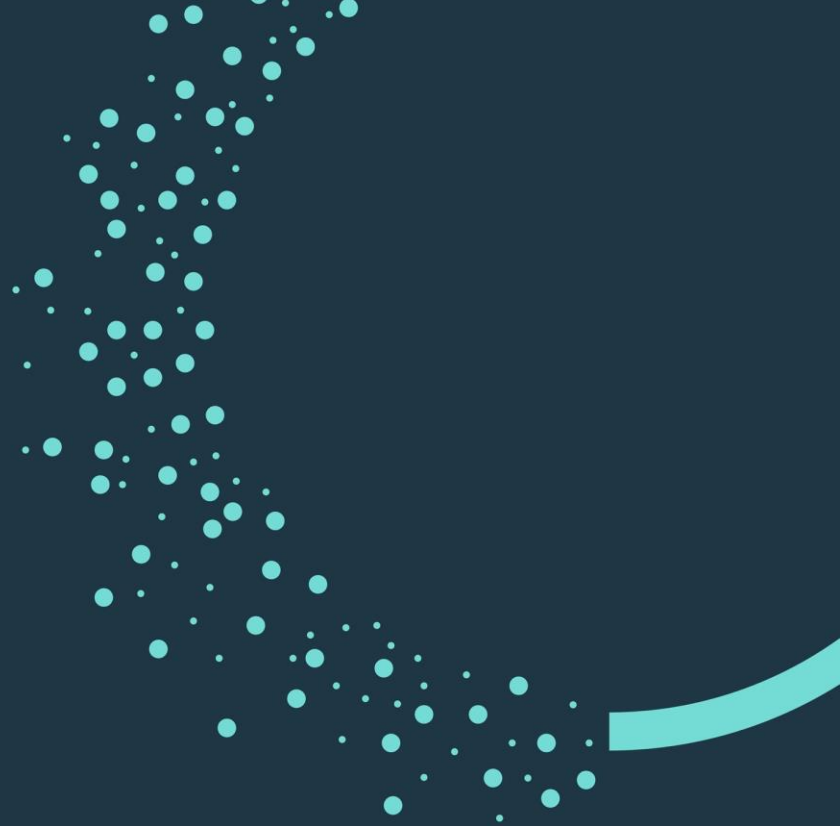


seL4 Microkernel for virtualization use-cases: The importance of a standard VMM

Everton de Matos
Jason Sebranek

Introduction

- **seL4**
  - Small and assured microkernel-based **type-1 hypervisor**
- **Fragmentation problem**
  - Different companies are using their own closed-source tools and user-space libraries
  - Short-term benefits
  - Does not allow the community to benefit from commonality
  - Quickly get old and out of sync with the mainline
- **Gaps in the seL4 stack, specifically the VMM, user space, and tooling**
- **Gather the community in building a standardized VMM as a shared effort**
- **Key design principles and potential feature set support**

Philosophy of a Standard VMM

- Establish a **driving philosophy** for the design of a baseline VMM **rather than prescribe a specific system architecture**
- **Discuss the missing features** of the existing seL4 VMM, more so than a prescription for the right way to do it
- Recommending high-level architecture patterns but cannot lock an adopter into specific implementations
- **One size does not fit all**
  - Close the gap between the current VMM baseline and the point of necessary deviation

Background

- **seL4 VMM**
  - *libsel4vm*
    - Guest hardware virtualization library for x86 (ia32) and ARM (ARMv7/w virtualization extensions & ARMv8) architectures
  - *libsel4vmmplatsupport*
    - library containing various VMM utilities and drivers that can be used to construct a guest VM on a supported platform
- Construct VMM servers through providing useful interfaces to create VM instances, manage guest physical address spaces and provide virtual device support (e.g., VirtIO Net, VirtIO PCI, VirtIO Console)
- **Missing features!**

Background

- **seL4 VMM**
  - *libsel4vm*
    - Guest hardware virtualization library for x86 (ia32) and ARM (ARMv7/w virtualization extensions & ARMv8) architectures
  - *libsel4vmmplatsupport*
    - library containing various VMM utilities and drivers that can be used to construct a guest VM on a supported platform
- Construct VMM servers through providing useful interfaces to create VM instances, manage guest physical address spaces and provide virtual device support (e.g., VirtIO Net, VirtIO PCI, VirtIO Console)
- **Missing features!**

What are the missing features on the current seL4 VMM in your opinion?

Case Study

- **Cog Systems** and **TII** worked together on a research project to create a virtualization framework on the *Qualcomm Snapdragon 845*
  - Brought up two Android VMs and a Linux VPN VM
  - vBlock and vNet implementations
  - Got bogged down and suffered from lack of VMM features

- **Cog System**s did similar work with **Hensoldt** on a *Qualcomm Snapdragon 670*
  - SMMU virtualization was a goal, but we had limited success
  - SMMU is undocumented and hidden behind black boxes
- Qualcomm SoCs already have enough challenging to deal with
- Decomposition of Android is another challenging endeavor
- Would be helpful to have a reliable and feature-rich VMM to use as baseline

seL4 Microkernel for virtualization use-cases:
Potential directions towards a standard VMM

https://arxiv.org/abs/2210.04328

Philosophy of a Standard VMM

- For the purposes of this discussion, driving philosophical concepts can be roughly binned into:
  - **Design Principles**
    - **Official and Open Source**
    - **Modular**
    - **Portable**
    - **Scalable**
    - **Secured by Design**

Philosophy of a Standard VMM

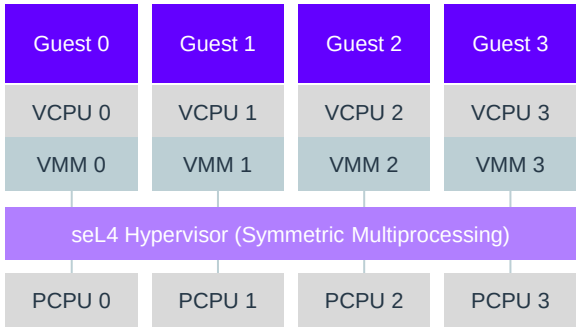
- For the purposes of this discussion, driving philosophical concepts can be roughly binned into:
  - **Design Principles**
    - **Official and Open Source**
    - **Modular**
    - **Portable**
    - **Scalable**
    - **Secured by Design**
  - **Feature Support Tenets**
    - **System Configuration**
    - **Multicore & Time Isolation**
    - **Memory Isolation**
    - **Hypervisor-agnostic I/O Virtualization and its derivations**

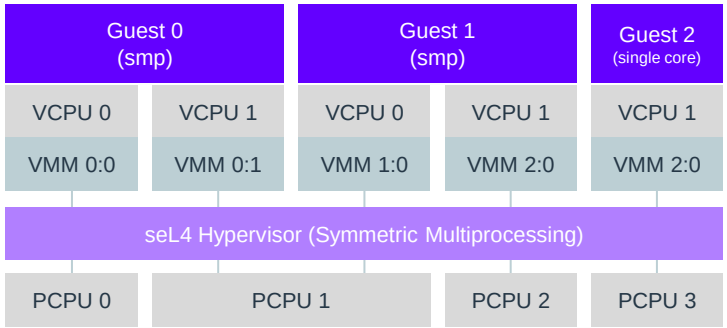
Feature Set Support Tenets

- **System Configuration**
  - Available frameworks
    - **CAmkES**
      - Static
    - **Core Platform**
      - Static & dynamic
      - Work in progress
  - Dynamis
    - It should be possible to use the seL4 mechanisms for transferring/revoking capabilities to the entities during runtime

Feature Set Support Tenets

- **Multicore & Time Isolation**
- Potential multi-core configurations that a standard VMM should be able to support:
- **Direct Mapping Configuration:**
  multiple single-core VMs running concurrently and physically distributed over dedicated CPUs
- **Hybrid Multiprocessing:**
  it can have multiple single-core VMs running in dedicated CPUs as the AMP configuration, however, it can also have multicore VMs running in different CPUs



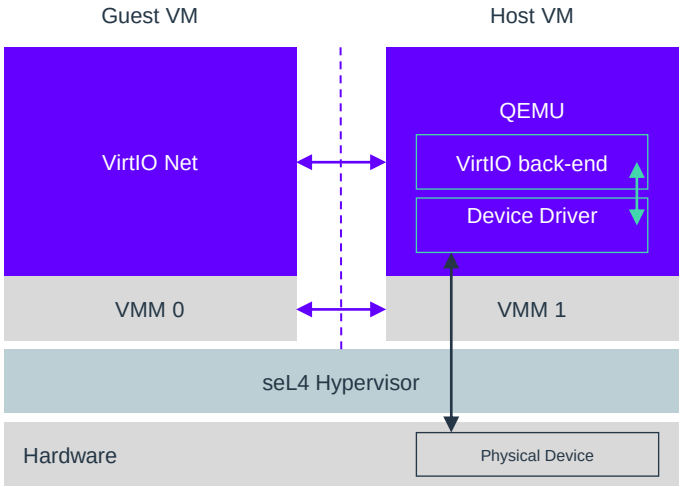


Feature Set Support Tenets

- **Memory Isolation**
- Memory isolation is critical to enforce the security properties such as VMs confidentiality and integrity
- Support for hardware-assisted virtualization (extended Page Tables or second-stage) MMU should be an integral part of the standard VMM
- Features for future reference that can leverage such hardware for memory isolation:
  - IOMMU
  - Configurable VM Virtual Address Space (VAS)
  - Cache Isolation

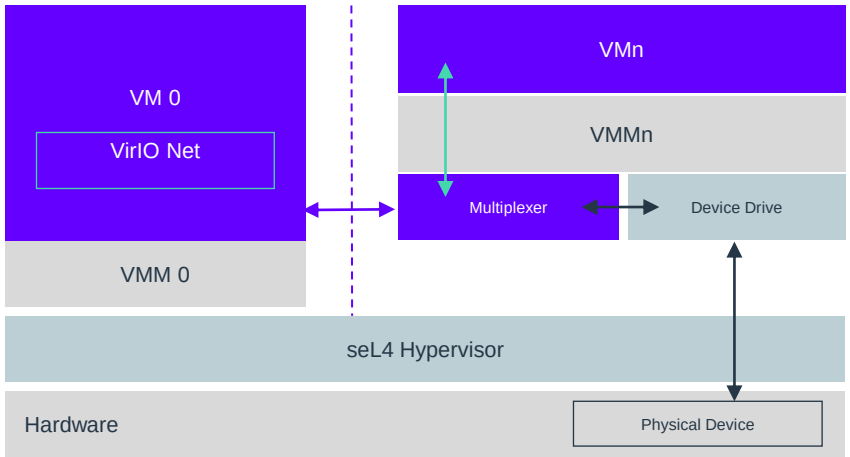
Feature Set Support Tenets

- **Hypervisor-agnostic I/O Virtualization and its derivations**
- VirtIO can be used for interfacing VMs with host device drivers
- It can support VirtIO driver *backends* and *frontends* on top of seL4. VirtIO interfaces can be connected to open-source technologies such as QEMU, *crosvm*, and *firecracker*, among others.
- This approach helps in achieving reusability, portability, and scalability



Feature Set Support Tenets

- **Hypervisor-agnostic I/O Virtualization and its derivations**
- VirtIO interfaces can be connected to formal verified native device drivers
- The verified device drivers can be multiplexed to different accesses, switching device access between multiple clients



Discussion Topics

- **VMM API**
  - Programmable API rather than something configured with static Domain Specific Language (DSL) during compilation (e.g., CAmkES)
  - An API would make it possible for some elements of the VMM not to be wrapped in a runtime context
  - There is a certain minimal subset that a VMM must handle, like handling the hardware virtualization of Generic Interrupt Controller Architecture (GIC) and handling faults. However, it should also be possible to define where VirtIO-console should be handled or that VirtIO-blk device must be handled by QEMU in some VM
  - Different flavors of VMMs, such as QEMU, crosvm, and cloud-hypervisor
- **Formal Methods**
  - **seL4 Core Platform**
  - **seL4 Device Driver Framework (sDDF)**
    - "*Formal verification of device drivers is our long-term goal, which is enabled by the strong isolation provided for usermode drivers on seL4, which allows verifying drivers in isolation*"

Summary

- The existing VMM baseline is **not ideal** and **lacks support** for many **useful design features**

- This can be remedied by **helping to build a new VMM standard**, unified under the **principles** laid out in this discussion talk. Ultimately, this standard must be put to the test by making a concerted effort to build a **real-world proof of concept** around it

- Considering the seL4 ecosystem, a step towards defining a standardized VMM would be the **creation of an RFC for community discussion and approval**

- **Creation of work groups within the seL4 Community, around topics of interest**

What are the missing features on the current seL4 VMM in your opinion?

seL4 Microkernel for
virtualization use-cases:
The importance of a standard VMM

Everton de Matos - everton.dematos@tii.ae
Jason Sebranek - jason@cog.systems

Design Principles

- **Official and Open Source**
  - The existing seL4 VMM employs open-source license (BSD-2-Clause), and any new implementations under the proposed standard should remain in accordance with this approach
  - Individual integrators should always retain the ability to keep closed-sourced their highly specialized or trade secret modification
  - This strikes a **balance** between business needs such as maintaining a competitive edge and fully participating in a collaborative community around a common baseline

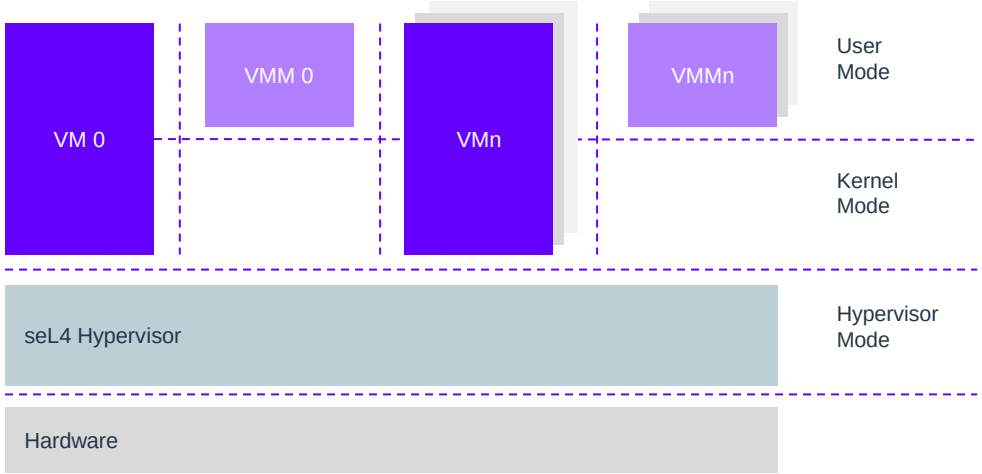
Design Principles

- **Modular**
  - Heterogeneous environments and scenarios under quite varied use cases
  - Flexibility in aspects such as system architecture, hardware requirements, performance
  - Decomposes the system so community members can pick and choose components for their project

- **Portable**
  - Hardware independence
  - VMM should also be generic enough to support different platforms
  - There is the need for a minimal and modular VMM that could be easily moved from 4 core ARM SoC (big.LITTLE) to a 48-core Thread Ripper AMD x86, as an example
    - **Baseline** for Proof-of-concept implementation and learning
    - Need to be extended to take advantage of all hardware characteristics

Design Principles

- **Scalable**
  - It needs to be able to support several applications running on top for different specific purposes
  - It is essential that the VMM supports from one to an arbitrary number of processing units or cores
- **Secured by Design**
  - A standard seL4 VMM implementation should support one VMM instance per VM



Feature Set Support Tenets

- System Configuration
- Multicore & Time Isolation
- Memory Isolation
- Hypervisor-agnostic I/O Virtualization and its derivations

Feature Set Support Tenets

- **Multicore & Time Isolation**
  - Ideally, it would be up to the system designer to decide which configuration to use. It could be either static or dynamic, enabling switching from a given configuration to another in run-time
  - Features that are likely required by multicore design are:
    - **vCPUs Scheduling**
      - Schedule threads on each pCPU based on their priority, credit-based time slicing, or budgeting
      - Extend seL4 kernel scheduler
    - **pIRQ/vIRQs ownership**
      - Protocol with acquire/release ownership of interrupts
    - **Inter-vCPU/inter-pCPU communication**
      - Ability to communicate between CPUs

Feature Set Support Tenets

- **Memory Isolation**
- IOMMU
  - Device memory isolation by hardware-support or purely software
  - Some requirements could be met in a standard VMM:
    - A device can only access the memory of the VM it belongs to
    - The device could understand the Virtual Address Space of its VM
    - The virtualization layer could intercept all accesses to the device and decode only those that intend to configure its DMA engine in order to do the corresponding translation if needed, and control access to specific physical memory regions
  - In order to meet these three requirements a standard VMM requires support for either an IOMMU (with one or two stage translation regimes) or software mechanisms for mediation

Feature Set Support Tenets

- **Memory Isolation**
- Configurable VM VAS
  - VM memory isolation using second-stage MMU
  - There should be a different VAS for different software entities: Hypervisor, VMM, and their respective VMs
  - User-controlled and configurable address spaces are important features for VMs
- Cache Isolation
  - Page-coloring
    - Map frame pages to different VMs without colliding into the same allocated cache line
  - It would be possible to assign a color to a set of VMs based on their criticality level

Feature Set Support Tenets

- **Hypervisor-agnostic I/O Virtualization and its derivations**
  - It is recommended to adopt VirtIO implementations for multiple interfaces
  - In architectures with more than one guest OS accessible from the user perspective, VMM VirtIO servers could also handle multiplexing access to various devices between VMs