ARM TrustZone and KVM Coexistence with RTOS For Automotive

Michele Paolino
m.paolino@virtualopensystems.com

Automotive-grade Linux Summit,
2015-06-01, Tokyo, Japan

Authorship and sponsorship

Michele Paolino, software architect at Virtual Open Systems (VOSYS). His experience includes Linux kernel drivers, KVM hypervisor, QEMU programming, libvirt, API remoting, GP/GPU, TrustZone security and OpenStack.

Virtual Open Systems is a high-tech start-up company active in open source virtualization solutions and custom services for complex mixed-criticality automotive, NFV networking infrastructures, consumer electronics, 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 Trusted APPs for CPS (TAPPS) project (www.tapps-project.eu).

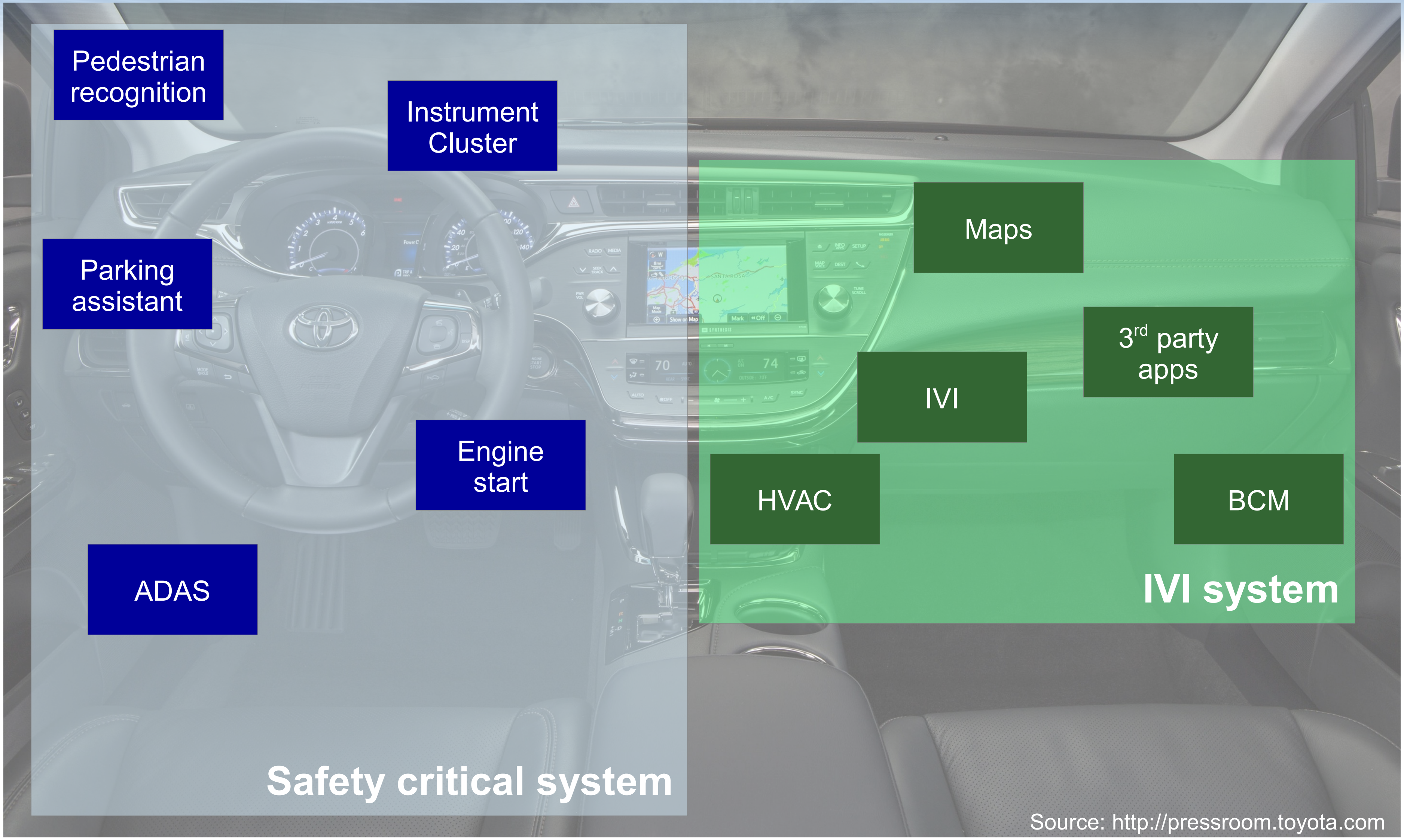
- **State of the art**
- Beyond the state of the art
- Status of the work and benchmark
- Next steps
- Conclusion

Introduction

Cars are getting smarter and always connected, combining safety critical applications with In-Vehicle Infotainment (IVI):

- Instrument Cluster management
- Park assistant, Heating, Ventilating, and Air Conditioning (HVAC)
- Advanced Driver Assistance Systems (ADAS)
- Key-less engine start, Body Control Module (BCM)
- Web browsing and social networking
- Internet of Things (IoT), Cyber Physical Systems (CPS) and cloud computing services
- Third party Apps (Maps, Games, Video, etc.)

Connected cars



Source: http://pressroom.toyota.com

The challenge

Such a concept of the future car, brings new and unprecedented challenges to the automotive industry:

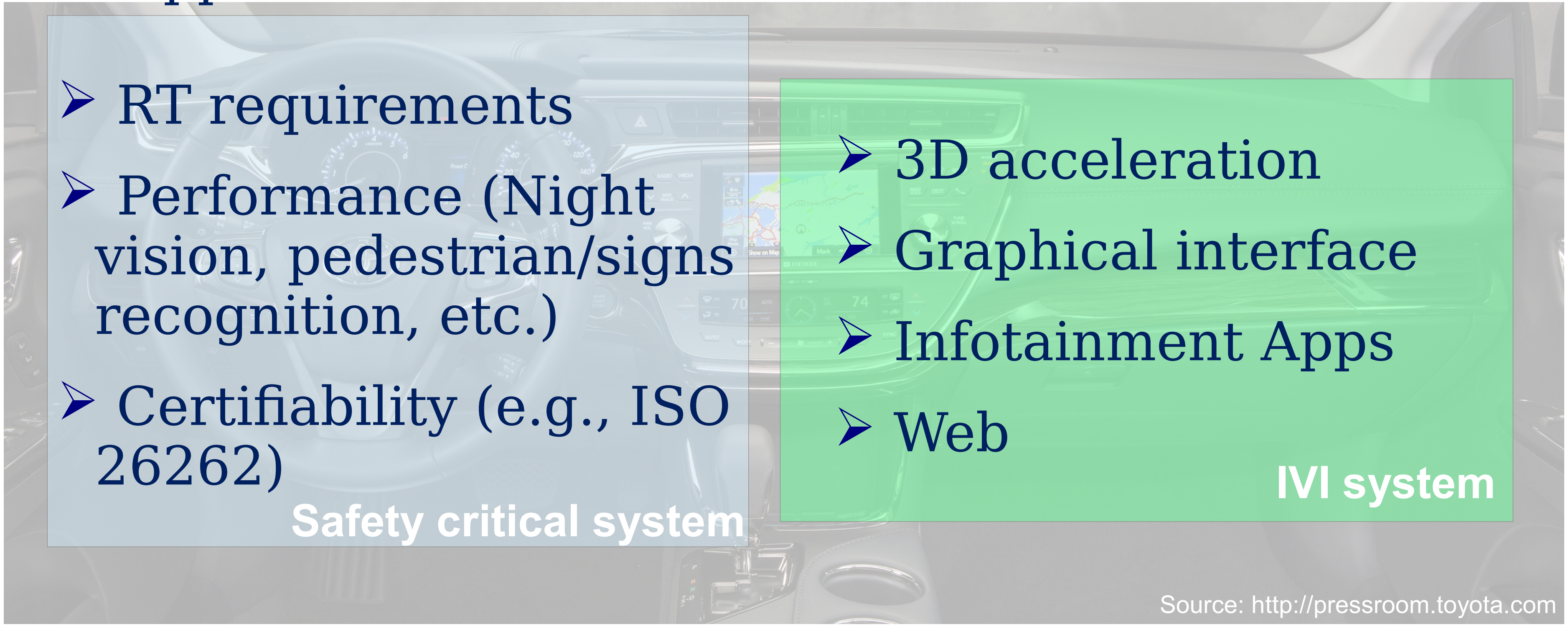
- Mixed criticality environment with RT requirements
- Security and trustworthiness of the software
- Secure the connected IVI environment

  (Apps installation, web browsing, shared devices, etc.)
- High performance

  (object recognition, DRM encoding, 3d acceleration, etc.)

State of the art

Today's cars are addressing these challenges by means of two platforms, one for the IVI and the other for safety critical applications:



- RT requirements
- Performance (Night vision, pedestrian/signs recognition, etc.)
- Certifiability (e.g., ISO 26262)

Safety critical system

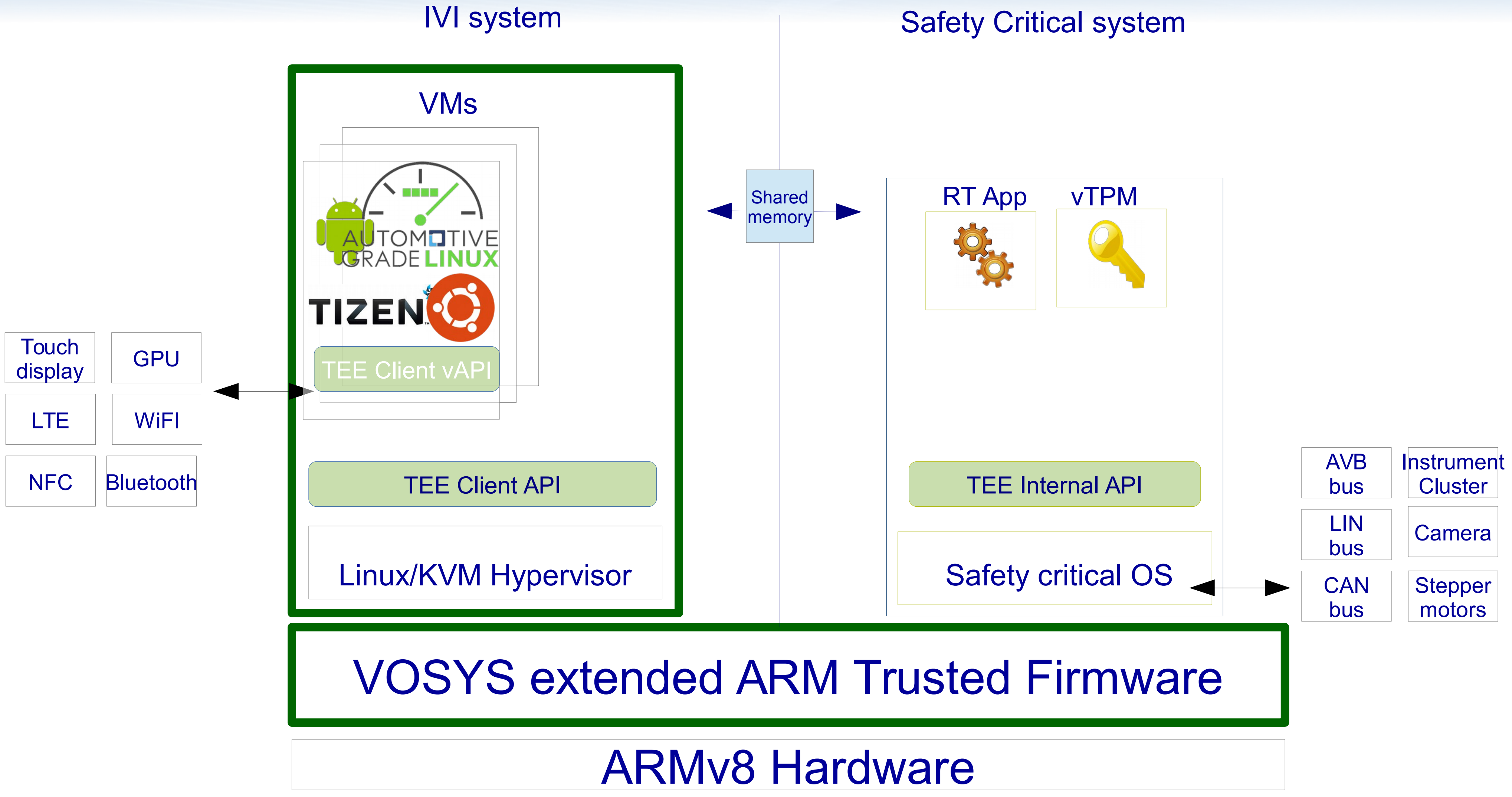
- 3D acceleration
- Graphical interface
- Infotainment Apps
- Web

IVI system

Source: http://pressroom.toyota.com

Two platforms are costly (hardware, cabling, space, weight etc.) and difficult to maintain/extend.

Beyond the state of the art



Beyond the State of the art (2)

Extending open source projects and innovative technologies it is possible to run virtualized IVI and safe-critical systems on the same hardware, by means of:

- **Hardware accelerated virtualization**

  (performance, isolation, OS concurrency)
- **ARM TrustZone**

  (security, trusted computing, mixed criticality)
- **Real Time Operating System support**

  (safety critical functions)

Hardware accelerated virtualization

Virtualization enables the execution of different operating systems concurrently.

- VMs isolation
- Support for multiple IVI guests: AGL distro, Android, Ubuntu, Tizen, etc.
- Direct hardware assignment and support for hardware acceleration (FPGA, GPU, DSP, etc.)
- VMs support for LTE and AVB/CAN bus connections
- Migration, over the air updates, App Store

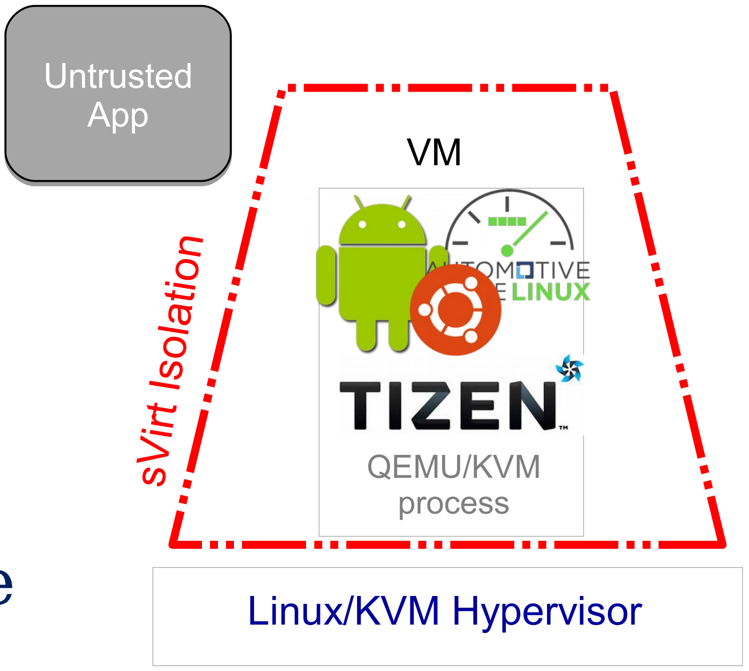
Hardware accelerated virtualization: KVM and SELinux isolation

Full Virtualization is the ability of a system to run different partitions concurrently with unmodified software.

VMs that exploit the CPU virtualization extensions (i.e. ARM VE, Intel VT), are hardware isolated regarding:

- Memory
- Interrupts
- Exceptions

sVirt uses security kernel features like SELinux to go beyond Discretionary Access Control, using the Mandatory Access Control security policy to isolate VMs.



Hardware accelerated virtualization: Virtualized AGL/Tizen support

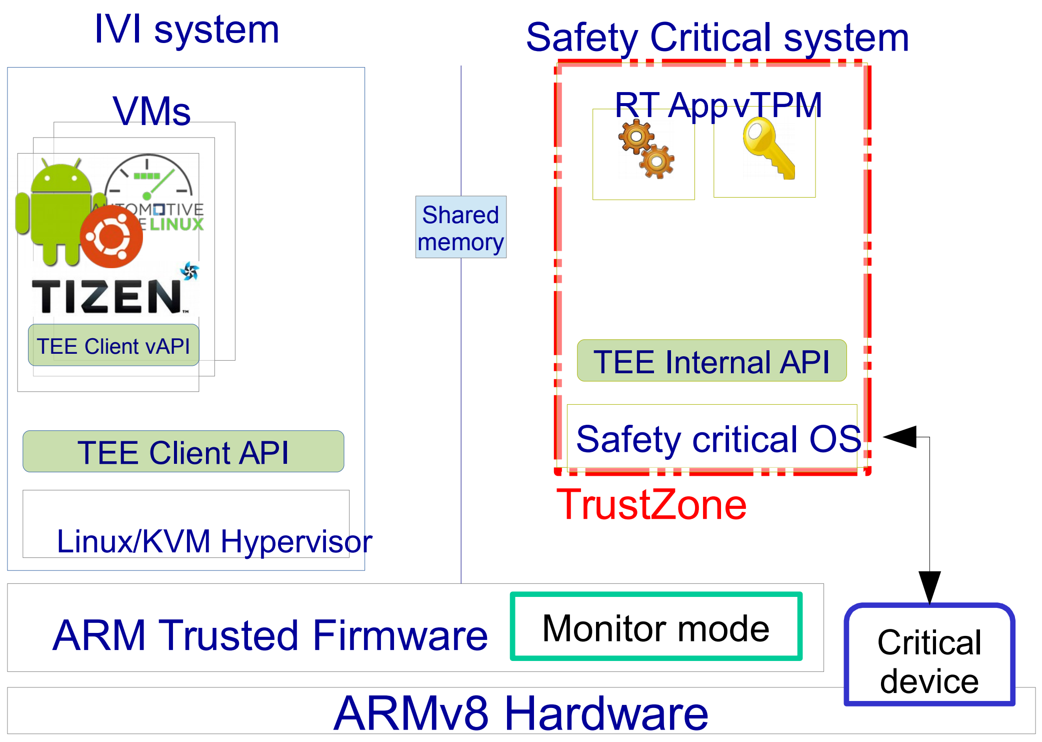
The AGL community is now working to build an AGL distribution. One of the points in the agenda is the support of QEMU, whose challenges on ARM are:

- Video acceleration (OpenGL)
- QEMU ARM audio support
- Short boot time
- Security and secure interaction with the safe critical OS

ARM TrustZone

TrustZone safely runs two OSes by defining a secure operational mode completely isolated from the rest of the system:

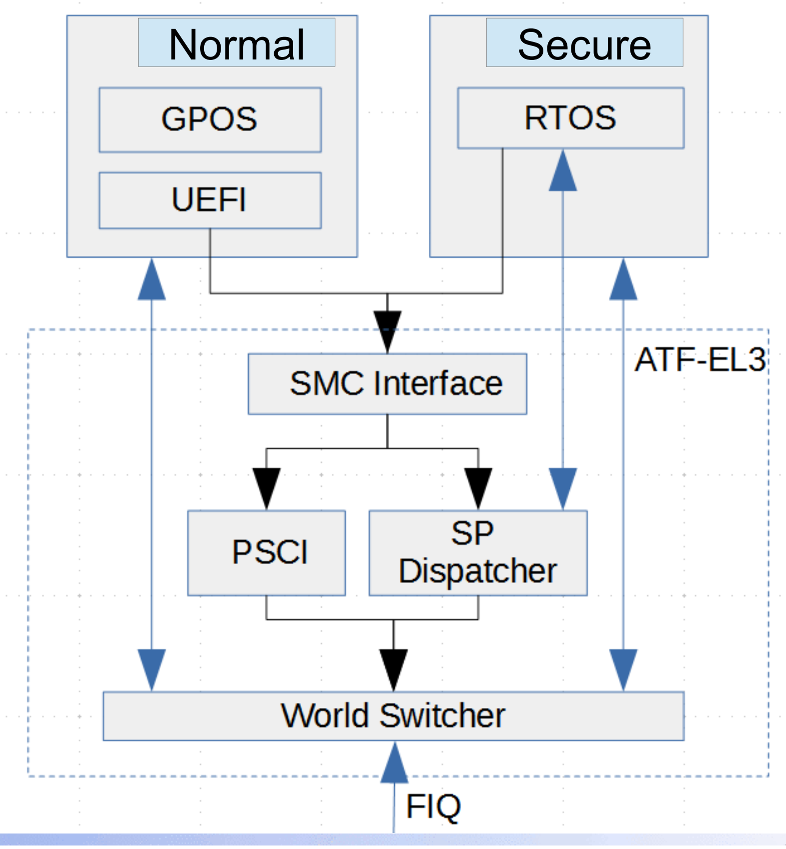
- The two OSes are fully independent
  - if the IVI part crashes, the safety critical OS runs normally
- TrustZone implements a secure context switch mechanism through the TrustZone Monitor



TrustZone: ARM Trusted Firmware

Arm Trusted Firmware (ATF) includes a Secure Monitor (EL3) software implementation for ARMv8-A platforms, which handles the boot procedure and interrupts

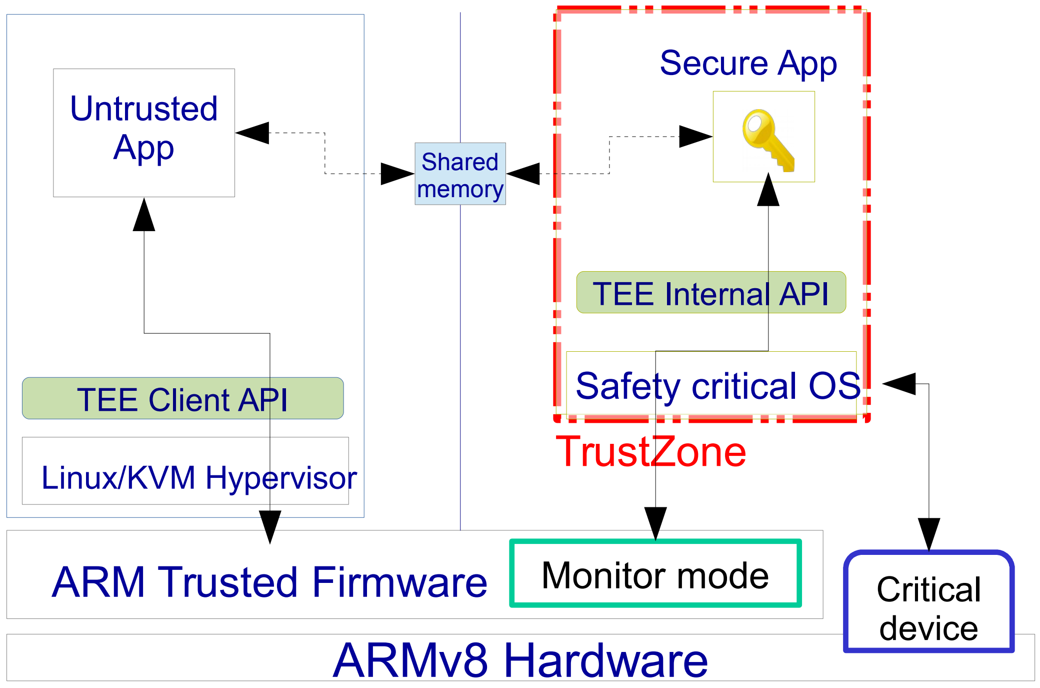
- Modular design
- Secure world Initialization
- S-EL1 payload dispatcher
- Initialize Secure/Normal world isolation
- SMC (Secure monitor Call) Handling
- PSCI for secondary core bring-up



TrustZone: GlobalPlatform TEE

GlobalPlatform's Trusted Execution Environment (TEE) is a secure area that guarantees that sensitive data are stored, processed and protected in a trusted environment.

- The TEE APIs are standardized set of APIs for Trusted Execution Environment
  - The TEE Client API
  - The TEE Internal API



RTOS support

The proposed architecture, enables the execution of custom or legacy operating systems in the safety critical system:

- FreeRTOS, ARC CORE, QNX, etc.
- The safety critical system boots before the IVI system, and is able to control and attest its execution.
- The isolation provided by TrustZone to the RTOS is based on security hardware extensions (e.g., cpu signals)
- System critical devices are allocated exclusively to the RTOS and are not visible to the IVI system

RTOS support: why TrustZone?

The other approaches which aim to the integration of a safety critical OS with IVI use virtualization to isolate the two systems:

- Is available in all the latest SoCs
- Isolates in hardware virtual machines
- Is a well-known and mature technology
- Supports the execution of many OSes concurrently
- Examples are: XEN Automotive Hypervisor and QNX hypervisor

- But..

RTOS support: why TrustZone?

The other approaches which aim to the integration of a safety critical OS with IVI use virtualization to isolate the two systems:

- Is available in all the latest SoCs
- Isolates in hardware virtual machines
- Is a well-known and mature technology
- Supports the execution of many OSes concurrently
- Examples are: XEN Automotive Hypervisor and QNX hypervisor

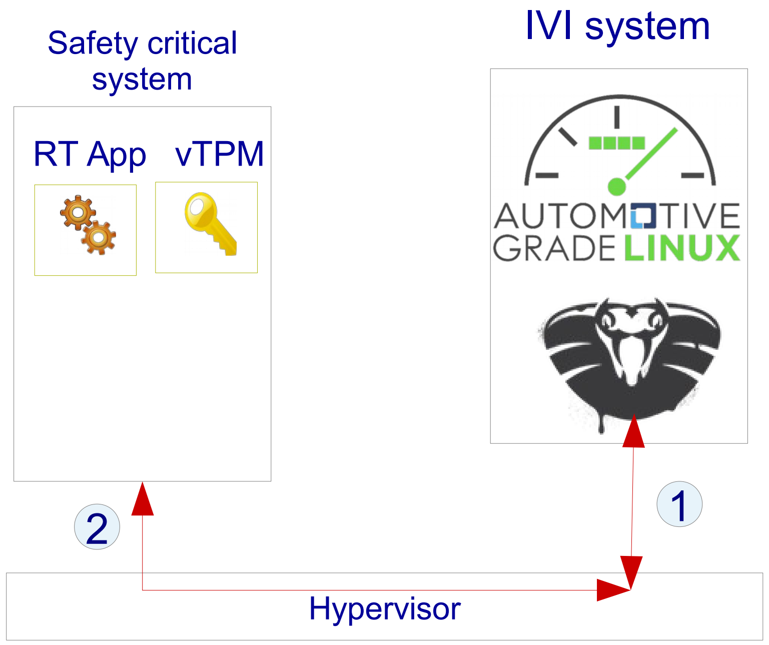
- But..

Source: http://venom.crowdstrike.com/

RTOS support: the VENOM vulnerability

Virtualization is cheap and provides nice features for automotive, but it could have important security problems

- VENOM, CVE-2015-3456, is a security vulnerability in the QEMU virtual floppy drive
- It allows an attacker to escape from the VM isolation (step 1)
- VENOM could open access to the host and all other VMs, potentially giving adversaries significant elevated access to the adjacent systems (step 2)



Virtual Open Systems

Proof of Concept: current status

A prototype of the proposed architecture can be seen in the booth area of the Tokyo ALS2015.

- Runs on ARMv8 Hardware with Security and Virtualization extensions
  - Juno development board r0
  - ARMv8 Dual Cortex-A57@800MHz + Quad Cortex-A53@700MHz

- Software:
  - Normal World: Linux v4.1/KVM
  - Secure world: FreeRTOS/bare metal
  - TrustZone Monitor: VOSYS ARM Trusted Firmware

Virtual Open Systems

VOSYS extended ARM Trusted Firmware

By default, the ATF reference world switch procedure is time-triggered, Virtual Open Systems has extended ATF to:

- Quantify the world switch and FreeRTOS latency
- Give an insight in the communication overhead between RTOS/GPOS and VM/RTOS
- Modify the world switch procedure to respond to event-triggered signals
- Make usage of the Performance Monitoring Unit (PMU) to have a very detailed view of latency in terms of clock cycles

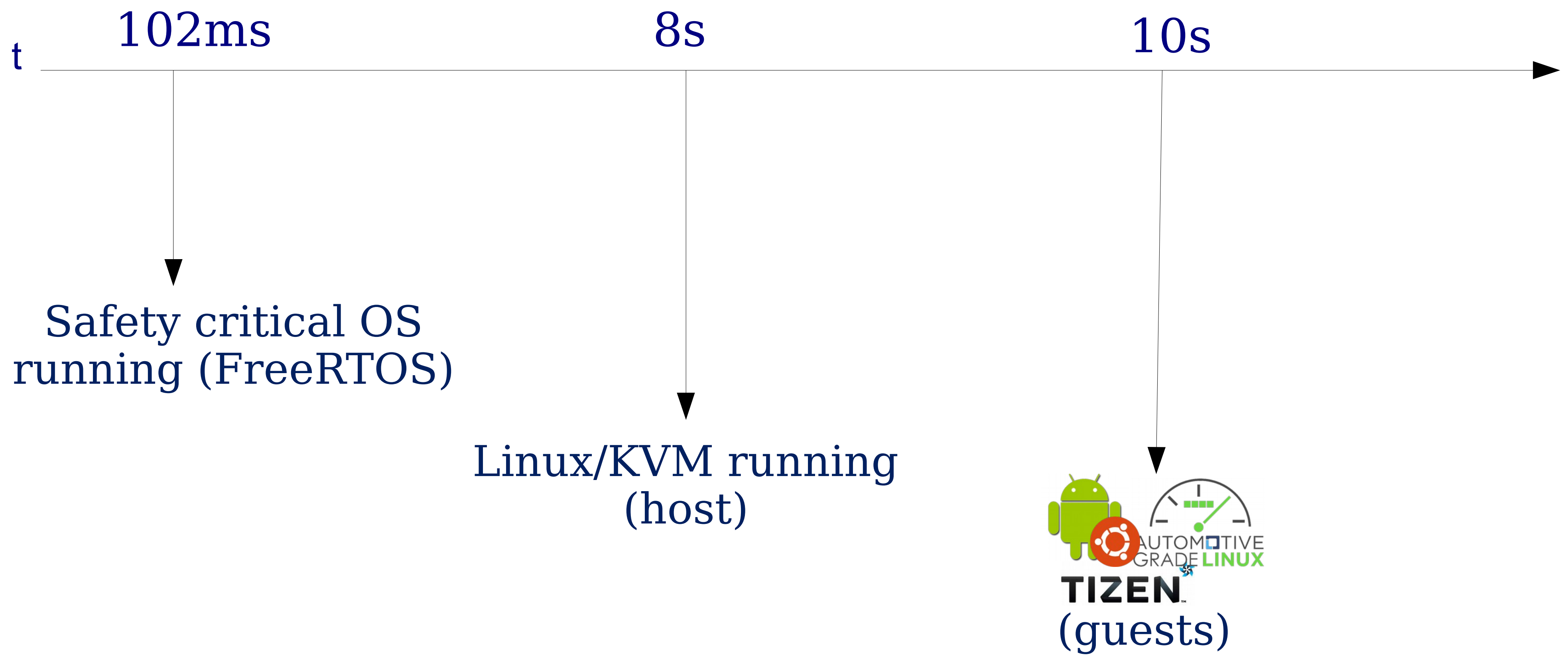
Performance measurements

Different performance measurements have been performed on the proof of concept presented at the ALS2015:

- Boot time test
- Interrupt Latency tests, which aim to assess the RTOS performance
- World switch latency, to measure the overhead introduced by TrustZone

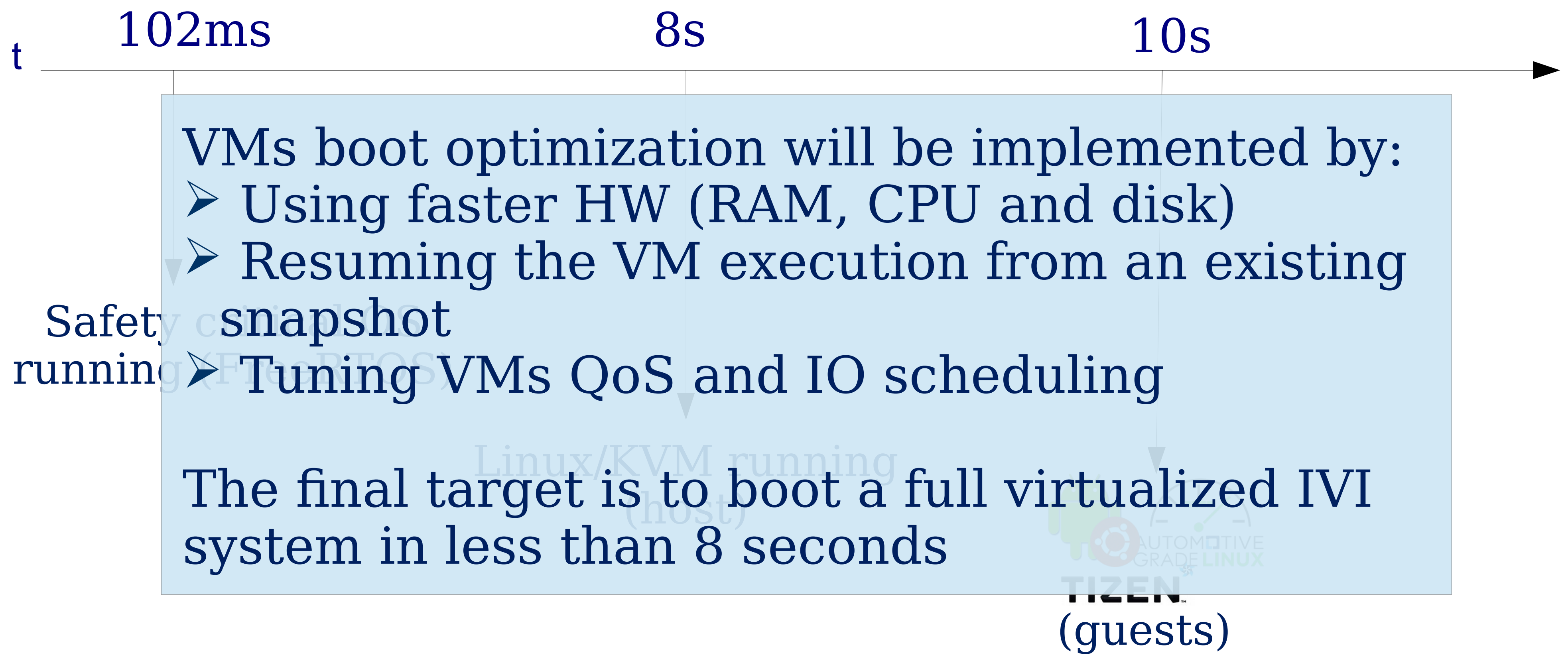
Boot time

Boot time is a critical factor both for user experience and for certifiability. CPU frequency is 800MHz for Cortex A57 and 700MHz for Cortex A53



Boot time

Boot time is a critical factor both for user experience and for certifiability. CPU frequency is 800MHz for Cortex A57 and 700MHz for Cortex A53



VMs boot optimization will be implemented by:

- Using faster HW (RAM, CPU and disk)
- Resuming the VM execution from an existing snapshot
- Tuning VMs QoS and IO scheduling

The final target is to boot a full virtualized IVI system in less than 8 seconds

Interrupt latency

The interrupt latency of the safety critical OS has been measured in the following cases:

- The interrupt arrives when the CPU is executing the safety critical OS (best case)
- The interrupt arrives when the CPU is running the IVI virtualized system (worst case)

	Description	Clock cycles	Time @700Mhz
Best case	Secure → Secure World	~1900	~2.7us
Worst case	Non Secure → Secure	~2500	~3.5us

World switch latency

The combination of ARM Virtualization and Security extensions adds isolation which provides high security, but at a cost of additional overhead when a VM wants to communicate with the Security critical OS:

- Secure Monitor Call (SMC) is the instruction defined by ARM ISA to jump from the Non Secure to the Secure World and vice versa.
- KVM traps the SMC call from a guest and forwards the request to the Secure World

World switch latency (2)

Following the path of a Secure Monitor Call (SMC) from the guest to the Secure World, there is a need to consider:

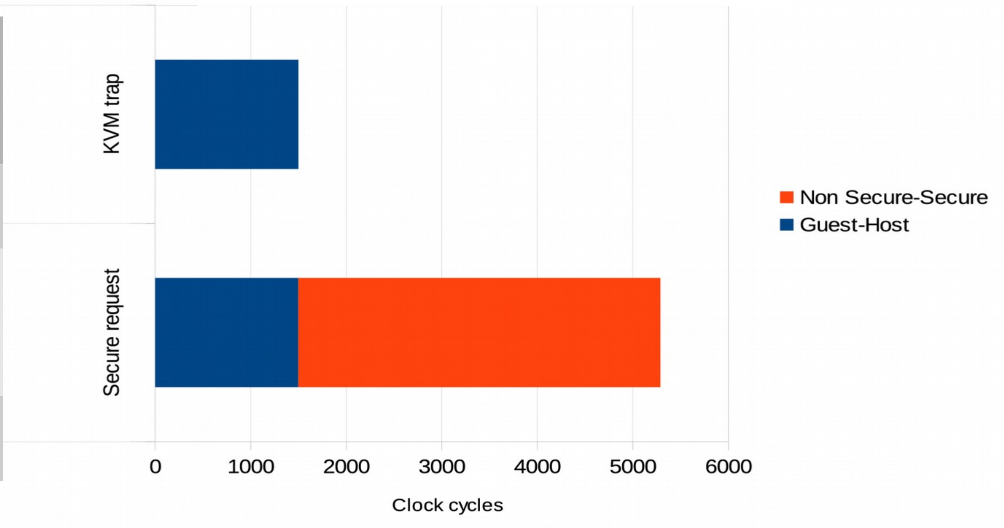
- Guest-Host context switch (SMC hypervisor trap)
  - Measured with a bare metal QEMU/KVM application which interacts with the Performance Monitoring Unit (PMU)
- Secure-Non Secure world context switch
  - Measured through a kernel module which executes the SMC assembly instruction causing a world switch into the secure World

World switch latency (3)

The performed tests aim to measure the minimal round-trip:

- the Secure World payload here is a bare metal application

	Clock cycles*	Time@700 MHz
Guest Host	~1400	~2us
Non Secure - Secure	~3700	~5.2us
Round-trip	**~5100**	~7.2us



*Source: "T-KVM: A Trusted architecture for KVM ARM v7 and v8 Virtual Machines", IARIA Cloud Computing 2015 awarded best paper

- State of the art
- Beyond the state of the art
- Status of the work and benchmark
- **Next steps**
- Conclusion

Next Steps

Virtual Open Systems is carrying on the development of a complete and powerful automotive software stack solution, which includes:

- High performance for VMs and safety critical OS (hardware accelerators)
- VMs and safety critical OS QoS (coordinated scheduling)
- Security and safety (vTPM)

High performance - HW accelerators

Both the VMs and the safety critical OS, leverages on hardware accelerators (such as FPGA, GPU or DSP) to:

- 3D rendering and computer vision algorithms
- Multimedia audio and video codecs
- Security (encrypted communication, driver recognition, remote connection, etc.)
- Advanced driving functions (ADAS, park assistant, driver recognition, etc.)

HW accelerators – Safety critical OS

Safety critical functions can be accelerated by:

Exclusively using a device

- ARM TrustZone is used to isolate in hardware the access to a device
- This device is not seen by the hypervisor and the VMs

Securely sharing a device with the VMs

- The safety critical OS acts as an arbiter for the VMs requests to the device
- VMs access to the device has a lower priority

HW Accelerators - VMs

Virtual machines exploit GPUs, DSPs and FPGAs to provide high performance to guests operating systems by means of:

	Direct Assignment (e.g. VFIO)	**API Remoting (e.g. OpenCL, CUDA)**	**HW assisted virtualization (e.g. SRIOV)**
Description	Static allocation of the device to a single VM	Virtualization at the API level	Dynamic allocation of the device to multiple VMs
Pros	Performance	Flexibility, migration, independence from the hardware	Performance, fexibility
Cons	Space/power consumption, flexibility, migration	performance	Requires HW support, migration

High performance - QoS

Quality of Service capabilities are important to provide always the right user experience to the driver:

The safety critical OS:

- Has always higher priority
- Can not be affected by incorrect or malicious software running in the VMs.

VMs QoS:

- A certain VM can be prioritized over the other
- Priority metrics are: network bandwidth, hardware utilization (CPUs and other devices), etc.

VMs QoS

When a virtualized system is saturated by concurrent workloads (host and guest), VMs applications start to see an increase in latency.

A guest-host scheduler missing link exacerbates these latency issues for:

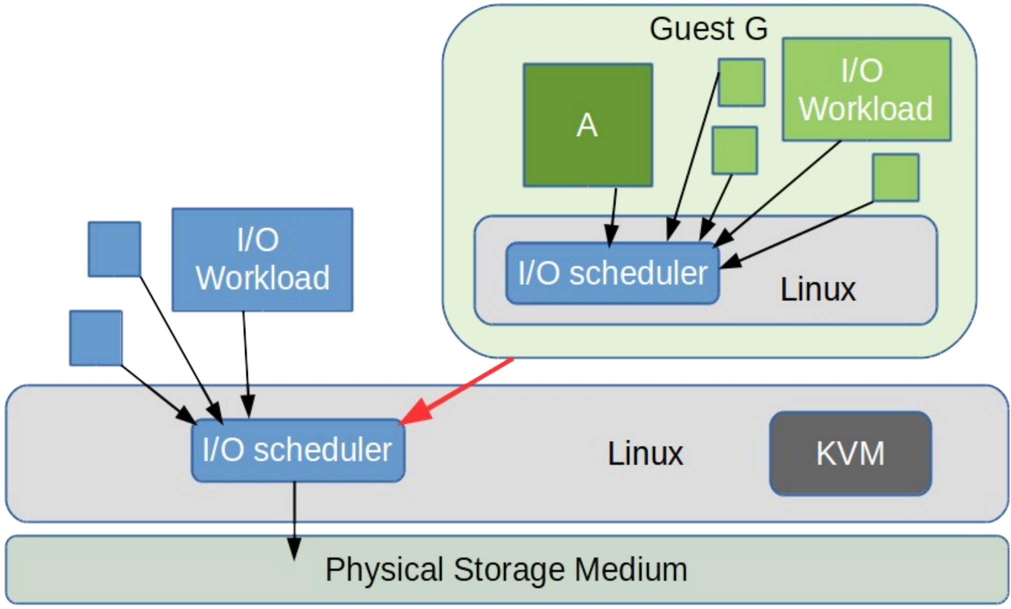
- I/O scheduling
- Process scheduling
- Interrupt handling and network packet scheduling

Highlighting the problem: I/O disk latency

The same concept is valid for I/O Disk as well.

- Consider a system running a guest operating system (guest **G**)
- A new application A, is being started in guest **G**
- Other applications are already performing I/O in the guest/host

- The cumulative I/O request pattern of guest G, may exhibit no special property that allows the I/O scheduler in the host, to realize that an application is being loaded in the guest.
- So, the latency of application A is higher (worse startup time and responsiveness).



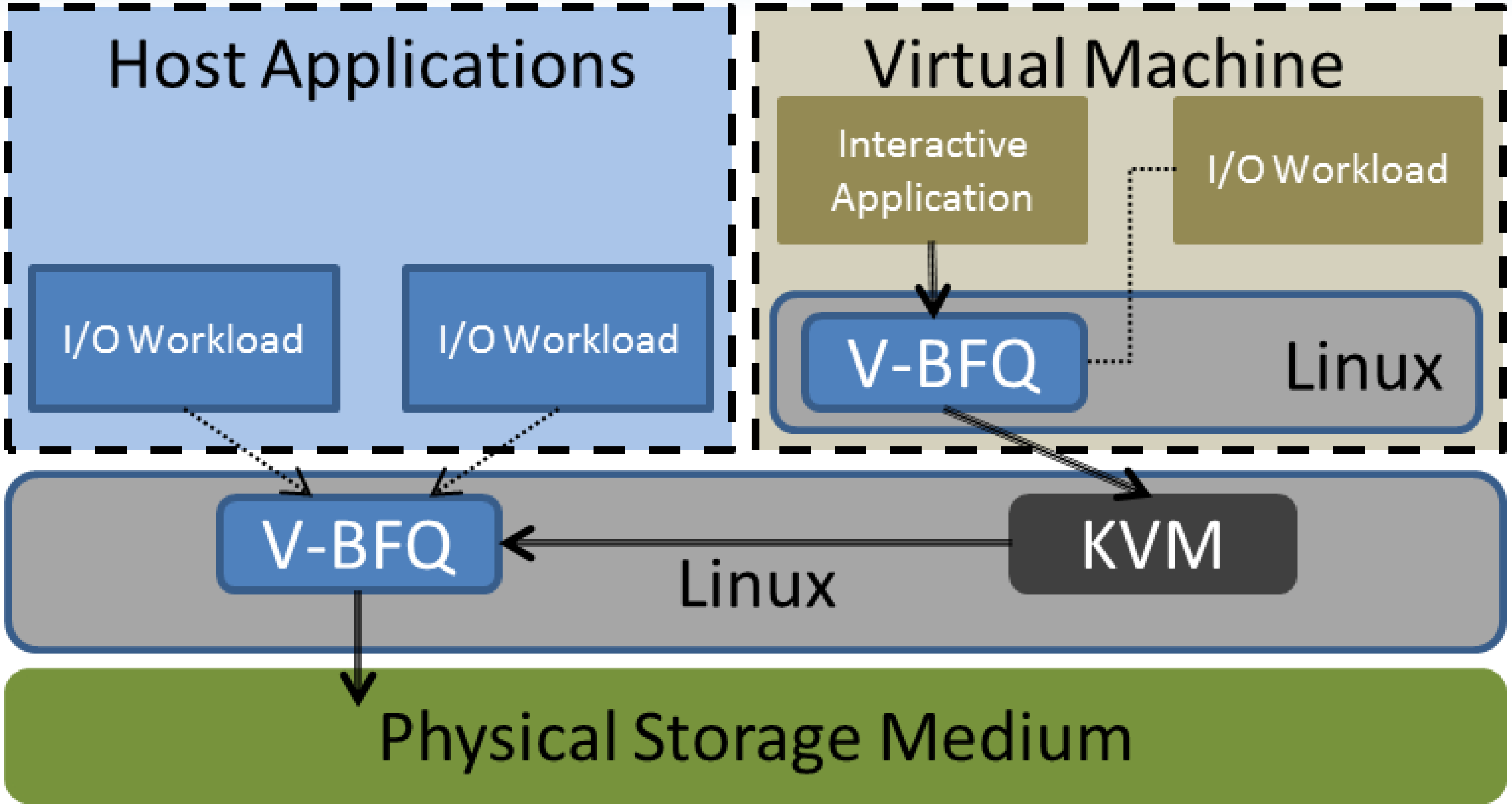
Virtual Open Systems

Solution: Coordinated Scheduling

The concept of coordinated scheduling is based on the idea of fast and direct communication of the guest scheduler with the host

- The guest scheduler can quickly signal the host scheduler that there is a need to increase the priority of the guest
- First PoC with I/O scheduling: V-BFQ (based on BFQ)
- I/O host/guest scheduler is extended with virtualized systems in mind
- Communication between guest V-BFQ and host V-BFQ is done by the Hypervisor-Call ARM instruction (HVC)
  - hvc #1 to indicate the guest needs to be privileged
  - hvc #0 to indicate the guest does not need to be privileged anymore

Coordinated Scheduling: v-BFQ



Coordinated Scheduling: benchmarks

A set of benchmark have been performed modifying the disk and CPU schedulers, resulting in:

- From 28% to 67% disk latency improvement if compared with BFQ
- Minimum 58% disk latency improvement CFQ - the default Linux scheduler
- On a 2 CPUs systems, running a VM with 1 VCPU, CPU latency ~20times better than CFS
- Maximum latency of 71µs, whatever the number of workloads, against a CPU latency from 71µs up to >6000µs with vanilla CFS

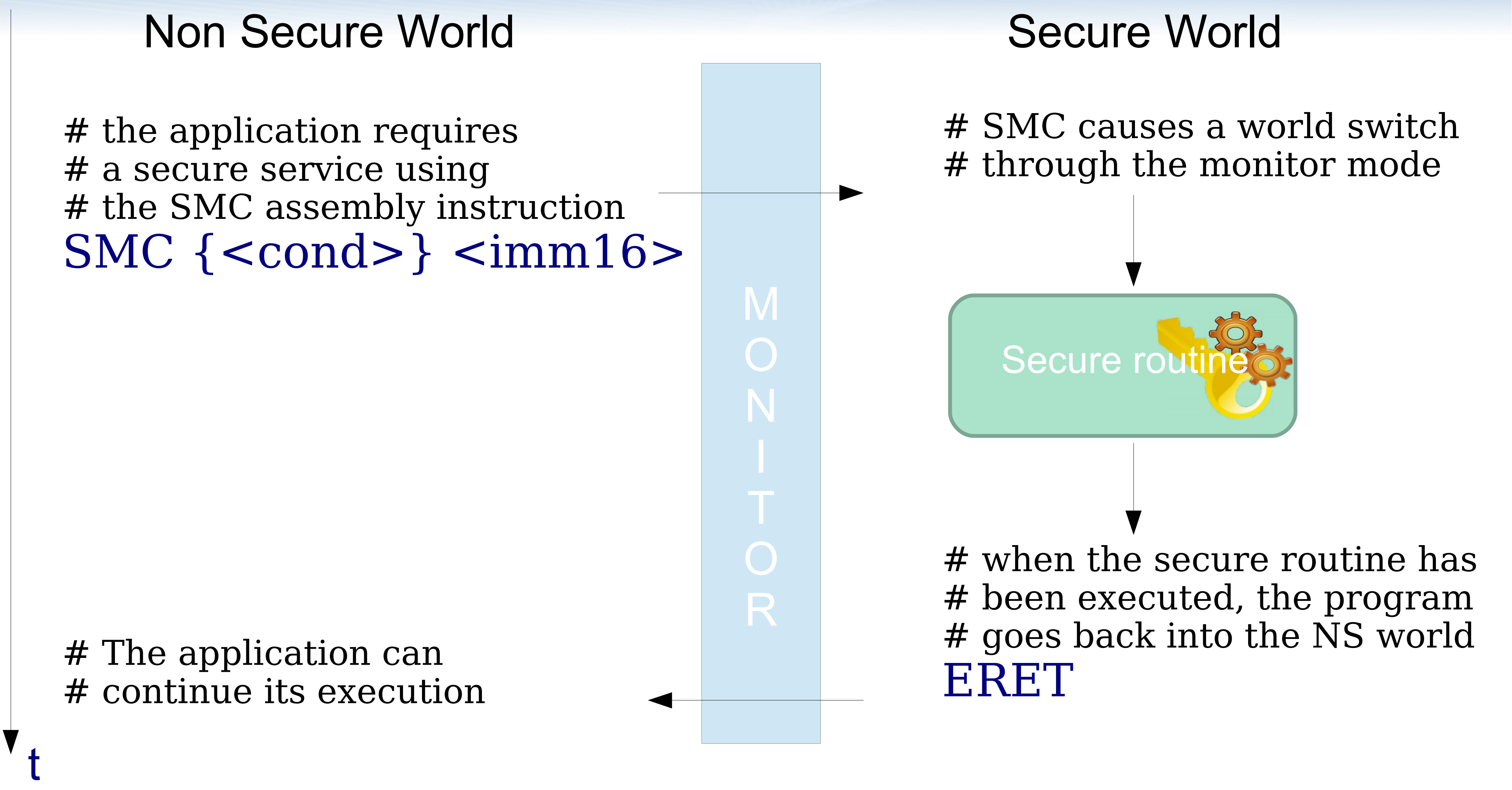
Security and Safety: Trusted Computing

VMs security, attestation and verification is of pivotal importance for connected cars (3rd party apps, IoT, Cloud Computing, etc.), and can be provided using a virtual Trusted Platform Module (vTPM) based on TrustZone:

- Enables software TPM (vTPM) implementations for Virtual Machines
- The guest Operating System accesses the TPM functions through the TEE API
- Supports Tizen, Android, Ubuntu guests

Historically, TPMs are custom non-virtualizable solutions which require specific development.

TPM for VMs
The typical TrustZone program flow



Non Secure World

Secure World

```
# the application requires
# a secure service using
# the SMC assembly instruction
SMC {<cond>} <imm16>
```

```
# SMC causes a world switch
# through the monitor mode
```

```
# when the secure routine has
# been executed, the program
# goes back into the NS world
ERET
```

```
# The application can
# continue its execution
```

Virtual Open Systems

TrustZone and virtualization challenges

The TrustZone technology has been designed by ARM as a security extension for resource constrained embedded systems.

- The Secure World is not able to run HW accelerated VMs
- The SMC does not know the identity of the VM which is requesting the secure service
- A secure service request (SMC call) performed in the guest should not affect the host Secure World state

.. is TrustZone virtualizable?

TPM for VMs implementations

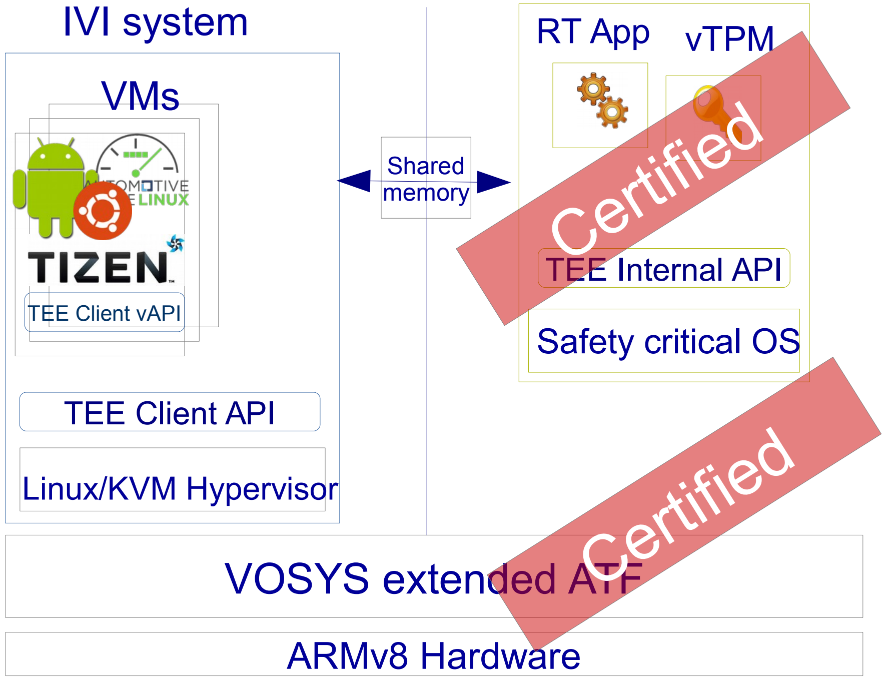
The VM access to the vTPM can be implemented either trapping the SMC call, or using an API remoting approach

	API remoting	SMC
Description	The secure service requests are handled by a specific API	The hypervisor traps the guests SMC calls
Pros	Flexibility, hardware independence	Guest applications can execute the SMC call directly
Cons	Guest OS needs to be modified accordingly to support API remoting	ARM TrustZone is a mandatory requirement

Security and safety: Certification

The certification of the system (RTOS, drivers, apps) is of pivotal importance for the automotive market

- ISO 26262 is the target for the VOSYS extended ATF (TrustZone Monitor layer), which supports already existing certified safety critical OS



- State of the art
- Beyond the state of the art
- Status of the work and benchmark
- Next steps
- **Conclusion**

Conclusion

In this presentation, a novel open source architecture based on TrustZone and KVM has been proposed and implemented as a proof of concept

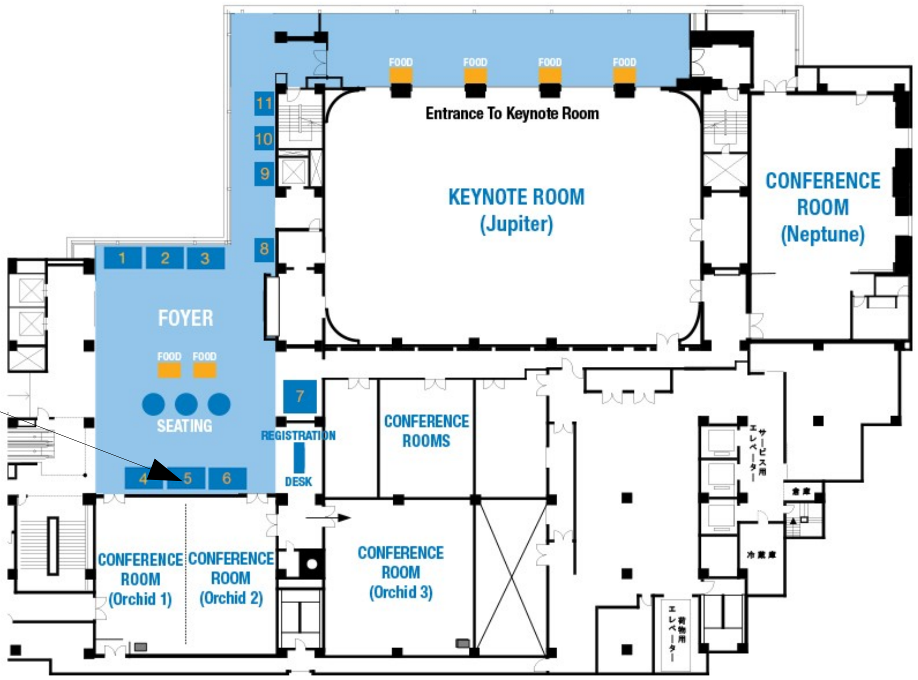
- Many different open source projects are involved, with planned future work
  - QEMU/KVM support for Tizen/AGL distribution
  - An open source virtualized TEE implementation
  - VM HW acceleration and QoS

Virtual Open Systems will actively contribute to the future AGL distribution for IVI and safety critical OSes convergence and HW accelerators virtualization

Thank You

come to visit us and see the demo to the Automotive Grade Linux 2015 booth 5!

contact@virtualopensystems.com



Virtual Open Systems