and ARMv8
- Gives the possibility to extend/create new hardware platforms with custom devices (System C), system buses, processors, etc.
- It is not cycle accurate
- FastModels has a library of ARM components which can be used to model new platforms
- FastModels provides a very well designed debugger

Lab Session

1) Create a new account on www.virtualopensystems.com

2) follow the guide: http://www.virtualopensystems.com/en/solutions/guides/kvm-on-armv8/



How to run an ARMv8 virtual machine

1) Create an account on the VOSYS website (http://www.virtualopensystems.com/)

2) Create an account on the ARM website

3) Get the 64-bit ARM Cross Compiler

4) Install ARMv8 Foundation Model (https://silver.arm.com/download/download.tm?pv=2033755)

5) Host and Guest 64-bit Userspace Images (https://releases.linaro.org/14.02/openembedded/aarch64/linaro-image-minimal-genericarmv8-20140223-649.rootfs.tar.gz)

6)Compile ARM64 Host and Guest Kernels (git clone git://github.com/virtualopensystems/linux-kvm-arm.git -b kvm-arm64-stable)

7)Download bootwrapper (git clone git://github.com/virtualopensystems/boot-wrapper.git -b aarch64)

Presentation overview

- Introduction to virtualization
- KVM on ARM
- Lab session
- **Company overview**

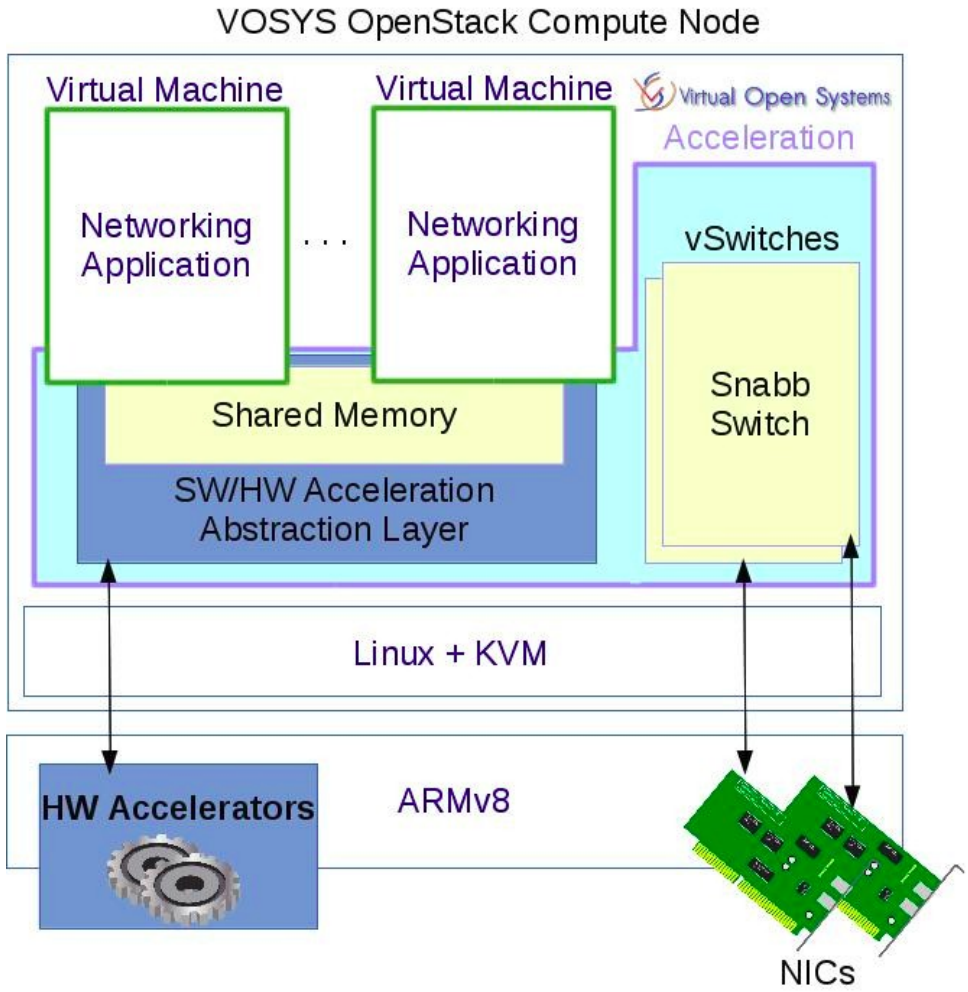
Virtual Open Systems overview

Company Overview

- Virtual Open Systems (VOSYS) is a French fully independent private high-tech start-up company created in Jan 2011.
- The core activity is on custom virtualization solutions for complex mixed criticality, heterogeneous multi-core ARM based SoCs.
- Virtual Open Systems is active in open source software development for networking, automotive and mobile.
- Among the open source contributions, there are:
  - Linux kernel / KVM / VFIO
  - QEMU / vhost-user / libvirt / OpenStack
- User space virtual switch (SnabbSwitch) for NFV.
- Member of: ETSI NFV, OVA, Linux Foundation, Automotive Grade Linux; HSA and MEC (submitted application)

Virtual Open Systems: Mastered Technologies

- KVM, VFIO
- Shared memory
- QoS in virtualized systems
- API Remoting
- GPU virtualization
- FPGA Accelerators virtualization
- Security in virtualized systems
- High performance virtual switch
- VNFs acceleration
- Accelerated OpenStack NFV compute node
- Virtual switches and networking



Virtual Open Systems: research activities

Virtual Open Systems is involved in different research activities:

- High performance computing and QoS

  (http://www.virtualopensystems.com/en/solutions/demos/virtual-bfq-action/)

- Networking Virtualization and NFV
(http://www.virtualopensystems.com/en/solutions/demos/vosyswitch-perf-openstack-integration/)

- Virtual Machines and hypervisor security

- Virtualization of hardware accelerators

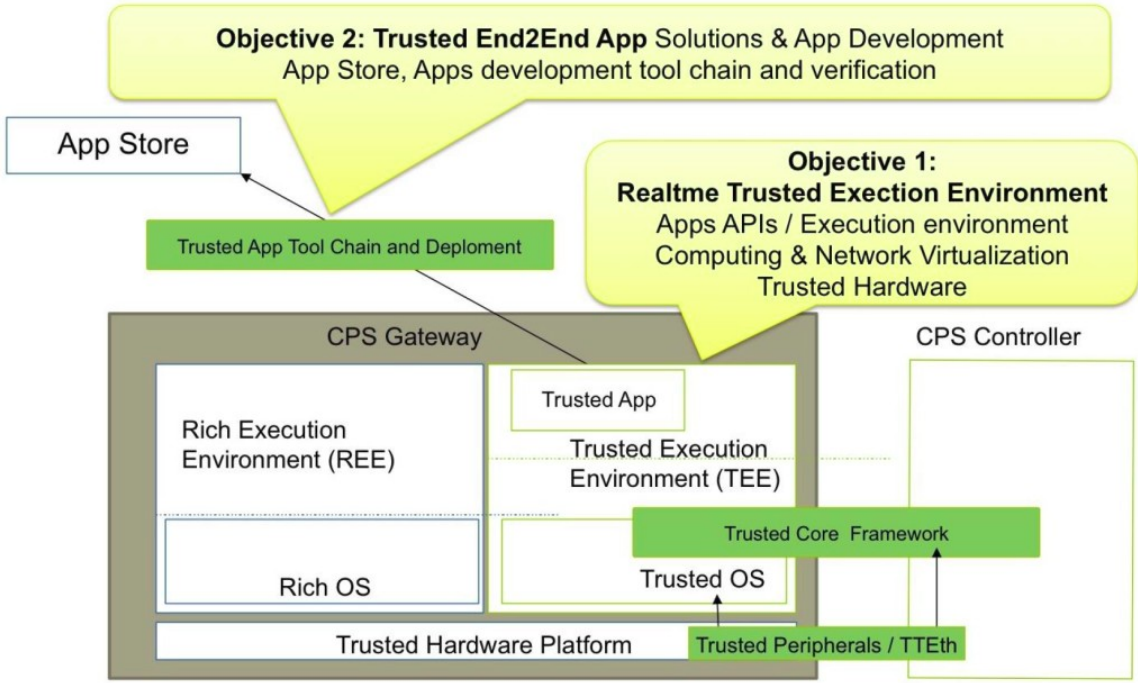
  (http://www.virtualopensystems.com/en/solutions/demos/vosyshmem-api-remoting/)

- Automotive and Dual OS systems

Virtual Open Systems Automotive research: the TAPPS project

The TAPPS project (http://www.tapps.eservices4life.org/) aims to combine applications with different criticality on the same hardware platform in automotive and healthcare, e.g.,:

- Brake control system and infotainment
- Nurses assistance and patients information



Call for thesis/stage

Virtual Open Systems proposes master thesis/stages in the areas of computing and networking virtualization coupled with hardware accelerators and security

More info at: www.virtualopensystems.com

Thank you

m.paolino@virtualopensystems.com

Virtual Open Systems