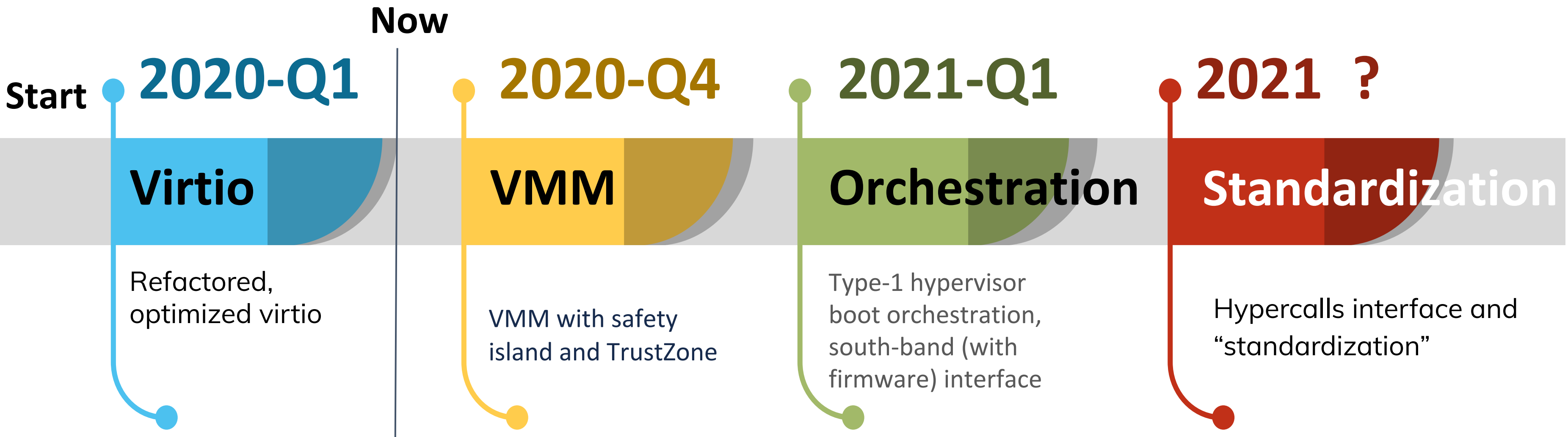
Linaro & AGL related timeline

Describe the relationship of the work Linaro is involved in with the AGL agenda

https://wiki.automotivelinux.org/eg-virt
https://www.linaro.org/projects/



Now

Start

2020-Q1

Virtio

Refactored, optimized virtio

2020-Q4

VMM

VMM with safety island and TrustZone

2021-Q1

Orchestration

Type-1 hypervisor boot orchestration, south-band (with firmware) interface

2021 ?

Standardization

Hypercalls interface and “standardization”

AGL requirements addressed

Inter VM communication (Shared memory, network, character)

Create standard interfaces for inter VM communication

Standard way of sharing hardware (Virtio) is very important for fast prototyping and portability

AGL requirements addressed

Coordinate startup and shutdown of all VMs

AGL requirements addressed

VirtIO interface works as “well-defined HAL”; SoCs are becoming hypervisor ready; Virtualization Ready BSP

AGL requirements addressed:

Standard API of I/O Virtualization. Common I/O Para-Virtualization API

Refactored, optimized virtio (1 of 2)

- Refactored, optimized virtio; virtio over SPCI/OpenAMP; Android as virtio backend, communication infrastructure:
- VirtIO backends that run in a de-privileged environment
- Allow better propagation of virtio updates in all hypervisors and VMMs
- Leverage DTE project for platform device assignment to VMs
- SPCI memory backend for virtio
- Maximized zero-copy mechanisms
- Non Linux backends: Android

Refactored, optimized virtio (2 of 2) - Virtio

Enable front-end and back-end drivers abstractions for SoC devices , with high performance and minimal memory footprint between the guest and virtio backend

Implement VirtIO interfaces

- VirtIO RPMB - In Progress
- VirtIO Audio
- VirtIO Watchdog
- VirtIO SMMUv
- VirtIO SPI
- VirtIO SCMI and other resource management approaches
- Prototype a minimal memory profile virtio backed with front end changes
- Create a common virtio library for use by programs implementing a backend

VMM with safety island and TrustZone

- Rust-VMM based VMM with safety island and TrustZone components, north bound (with guest OSes, VMM) interfaces:
- Rust based VMM (replacing QEMU) in Xen
- Split VMM (portions of VMM running in TrustZone or Safety Island or special domain
- Zephyr (on cortex-A or M) as VMM or partial VMM or as abstract device backend
- CortexM/R or TrustZone as part of VM control
- Hypercalls for security (incl. Access to security hardware), VM creation, orchestration/VMM, firmware control, system control (power off, sleep…)
- White paper on certification requirements impact on hypervisors

Hypervisor boot orchestration (type-1)

- Type-1 hypervisor boot orchestration, south-band (with firmware) interfaces:
- System Device Tree bindings for static partition configuration
- System Device Tree tooling to simplify device assignment
- Safety island wrapping into a partition
- Secure Monitor/SPCI services for hypervisors (memory assignment, Secure IRQs...)
- Hypervisor as BL33 payload (proof of concept exist with Linux as BL33)

Hypercalls interface and “standardization”

- Hypercalls interface description documents and “standardization”
- Produce a document to describe in details a hypercall interface to perform common operations, using the existing Xen FuSa effort as a starting point
- The hypercall interface should be implementable by multiple vendors/hypervisors
- The documents will be written in a way so that they can be used as a base for Safety Certification requirement documents

Thank you

Linaro