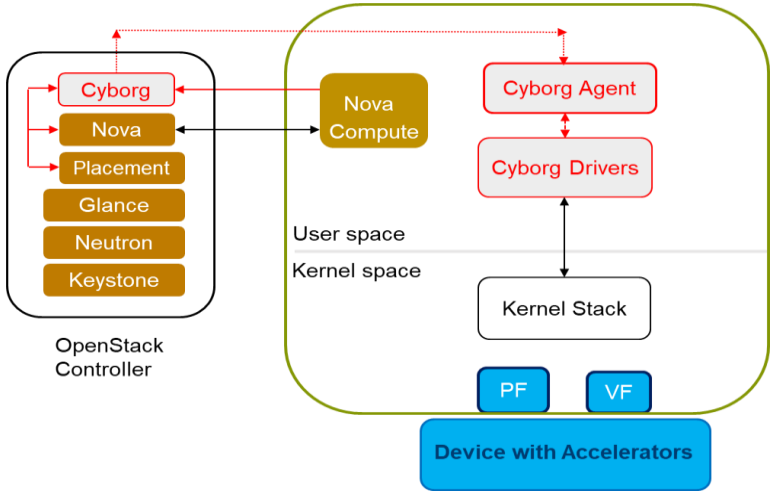
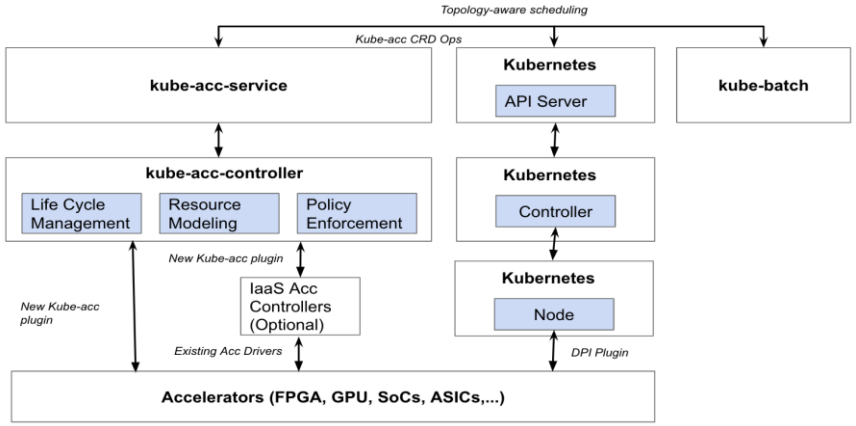
RISC-V For Heterogeneous Computing

Zhipeng Huang, Huawei
Justin Cormack, Docker

Recap from Zurich Workshop



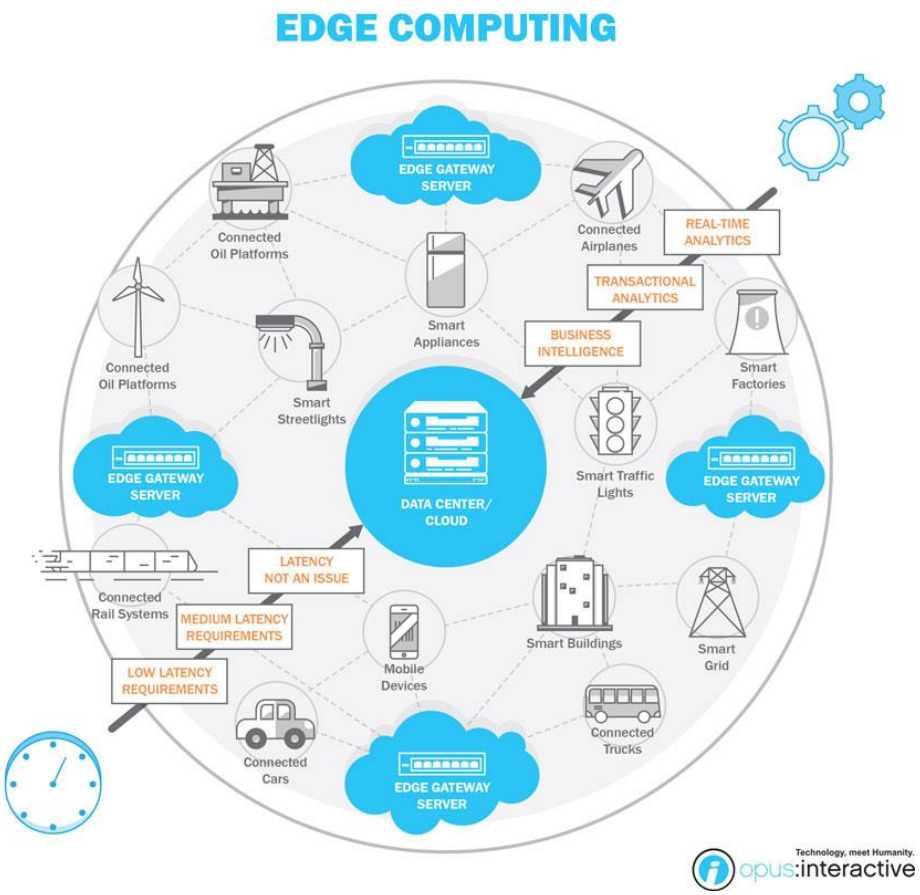


Cloud Management (OpenStack,Kubernetes, etc...)

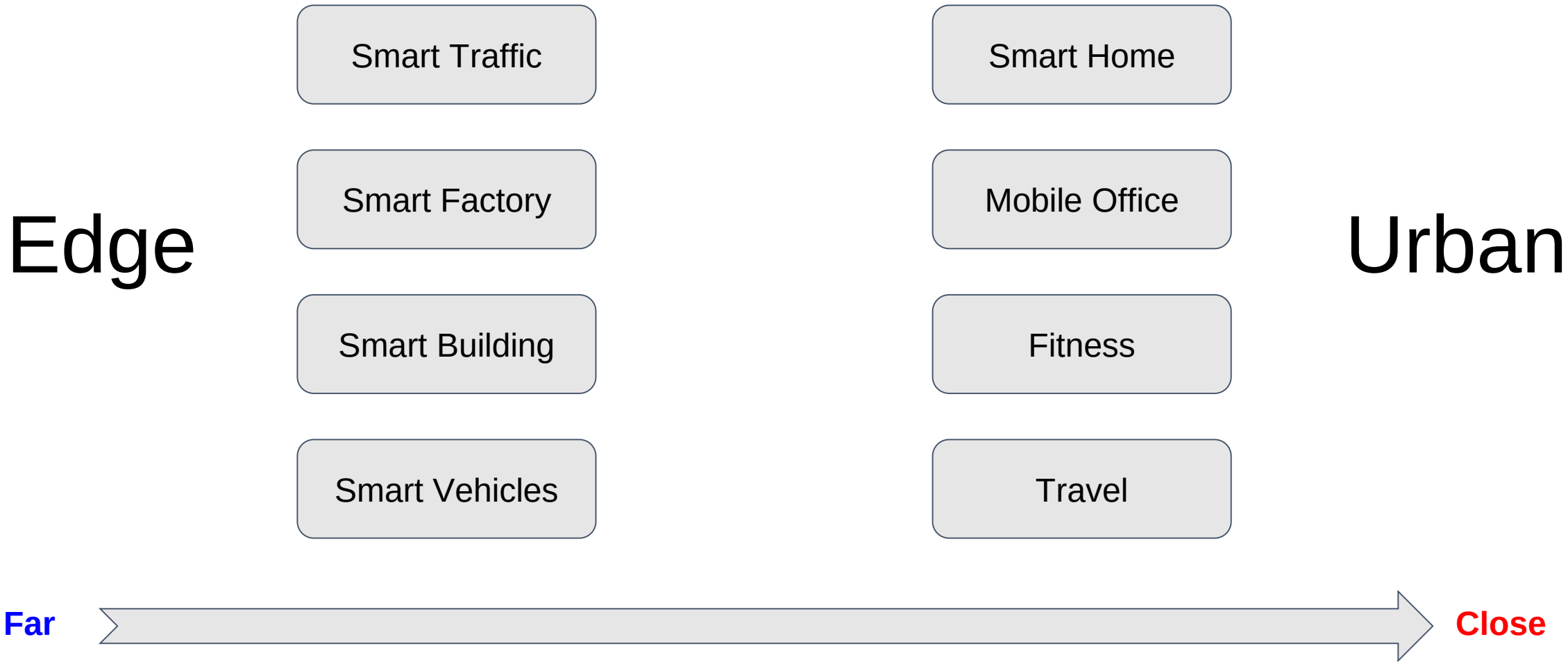
RISC-V Core Capabilities (topology, socket closeness, affinity, power, ...)

RISC-V Core Based Accelerator

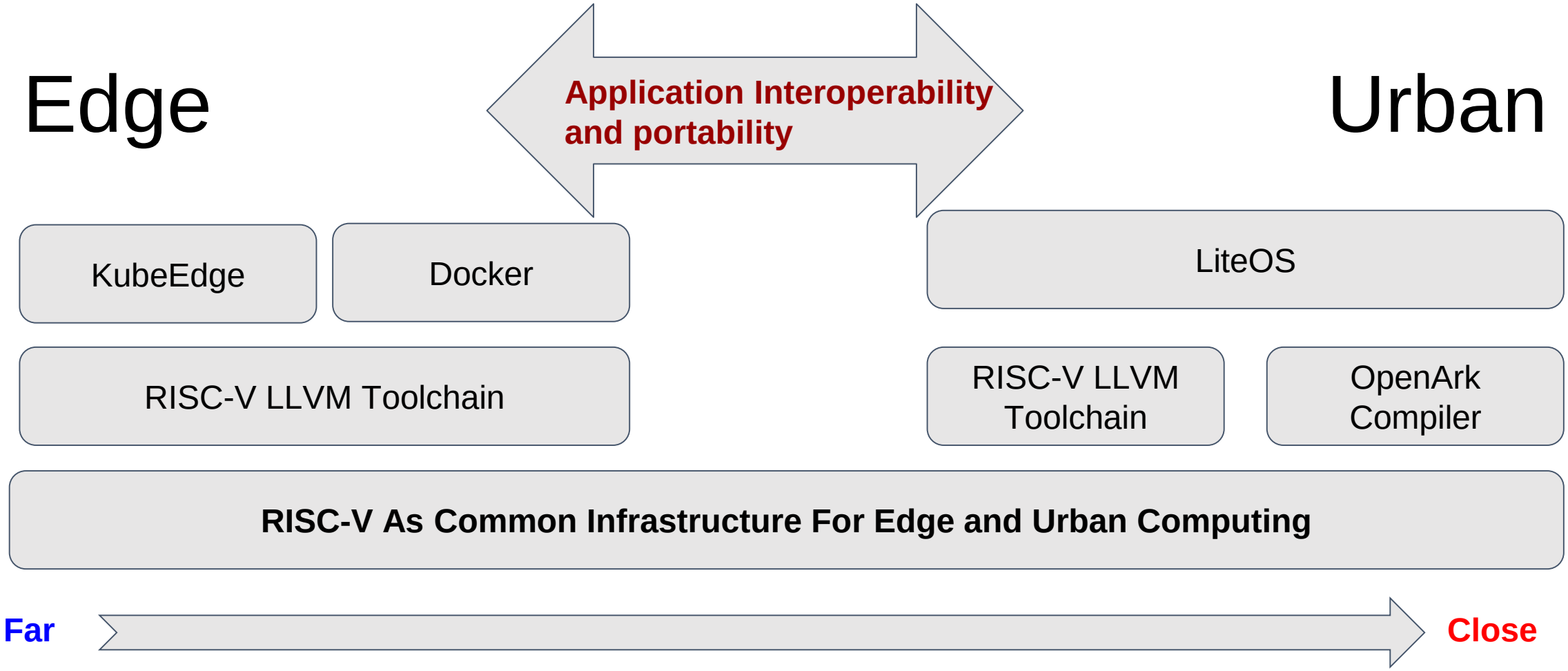
Heterogeneous Computing at Edge/Urban



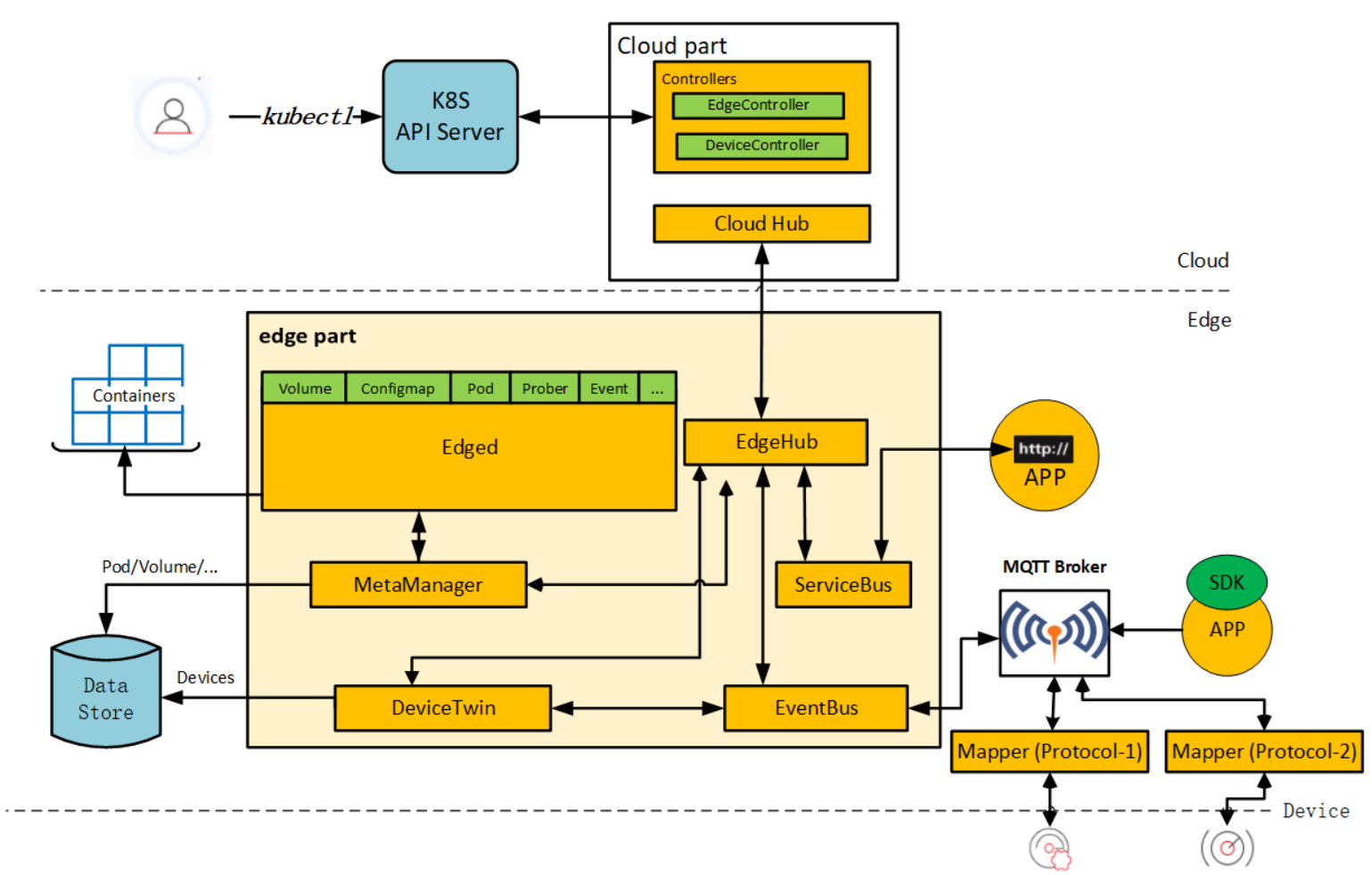
Heterogeneous Computing at Edge/Urban



Heterogeneous Computing at Edge/Urban Open Source Ecosystem



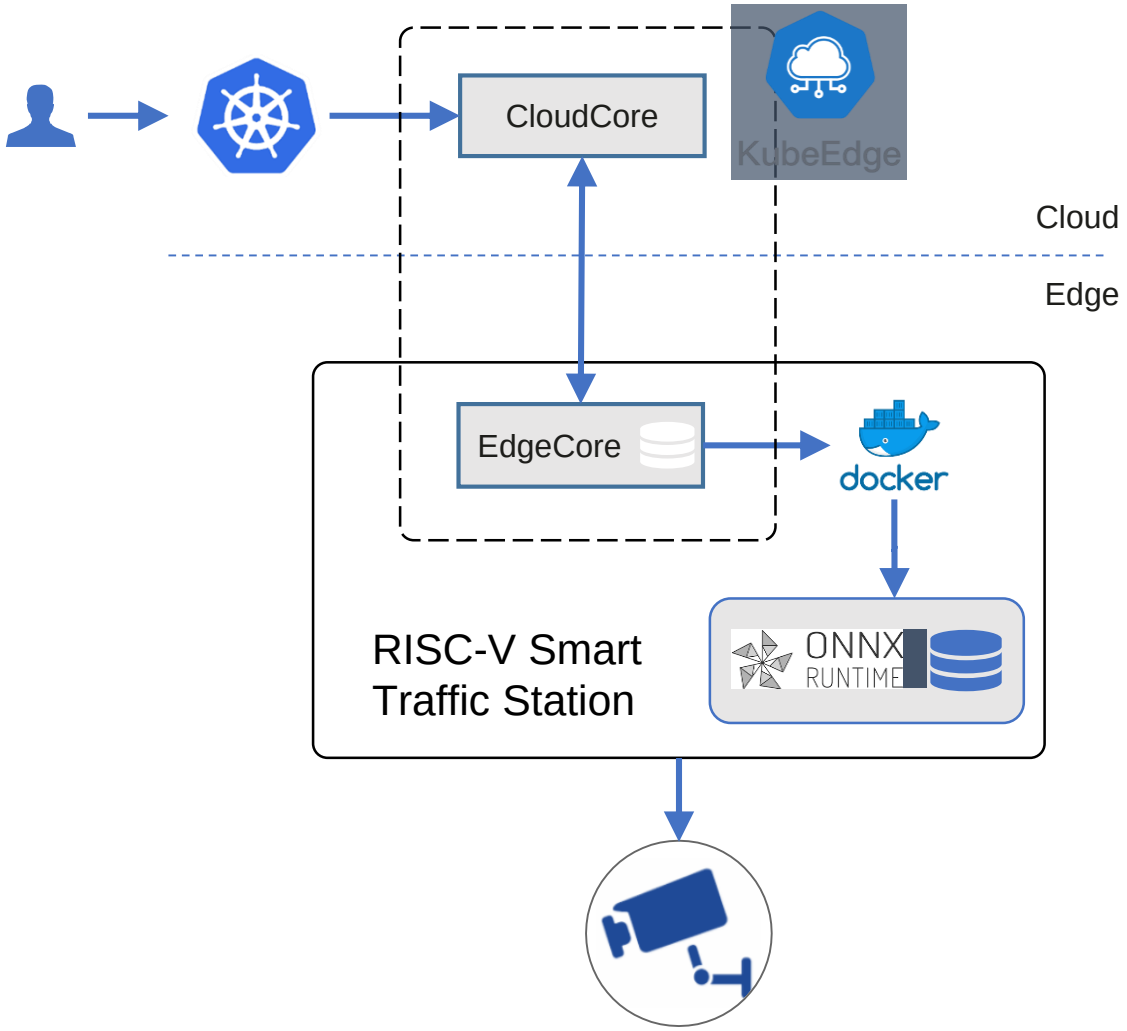
KubeEdge Overview



KubeEdge

https://github.com/kubeedge/kubeedge

KubeEdge RISC-V Support



- KubeEdge RISC-V Support mostly depends on the EdgeCore part.
- Current Docker RISC-V build could solve most of the problem, with the exception of syscall and MQTT driver support
- ONNX Runtime RISC-V build support would be another great to have
- Thanks Carlos for all the hard work at https://github.com/carlosedp/riscv-bringup

LiteOS: One Lightweight Kernel & Multiple Frameworks For Industrial IoT Device

https://github.com/LiteOS/LiteOS

Open APIs

Connectivity Framework

Connectivity and auto-networking of multi-protocol devices

- Application profile
- Auto-networking APIs
- Device-to-Cloud protocol HTTP/CoAP/MQTT/LWM2M
- Network protocol stack uIP/lwIP/RPL
- Communication protocol BLE/WiFi/6LowPAN/Zigbee/ PLC/NB-IoT

Sensor Framework

Unified sensor management

- Sensing algorithm library
- Sensor management
- Driver management

Security Framework

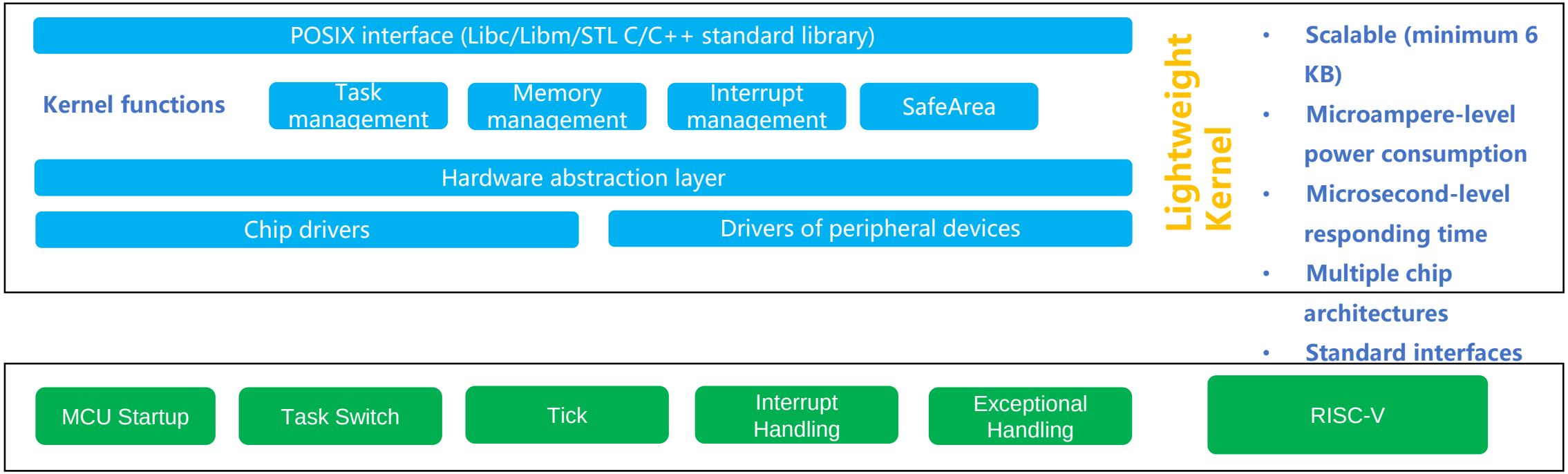
Security protection for terminals

- API authentication
- Bidirectional device authentication
- Encryption algorithm library

Kernel Functions

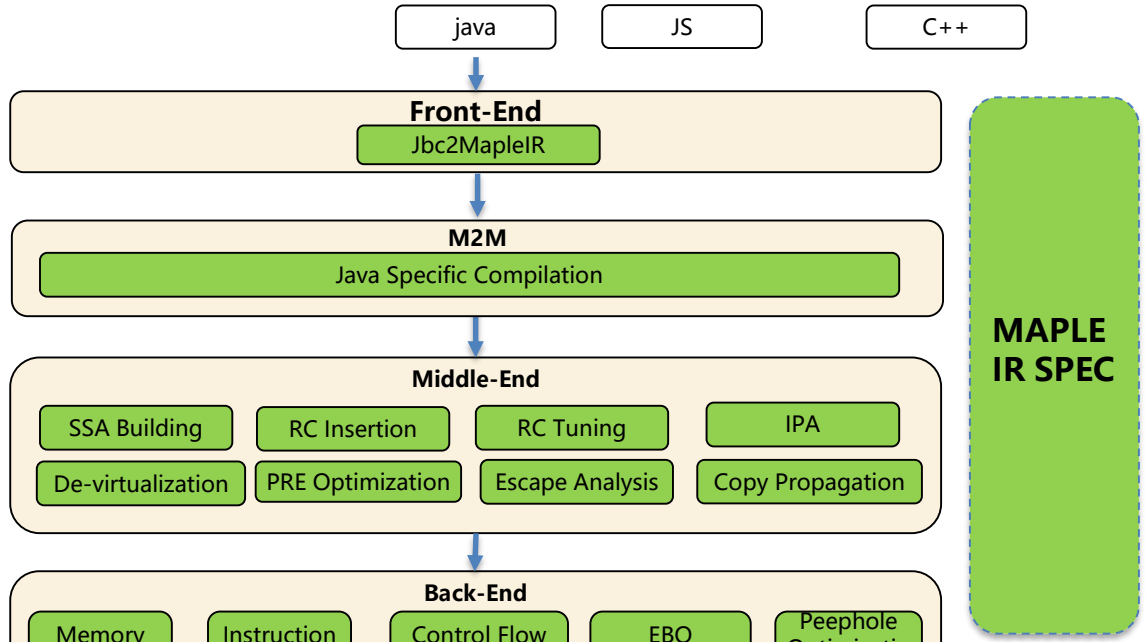
RISC-V | ARM Cortex-M0, Cortex-M3, Cortex-M4, Cortex-M7 | ARM Cortex-A7, Cortex-A17, Cortex-A53 | ARM11 | DSP

LiteOS: RISC-V Support



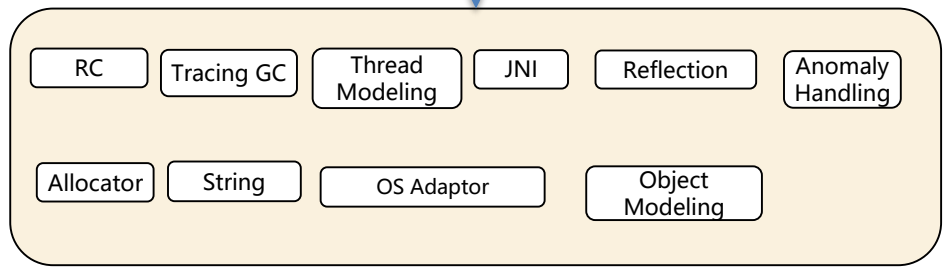
OpenArkCompiler Overview

Compiler



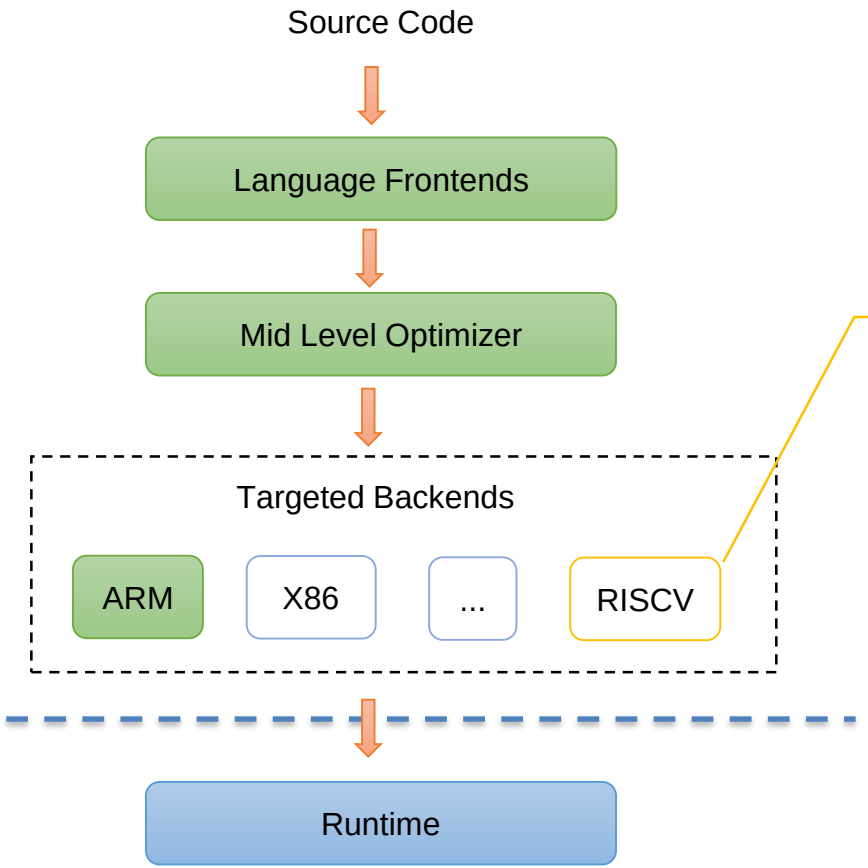
MAPLE IR SPEC

Runtime



https://gitee.com/harmonyos/OpenArkCompiler

RISC-V Support in OpenArkCompiler



TODO List:

- Instruction Selection
- Frame Lowering
- Calling Conventions
- Register Allocation
- Instruction Scheduling
- Optimizations
- ...

RISC-V Toolchain Development

Work In Progress

- Vector Extension Based On LLVM
  - Assembly Support（current stage）=> Instrinsic Function Support => Vectorization
  - https://github.com/isrc-cas/rvv-llvm
- Eternal Balance (Benchmark CI)
  - https://github.com/isrc-cas/Eternal-Balance
- Tiger Compiler for RISC-V
- LLVM CodeGen Analysis
  - Register Allocation
  - Instruction Scheduling
  - Opportunistic Scheduling
  - Soft Pipeline
  - Target Description Model

Future Work

- RISC-V Code Size Optimization
- RISC-V backend micro-architecture optimizations
- Linker Relax Optimization
- Linker Time Optimization
- Exchange between LLVM IR and OpenArkCompiler MAPLE IR
- ps abi
- Code Obfuscation

*Thanks to PLCT Lab team effort