

开放·连接·预见



K

# 2021 K+

## 全球软件研发行业创新峰会

重新探讨以eBPF为中心的超融合  
基础设施及边缘计算新方案

主讲人：李枫 (hkli2013@126.com)



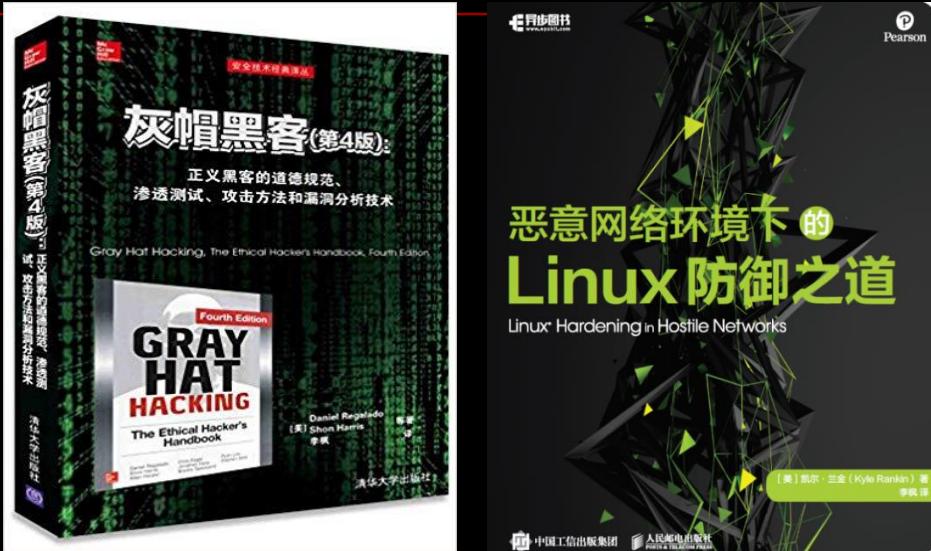
鲜卑拓跋枫



扫一扫上面的二维码图案，加我微信

## Who Am I

- The main translator of the book «Gray Hat Hacking The Ethical Hacker's Handbook, Fourth Edition» (ISBN: 9787302428671) & «Linux Hardening in Hostile Networks, First Edition» (ISBN: 9787115544384)



- Pure software development for ~15 years
- Actively participate in various activities of the open source community
  - <https://github.com/XianBeiTuoBaFeng2015/MySlides/tree/master/Conf>
  - <https://github.com/XianBeiTuoBaFeng2015/MySlides/tree/master/LTS>
- Recently, focus on infrastructure of Cloud/Edge Computing, AI, Virtualization, Program Runtimes, Network, 5G, RISC-V, EDA...



# Agenda

## I. Background & Updated Tech Stack

- Background
- 

- What's New

- An eBPF overview

- HW/SW Co-design and Co-development

## II. eBPF-centric Lightweight Solution for HCI/EC

- Design Goals and Principles

- Overall Design

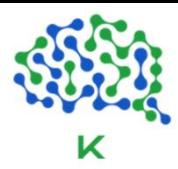
- Testbed

## III. Programming language and runtime

- Rust for System Programming

- Unified runtime for eBPF and Wasm

- Smart Runtime



## IV. Networking

- Emerging Networking Technologies
  - eBPF for Cloud-native Networking
  - eBPF/XDP Hardware Offload to SmartNIC/DPU
  - P4 and eBPF
- 

## V. Messaging & RPC

- Lightweight RPC
- Hardware-accelerated RPC
- eBPF-inspired Messaging and RPC

## VI. HPC

- Standards and Frameworks
- New Runtime for HPC

## VII. Storage

- Background
- eBPF-powered Userland Filesystems
- eBPF in Computational Storage
- eBPF-based In-kernel Storage



## VIII. Distributed AI

- Data Processing
- Project Ray

## IX. Blockchain

---

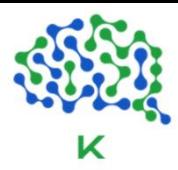
- eBPF in Blockchain

## X. Security

- Overview
- eBPF-based Linux System Security
- eBPF in Firewall
- eBPF for Cloud-native Security

## XI. System Debugging, Tuning, and Monitoring

- eBPF-based Unified Debugger
- eBPF in System and Application Profiling
- eBPF for System Observability and Monitoring



## XII. Cloud-native

- eBPF for Containers
  - eBPF in Kubernetes Ecosystem
  - eBPF in Microservices
  - Project Cilium
  - Project BumbleBee
- 

## XIII. 5G-6G

- 5G & 6G
- XDP for 5G UPF

## XIV. xBPF

- Overview
- Femto-Containers



## XV. Tittle-tattle

- eBPF for more Linux subsystems
- eBPF in Edge Computing
- Accelerating Python

---

- Lua
- Integration
- Emerging Technologies

## XVI. Rethinking the future

- Unikernel
- Rust-based Cloud Infrastructure
- Innovations in Edge Infrastructure
- Hardware Acceleration
- Beyond Kubernetes
- Better System Programming Language

## XVII. Wrap-up



# I. Background & Updated Tech Stack

## 1) Background

### 1.1 Previous Talk

■ **OpenInfra Days China**  
Shanghai 2019

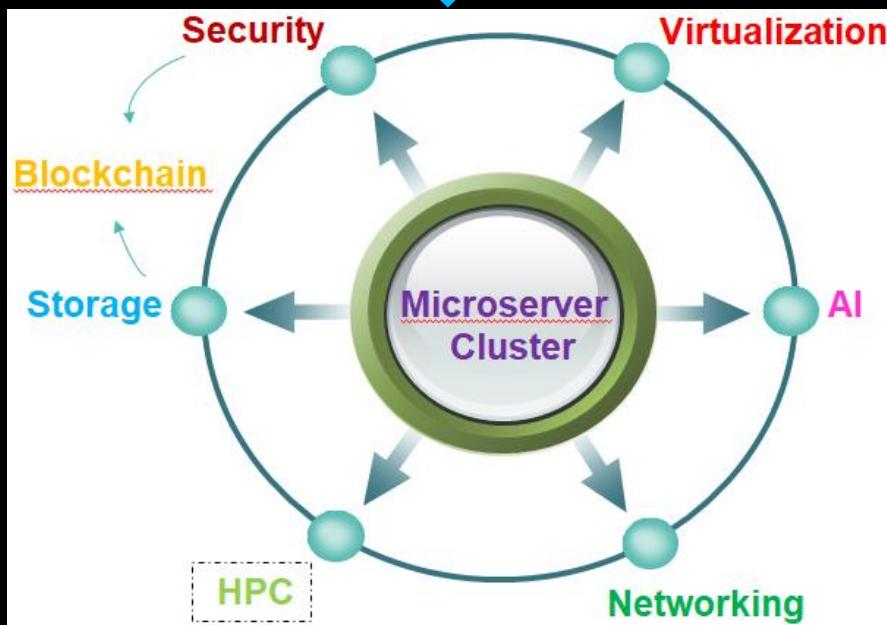
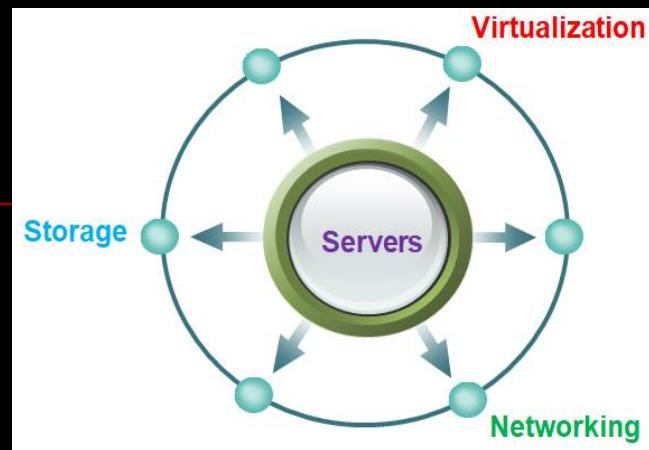
**Rethinking Hyper-  
Converged Infrastructure  
for Edge Computing**

Feng Li (李枫)

hkli2013@126.com

Nov 5, 2019

## ■ Trend (from my point of view)





## Key Points

### ■ 4) Summary

- Global Edge Computing Market is keep on growing, which mainly driven by the burst of Internet of Thing

- 
- Hardware and software vendors in Edge computing like "Let a hundred flowers bloom"

- Currently, there is no dominant solution provider at Edge like what AWS, GCP, and Aliyun does at Cloud

- Due to the diverse requirement for Edge Computing, and considering it may meet the resource limitation problem when comparing with Cloud Computing, so a lightweight and flexible solution with high cost–performance ratio is still very attractive

### ■ 2) Hardware Platform

- ARM is the best choice in current stage (from my point of view)  
high cost–performance ratio development boards  
an increasingly mature ecosystem

---

- a lot of vendors to choose from

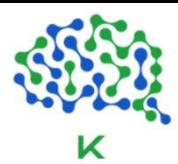
- lower power consumption

- ARM is ruling IoT and Embedded

- the major FPGA vendor Xilinx uses ARM as hardware cores on their Reconfigurable Computing platform

- ...

- may migrate to ARM, X86, RISC-V Hybrid Architecture in the near future, but ARM is still first class



## ■ II. Overall Design

### 1) Design Goals & Principles

#### Goals

- a **lightweight Edge Computing solution with high cost–performance ratio**
- **flexible and modular architecture**
- **scale out, not scale up**
- **meet the trend for Hyper-Converged Infrastructure at Edge**
- ...

#### Principles

- **no JVM based project is considered  
(so Spark, Flink, Kafka, ElasticSearch are excluded...)**
- **reduce the dependencies for Go-based project to least  
(though it is difficult to do so...)**
- **make project code reusable and self-contained as much as possible**
- ...



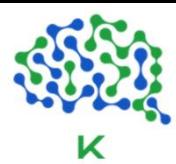
## 1.2 Original Design

- An eBPF-centric new lightweight for Edge Computing that devided into four stages

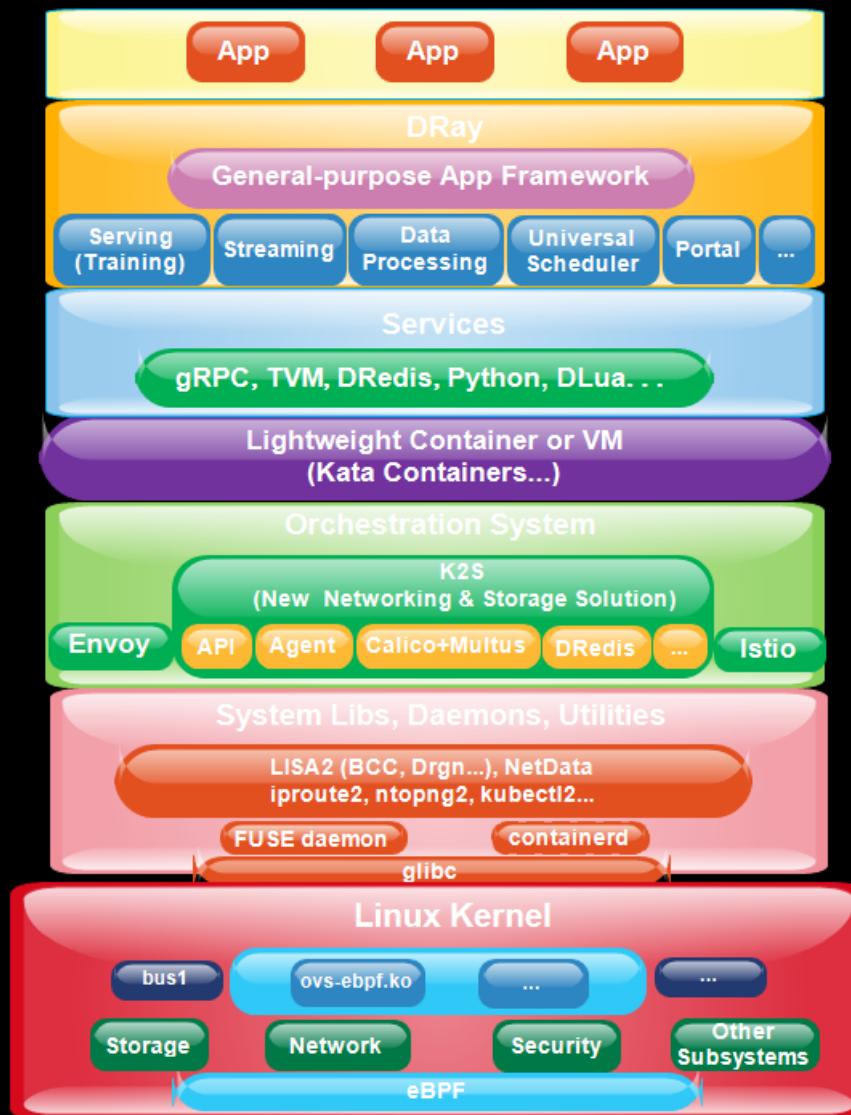
### Stage I



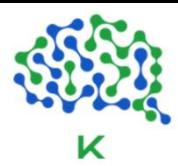
Source: “Rethinking Hyper-Converged Infrastructure for Edge Computing”, Feng Li, OpenInfra Days China 2019



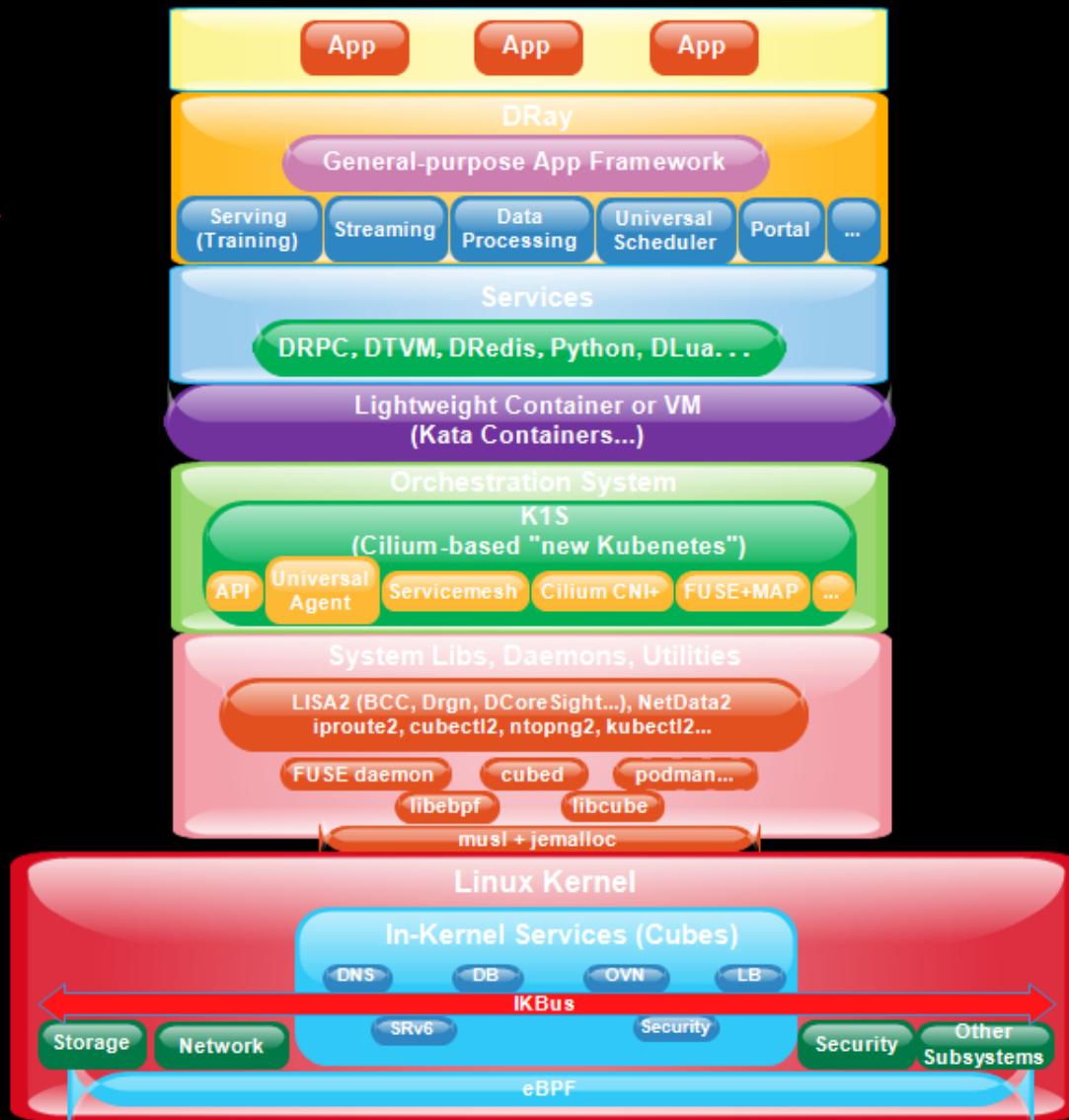
## Stage II



Source: “Rethinking Hyper-Converged Infrastructure for Edge Computing”, Feng Li, OpenInfra Days China 2019

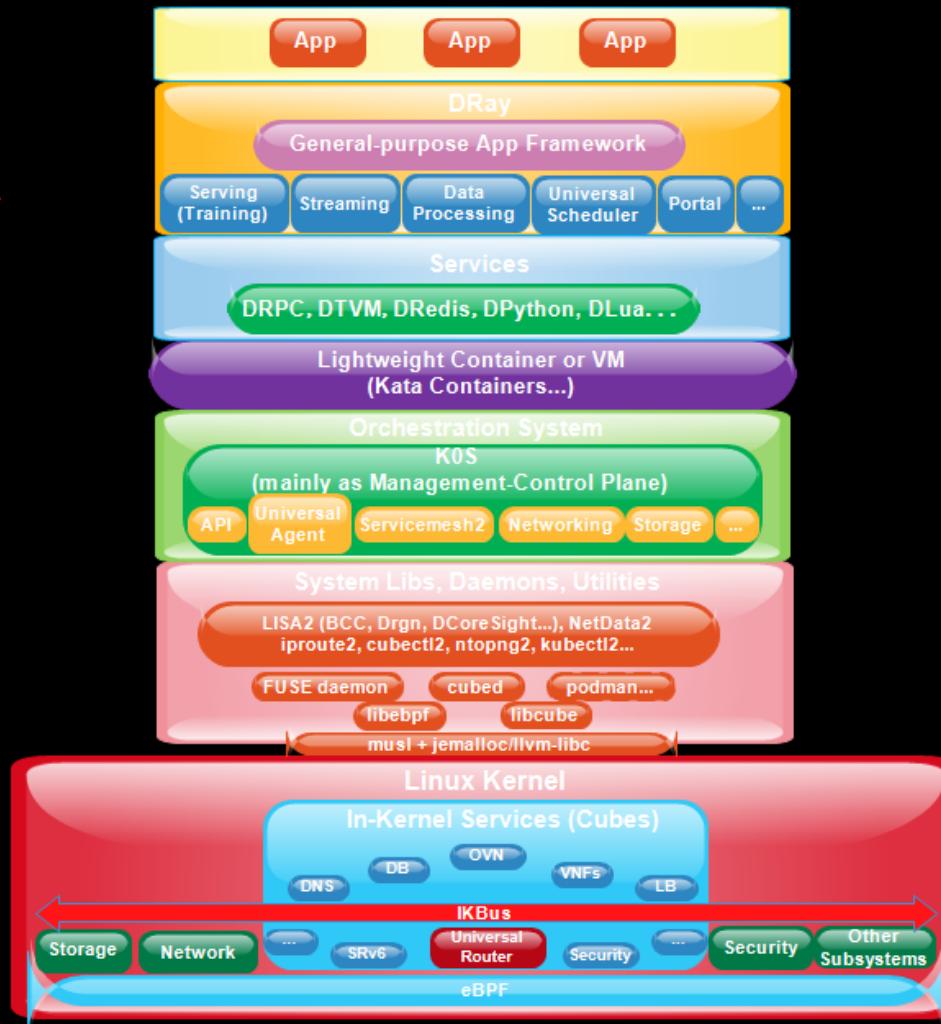


## Stage III



Source: “Rethinking Hyper-Converged Infrastructure for Edge Computing”, Feng Li, OpenInfra Days China 2019

## Stage 4



Source: "Rethinking Hyper-Converged Infrastructure for Edge Computing", Feng Li, OpenInfra Days China 2019

- While so much has changed over the past two years!



## 2) What's New

### 2.1 eBPF

#### 2.1.1 Official Site

■ <https://ebpf.io/>

The screenshot shows the official eBPF website at <https://ebpf.io/>. The page features a large eBPF logo with a bee icon and the text "eBPF". Below the logo are two yellow buttons: "What is eBPF?" and "Projects". A brief introduction explains that eBPF is a revolutionary technology used to safely and efficiently extend kernel capabilities. It contrasts eBPF's innovation with the traditional low rate of innovation at the operating system level. The page also includes a diagram illustrating the eBPF ecosystem across User Space and Kernel.

The diagram illustrates the eBPF ecosystem across three layers:

- User Space:** Contains sections for Networking, Security, and Observability & Tracing, each connected to specific projects like Katran and cilium.
- Kernel:** Contains sections for Verifier & JIT, Maps, OS Runtime, and Kernel Helper API, all connected to the Kernel Runtime layer.
- Application:** A central box connected to both the User Space and Kernel layers, listing features such as Tracing, Profiling, Monitoring, Observability, Security Controls, Load Balancing, and Behavioral Security.

Key components shown in the diagram include:

- Projects:** Katran, cilium, Falco.
- SDKs:** Go, C, Python.
- Verifier & JIT:**
- Maps:**
- OS Runtime:**
- Kernel Helper API:**
- Observability & Tracing:** Tracing, Profiling, Monitoring, Observability, Security Controls, Load Balancing, Behavioral Security.



## ■ eBPF Foundation

<https://www.linuxfoundation.org/press-release/facebook-google-isovalent-microsoft-and-netflix-launch-ebpf-foundation-as-part-of-the-linux-foundation/>  
<https://ebpf.io/foundation/>

### What is eBPF Foundation?

The number of eBPF-based projects has exploded in recent years and many more have been announcing intent to start adopting the technology. eBPF is quickly becoming one of the most influential technologies in the infrastructure software world. As such, the demand is high to optimize collaboration between projects and ensure that the core of eBPF is well maintained and equipped with a clear roadmap and vision for the bright future ahead of eBPF. This is where the eBPF Foundation comes in, and establishes an eBPF steering committee to take care of the technical direction and vision of eBPF.

If you are interested to collaborate with the eBPF Foundation, please join [#ebpf-foundation](#) on [ebpf.io/slack](#).

### eBPF Steering Committee

The eBPF Steering Committee (BSC) is responsible for the technical direction and overall vision of eBPF, the collaboration among projects, making recommendations to the governing board, defining the minimal requirements of eBPF runtimes, overseeing community events, maintaining project lifecycle procedures, and communicating on behalf of the eBPF community.

...

### Members

#### Platinum

[FACEBOOK](#)

[Google](#)

 [ISOVALENT](#)

 [Microsoft](#)

 [FUTUREWEI](#)

#### Silver

[NETFLIX](#)

 [DaoCloud](#)



- **eBPF Summit**

<https://ebpf.io/summit-2020/>

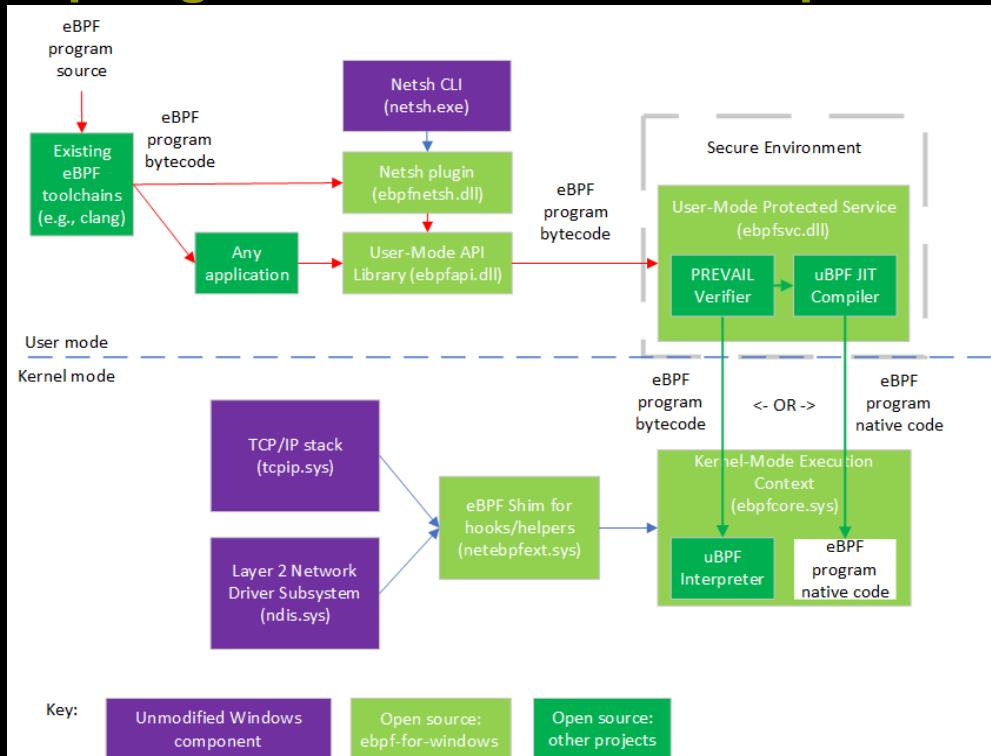
<https://ebpf.io/summit-2021/>

---



## 2.1.2 Cross-Platform eBPF on Windows

- <https://cloudblogs.microsoft.com/opensource/2021/05/10/making-ebpf-work-on-windows/>
- <https://github.com/microsoft/ebpf-for-windows>



- <https://lwn.net/Articles/857215/> // Implementing eBPF for Windows
- <https://thenewstack.io/microsoft-brings-ebpf-to-windows/>



# Status

Platform →	Linux		MacOSX		Android		FreeBSD		Windows		TockOS (embedded)	
↓ Project	Kernel	User	Kernel	User	Kernel	User	Kernel	User	Kernel	User	Kernel	User
Linux	2014											
uBPF		2015										
rbpf		2017		2017							2018	
Android					2017							
Generic eBPF	2017	2017		2017			2017	2017				
eBPF for Windows									2021			
Tock											2021	

Source: “The Cross-Platform Future of eBPF”, Dave Thaler(Microsoft), Cloud Native eBPF Day North America 2021

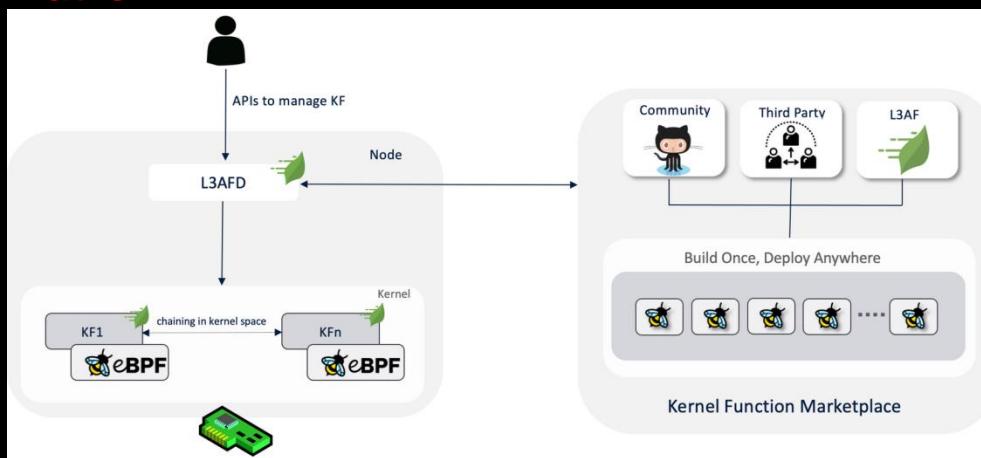


## 2.1.3 KFaaS (Kernel Functions as a Service)

- <https://www.linuxfoundation.org/press-release/walmart-moves-production-grade-networking-project-l3af-to-the-linux-foundation/>
- <https://l3af.io/>
- <https://github.com/l3af-project>
- **Features**

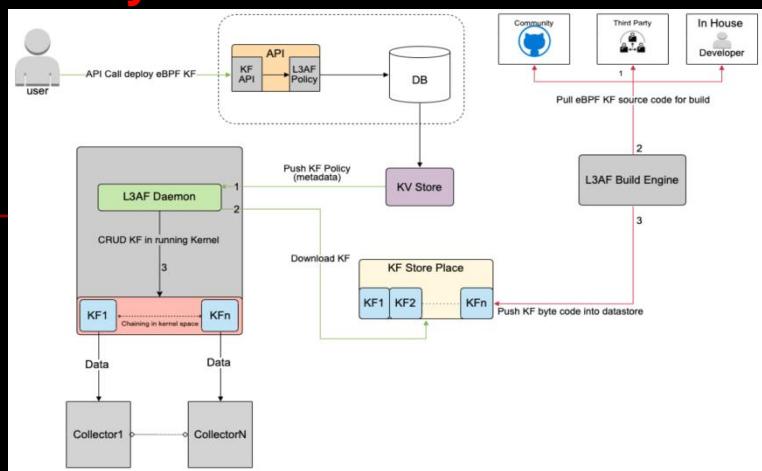
Innovation	Empowerment	Flexibility
Industry-first "Kernel Functions as a Service"	Simple API to add, remove, and reorder kernel functions on the fly	Distributed model to manage and configure kernel functions on a per-node basis
Multiple independent kernel functions executing in a chain	Configurable metrics are gathered for each kernel function	Compose kernel functions to fit business needs
More to come, including a community-driven kernel function marketplace	Replace proprietary applications and hardware with blazing fast eBPF code	Cloud and vendor agnostic

- **Platform**



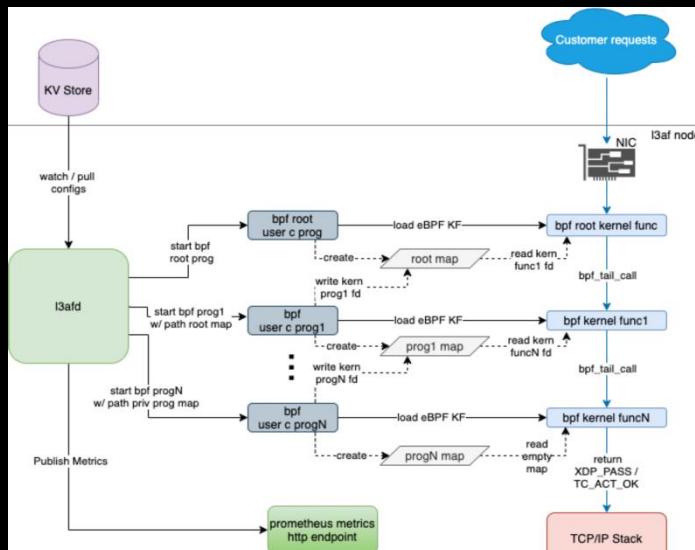


## Ecosystem



Source: <https://github.com/l3af-project/l3af-arch>

## Orchestration



Source: <https://github.com/l3af-project/l3af-arch>



## 2.1.4 New Features

- <https://github.com/iovisor/bcc/blob/master/docs/kernel-versions.md>  
**JIT compiling**

RISC-V RV64G	5.1	<a href="#">2353ecc6f91f</a>
RISC-V RV32G	5.7	<a href="#">5f316b65e99f</a>

### Main features

BPF cgroup sysctl	5.2	<a href="#">7b146cebe30c</a>
BPF raw tracepoint writable	5.2	<a href="#">9df1c28bb752</a>
BPF trampoline	5.5	<a href="#">fec56f5890d9</a>
BPF LSM hook	5.7	<a href="#">fc611f47f218</a> <a href="#">641cd7b06c91</a>
BPF iterator	5.8	<a href="#">180139dca8b3</a>
BPF socket lookup hook	5.9	<a href="#">e9ddb7707ff</a>
Sleepable BPF programs	5.10	<a href="#">1e6c62a88215</a>

...

- [https://kernelnewbies.org/Linux\\_5.17](https://kernelnewbies.org/Linux_5.17)  
[https://kernelnewbies.org/Linux\\_5.16](https://kernelnewbies.org/Linux_5.16)

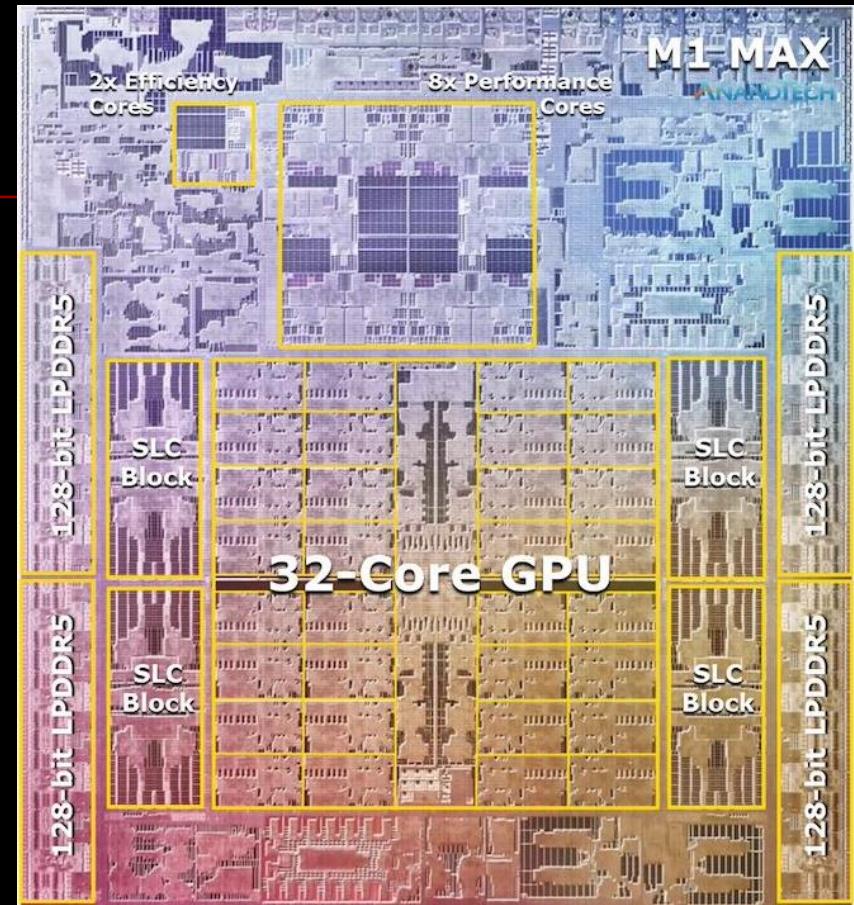
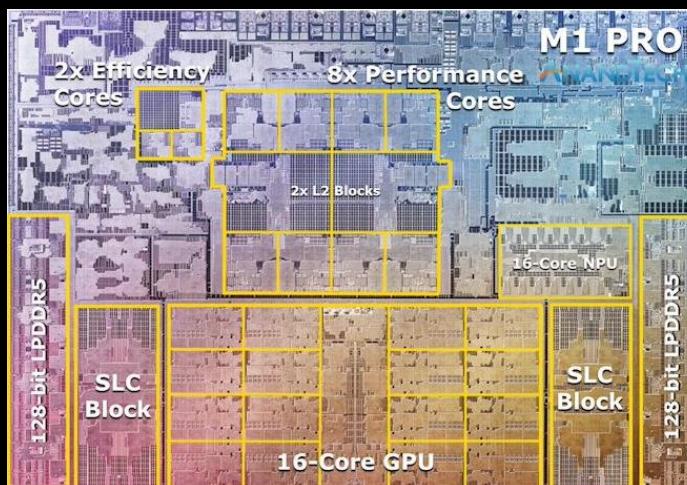
...



## 2.2 Hardware Platforms

### 2.2.1 ARM

#### 2.2.1.1 Apple

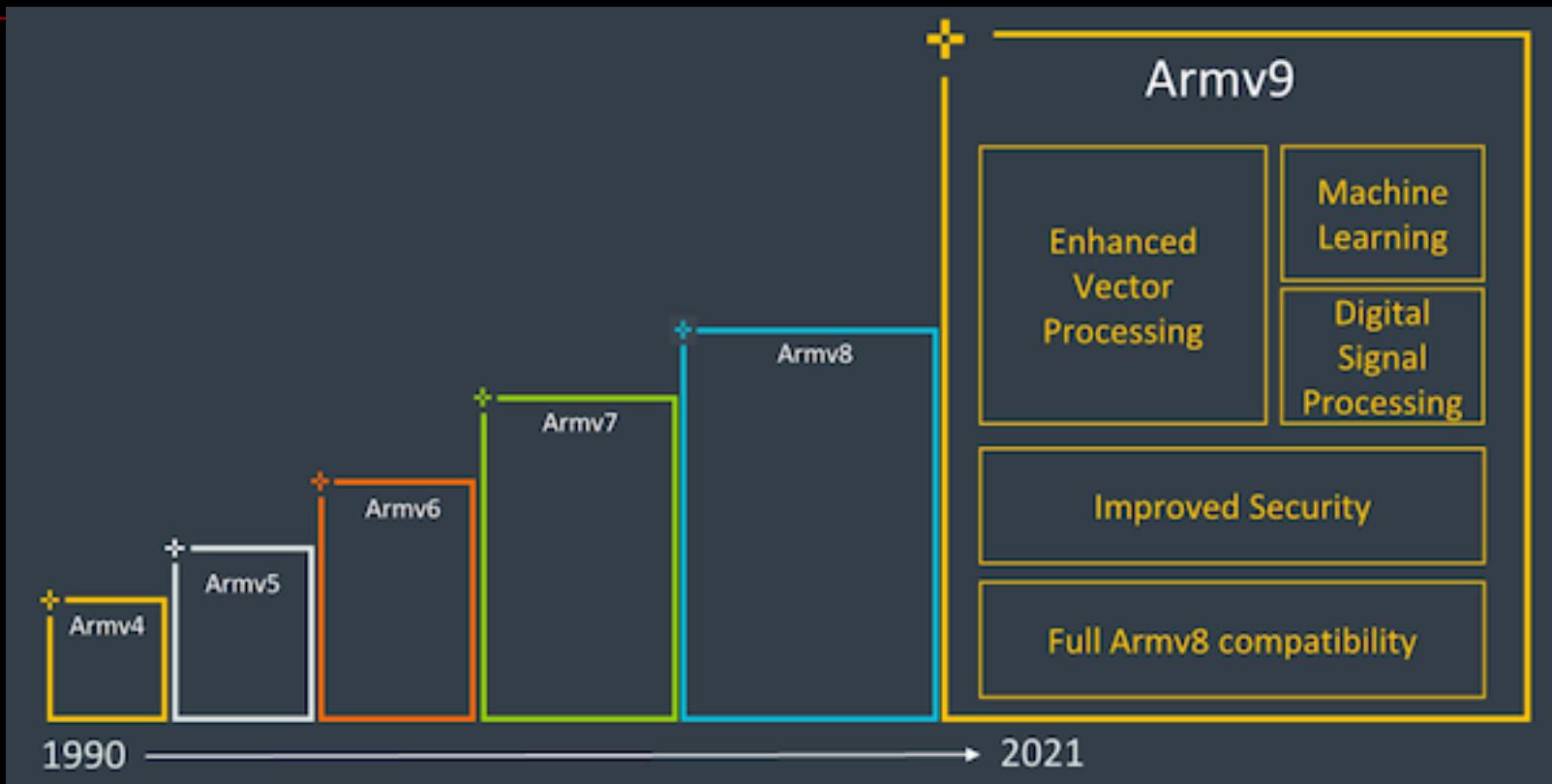


- <https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/>
- <https://www.apple.com/newsroom/2021/10/introducing-m1-pro-and-m1-max-the-most-powerful-chips-apple-has-ever-built/>
- ...



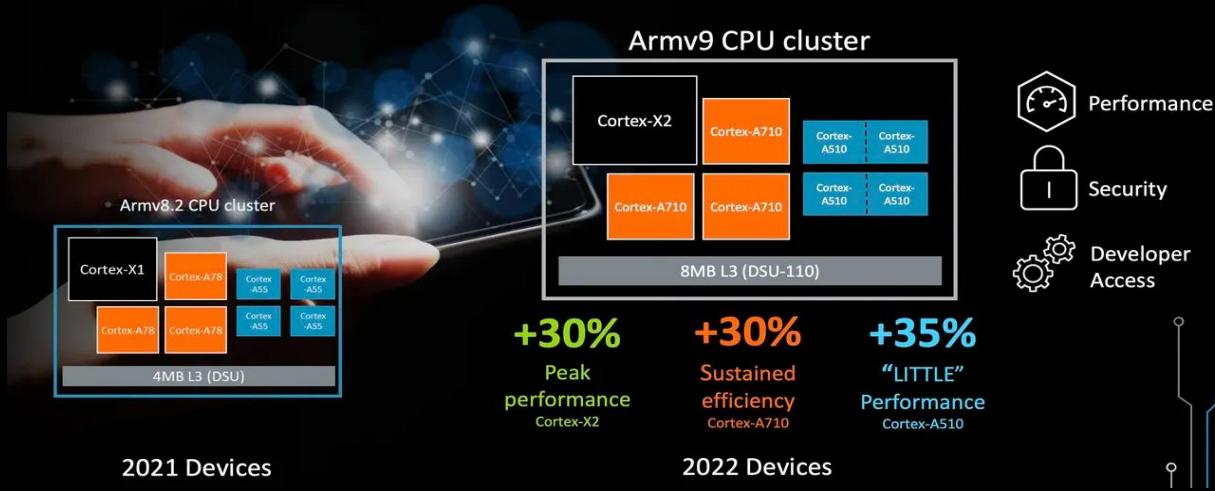
## 2.2.1.2 ARM v9

- <https://www.arm.com/company/news/2021/03/arms-answer-to-the-future-of-ai-armv9-architecture>
- <https://www.arm.com/blogs/blueprint/armv9>





## ■ Armv9 CPU Cluster: A Step Change in Premium Mobile



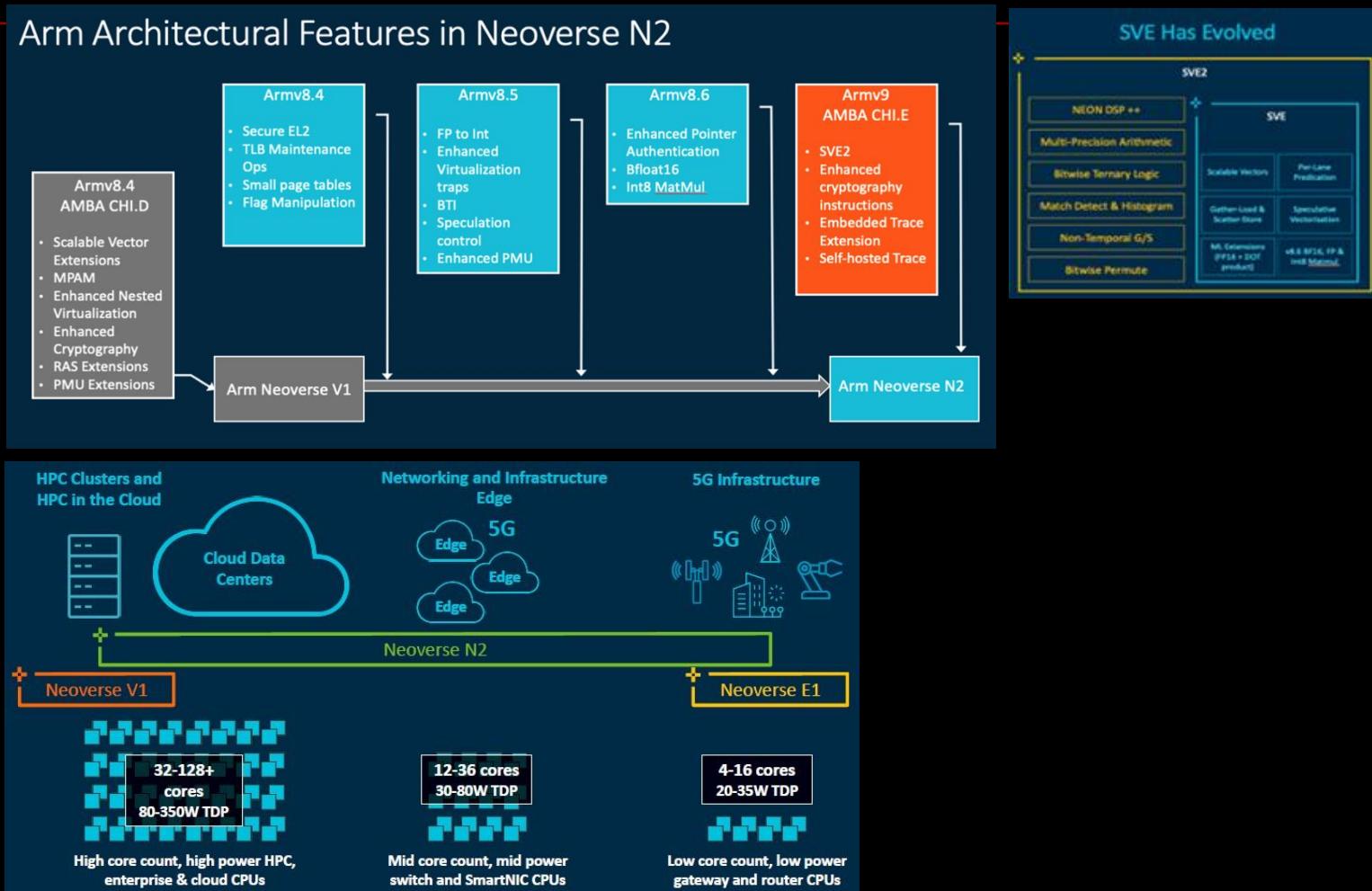


## Neoverse N2

### ■ Neoverse N2

<https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/arm-neoverse-n2-industry-leading-performance-efficiency>

#### Arm Architectural Features in Neoverse N2

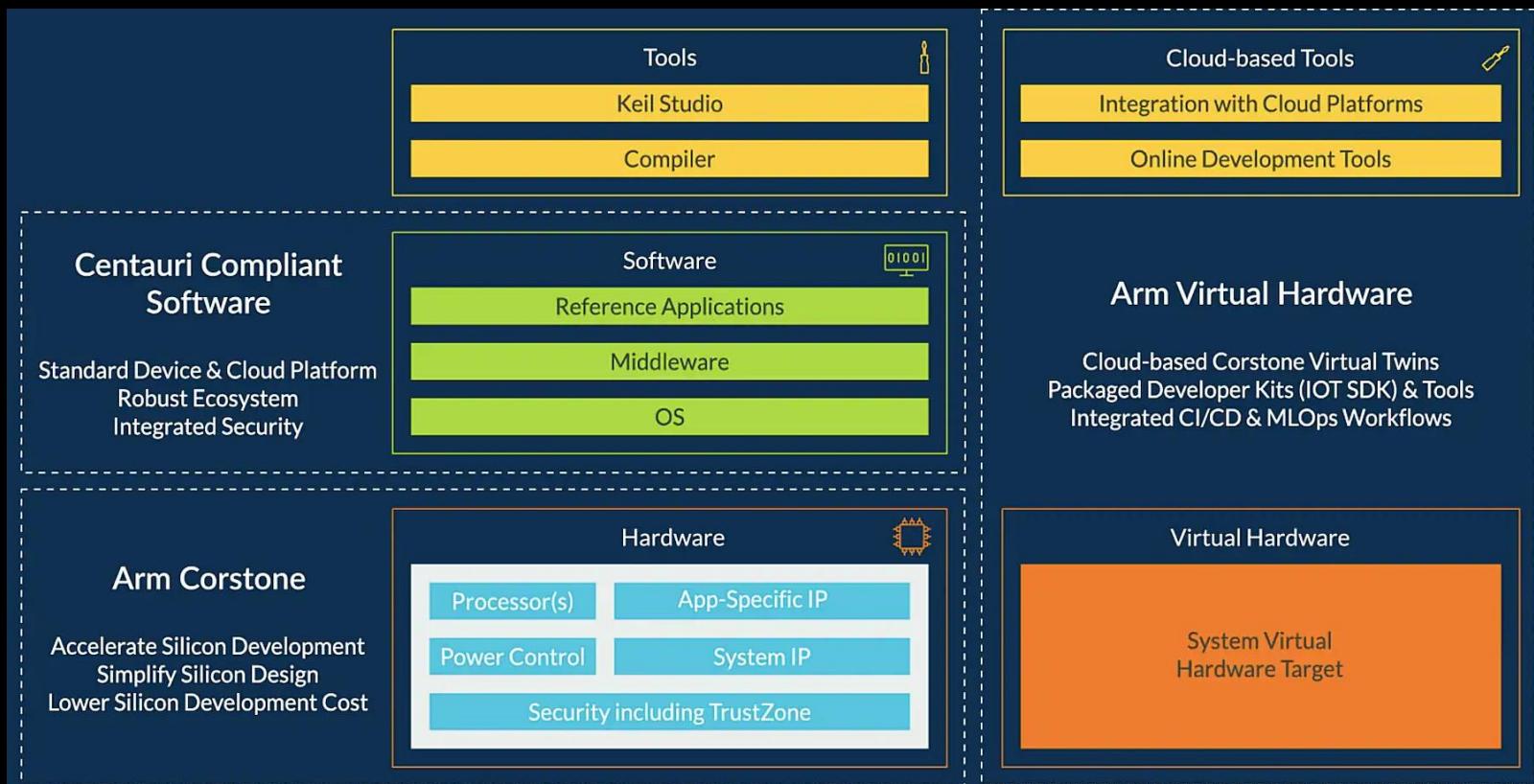




## 2.2.1.3 ARM Total Solutions for IoT

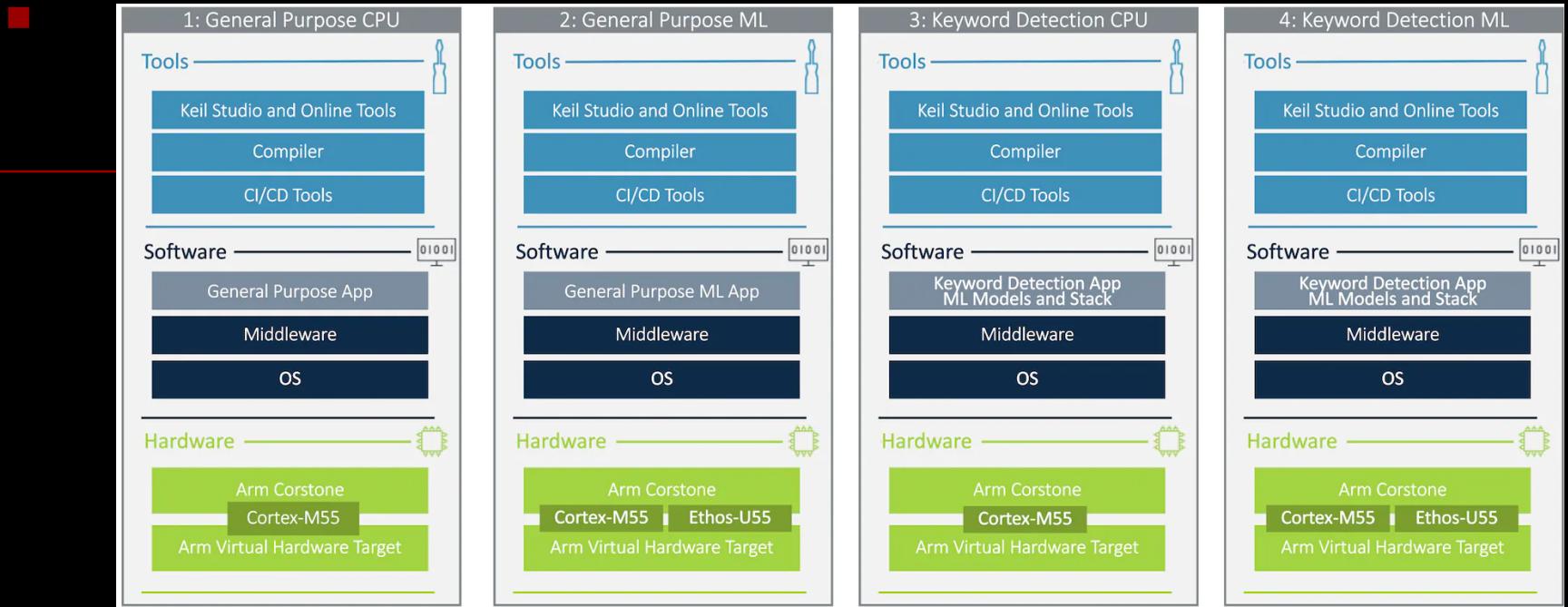
- <https://www.arm.com/solutions/iot/total-solutions-iot>

Developing IoT systems is incredibly complex. To fulfil the potential of IoT, we need to simplify and accelerate development for the entire value chain. ARM Total Solutions for IoT is an industry first, bringing together specialized processing capabilities with standardized, secure software, and innovative approaches to tooling and development.





## ARM's First Total Solutions for IoT



■ <https://www.arm.com/products/silicon-ip-subsystems/corstone-300>



## Enabling Software-Hardware Co-Development for IoT and ML

### ■ Total Solutions SDK

The first configurations of Arm Total Solutions for IoT are available as Software Development Kits (SDKs) free on the Arm Github. Total Solutions SDKs and other software can be run on the first beta release of Arm Virtual Hardware, available as an Amazon Machine Image (AMI) on AWS Marketplace.

**<https://github.com/arm-software/ATS-Keyword>**

This repo contains Arm's first [IoT Total Solution](#), "Keyword Detection". It provides general-purpose compute and ML workload use-cases, including an ML-based keyword recognition example that leverages the DS-CNN model from the [Arm Model Zoo](#).

The software supports multiple configurations of the Arm Corstone™-300 subsystem, incorporating the Cortex-M55 processor and Arm Ethos™-U55 microNPU. This total solution provides the complex, non differentiated secure platform software on behalf of the ecosystem, thus enabling you to focus on your next killer app.

This repo also supports a GitHub runner CI/CD workflow right out of the box which leverages [Arm Virtual Hardware](#), a simulation environment that enables software development without the need of physical SoCs.

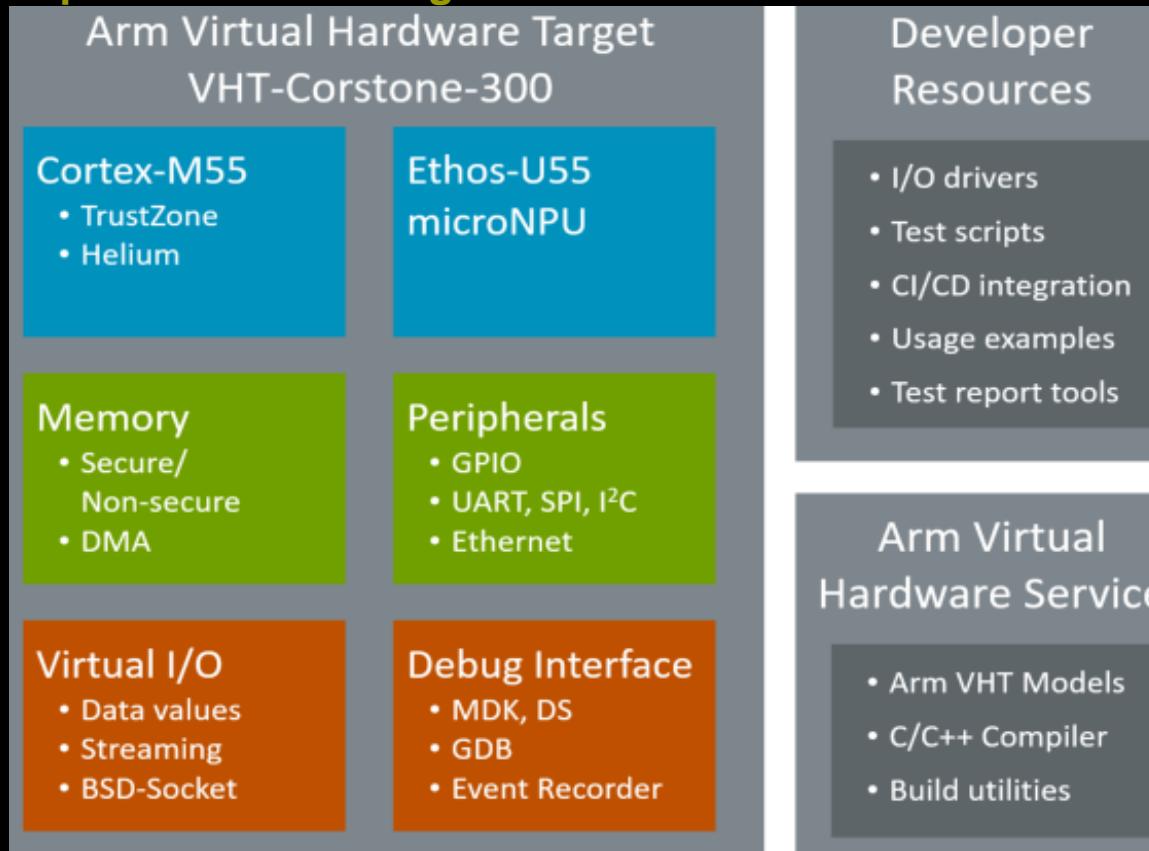


## ■ ARM Virtual Hardware

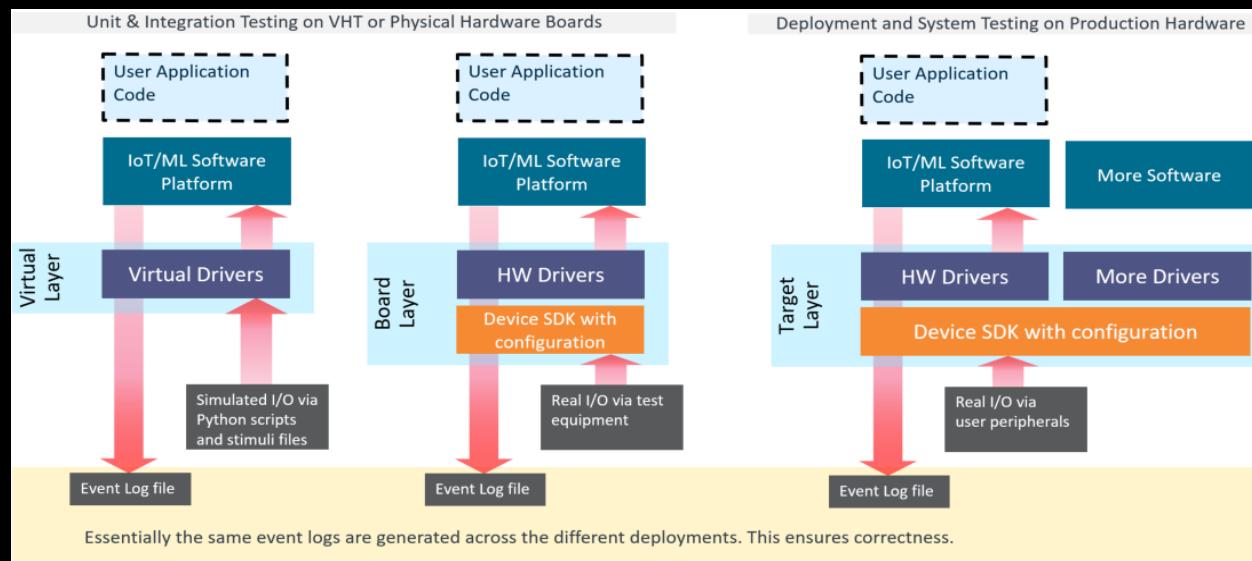
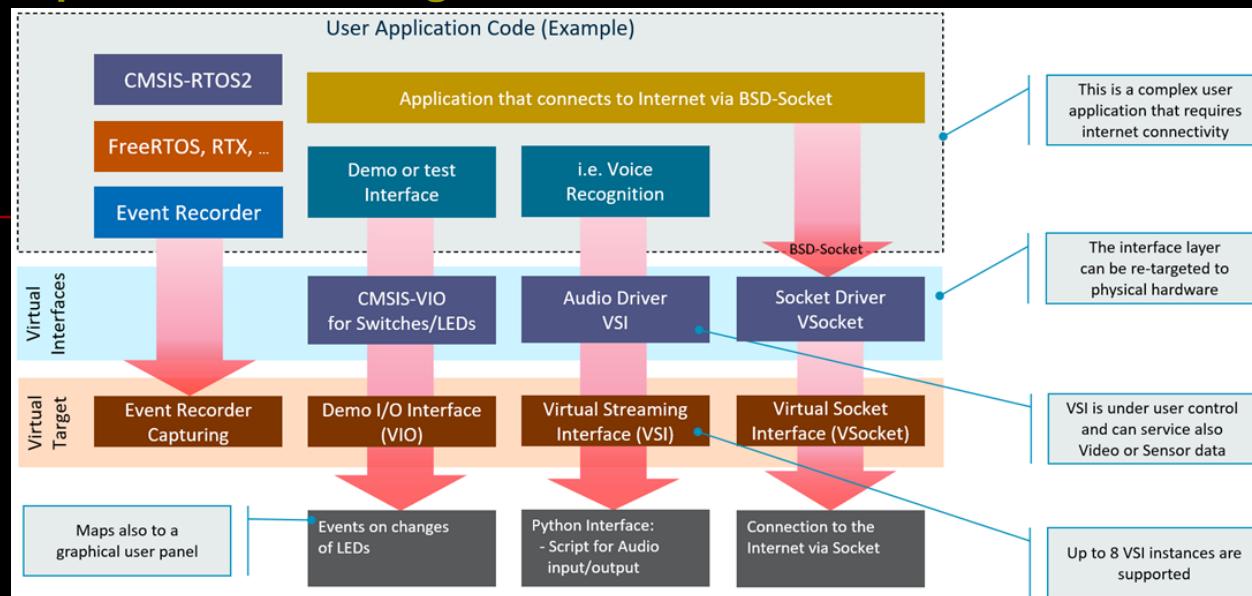
<https://www.arm.com/products/development-tools/simulation/virtual-hardware>

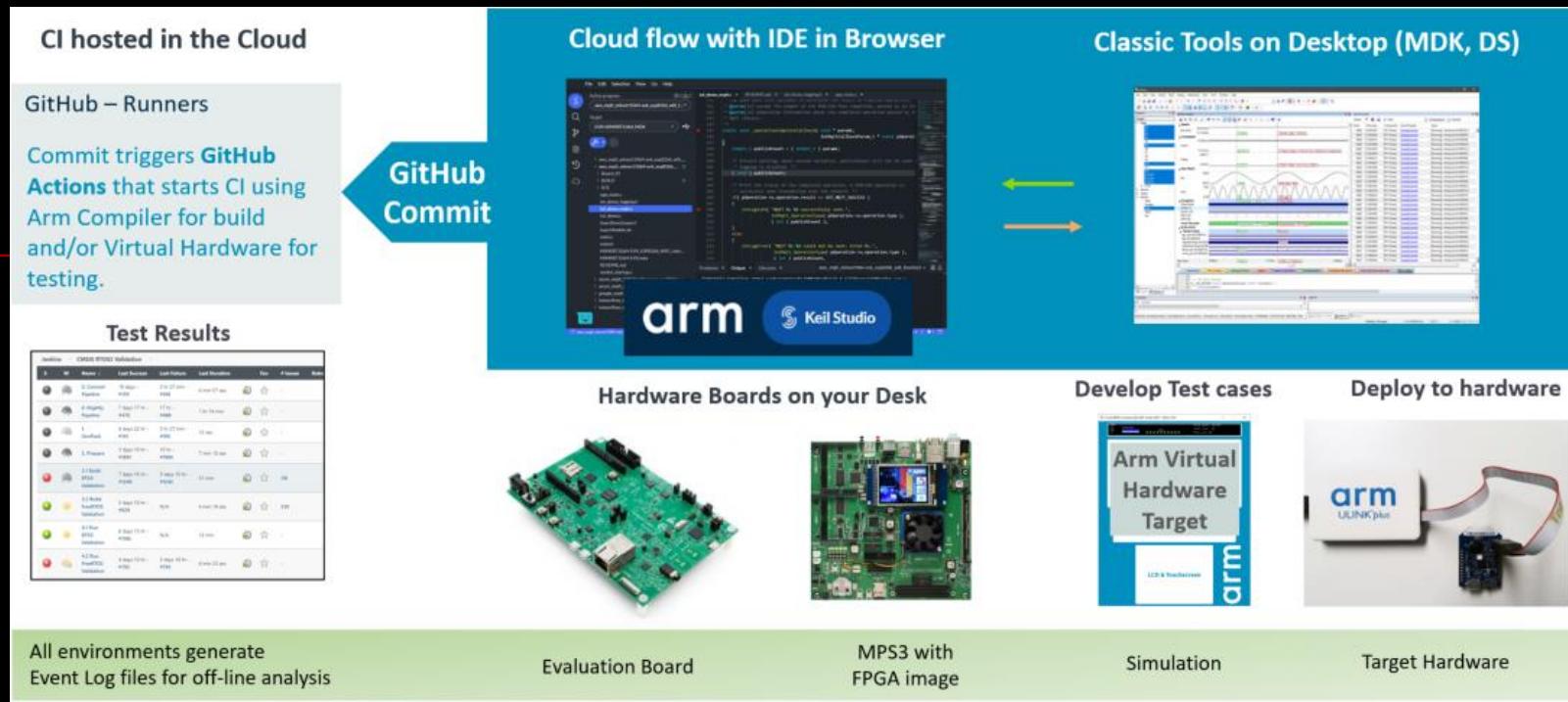
Arm Virtual Hardware is an evolution of Arm's modelling technology delivering accurate models of Arm-based SoCs for application developers to build and test software before and after silicon and hardware availability. It runs as a simple application in the cloud for simulating memory and peripherals, removing the complexity of building and configuring board farms for testing using modern agile software development practices such as continuous integration and continuous development CI/CD (DevOps) and MLOps workflows.

<https://arm-software.github.io/VHT/main/overview/html/index.html>



<https://arm-software.github.io/VHT/main/simulation/html/index.html>





<https://github.com/ARM-software/VHT>

<https://github.com/ARM-software/VHT-AMI>

<https://github.com/ARM-software/VHT-GetStarted>

...



# Embracing Cloud-Native Approaches for IoT

## ■ Project Centauri

<https://www.arm.com/solutions/iot/project-centauri>

Project Centauri draws upon and includes Arm's rich portfolio of Cortex-M software, bringing together complementary initiatives under a single MCU software strategy. These initiatives have laid the foundation for [Arm Total Solutions for IoT](#), giving the entire value chain foundational standards, such as Open-CMSIS-Pack, secure device management with PSA Certified and Trusted Firmware-M and ecosystem support for use case specific IoT solutions. Silicon partners, ODMs, OEMs, software and OS vendors, system integrators and hyperscalers will all benefit from interchangeable baseline software components that are deployable, updateable, and secure at scale.

## Components



### Foundational Standards

- ✚ Enable a standard cloud service-to-device specification
- ✚ Common interfaces required by CSP middleware



psacertified™



Trusted Firmware-M

### Secure Device Management

- ✚ Security model and implementation
- ✚ Scalable 3<sup>rd</sup> party certification
- ✚ Crypto primitives
- ✚ Secure provisioning
- ✚ Secure OTA updates

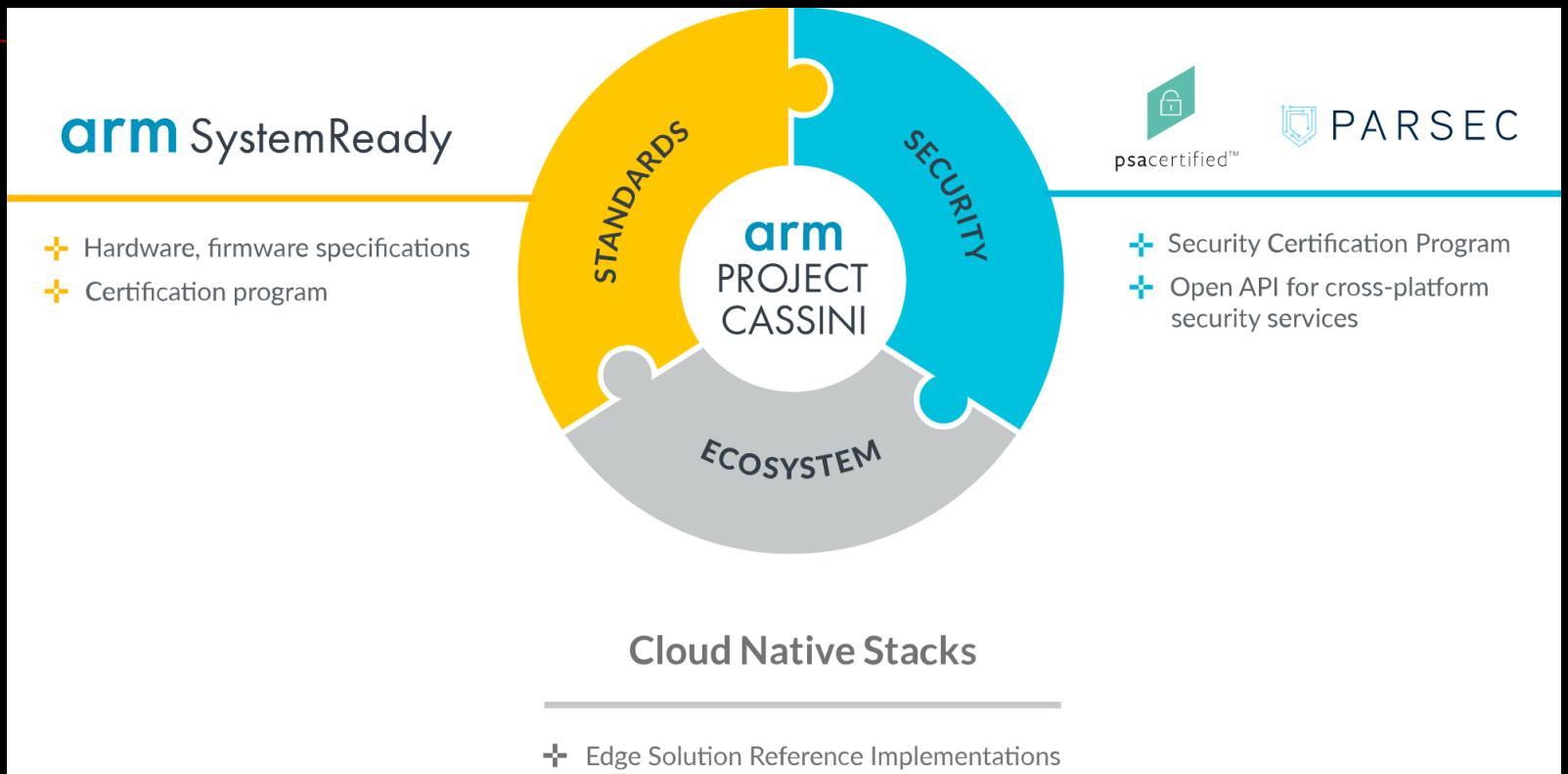
- ✚ Platform software and middleware
- ✚ Broad developer tooling enablement
- ✚ 300+ member partner ecosystem delivering capabilities in ML, cloud and more



## ■ Project Cassini

<https://www.arm.com/solutions/infrastructure/edge-computing/project-cassini>

Project Cassini is the open, collaborative, standards-based initiative to deliver a seamless cloud-native software experience for devices based on Arm Cortex-A. Developers of IoT and infrastructure edge solutions can access the power of Cassini today with Arm SystemReady and PSA Certified silicon and development boards and OS Linux support from the Arm ecosystem.





## ***Technologies to Unleash IoT Innovation***

- <https://www.arm.com/solutions/iot/iot-technology>

### **Custom Instructions**

Add unique application-specific features to enhance differentiation, performance and the efficiency of embedded devices. Arm Custom Instructions ensure easy integration with the existing software ecosystem for innovation without fragmentation.

### **DSP Extensions**

Achieve efficient signal processing capability for a wide range of applications with Arm's DSP extensions in Cortex processors. They provide a unique combination of compute scalability, power efficiency, and determinism, for simplified designs, reduced system complexity, and lower development costs.

### **Helium Technology**

Enhance compute capabilities with Arm Helium technology, the M-Profile Vector Extension (MVE) for Cortex-M CPUs. Designed for the Armv8.1-M architecture, it also provides a significant uplift in DSP and ML performance, combined with the energy-efficiency required for IoT devices.

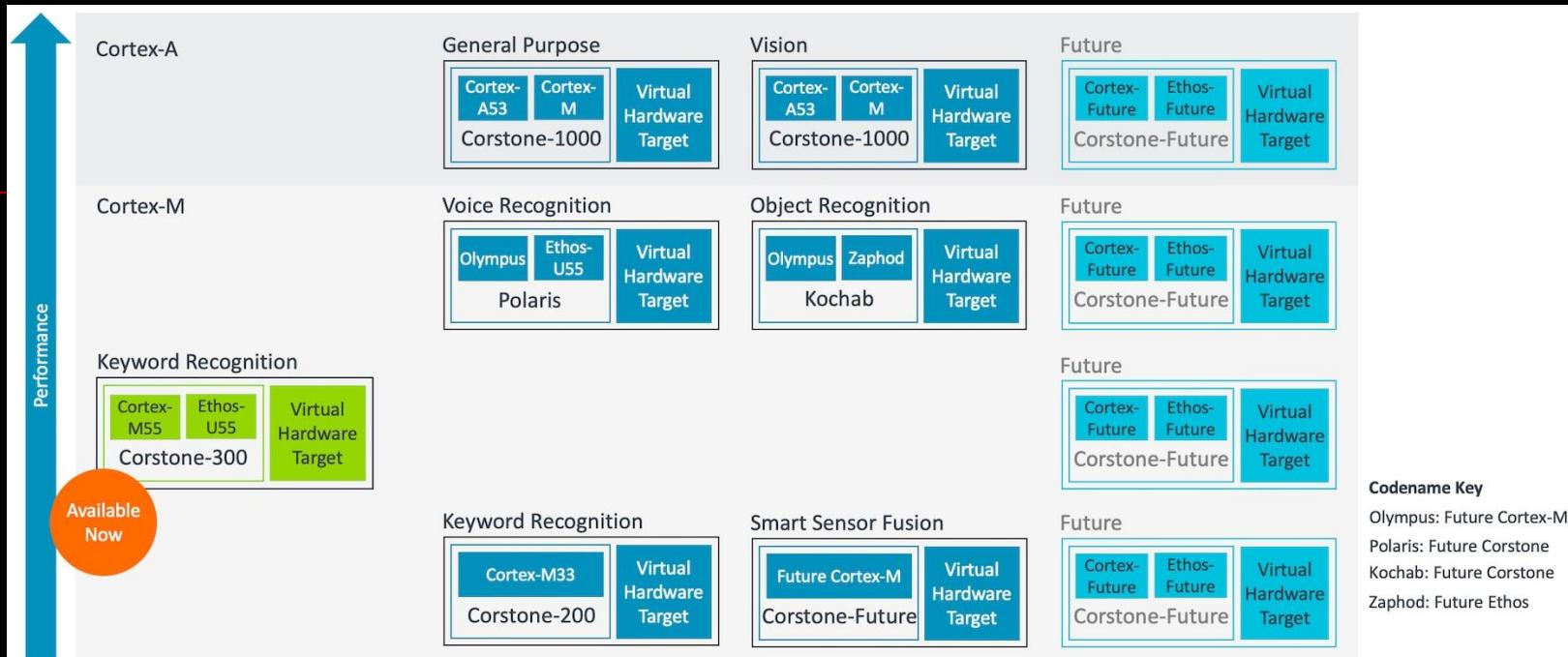
<https://www.arm.com/technologies/custom-instructions>

<https://www.arm.com/technologies/dsp>

<https://www.arm.com/technologies/helium>



# Delivering a Roadmap for Success





## Good Resources

- <https://www.arm.com/solutions/iot>
- <https://www.arm.com/solutions/iot/iot-technology>
- <https://developer.arm.com/solutions/internet-of-things>
- <https://www.arm.com/company/news/2021/10/arm-transforms-the-economics-of-iot-with-virtual-hardware-and-new-solutions-led-offering>
- <https://staceyoniot.com/arm-tries-to-solve-the-iots-fragmentation-problem-again/>
- <https://thenewstack.io/arm-looks-to-supercharge-iot-software-development/>
- ...



## 2.2.2 RISC-V

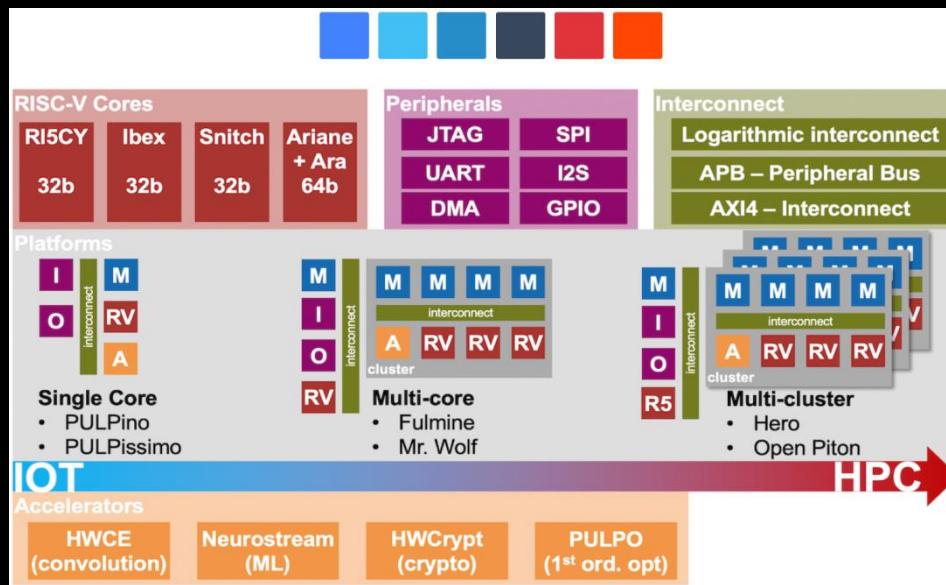
- <https://en.wikipedia.org/wiki/RISC-V>
- <https://riscv.org/>

### Development

- <https://riscv.org/exchange/>
- <https://github.com/riscv/riscv-isa-manual>

### Cores & SoC

- <https://github.com/riscv/riscv-cores-list>
- <https://riscv.org/exchange/cores-socs/>
- 

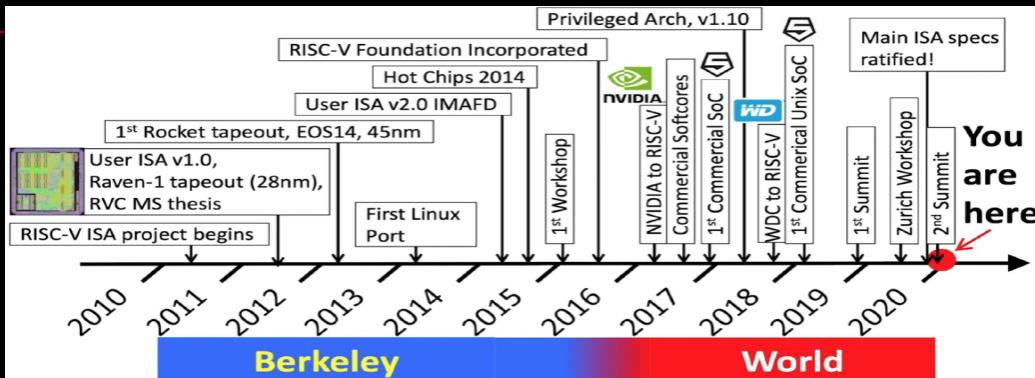


Source: <https://fuentitech.com/what-is-the-risc-v-ecosystem/19227/>



## Ecosystem

- <https://riscv.org/exchange/>
- <https://community.riscv.org/events/details/risc-v-foundation-bay-area-risc-v-group-presents-2021-risc-v-ecosystem-updates/>



Source: "Linux on RISC-V", D. Fustimi, ELC2020

## Incredible industry progress

- The European Processor Initiative finalized the first version of its **RISC-V accelerator architecture** and will deliver test chip in 2021.
- The RIOS Lab announced PicoRio, an affordable **RISC-V open source small-board computer** available in 2021.
- Imperas announced first **RISC-V verification reference model with UVM encapsulation**.
- Seagate announced **hard disk drive controller** with high-performance RISC-V CPU.
- GreenWaves **ultra-low power GAP9 hearables platform** enabling scene-aware and neural network-based noise reduction.
- Alibaba unveiled RV64GCV core in its Xuantie 910 processor for **cloud and edge servers**.
- Microchip released the first **SoC FPGA development kit** based on the RISC-V ISA.
- Andes released **superscalar multicore and L2 cache controller** processors.
- StarFive released the world's first **RISC-V AI visual processing platform**
- SiFive unveiled world's fastest development board for **RISC-V Personal Computers**.
- Micro Magic announced an incredibly **fast 64-bit RISC-V core** achieving 5GHz and 13,000 CoreMarks at 1.1V.

Source: "RISC-V: The Open Era of Computing", Calista Redmond, The Linux Foundation Spring Member Meeting 2021.



- <http://www.andestech.com/en/2022/02/07/looking-back-on-2021-strong-growth-momentum-of-risc-v-market/>
  - <https://www.techrepublic.com/article/risc-vs-open-chip-processors-expected-to-double-in-2022-and-double-again-in-2023/>
  - ...
-



## ***Good Resources***

- <https://riscv.org/exchange/>
  - <https://github.com/riscvarchive/riscv-software-list>
  - ...
-



## X86, ARM, and RISC-V

- <https://riscv.org/whats-new/2022/02/intel-corporation-makes-deep-investment-in-risc-v-community-to-accelerate-innovation-in-open-computing/>

---

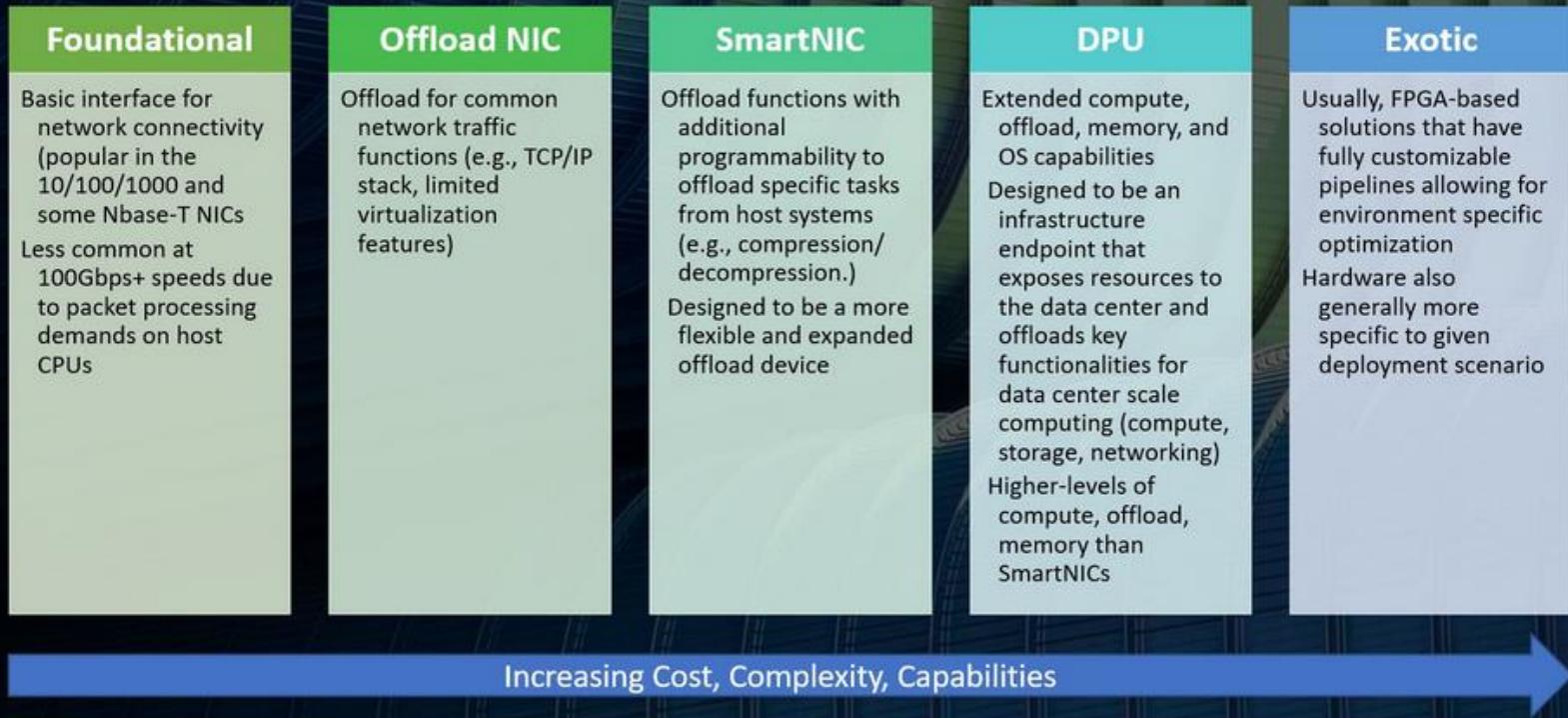
...

## 2.2.3 SmartNIC/DPU/IPU Overview

- <https://www.servethehome.com/dpu-vs-smartnic-sth-nic-continuum-framework-for-discussing-nic-types/>



# 2021 STH NIC Continuum





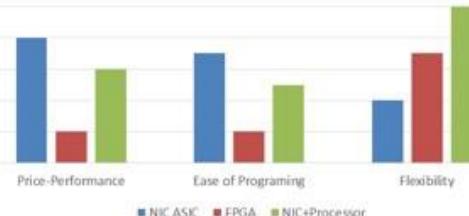
## 2.2.3.1 SmartNIC

- <https://blogs.nvidia.com/blog/2021/10/29/what-is-a-smartnic/>
- <https://www.nextplatform.com/2018/08/06/living-in-the-smartnic-future/>

Capability	Workloads Accelerated	Basic NIC	Cloud NIC	SmartNIC
<b>Basic Virtualization and Data Movement</b>				
TCP/IP Acceleration	Enterprise workloads	✓	✓	✓
SRIOV (NIC-level virtualization)	Enterprise workloads	✓	✓	✓
VXLAN, NVGRE (Network-level virtualization)	Multi-Tenant workloads	✓	✓	✓
<b>Data Transport Acceleration</b>				
RDMA and RoCE	Big Data, Storage, AI/ML, Virtual Machines		✓	✓
ACLs and QoS support	Web serving, Content Distribution (CDN)		✓	✓
OVS hardware acceleration	Efficient, Scalable Virtualized Apps		✓	✓
Storage functions acceleration (NVMe-oF, RAID, T10)	High-performance, low-latency Storage		✓	✓
Flow match/action engine	Software Defined Networking (SDN)		✓	✓
Flow monitoring/reporting	Visibility, Network Packet Broker, IBN		✓	✓
<b>Smart Networking, Security &amp; Virtualization</b>				
Stateful n-tuple / ACL filtering	Load Balancing, IPS/IDS, Unified Threat Management			✓
Isolated OVS control plane, Fault Isolation & Response	Bare Metal Cloud, High-Availability Datacenter			✓
Analytics Engine	DPI, Network Monitoring and Diagnostics			✓
On-board Security VNFs	Firewall, Anti-DDoS, Anti-Malware, Host Introspection			✓
datapath encryption/decryption	Secure data-at-rest or data-in-flight			✓
Public Key crypto, hardware RNG	Secure Authentication & Key Exchange			✓



NIC Implementation Comparison



- ASIC Based

- Excellent price-performance
- Vendor development cost high
- Programmable and extensible
- Easy to program but flexibility is limited to pre-defined capabilities



- FPGA Based

- Good performance but expensive
- Very difficult to program
- Workload specific optimization



- SOC Based

System on Chip - NIC + CPU

- Good price-performance
- C Programmable Processors
- Highest Flexibility
- Easiest programmability

- <https://www.electronicdesign.com/industrial-automation/article/21134459/xilinx-why-is-a-smartnic-better-than-a-regular-nic>





## 2.2.3.2 DPU

### ■ [https://en.wikipedia.org/wiki/Data\\_processing\\_unit](https://en.wikipedia.org/wiki/Data_processing_unit)

A **data processing unit (DPU)** is a programmable specialized electronic circuit with **hardware acceleration** of **data processing** for **data-centric computing**.<sup>[1]</sup> The data is transmitted to and from the component as multiplexed packets of information. A DPU generally contains a **CPU**, **NIC** and programmable data **acceleration engines**.<sup>[2][3]</sup> This allows DPUs to have the generality and the programmability of **central processing units** while being specialized to operate efficiently on **networking packets**, storage requests or analytics requests.<sup>[4][5][6]</sup>

The data **acceleration engines** differentiates itself from a CPU by a larger degree of parallelism (required to process many requests) and from a **GPU** by a **MIMD** architecture rather an **SIMD** architecture (required as each request needs to make different decisions and follow a different path through the chip).<sup>[7][8]</sup> DPUs can be either **ASIC-based**, **FPGA-based** or **SoC-based**.<sup>[9]</sup> DPUs have been increasing used in **data centers** and **supercomputers** since their introduction in the 2010s due to the raise to **data-centric computing**, **big data**, and **artificial intelligence/machine learning/deep learning**.<sup>[10]</sup> DPUs are designed to be independent infrastructure endpoints.<sup>[11]</sup>

### ■ <https://blogs.nvidia.com/blog/2020/05/20/whats-a-dpu-data-processing-unit/>

### ■ **Vendors & product lines**

- [Nvidia/Mellanox Technologies](#): BlueField, ConnectX, Innova<sup>[12]</sup>
- [Marvell Technology](#): OCTEON and ARMADA<sup>[13]</sup>
- [Fungible](#): F & S Series<sup>[14]</sup>
- [Pensando](#): Capri, Elba & DSC<sup>[15][3]</sup>
- [Broadcom](#): Stingray<sup>[16]</sup>
- [Intel](#): Infrastructure Processing Unit (IPU)<sup>[17]</sup>

...

...



## ■ Elements of a DPU

While many other features may be present on a DPU, the common features of DPU chips seem to be:

High-speed networking connectivity (25Gbps minimum, usually 100Gbps+)

High-speed packet processing with specific acceleration and often programmable logic (P4/ P4-like is common)

A CPU core complex (often Arm or MIPS based in this generation)

Memory controllers (commonly DDR4 but we also see HBM and DDR5 support)

Accelerators (often for crypto or storage offload)

PCIe Gen4 lanes (run as either root or endpoints)

Security and management features (offering a hardware root of trust as an example)

Runs its own OS separate from a host system (commonly Linux, but the subject of [VMware Project Monterey ESXi on Arm](#) as another example)

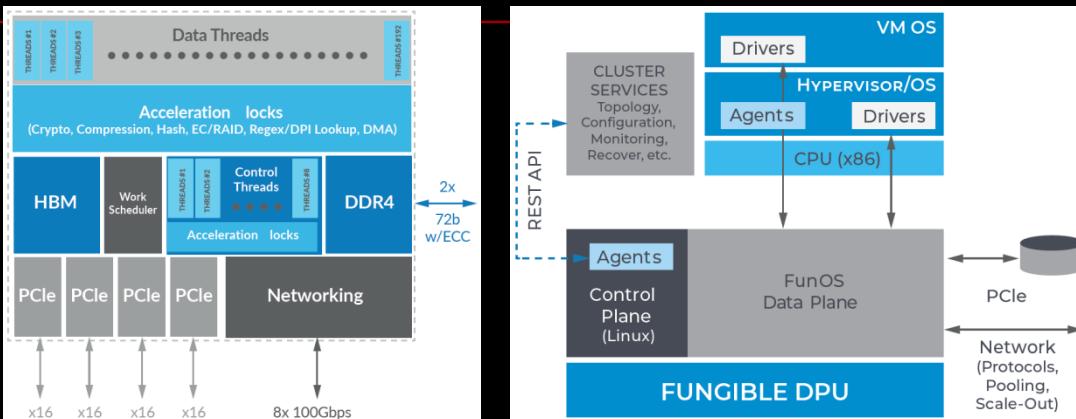
Designed to be an infrastructure endpoint that exposes resources to the data center and offloads key functionalities for data center scale computing

Source: <https://www.servethehome.com/dpu-vs-smartnic-sth-nic-continuum-framework-for-discussing-nic-types/>

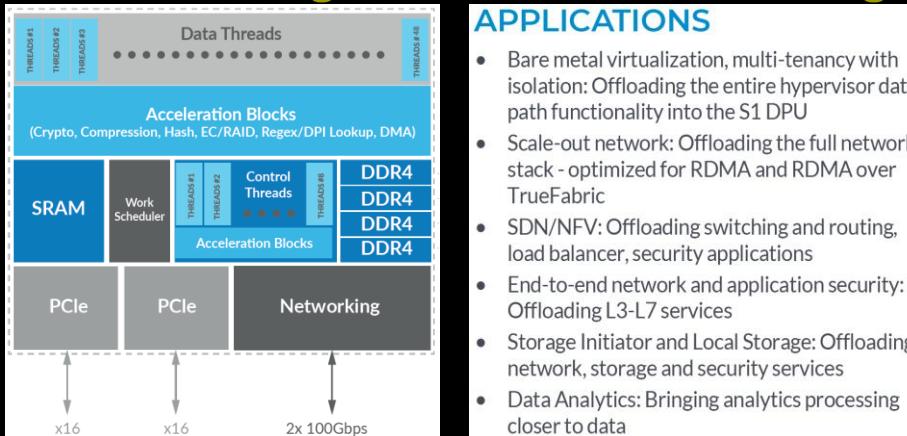


## DPU from Fungible

- <https://www.fungible.com/product/dpu-platform/>
- <https://www.fungible.com/wp-content/uploads/2021/09/PB0028.02.12020914-Fungible-F1-Data-Processing-Unit.pdf>



- <https://www.fungible.com/wp-content/uploads/2021/09/PB0029.03.12020914-Fungible-S1-Data-Processing-Unit.pdf>





## DPU from Nvidia

- <https://www.nvidia.com/en-us/networking/products/data-processing-unit/>
- <https://blogs.nvidia.com/blog/2020/05/20/whats-a-dpu-data-processing-unit/>
- <https://nvidianews.nvidia.com/news/nvidia-extends-data-center-infrastructure-processing-roadmap-with-bluefield-3>  
**First 400Gb/s DPU with Line-Rate Processing of Software-Defined Networking, Storage and Cybersecurity**
- <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf>



ARM	16 Arm A78 cores Fully programmable OS Apps/services, service chaining Control Path / Slow Path Memory to Memory Accelerators
Datapath Accelerator	16 cores, 256 threads Programmability through DOCA Heavy multi-threading application acceleration
ASAP <sup>2</sup>	Programmable packet processor flow pipeline Flow table based Data Path
...	



# Features

## Network and Host Interfaces

### Network Interfaces

- > Ethernet - 1, 2, 4 ports with up to 400 Gb/s connectivity
- > InfiniBand - Single port of NDR (400Gb/s), or dual ports of NDR200 / HDR (200Gb/s)

### PCI Express Interface

- > 32 lanes of PCIe Gen 5.0
- > PCIe switch bi-furcation of up to 16 downstream ports
- > Non-transparent bridging (NTB) support

## Compute and Memory

### Arm CPU Cores

- > Up to 16 Armv8.2+ A78 Hercules cores (64-bit)
- > 8MB L2 cache
- > 16MB LLC system cache

### Programmable Datapath Accelerator

- > 16 cores, 256 threads
- > Programmability through DOCA
- > Heavy multi-threading applications acceleration

### DDR DIMM Support

- > Dual DDR5 5600MT/s DRAM controllers
- > 16GB on-board DDR5
- > ECC error protection support

## Hardware Accelerations

### Security

- > Secure boot with Public key accelerator (PKA) root-of-trust
- > Secure firmware update
- > Flash encryption
- > Cerberus compliant

- > Functional isolation layer
- > Regular expression [RegEx] matching processor
- > MACsec/IPsec/TLS data-in-motion encryption
  - > AES-GCM 128/256-bit key
  - > AES-XTS 256/512-bit data-at-rest encryption
- > Connection tracking for stateful firewall
- > Public key accelerator (PKA)
  - > RSA, Diffie-Hellman, DSA, ECC, EC-DSA, EC-DH
- > True random number generator (TRNG)

### Storage

- > BlueField SNAP - Elastic block storage - NVMe™ and VirtIO-blk
- > NVMe-oF™ and NVMe/TCP™ acceleration
- > Decompression engine
- > Erasure coding for RAID implementation
- > M.2 / U.2 connectors for direct attached storage

### Networking

- > RoCE, Zero Touch RoCE
- > ASAP<sup>2</sup> - Accelerated Switch and Packet Processing® for SDN and VNF acceleration
- > Single Root I/O Virtualization (SR-IOV)
- > VirtIO acceleration
- > Overlay network acceleration
  - > VXLAN, GENEVE, NVGRE
- > Programmable flexible parser: user defined classification
- > Connection tracking (L4 firewall)
- > Flow mirroring, sampling and statistics
- > Header rewrite
- > Hierarchical QoS
- > Stateless TCP offloads

### HPC/AI Accelerations

- > HPC / AI All-to-All engine
- > NVIDIA GPUDirect
- > NVIDIA GPUDirect Storage (GDS)

- > HPC MPI Tag Matching

### Advanced Timing and Synchronization

- > IEEE 1588v2 (any profile)
- > G.8273.2 Class C
- > PTP hardware clock (PHC)
- > Line rate hardware timestamp
- > SyncE
- > G.8262.1 (eEEC)
- > Configurable PPS In and PPS Out
- > Time triggered scheduling
- > Time-based SDN acceleration

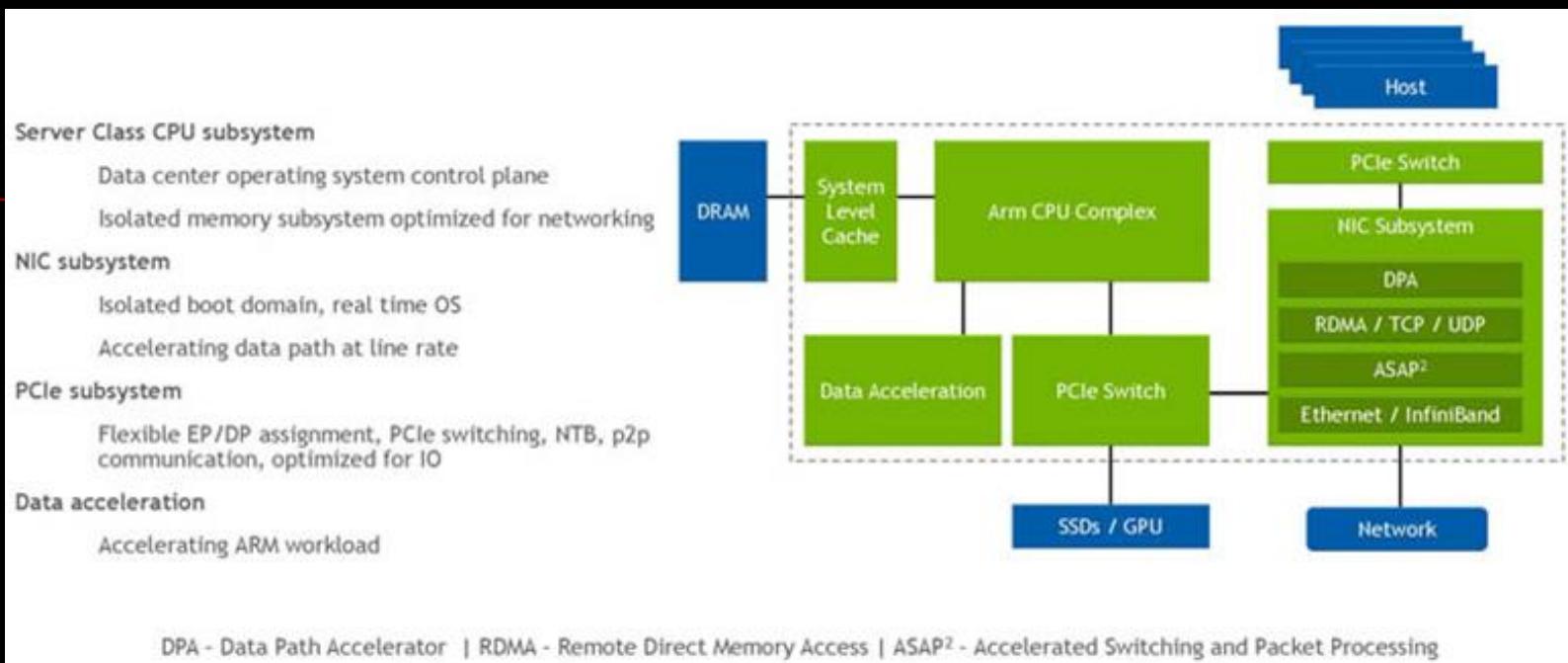
### Boot Options

- > Secure boot (RSA authenticated)
- > Remote boot over Ethernet
- > Remote boot over iSCSI
- > PXE and UEFI

### Management

- > 1GbE out-of-band management port
- > NC-SI, MCTP over SMBus, and MCTP over PCIe
- > PLDM for Monitor and Control DSP0248
- > PLDM for Firmware Update DSP026
- > I<sup>2</sup>C interface for device control and configuration
- > SPI interface to flash
- > eMMC memory controller
- > UART
- > USB

## ■ System Architecture



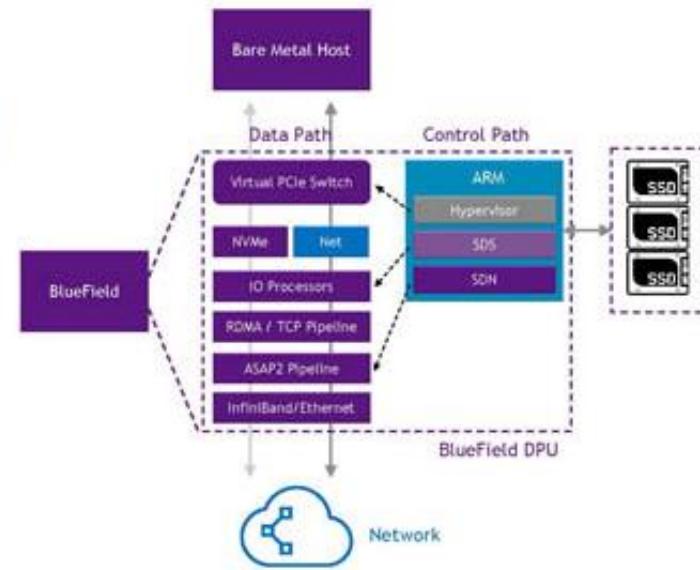
Source: <https://www.servethehome.com/nvidia-bluefield-3-dpu-architecture-at-hot-chips-33/>



## DPU Accelerated Switching And Packet Processing

Programmable Data Path | Software-Defined Orchestration

Accelerated	Software Defined
✓ Virtio-Net/Other Emulation	✓ eSwitch management
✓ QoS & scheduling	✓ Connection Establishment
✓ Telemetry and statistics	✓ Key Association
✓ Micro Segmentation	✓ Monitoring & Stats
✓ Encryption (Ipsec / MACsec)	✓ IDS / IPS / WAF
✓ Tunneling (VXLAN / GRE)	
✓ NAT	
✓ Routing	
✓ ACL	



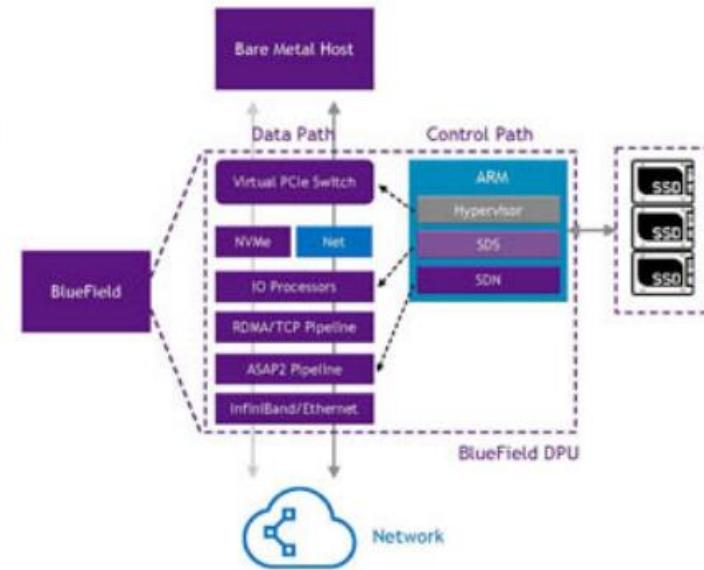
Source: <https://www.servethehome.com/nvidia-bluefield-3-dpu-architecture-at-hot-chips-33/>



## DPU Accelerated Storage Processing

Programmable Data Path | Software Defined Orchestration

Accelerated	Software Defined
✓ NVMe / Virtio-Block Emulation	✓ LVM / RAID control plane
✓ Data Reduction, Erasure Coding	✓ Key Association
✓ Data at Rest Crypto & Integrity	✓ QoS Monitoring & Stats
✓ RDMA - RoCE / InfiniBand, TCP	✓ Namespace/controller management
✓ Data-in-Flight Encryption	



Source: <https://www.servethehome.com/nvidia-bluefield-3-dpu-architecture-at-hot-chips-33/>



## DPU Enables Cloud-native Supercomputing

### Multi-Tenancy with Zero-Trust Security

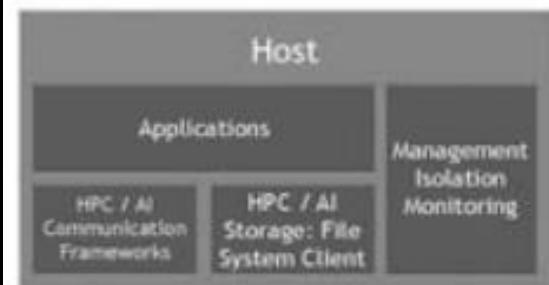
Collective offload with UCC accelerator

Smart MPI progression

User-defined algorithms

1.4X higher application performance

### Traditional Supercomputing



### Cloud-Native Supercomputing



Source: <https://www.servethehome.com/nvidia-bluefield-3-dpu-architecture-at-hot-chips-33/>

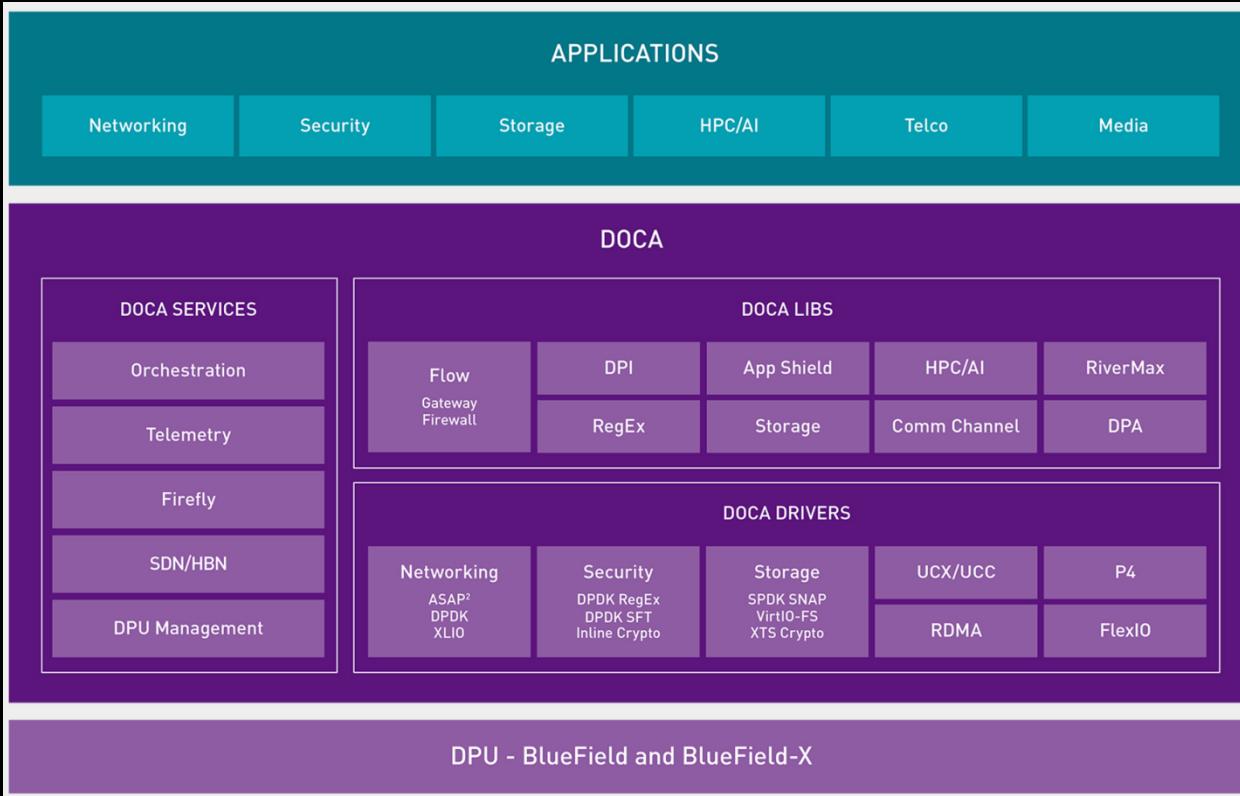
# DOCA

- <https://developer.nvidia.com/networking/doca>
- **Data Center Infrastructure-on-a-Chip Architecture**



NVIDIA® DOCA™ is the key to unlocking the potential of the NVIDIA BlueField® data processing unit (DPU) to offload, accelerate, and isolate data center workloads. With

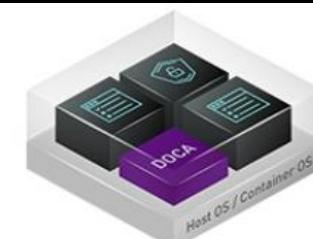
DOCA, developers can program the data center infrastructure of tomorrow by creating software-defined, cloud-native, DPU-accelerated services with zero-trust protection to address the increasing performance and security demands of modern data centers.





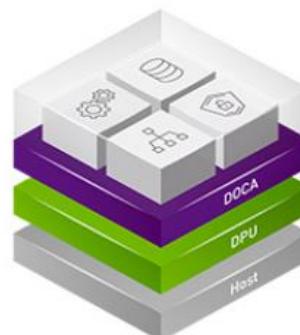
## ■ Key Components

- Industry-standard APIs: DPDK, SPDK, P4, Linux Netlink
- Network acceleration SDK: NVIDIA Accelerated Switching and Packet Processing (ASAP2)<sup>TM</sup> software-defined networking (SDN), emulated VirtIO, P4, 5T for 5G technology, Firefly time synchronization
- Security acceleration SDK: inline cryptography, deep packet inspection
- Storage acceleration SDK: storage emulation and virtualization, crypto and compression
- Remote direct-memory access (RDMA) acceleration SDK: unified communications and collaboration (UCC) and Unified Communication X (UCX), RDMA verbs, GPUDirect®
- Management SDK: deployment, provisioning, service orchestration
- User space and kernel



Business Application Domain

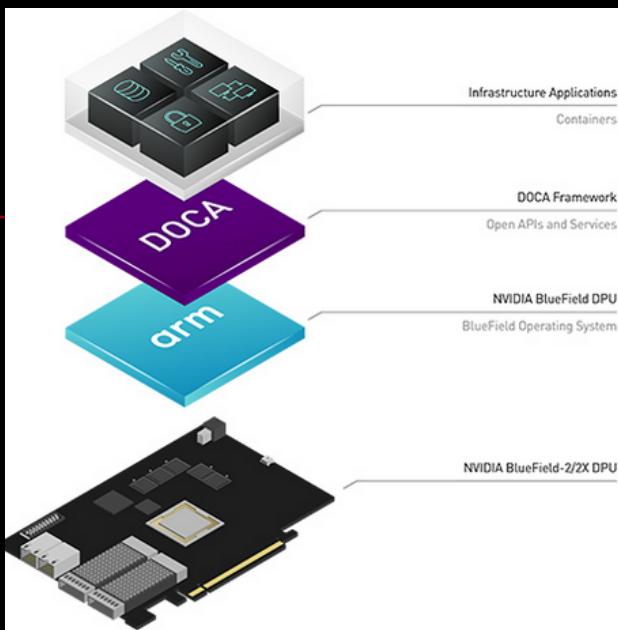
Functional Isolation



Infrastructure Services Domain



## ■ BlueField DPU OS and Drivers



- The BlueField OS includes the bootloader, OS kernel, necessary network interface card (NIC) firmware, NVIDIA drivers, sample filesystem, and toolchain—all certified as part of the NVIDIA NGC™ catalog.
- Ubuntu Server 20.04 ships with NVIDIA BlueField DPUs as commercial-grade Linux distribution with continuous OS and security updates.
- DOCA software is available on every leading operating system as a standalone package without a bundled OS for Arm® and x86 architectures.



## Ecosystem



Source: <https://www.servethehome.com/nvidia-bluefield-3-dpu-architecture-at-hot-chips-33/>



## 2.2.3.3 IPU from Intel

- <https://www.intel.com/content/www/us/en/products/network-io/smartnic.html>



- Intel® Infrastructure Processing Unit (Intel® IPU)

**Intel® IPU C5000X-PL Platform**  
High-performance Cloud Infrastructure acceleration platform with 2x25GbE network interfaces. It has the capability to support Cloud Infrastructure workloads such as Open vSwitch, NVMe over Fabrics and RDMA over Converged Ethernet v2 (RoCEv2).

[Learn more >](#)

[Accelerate Your Data Center with Intel® FPGAs >](#)

[IPU-Based Cloud Infrastructure: The Fulcrum for Digital Business >](#)

**Intel® IPU Platform Codenamed Oak Springs Canyon**  
Intel's next-generation high-performance Cloud Infrastructure acceleration platform with 2x100GbE network interfaces. In addition to supporting Cloud Infrastructure workloads such as Open vSwitch, NVMe over Fabrics and RDMA over Converged Ethernet v2 (RoCEv2), Oak Springs Canyon has an integrated hard crypto block, which enables securing these workloads at line rate.

[Learn more >](#)

[Accelerate Your Data Center with Intel® FPGAs >](#)

[IPU-Based Cloud Infrastructure: The Fulcrum for Digital Business >](#)

**Intel® IPU SoC Codenamed Mount Evans**  
200G IPU, co-developed with a top cloud provider and designed for performance at scale.

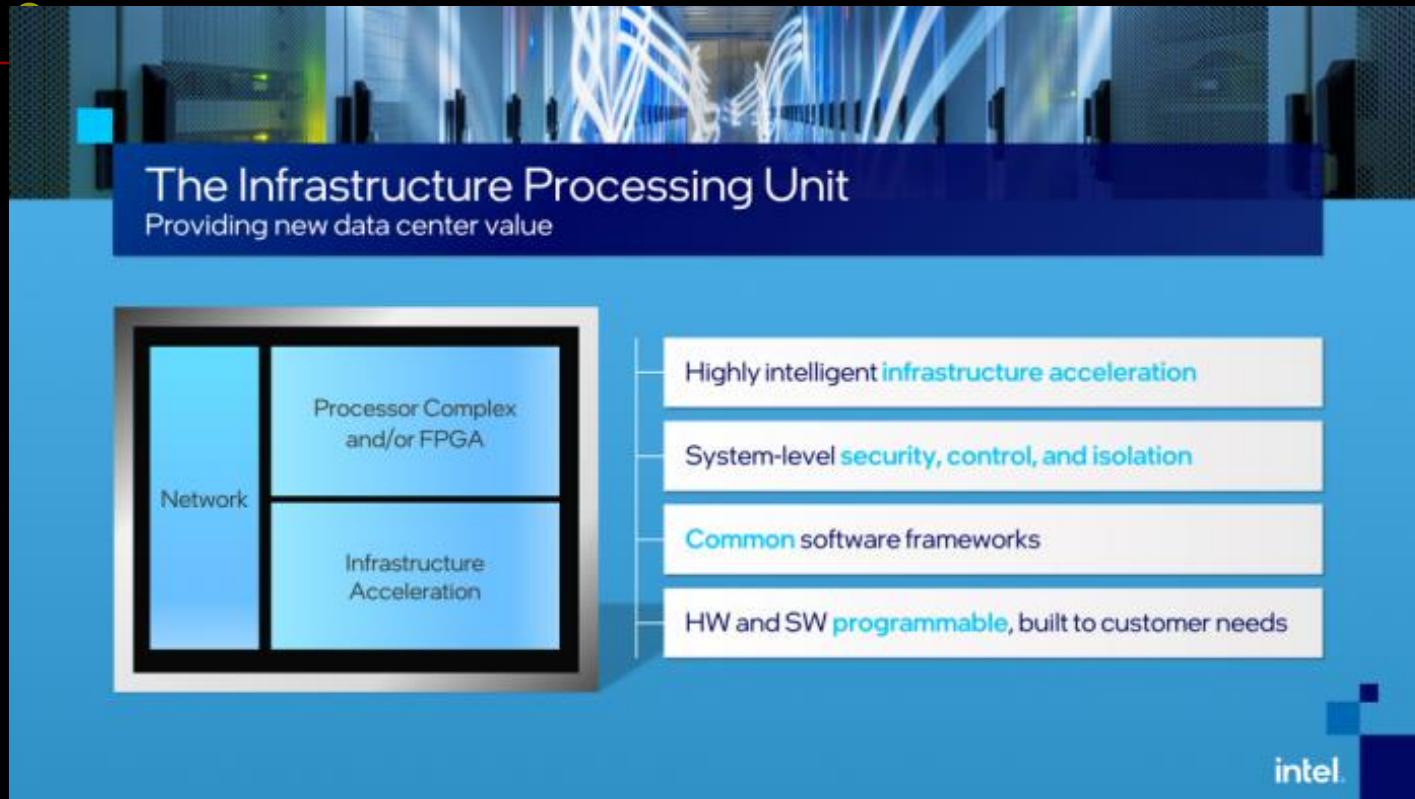
[Learn more >](#)

[IPU-Based Cloud Infrastructure: The Fulcrum for Digital Business >](#)



## What is it

- An IPU is an advanced networking device with hardened accelerators and Ethernet connectivity that accelerates and manages infrastructure functions using tightly coupled, dedicated, programmable cores. An IPU offers full infrastructure offload and provides an extra layer of security by serving as a control point of the host for running infrastructure applications.



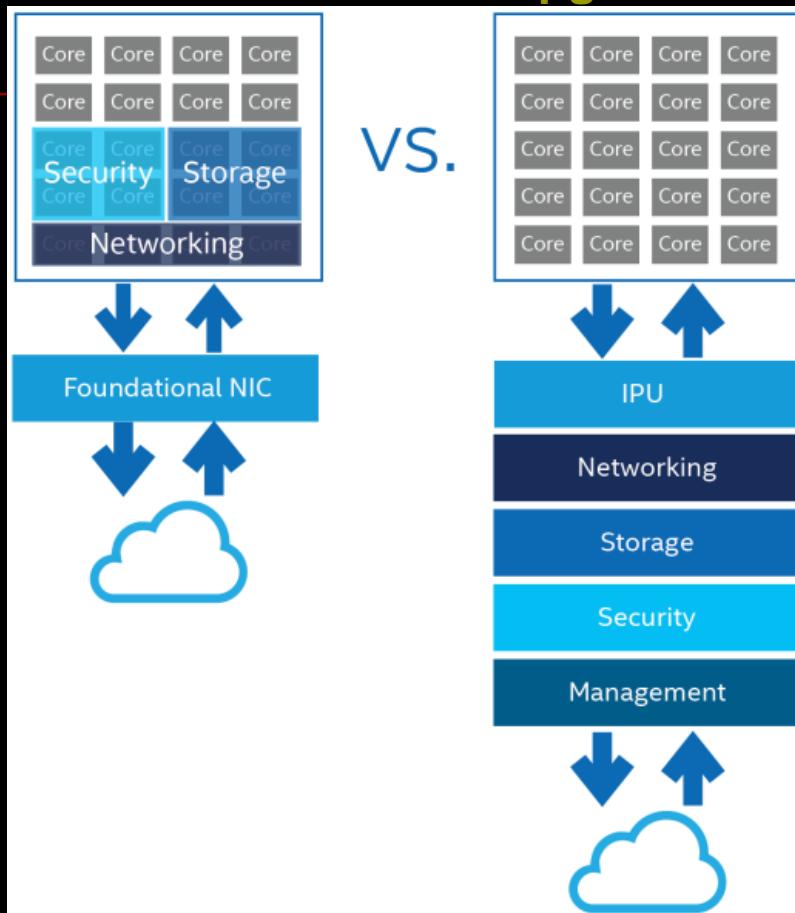
## What is the difference between a SmartNIC and IPU?

- An IPU is capable of offloading the entire infrastructure stack from the host and can control how the host attaches to this infrastructure. This gives the service provider an extra layer of security and control, enforced in hardware by the IPU. A SmartNIC has similar networking and offload capabilities as the IPU but remains under the control of the host as a peripheral.



## ***IPU can help increase data center performance***

- <https://www.intel.com/content/www/us/en/products/programmable/accelerate-data-center-fpgas.html>



**Figure 1.** IPUs based on the Intel FPGA IPU C5000X platform can help increase data center performance through increased network throughput, lower latency, and better server CPU utilization



#### 2.2.3.4 Good Resources

- <https://premioinc.com/blogs/blog/what-is-a-dpu-data-processing-unit>
  - ...
-



## 2.2.4 Chiplet Overview

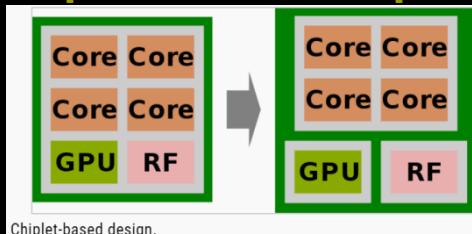
### ■ <https://en.wikipedia.org/wiki/Chiplet>

A chiplet<sup>[1][2][3][4]</sup> is a tiny integrated circuit (IC) that contains a well-defined subset of functionality. It is designed to be combined with other chiplets on an interposer in a single package. A set of chiplets can be implemented in a mix-and-match "LEGO-like" assembly. This provides several advantages over a traditional system on chip (SoC):

- Reusable IP (Intellectual Property):<sup>[5]</sup> the same chiplet can be used in many different devices
- Heterogeneous integration:<sup>[6]</sup> chiplets can be fabricated with different processes, materials, and nodes, each optimized for its particular function
- Known good die:<sup>[7]</sup> chiplets can be tested before assembly, improving the yield of the final device

Multiple chiplets working together in a single integrated circuit may be called a multi-chip module (MCM), hybrid IC, or 2.5D IC.

### ■ <https://en.wikichip.org/wiki/chiplet>



Chiplet-based design.

#### Overview [edit]

Chiplets refer to the independent constituents which make up a large chip built out of multiple smaller dies. Historically the need to go with multiple chips was driven by the [reticle limit](#) which dictated the maximum size of chip possible to be [fabricated](#). Designs that exceeded the reticle limit had to be split up into smaller dies in order to be manufacturable. As [process technologies](#) continued to enable higher integration, multiple dies were merged into single, more complex integrated circuits. More recently, economics has resulted in a reversal of that trend. The desire to move to a chiplet-based design has been driven by the increasing cost of manufacturing devices on leading-edge process nodes. Due to the cost associated with leading-edge nodes, it became advantageous to, once again, de-integrate and break down a large die into smaller 'chiplets' in order to improve [yield](#) and [binning](#).

#### Motivation [edit]

As the industry moves to smaller [process nodes](#), costs for yielding large dies continues to increase. Compared to 250 mm<sup>2</sup> die on the 45 nm process, the 16 nm process more than doubles the cost/mm<sup>2</sup> and the 7 nm process nearly double that to 4x the cost per yielded mm<sup>2</sup>. Moving to the 5 nm and even 3 nm nodes, the cost is expected to continue to increase. Fabricating large monolithic dies will become increasingly less economical. One solution to easing the economics of manufacturing chips with a large amount of [transistors](#), the industry has started shifting to chiplet-based design whereby a single chip is broken down into multiple smaller chiplets.

#### Example [edit]

Consider a [DO](#) of 0.1 defects per cm<sup>2</sup>. Now, consider a medium-sized die 18 mm x 20 mm (360 mm<sup>2</sup>). On a standard 300-millimeter [wafer size](#), up to 150 [dies](#) can be fabricated.



At IEDM 2017, AMD CEO Dr. Lisa Su reported cost <sup>53</sup> per yielded mm<sup>2</sup> for a 250 mm<sup>2</sup> die.

## ■ Interfaces

Standard	Source	Bandwidth Density/edge	Throughput /lane	Delay	PHY energy/bit
Advanced Interface Bus (AIB)	Intel <sup>7</sup>	504 Gbps/mm	Up to 2 Gbps	<5ns	0.85 pJ/bit
Multi-Die IO (MDIO)	Intel	1600 Gbps/mm	Up to 5.4 Gbps		0.5 pJ
High Bandwidth Memory (HBM3)	JEDEC <sup>8</sup>		4.8 Gbps		0.37 pJ
XSR/USR	Rambus/OIF		112 Gbps		
Lipincon	TSMC <sup>9</sup>	536 Gbps/mm	2.8 Gbps	<14 ns	0.486 pJ
Bunch of Wires (BOW)	OCP/ODSA <sup>10</sup>	1280 Gbps/mm	Up to 16 Gbps	<5ns	0.7 pJ
Bandwidth Engine	Mosys <sup>11</sup>		Up to 10.3 Gbps	<2.4 ns	
Infinity Fabric	AMD <sup>12</sup>		10.6 Gbps		



## Good Resources

- <https://www.electronicdesign.com/technologies/embedded-revolution/article/21235774/electronic-design-building-a-chiplet-ecosystem>
- <https://semiengineering.com/waiting-for-chiplet-standards/>
- <https://www.ttieuropa.com/content/ttieuropa/en/resources/marketeye/categories/new-technology/me-slovick-20211124.html>
- <https://www.mdpi.com/2079-9292/9/4/670/htm>
- <https://www.intrinsix.com/blog/chiplet-interfaces>
- <https://www.synopsys.com/glossary/what-is-die-to-die-interface.html>
- <https://openfive.com/protocol-and-interface-agnostic-universal-d2d-controller-for-hpc-and-chiplets/>
- <https://openfive.com/interface-agnostic-universal-d2d-controller-for-hpc-and-chiplets-bow-part-2/>
- ...



## 2.2.5 OCP

### Overview

- <https://www.uciexpress.org/>  
**Universal Chiplet Interconnect Express**
-

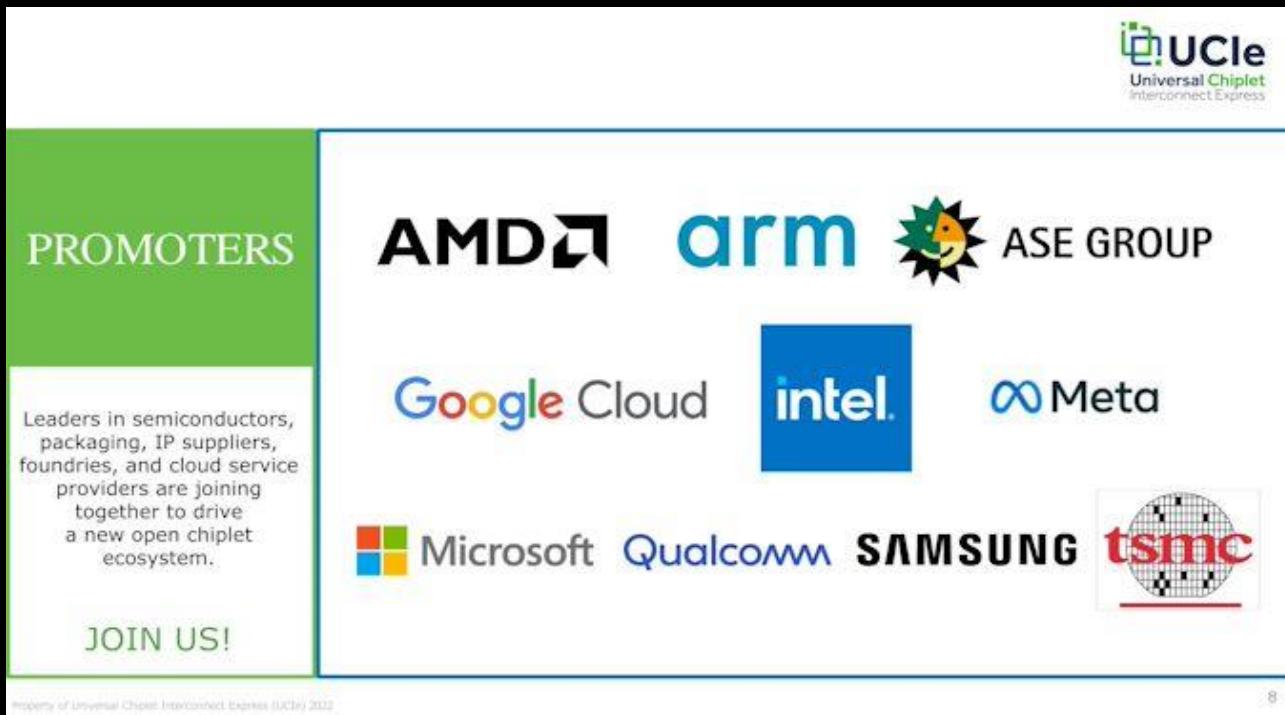
## 2.2.6 UCIe Overview



- <https://www.uciexpress.org/>  
**Universal Chiplet Interconnect Express**

UCIe – Universal Chiplet Interconnect Express – addresses customer requests for a more customizable, package-level integration – combining best-in-class die-to-die interconnect and protocol connections from an interoperable, multi-vendor ecosystem.

This new open industry standard establishes a universal interconnect at the package-level.



The image shows the UCIe logo at the top right, followed by a grid of logos for various industry promoters. The grid is divided into two main sections: 'PROMOTERS' on the left and a list of companies on the right. The 'PROMOTERS' section contains a box of text and a 'JOIN US!' button. The companies listed are AMD, arm, ASE GROUP, Google Cloud, intel, Meta, Microsoft, Qualcomm, SAMSUNG, and tsmc.

**PROMOTERS**

Leaders in semiconductors, packaging, IP suppliers, foundries, and cloud service providers are joining together to drive a new open chiplet ecosystem.

JOIN US!

Properties of Universal Chiplet Interconnect Express. ©2021 2022

8

UCIe  
Universal Chiplet  
Interconnect Express

AMD arm ASE GROUP

Google Cloud intel Meta

Microsoft Qualcomm SAMSUNG tsmc

## Open Chiplet: Platform on a Package

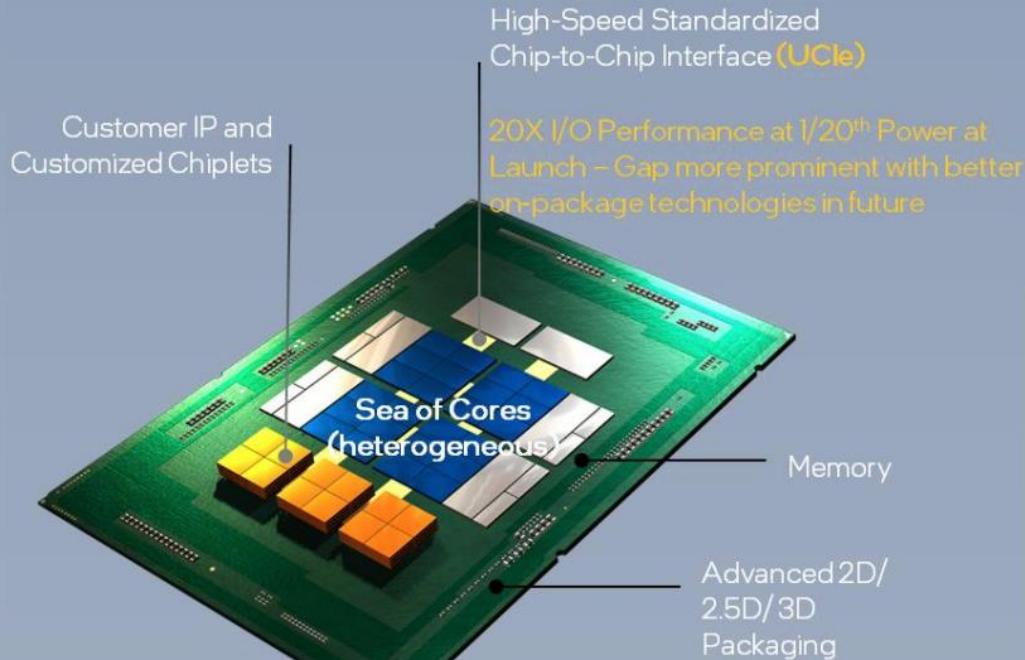


Figure 1: UClie to enable an Open Chiplet Ecosystem delivering Platform on a Package

Source: [https://www.uciexpress.org/\\_files/ugd/0c1418\\_c5970a68ab214ffc97fab16d11581449.pdf](https://www.uciexpress.org/_files/ugd/0c1418_c5970a68ab214ffc97fab16d11581449.pdf)

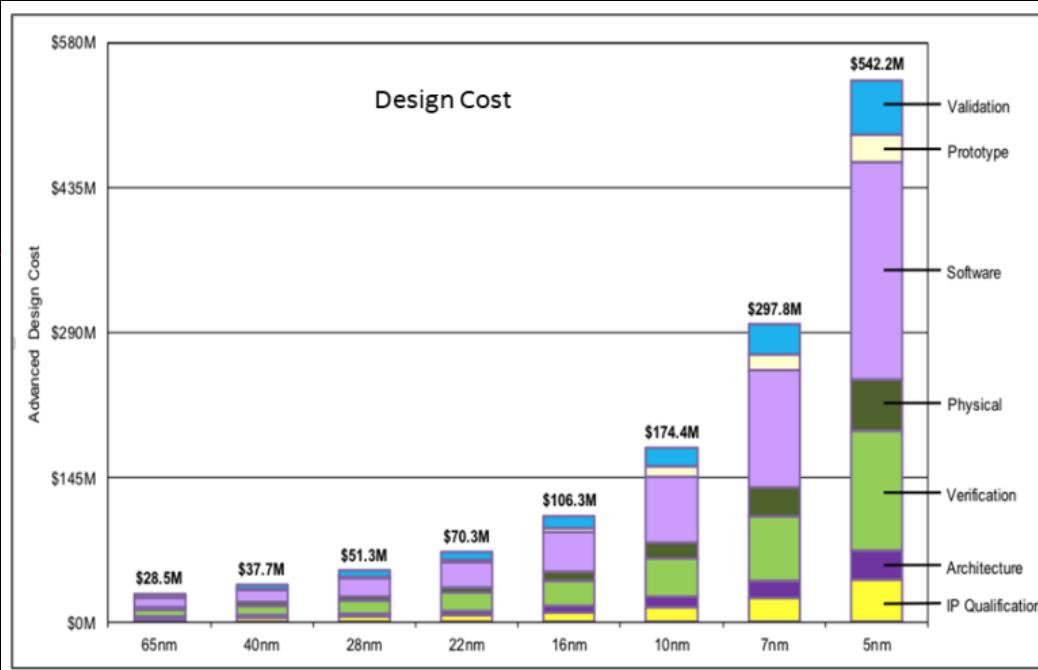


Figure 2: Design cost across different process nodes (Source: IBS, as cited in IEEE Heterogeneous Integration Roadmap)

Source: [https://www.uciexpress.org/\\_files/ugd/0c1418\\_c5970a68ab214ffc97fab16d11581449.pdf](https://www.uciexpress.org/_files/ugd/0c1418_c5970a68ab214ffc97fab16d11581449.pdf)



## UCIe 1.0

### ■ Characteristics and Key Metrics

Characteristics / KPIs	Standard Package	Advanced Package	Comments
<b>Characteristics</b>			
Data Rate (GT/s)	4, 8, 12, 16, 24, 32		Lower speeds must be supported -interop (e.g., 4, 8, 12 for 12G device)
Width (each cluster)	16	64	Width degradation in Standard, spare lanes in Advanced
Bump Pitch (um)	100 – 130	25 - 55	Interoperate across bump pitches in each package type across nodes
Channel Reach (mm)	<= 25	<=2	
<b>Target for Key Metrics</b>			
B/W Shoreline (GB/s/mm)	28 – 224	165 – 1317	Conservatively estimated: AP: 45u for AP; Standard: 110u;
B/W Density (GB/s/mm <sup>2</sup> )	22-125	188-1350	Proportionate to data rate (4G – 32G)
Power Efficiency target (pJ/b)	0.5	0.25	
Low-power entry/exit	0.5ns <=16G, 0.5-1ns >=24G		Power savings estimated at >= 85%
Latency (Tx + Rx)	< 2ns		Includes D2D Adapter and PHY (FDI to bump and back)
Reliability (FIT)	0 < FIT (Failure In Time) << 1		FIT: #failures in a billion hours (expecting ~1E-10) w/ CXi Flit Mode

Source: [https://www.uciexpress.org/\\_files/ugd/0c1418\\_c5970a68ab214ffc97fab16d11581449.pdf](https://www.uciexpress.org/_files/ugd/0c1418_c5970a68ab214ffc97fab16d11581449.pdf)



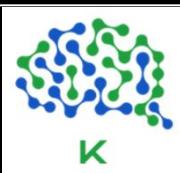
## Good Resources

- <https://www.anandtech.com/show/17288/universal-chiplet-interconnect-express-ucie-announced-setting-standards-for-the-chiplet-ecosystem>
- <https://github.com/riscvarchive/riscv-software-list>
- ...

## 2.3 IoT Edge, 5G, and Self-driving

### 2.3.1 IoT Edge

- ...
- 





## 2.3.1.1 TinyML

- <https://www.tinyml.org/>

Tiny machine learning is broadly defined as a fast growing field of machine learning technologies and applications including hardware, algorithms and software capable of performing on-device sensor data analytics at extremely low power, typically in the mW range and below, and hence enabling a variety of always-on use-cases and targeting battery operated devices.

- <https://semiengineering.com/why-tinyml-is-such-a-big-deal/>

### TinyML on ARM

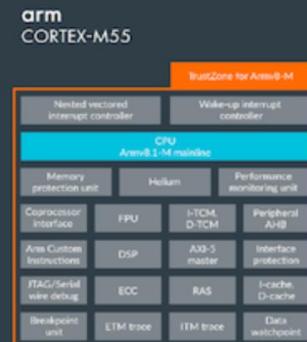
- <https://www.arm.com/blogs/blueprint/tinyml>
- **Cortex-M55**

<https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m55>

**First Helium and Custom Instruction capable CPU core**

#### Cortex-M55: The Most AI-capable Cortex-M Processor

- ✓ First CPU based on Arm Helium technology
  - Energy-efficient and configurable with vector processing capabilities
  - Delivers up to 5x DSP performance and up to 15x ML performance\*
  - Versatile capability for both classical ML and NN inference
- ✓ Advanced memory interfaces for fast access to ML data and weights
- ✓ Arm TrustZone security, accelerating the route to PSA Certified





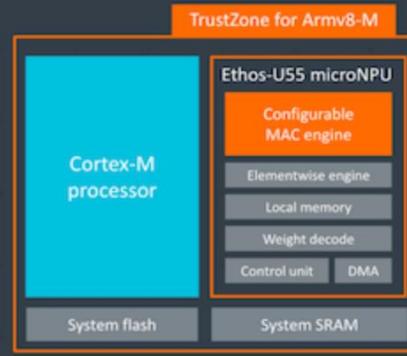
## Ethos: MicroNPU for ARM

<https://www.arm.com/product-filter?families=ethos%20npus&showall=true>

### Ethos-U55:

#### Ethos-U55: The First microNPU for Cortex-M

- ✓ Highest efficiency and small memory footprint
- ✓ 32, 64, 128, or 256 unit multiply-accumulate (MAC) engine
- ✓ Weight decoder and DMA for on-the-fly weight decompression
- ✓ Tooling available for offline optimization
- ✓ Works with a range of Cortex-M processors:
  - Cortex-M55      • Cortex-M7
  - Cortex-M33      • Cortex-M4



Key Features	Performance (At 1GHz)	64 to 512 GOP/s
	MACs (8x8)	32, 64, 128, 256
	Utilization on popular networks	Up to 85%
	Data Types	Int-8 and Int-16
	Network Support	CNN and RNN/LSTM
	Winograd Support	No
	Sparsity	Yes
Memory System	Internal SRAM	18 to 50 KB
	External On Chip SRAM	KB to Multi-MB
	Compression	Weights only
	Memory Optimizations	Extended compression, layer/operator fusion
Development Platform	Neural Frameworks	TensorFlow Lite Micro
	Operating Systems	RTOS or bare-metal
	Software Components	TensorFlow Lite Micro Runtime, CMSIS-NN, Optimizer, Driver
	Debug and Profile	Layer-by-layer visibility with PMUs
	Evaluation and Early Prototyping	Performance Model, Cycle Accurate Model, or FPGA Evaluations



## 2.3.1.2 Wasm

- ...

### WasmEdge

- ~~<https://wasmedge.org/>~~
- ~~<https://www.secondstate.io/articles/wasmedge-joins-cncf/>~~
- Key Features

WasmEdge is fully compatible with the W3C WebAssembly standard. Out of the box, it is supported by standard language and compiler tool chains, such as LLVM, Rustc and emscripten. WasmEdge is differentiated for its support for extensions, especially extensions for edge computing.

For starters, WasmEdge supports W3C optional WebAssembly features and proposals, such as WebAssembly System Interface(WASI) spec, Reference type, Bulk memory operations, and SIMD. We are also exploring the [wasi-socket](#) proposal to support network access in WebAssembly programs.

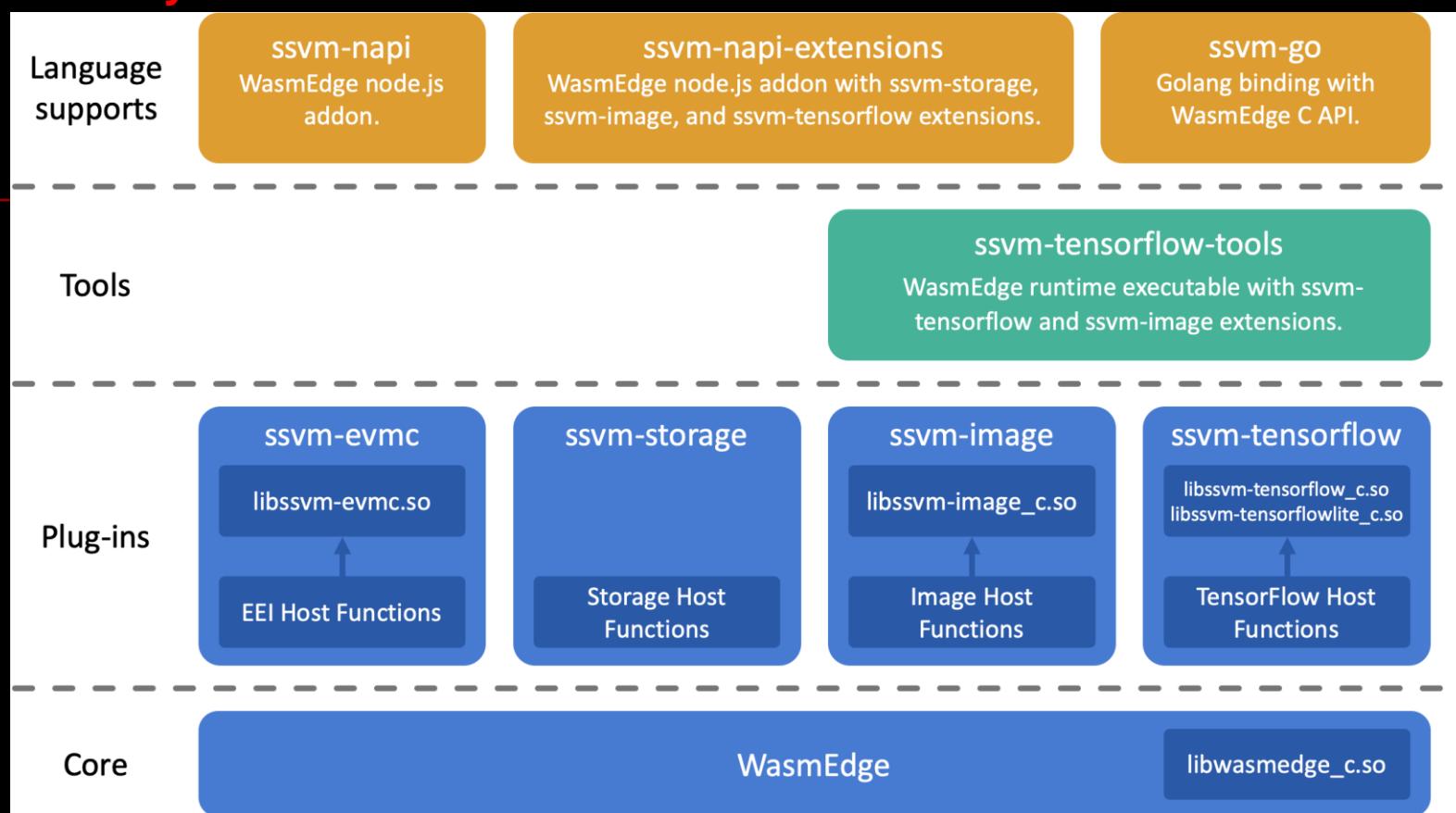
Furthermore, WasmEdge supports non-standard extensions designed for specific application scenarios.

- **Tensorflow.** Developers can write Tensorflow inference functions using a [simple Rust API](#), and then run the function securely and at native speed inside WasmEdge. We are working on supporting other AI frameworks.
- **Storage.** The WasmEdge [storage interface](#) allows WebAssembly programs to read and write a key-value store.
- **Command interface.** WasmEdge enables webassembly functions to execute native commands in the host operating system. It supports passing arguments, environment variables, STDIN / STDOUT pipes, and security policies for host access.
- **Ethereum.** The WasmEdge Ewasm extension supports Ethereum smart contracts compiled to WebAssembly. It is a leading implementation for Ethereum flavored WebAssembly (Ewasm).
- **Substrate.** The [Pallet](#) allows WasmEdge to act as an Ethereum smart contract execution engine on any Substrate based blockchains.

Last but not least, WasmEdge is a “cloud-native” WebAssembly VM. It supports the [OCI \(Open Container Initiative\)](#) specification, which will allow WasmEdge instances to be managed by cloud-native orchestration tools such as Kubernetes.



## Formerly called SSVM





## **aWsm/SLEdge**

### ■ **<https://github.com/gwsystems/aWsm>**

aWsm is a compiler and runtime for compiling WebAssembly (Wasm) code into LLVM bytecode, then into sandboxed binaries you can run on various platforms. It focuses on generating very fast code (best of breed for WebAssembly), having a simple and extensible code-base, and on portability.

### ■ **Features**

- *Performance.* aWsm is an ahead-of-time compiler that leverages the LLVM compiler to optimize code, and target different architectural backends. We have evaluated aWsm on x86-64, aarch64 (Raspberry Pi), and thumb (ARM Cortex-M4 and M7), and performance on the microprocessors is within 10% of native, and within 40% on the microcontrollers on Polybench benchmarks.
- *Simplicity.* The entire code base for the compiler and runtime is relatively small. The compiler is <3.5K lines of Rust, and the runtime (for all platforms) is <5K lines of C. It is nearly trivial to implement different means of sandboxing memory accesses. We've implemented seven different mechanisms for this!
- *Portability.* Both the compiler and runtime are mostly platform-independent code, and porting to a new platform only really requires additional work if you need to tweak stack sizes (microcontrollers), or use architectural features (e.g., MPX, segmentation, etc...). aWsm only links what is needed, so it's possible to avoid microcontroller-expensive operations such as f64, f32, and even dynamic memory.
- *Composability.* The final output of aWsm is simple \*.o elf objects that can be linked into larger systems. This enables the trivial composition of sandboxes together, and sandboxes into larger programs.

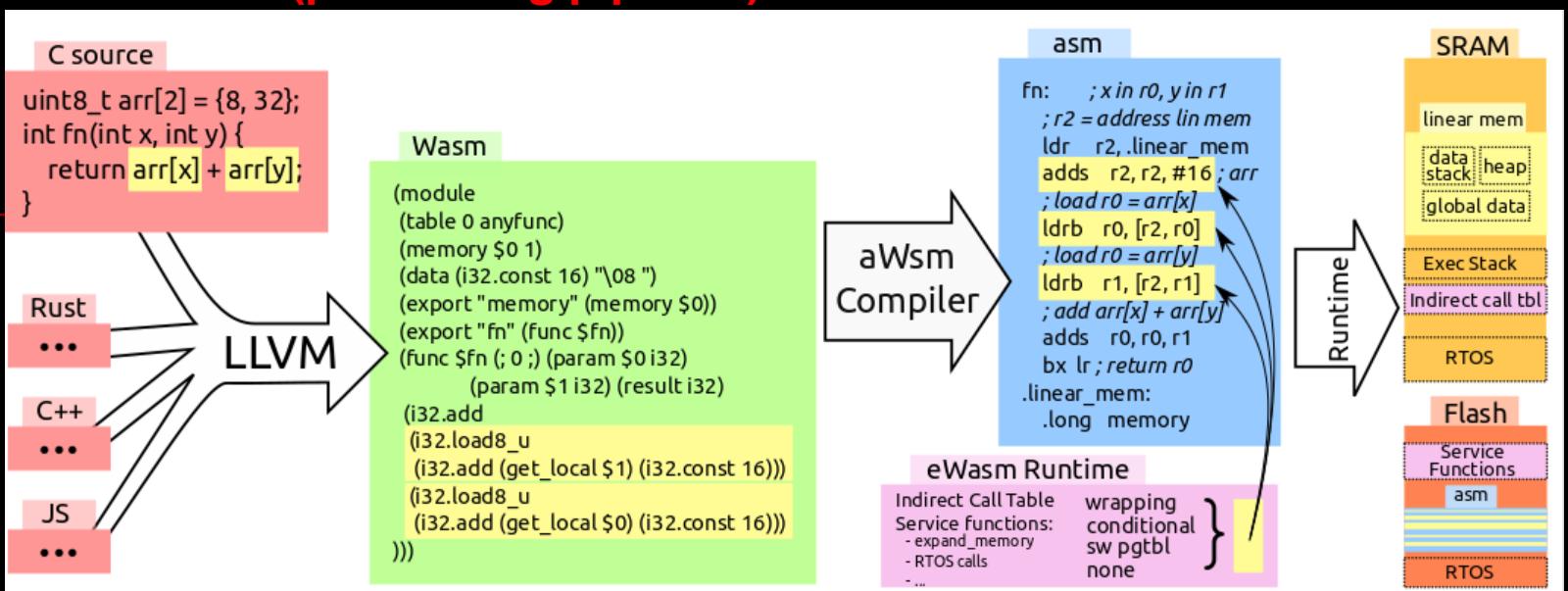
### ■ **Comparison**

**Following table is from the best of understanding of each system in July 2020:**

Runtime	Method	x86_64	x86	aarch64	thumb	URL
aWsm	AoT	✓	✓	✓	✓	You are here
Wasmtime	AoT	✓	✓	✓		<a href="https://github.com/bytecodealliance/wasmtime">https://github.com/bytecodealliance/wasmtime</a>
Wasmer	AoT	✓	✓	✓		<a href="https://github.com/wasmerio/wasmer">https://github.com/wasmerio/wasmer</a>
WAMR	Pseudo-AoT/Int	✓	✓	✓	✓	<a href="https://github.com/bytecodealliance/wasm-micro-runtime">https://github.com/bytecodealliance/wasm-micro-runtime</a>
Wasm3	Int	✓	✓	✓	✓	<a href="https://github.com/wasm3/wasm3">https://github.com/wasm3/wasm3</a>
Wasmi	Int	✓	✓	✓	✓	<a href="https://github.com/paritytech/wasmi">https://github.com/paritytech/wasmi</a>



## How it works(processing pipeline)



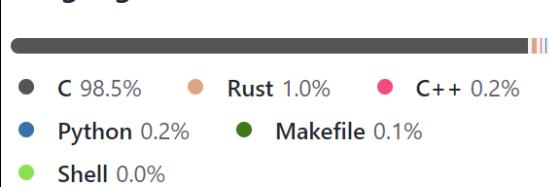
- Programming languages are compiled into Wasm, for example, using LLVM.
- Wasm has a binary representation and (as depicted) a s-expr representation.
- The aWsm compiler inputs binary Wasm, generates LLVM IR corresponding to the Wasm.
- This IR is compiled with the runtime to generate the final object that exports `wasm_main` to execute in the broader application.

In the Figure, we target Arm Cortex-M, and the yellow boxes emphasize how linear memory bounds checks transition throughout the process.

Source: <https://github.com/gwsystems/aWsm/blob/master/doc/design.md>

## Src

### Languages



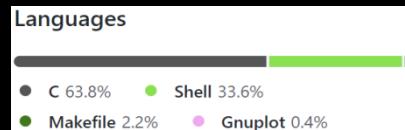


- <https://github.com/gwsystems/sledge-serverless-framework>  
**A lightweight Serverless solution suitable for Edge Computing.  
It builds on Wasm sandboxing provided by the aWsm compiler.**
- **Features**

- *Light-weight function isolation.* Sledge executes functions in light-weight Wasm sandboxes, removing the high overheads of traditional runtimes (e.g., VMs and containers), while still providing strong memory isolation.
- *Optimized function startup for high densities.* Given the repeated execution of the same functions across many tenants, Sledge optimizes function startup by decoupling the processing (linking and loading) of function binaries from function instantiation.
- *Decoupling work-distribution from temporal isolation.* Sledge leverages the short-lived execution properties of serverless to specialize system scheduling by decoupling both the work-distribution and load balancing across cores for scalability, from the scheduling logic to maintain fairness and temporal isolation.
- *Serverless execution performance evaluation.* We perform an extensive evaluation of Sledge serverless runtime using various Edge workloads to show that Sledge provides up to 4 times better latencies and throughputs compared to Nuclio [57], one of the fastest open-source serverless frameworks. We also evaluate our Sledge Wasm compiler and its runtime on x86\_64 and AArch64 architectures to show an average performance overhead within 13% of the native code execution for PolyBench/C [62] benchmarks and compare its efficiency with various existing LLVM- and Cranelift-based [19] Wasm compilers and runtimes.

Source: <https://conix.io/wp-content/uploads/pubs/3878/CONIX-Sledge-Poster.pdf>

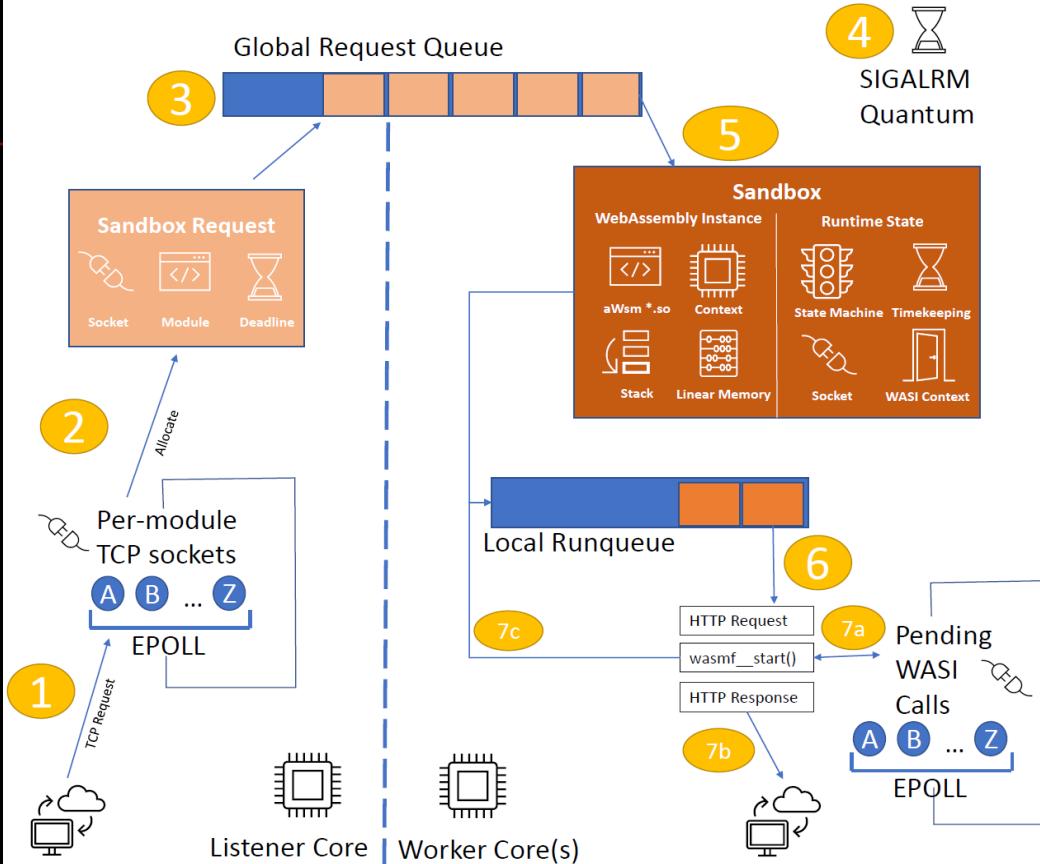
## Src





## Workflow of SLEdge

### SLEdge: Serverless for the Edge



Source: <https://conix.io/wp-content/uploads/pubs/3878/CONIX-Sledge-Poster.pdf>

1. Receive TCP request
2. Allocate sandbox request
3. Add sandbox request to the **global request queue**.
4. On **SIGALRM**, handle **epoll** events and execute scheduler based on configured policy (EDF, FIFO, multi-tenancy, etc.)
5. If the scheduler policy considers the head of the **global request queue** highest priority, allocate and initialize the sandbox and add it to the **local runqueue**.
6. Context switch to the sandbox at the head of the **local runqueue** and execute.
7. Context switch away from the sandbox due to one of the following conditions:
  - a) Making a **WASI call** that blocks (I/O)
  - b) Running to completion and sending a response to the client.
  - c) Exhausting the quantum as indicated by a **SIGALRM**.

■ For details, please refer to my previous talk "**AOT compilation based Wasm compiler and runtime for Serverless Edge computing**" at **OpenInfra Days China 2021(Beijing)** and upcoming follow-ups.



## Fastly & Bytecode Alliance

- <https://www.fastly.com/products/edge-compute/serverless>  
**Faster, simpler, and more secure serverless code.**
- **Benefits**



### **Execute code faster**

At 35.4 microseconds, Compute@Edge provides a 100x faster code execution startup time than other serverless solutions. Run your code on hundreds of servers located around the world simultaneously. There are no cold starts or roundtrip delays — just fast, always-on computing.



### **Build exceptional user experiences**

Better end user experiences are at the forefront of digital transformation. Write, deploy, and test your code on the Fastly edge using a powerful local development and debugging environment. From there, we manage everything required to scale it instantly and globally, as close to your end users as possible.



### **Enhance security and reliability**

Operating within microseconds, our isolation technology helps protect you from side-channel attacks and diminishes resource contention while offering consistent performance you can count on. By creating and destroying a sandbox for each request that comes through the platform, we limit the blast radius of buggy code or configuration mistakes from other users and can reduce the attack surface area.



### **Leverage familiar tools and languages**

Developer experience and intuitive tool sets matter. With Compute@Edge, you can program in familiar languages, like Rust and JavaScript, port your code across cloud providers, and get an up-to-the-second view of your services. Plus, Compute@Edge seamlessly integrates into your existing tech stack.

- <https://www.fastly.com/edge-cloud-platform/>

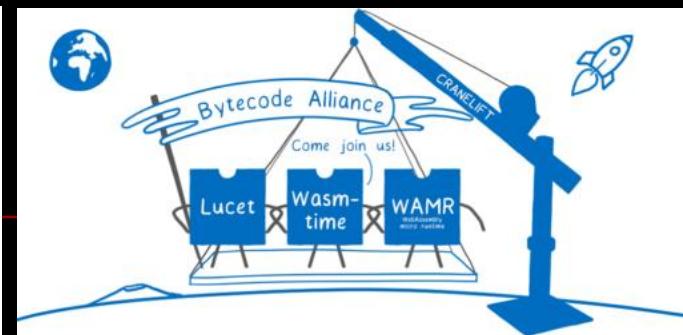


## ■ <https://bytecodealliance.org/>

The Bytecode Alliance is a nonprofit organization dedicated to creating secure new software foundations, building on standards such as [WebAssembly](#) and [WebAssembly System Interface \(WASI\)](#).

The Bytecode Alliance is committed to establishing a capable, secure platform that allows application developers and service providers to confidently run untrusted code, on any infrastructure, for any operating system or device, leveraging decades of experience doing so inside web browsers.

We have a [vision](#) for a secure-by-default WebAssembly ecosystem for all platforms.



## ■ Members

**arm**



**embark**

**fastly**

**Google**

**InfiniOn**

**igalia**

**intel**

**Microsoft**

**moz://a**

**Profian**

**SIEMENS**

**shopify**

**StackBlitz**

**SUBORBITAL**

## ■ <https://github.com/bytecodealliance>

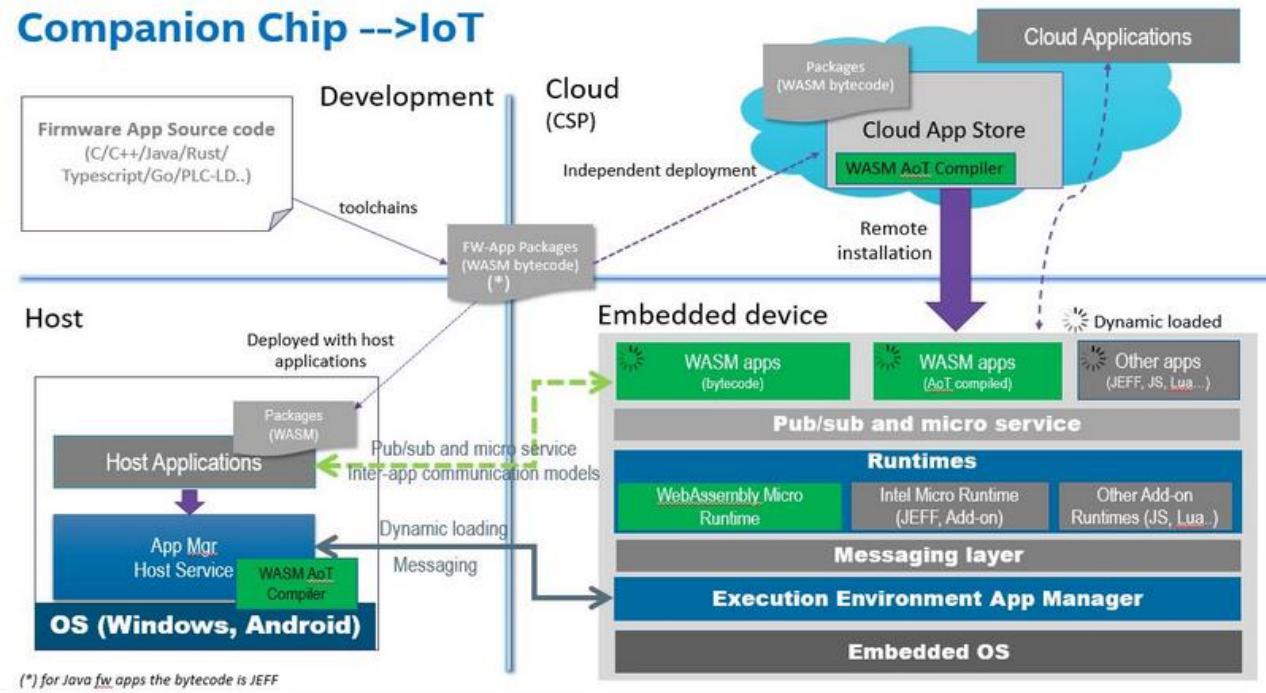
# WAMR(WebAssembly Micro Runtime)

By using the iwasm VM core, we are flexible to build different application frameworks for the specific domains.

The WAMR has offered a comprehensive application framework for device and IoT usages. The framework solves many common requirements for building a real project:

- Modular design for more language runtimes support
- Inter application communication
- Remote application management
- WASM APP programming model and API extension mechanism

## Companion Chip -->IoT





## 2.3.2 5G

- <https://en.wikipedia.org/wiki/5G>
- <https://www.3gpp.org/release-15>
- <https://www.3gpp.org/release-16>
- ...



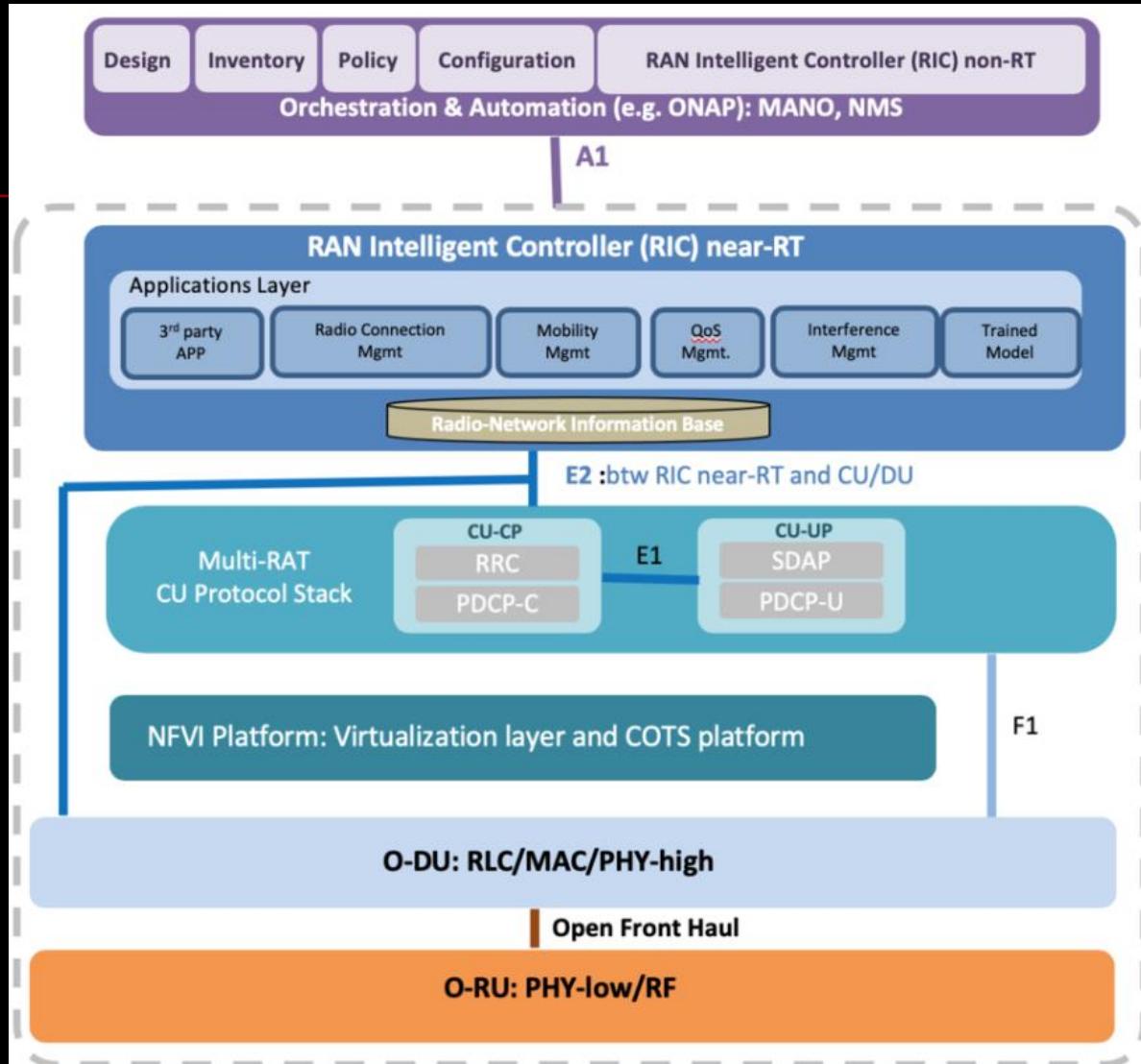
### 2.3.2.1 Open RAN

- <https://www.o-ran.org/>  
**Operator Defined Open and Intelligent Radio Access Networks.**
- <https://open-ran.org/#next>

- 
- 1 **Open RAN** is intelligent Radio Access Network(RAN) integrated on general purpose platforms with open interface between software defined functions.
  - 2 **Open RAN** ecosystem enables enormous flexibility and interoperability with a complete openness to multi-vendor deployments.
  - 3 **Open RAN** architecture is designed for building virtualized RAN with AI powered control, which is the key to tame the 5G complexity.

- <https://www.o-ran.org/specifications>
- <https://www.o-ran.org/software>
- <https://www.o-ran.org/resources>
- <https://wiki.o-ran-sc.org/>
- ...

## Architecture & Design



Source: <https://www.o-ran.org/>



## ■ Cloud Native Open RAN

<https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-ric-dep/en/latest/installation-guides.html>

<https://thenewstack.io/european-telecom-giants-prioritize-a-kubernetes-based-open-cloud-platform>

---

<https://www.calsoftinc.com/resources/ebriefs/open-ran-cloud-native-software-aspects-for-end-to-end-orchestration-automation>

...



## SW & Src

### ■ <https://o-ran-sc.org/>

The O-RAN Software Community (SC) is a collaboration between the O-RAN Alliance and Linux Foundation with the mission to support the creation of software for the Radio Access Network (RAN). The RAN is the next challenge for the open source community. The O-RAN SC plans to leverage other LF network projects, while addressing the challenges in performance, scale, and 3GPP alignment.

### <https://wiki.o-ran-sc.org/pages/viewpage.action?pageId=41452927>

- Amber Release (Nov 2019)
- Bronze Release (Jun 2020)
- Cherry Release (Dec 2020)
- D release (Jul 2021)
- E Release (Dec 2021)
  - E release timeline
- F Release
  - Vote Release Name

### ■ <https://github.com/o-ran-sc>

 Repositories 117

Top languages

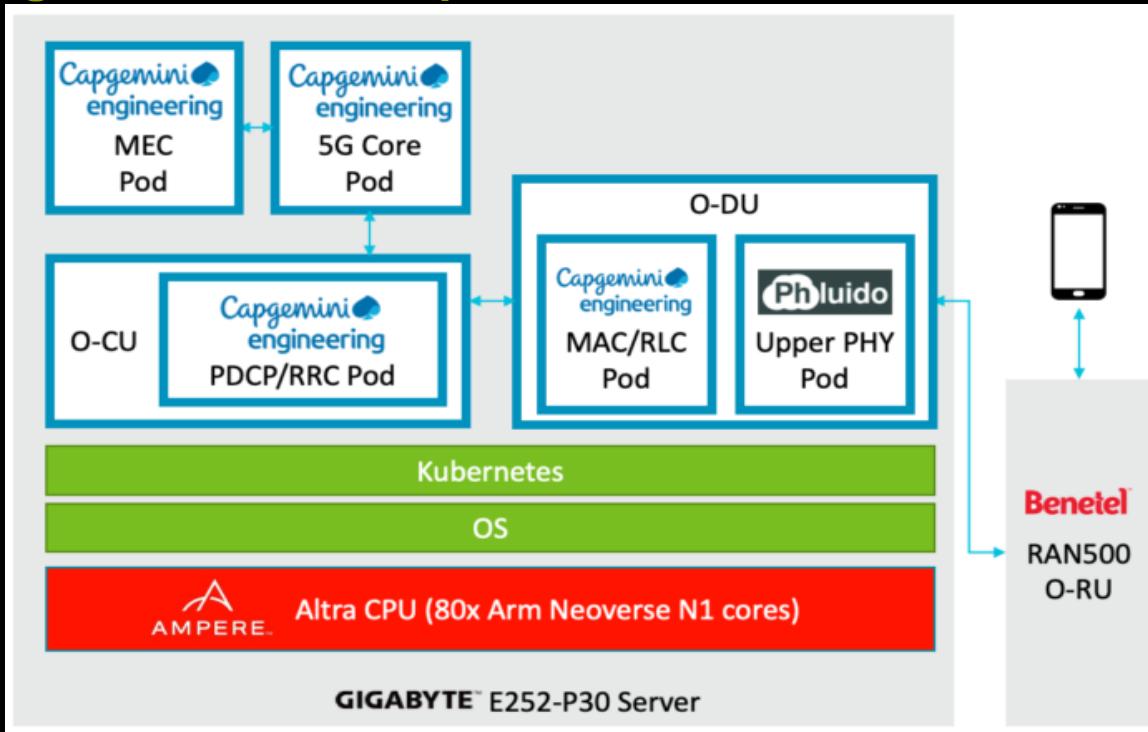
 Python  C  Go  C++  Shell

■ ...



## ARM for Open-RAN

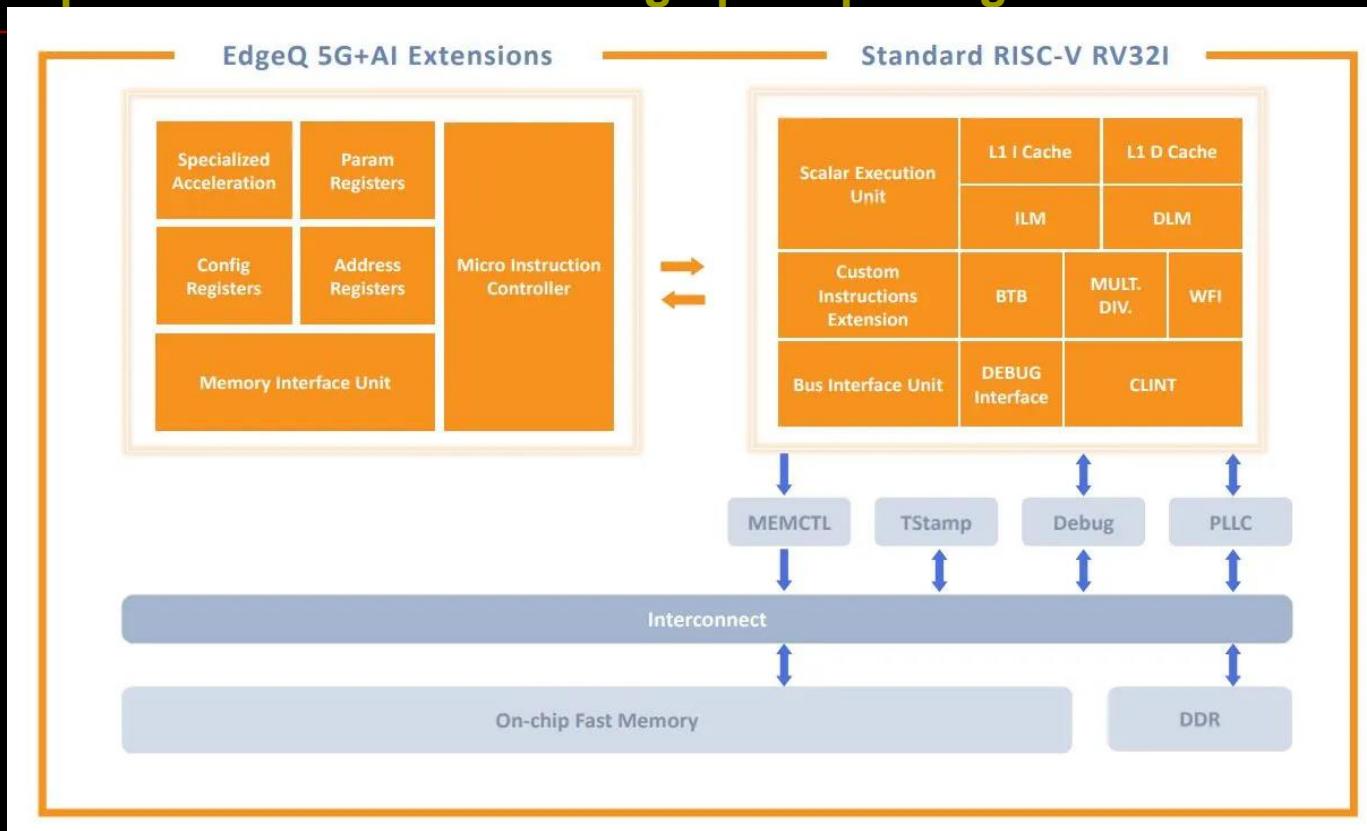
- <https://www.arm.com/campaigns/5g-solutions-lab>
- <https://developer.arm.com/solutions/infrastructure/developer-resources/5g/ran>
- <https://www.arm.com/company/news/2021/10/enabling-end-to-end-5g-networks-on-arm>
- <https://benetel.com/guest-post-deploying-power-performance-efficient-5g-solutions-with-openran-innovation/>





## RISC-V for Open-RAN

- <https://www.eenewseurope.com/en/picocom-samples-its-risc-v-openran-chip/>
- <https://www.eetimes.com/edgeq-samples-5g-basestation-on-a-chip/>



- <https://www.eenewseurope.com/en/fraunhofer-extends-risc-v-embedded-processor-for-edge-ai/>
- ...



## Good Resources

- <https://www.eenewseurope.com/en/vodafone-opens-e225m-european-rd-centre-in-spain/>
- [https://www.rcrwireless.com/20200708/open\\_ran/open-ran-101-ru-du-cu-reader-forum](https://www.rcrwireless.com/20200708/open_ran/open-ran-101-ru-du-cu-reader-forum)
- [https://www.3gpp.org/news-events/2150-open\\_ran](https://www.3gpp.org/news-events/2150-open_ran)
- ...

## 2.3.2.2 5G Chips

- ...
  - ...



### Picocom PC802

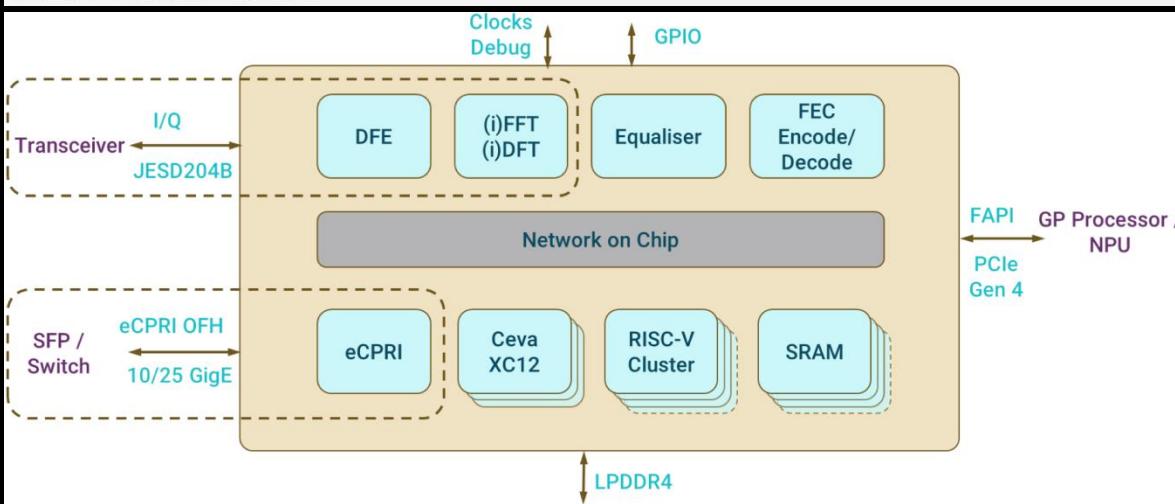
- <https://picocom.com/products/socs/pc802/>

#### Device description

The PC802 device's function in a small cell use case is, at the highest level, seen as transforming FAPI messages from the L2/3 into IQ samples for the radio, and vice versa. This includes driving the external physical and software interfaces, PHY processing and Digital Front End (DFE) functions.

The PC802 is also optimised to provide upper PHY processing in an O-RAN Alliance O-DU, by transforming the FAPI messages from the L2/3 into frequency domain IQ samples using eCPRI and Open fronthaul messages, which can be transmitted over the Ethernet to O-RU remote radio units.

The high-level PC802 architecture and functional block diagrams for the different use cases are illustrated below.

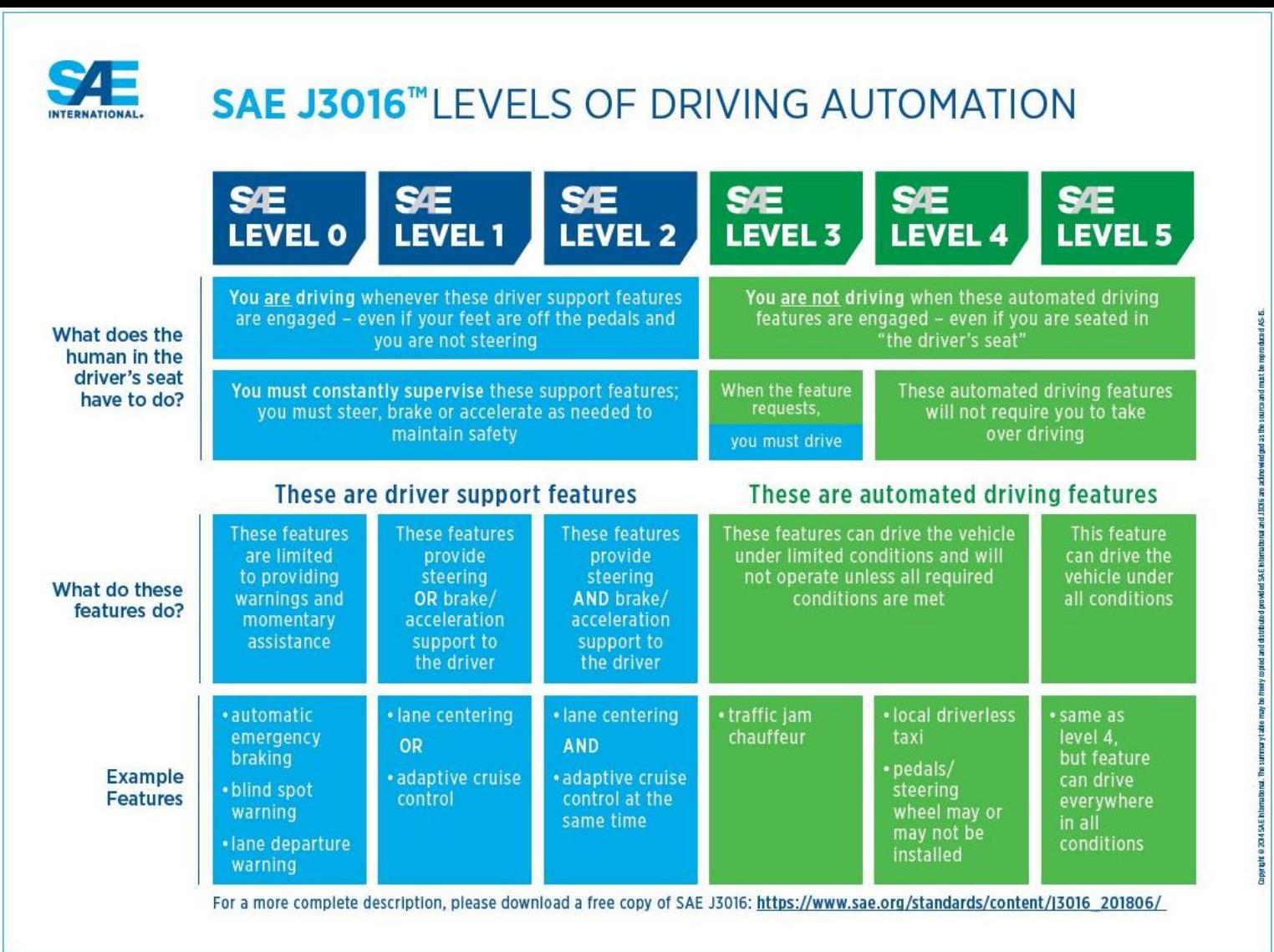




### 2.3.3 Self-driving

- <https://www.autonews.com/>
- [https://en.wikipedia.org/wiki/Self-driving\\_car](https://en.wikipedia.org/wiki/Self-driving_car)
- <https://wiki.automotivelinux.org/>
- ...

## 2.3.3.1 Levels of Driving Automation



Source: <https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic>



### 2.3.3.2 Software-defined Vehicle

- <https://www.arm.com/en/solutions/automotive/software-defined-vehicles>
- <https://www2.deloitte.com/cn/en/pages/consumer-business/articles/software-defined-cars-industrial-revolution-on-the-arrow.html>
- <https://www.bosch-mobility-solutions.com/en/mobility-topics/software-defined-vehicle/>
- <https://www.embedded.com/embedded-news-software-defined-vehicles-open-lidar-biowearables/>
- <https://www.cubictelecom.com/blog/ces-2022-the-future-is-software-defined-and-connected/>
- <https://www.just-auto.com/interview/ces-blackberry-on-the-software-defined-car/>
- <https://techcrunch.com/2022/02/03/aptivs-latest-investment-shows-that-software-defined-vehicles-are-here-to-stay/>
- <https://www.bigmarker.com/Embedded-Computing-Design/The-Next-generation-Software-Defined-Vehicle-Communication-Framework-Challenges>
- <https://www.smart2zero.com/en/the-new-generation-of-software-defined-vehicles/>
- <https://aws.amazon.com/cn/automotive/software-defined-vehicle/>
- <https://semiengineering.com/automotive-outlook-2022/>



## ■ <https://www.arm.com/solutions/automotive/software-defined-vehicles>

Software-defined vehicles (SDV) offer significant safety and convenience features, enabling new in-vehicle experiences and functions through software and delivering updates and services over-the-air (OTA). For automakers, OEMs, and tier one suppliers, SDVs can deliver major revenue and customer relationship benefits and help meet stringent functional safety standards.

The evolution from hardware-defined to software-defined compute requires new development paradigms to address advanced compute architectures and increasing software complexity to ensure seamless development and deployment from the cloud to the car. Arm and our partners are committed to making this a reality.

### Solutions to Bring the Software-Defined Vision into Focus

Together with leaders from across the automotive supply chain, Arm is working to address the challenges with software portability, cloud-native mixed-criticality, and seamless deployment to bring us closer to a software-defined future.



#### Scalable Open Architecture for Embedded Edge

SOAFEE offers a cloud-native architecture enhanced for mixed-criticality automotive applications. It includes an open-source reference implementation to enable commercial and non-commercial offerings.

Building on [Project Cassini](#) and SystemReady standards, which define standard boot and security requirements for Arm architecture, SOAFEE adds the functional safety and real-time capabilities required for autonomous workloads.



#### SOAFEE-Enabled Hardware Platforms

To enable automotive software porting, prototyping, and development today, Arm has partnered with ADLINK and Ampere to deliver high-performance, SOAFEE-enabled hardware platforms:

- ADLINK AVA Developer Platform supports lab-based development that can run autonomous workloads, including Autoware.
- ADLINK AVA-AP1 Platform is a ruggedized version with I/O to support a full sensor suite for in-vehicle prototyping and testing.

<https://armkeil.blob.core.windows.net/developer/Files/pdf/white-paper/linaro-automotive-white-paper.pdf>

<https://armkeil.blob.core.windows.net/developer/Files/pdf/white-paper/arm-aws-edge-environmental-parity-wp.pdf>

...



## ***MIH EV***

- **<https://developer.mih-ev.org/about/#what-is-sdv-software-defined-vehicle>**

“Software-defined vehicle” is a term that describes a vehicle whose features and functions are primarily enabled through software, a result of the ongoing transformation of the automobile from a product that is mainly hardware-based to a software-centric electronic device on wheels.

Premium vehicles today can already have up to 150 million lines of software code, distributed among as many as 100 electronic control units (ECUs) and a growing array of sensors, cameras, radar and light detection and ranging (lidar) devices. Mass-market vehicles are not far behind. Three powerful trends – electrification, automation and connectivity – are reshaping customer expectations and driving manufacturers to increasingly turn to software to address them.

In the past, vehicle manufacturers differentiated themselves with mechanical features such as horsepower and torque. Today, consumers are increasingly looking for features defined by software, such as driver assistance features, infotainment innovations and intelligent connectivity solutions. As driver-assistance features grow into more automated driving and toward fully autonomous driving, the need for more software also grows. As consumers expect richer content in their infotainment systems, that also increases the amount of digital content the vehicle must manage. And as vehicles become part of the internet of things (IoT), transmitting large amounts of data to and from the cloud, software will be required to process, manage and distribute all of that data.

- ...

## 2.3.3.3 ECU

### ■ [https://en.wikipedia.org/wiki/Engine\\_control\\_unit](https://en.wikipedia.org/wiki/Engine_control_unit)



An **engine control unit (ECU)**, also commonly called an **engine control module (ECM)**, is a type of electronic control unit that controls a series of **actuators** on an **internal combustion engine** to ensure optimal engine performance. It does this by reading values from a multitude of **sensors** within the engine bay, interpreting the data using multidimensional performance maps (called **lookup tables**), and adjusting the engine actuators. Before ECUs, air-fuel mixture, ignition timing, and idle speed were mechanically set and dynamically controlled by **mechanical and pneumatic** means.

If the ECU has control over the **fuel lines**, then it is referred to as an **electronic engine management system (EEMS)**. The **fuel injection** system has the major role of controlling the engine's fuel supply. The whole mechanism of the EEMS is controlled by a stack of sensors and actuators.

...

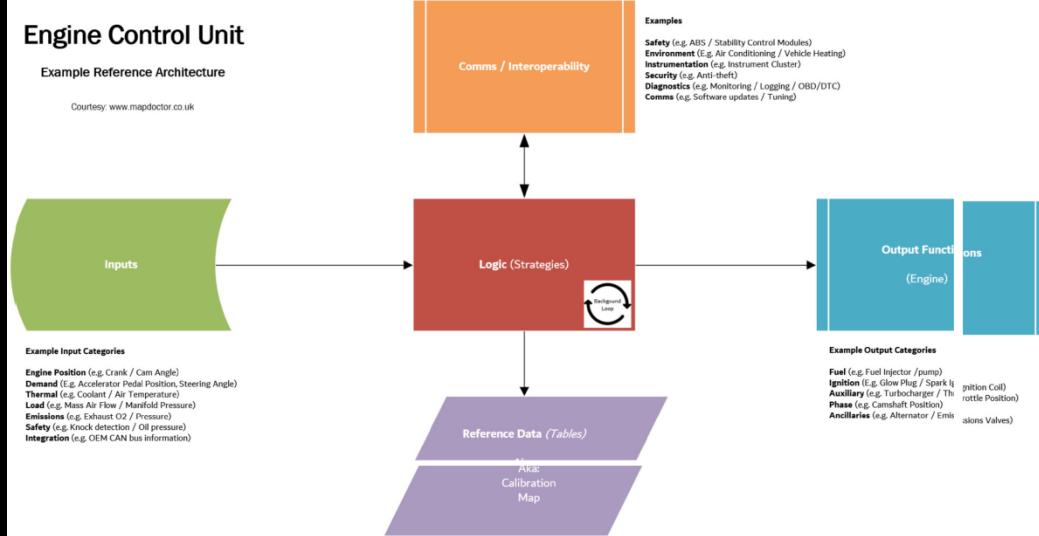
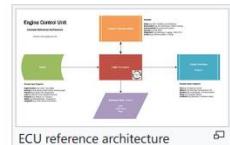
## Modern design

Modern ECUs use a **microprocessor** which can process the inputs from the engine sensors in **real-time**. An electronic control unit contains the hardware and software (**firmware**). The hardware consists of electronic components on a **printed circuit board (PCB)**, ceramic substrate or a thin laminate substrate. The main component on this circuit board is a **microcontroller chip (MCU)**. The software is stored in the microcontroller or other chips on the PCB., typically in **EPROMs** or **flash memory** so the CPU can be re-programmed by uploading updated code or replacing chips. This is also referred to as an (electronic) Engine Management System (EMS).

Sophisticated engine management systems receive inputs from other sources, and control other parts of the engine; for instance, some **variable valve timing** systems are electronically controlled, and **turbocharger** waste gates can also be managed. They also may communicate with **transmission control units** or directly interface electronically controlled **automatic transmissions**, **traction control systems**, and the like. The **Controller Area Network** or CAN bus automotive network is often used to achieve communication between these devices.

Modern ECUs sometimes include features such as **cruise control**, transmission control, anti-skid brake control, and anti-theft control, etc.

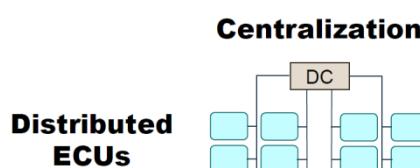
General Motors' (GM) first ECUs had a small application of hybrid digital ECUs as a pilot program in 1979, but by 1980, all active programs were using microprocessor based systems. Due to the large ramp up of volume of ECUs that were produced to meet the Clean Air Act requirements for 1981, only one ECU model could be built for the 1981 model year.<sup>[6]</sup> The high volume ECU that was installed in GM vehicles from the first high volume year, 1981, onward was a modern microprocessor based system. GM moved rapidly to replace carburation with fuel injection as the preferred method of fuel delivery for vehicles it manufactured. This process first saw fruition in 1980 with fuel injected Cadillac engines, followed by the Pontiac 2.5L I4 "Iron Duke" and the Chevrolet 5.7L V8 L83 "Cross-Fire" engine powering the Chevrolet Corvette in 1982. The 1990 Cadillac Brougham powered by the Oldsmobile 5.0L V8 LV2 engine was the last carbureted passenger car manufactured for sale in the North American market (a 1992 Volkswagen Beetle model powered by a carbureted engine was available for purchase in Mexico but not offered for sale in the United States or Canada) and by 1991 GM was the last of the major US and Japanese automakers to abandon carburetion and manufacture all of its passenger cars exclusively with fuel injected engines. In 1988 Delco (GM's electronics division), had produced more than 28,000 ECUs per day, making it the world's largest producer of on-board digital control computers at the time.<sup>[7]</sup>



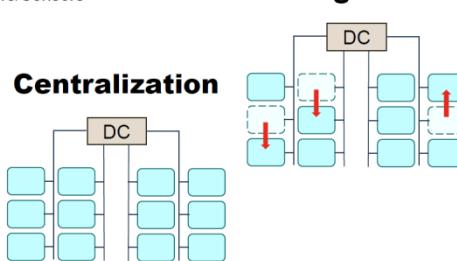
## Challenges in the Mobility Sector

### Driving Innovations in E/E Architectures

- Domain/Vehicle Controller
- Deeply Embedded ECUs
- Obsolete ECUs
- Integration Process
- Intelligent Actuators/Sensors



1970  
Mechanics



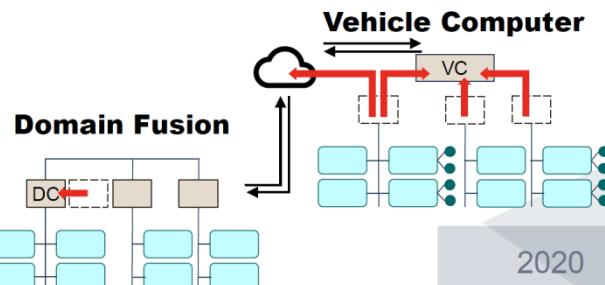
1980  
Electric Support

### Integration

1990  
Infotainment

### Domain Fusion

2000  
Linked Networks



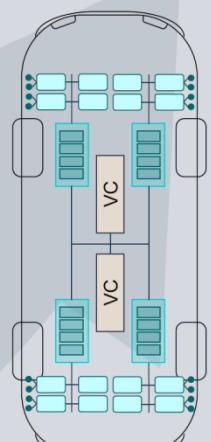
2010

90% of All  
E/E-Driven  
Innovations

2020

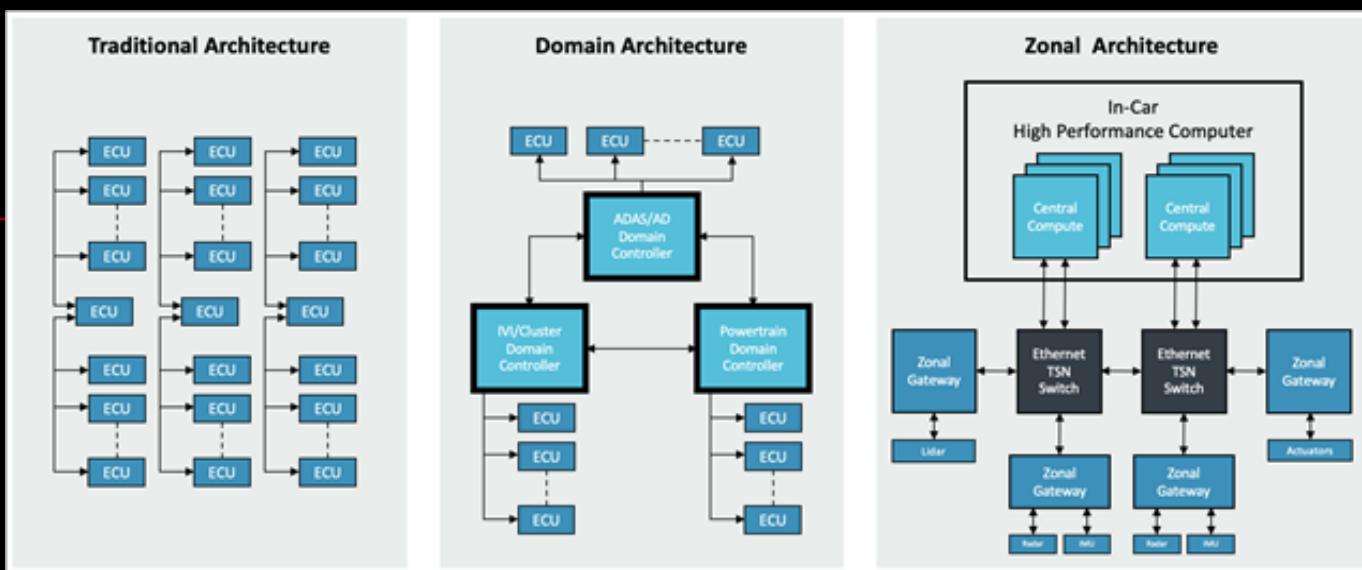
**Major E/E-driven Innovations**  
Vehicle-  
Backend  
Connection

2025  
**Zone  
Architecture**



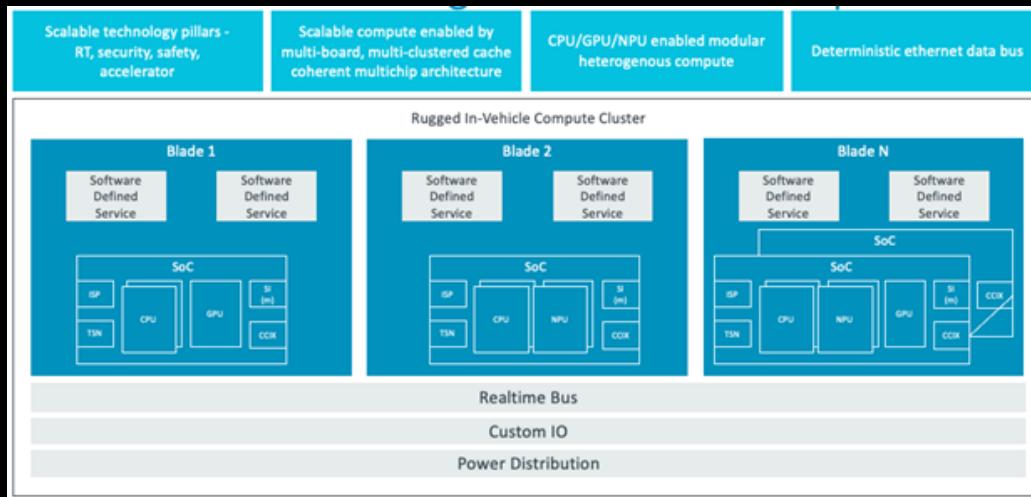
Source: [https://www.autosar.org/fileadmin/user\\_upload/AUTOSAR\\_EXP\\_Introduction\\_Part1.pdf](https://www.autosar.org/fileadmin/user_upload/AUTOSAR_EXP_Introduction_Part1.pdf)

## Automotive E/E Architecture Trend



Source: <https://community.arm.com/arm-community-blogs/b/embedded-blog/posts/cloud-native-approach-to-the-software-defined-car>

### Datacenter-On-Wheel





## 2.3.3.4 Automotive Ethernet TSN

- [https://handwiki.org/wiki/Time-Sensitive\\_Networking](https://handwiki.org/wiki/Time-Sensitive_Networking)

Time-Sensitive Networking (TSN) is a set of standards under development by the Time-Sensitive Networking task group of the IEEE 802.1 working group.<sup>[1]</sup> The TSN task group was formed in November 2012 by renaming the existing Audio Video Bridging Task Group<sup>[2]</sup> and continuing its work. The name changed as a result of the extension of the working area of the standardization group. The standards define mechanisms for the time-sensitive transmission of data over deterministic Ethernet networks.

development by the IEEE for real-time networking

The majority of projects define extensions to the IEEE 802.1Q – Bridges and Bridged Networks, which describes Virtual LANs and network switches.<sup>[3]</sup> These extensions in particular address the transmission of very low transmission latency and high availability. Applications include converged networks with real-time Audio/Video Streaming and real-time control streams which are used in automotive or industrial control facilities.

### IEEE 802.1AS Timing and Synchronization for Time-Sensitive Applications

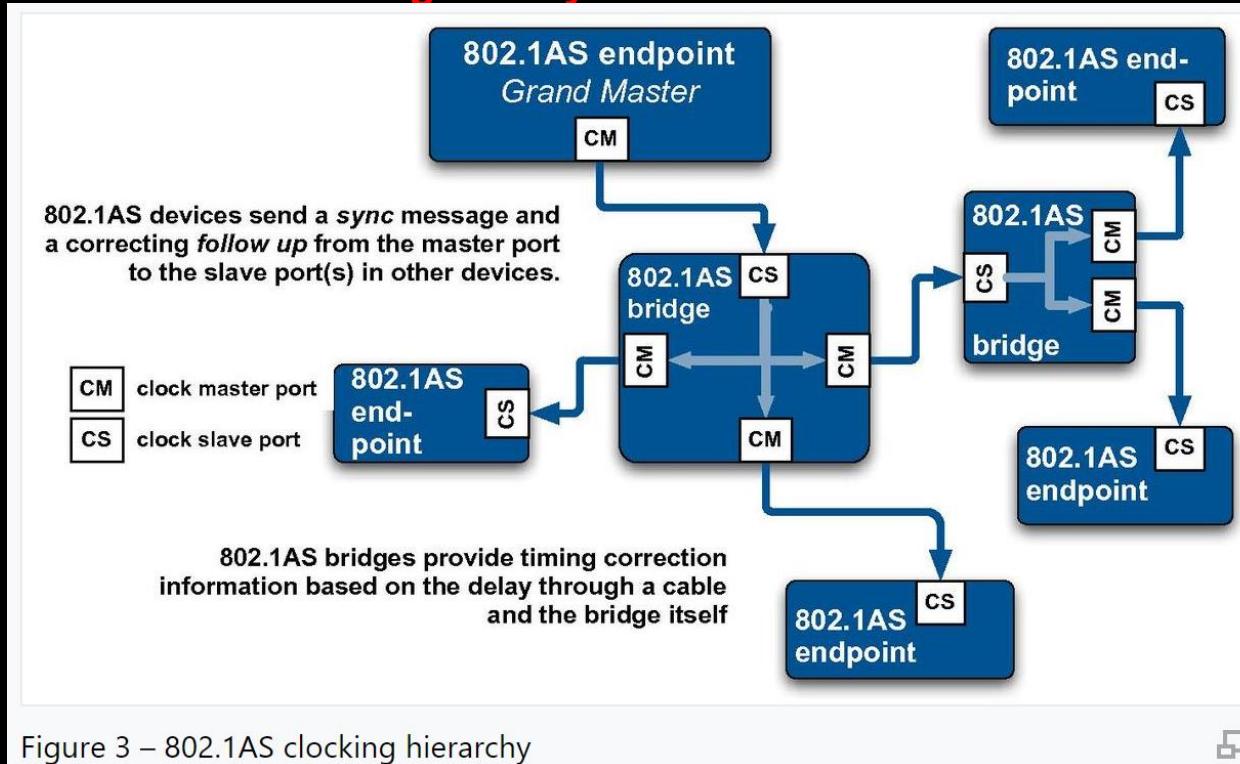


Figure 3 – 802.1AS clocking hierarchy

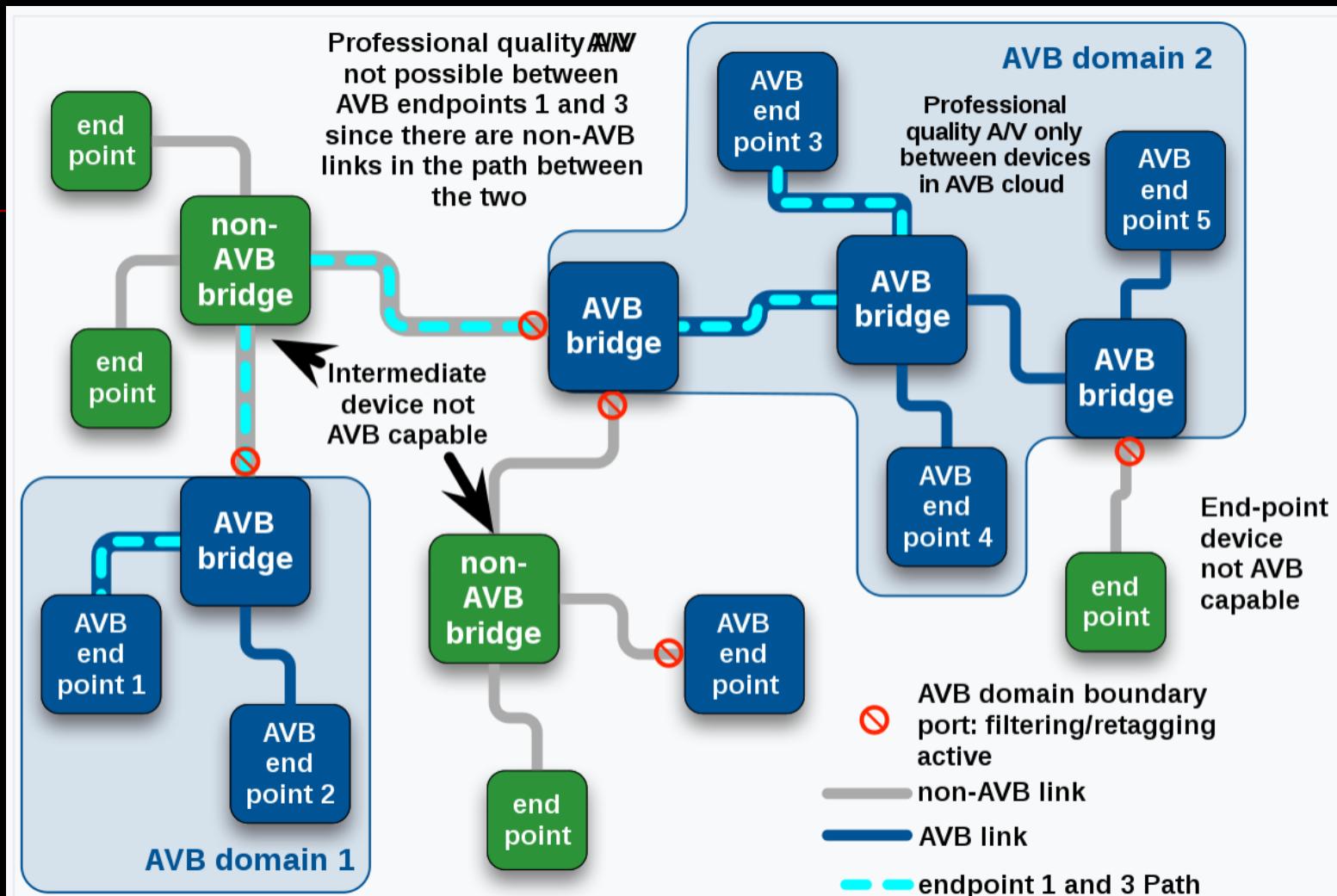


Figure 2 - AVB Connections





# Standards

Time-Sensitive Networking standards suite			
Standard	Title	Status	Publication Date
IEEE 802.1BA-2011	Audio Video Bridging (AVB) Systems	Current, amended by Cor1-2016 [12]	30 September 2011
IEEE 802.1AS-2020	Timing and Synchronization for Time-Sensitive Applications (gPTP)	Current [13][14]	30 January 2020
IEEE 802.1ASdm	Timing and Synchronization for Time-Sensitive Applications - Hot Standby	Draft 0.2[15]	15 March 2021
IEEE 802.1Qav-2009	Forwarding and Queuing Enhancements for Time-Sensitive Streams		5 January 2010
IEEE 802.1Qat-2010	Stream Reservation Protocol (SRP)		30 September 2010
IEEE 802.1aq-2012	Shortest Path Bridging(SPB)		29 March 2012
IEEE 802.1Qbp-2014	Equal Cost Multiple Paths (for Shortest Path Bridging)		27 March 2014
IEEE 802.1Qbv-2015	Enhancements for Scheduled Traffic	Incorporated into IEEE 802.1Q	18 March 2016
IEEE 802.1Qbu-2016	Frame Preemption		30 August 2016
IEEE 802.1Qca-2015	Path Control and Reservation		11 March 2016
IEEE 802.1Qch-2017	Cyclic Queuing and Forwarding		28 June 2017
IEEE 802.1Qci-2017	Per-Stream Filtering and Policing		28 September 2017
IEEE 802.1Q-2018	Bridges and Bridged Networks ( <i>incorporates 802.1Qav/Qat/aq/Qbp/Qbv/Qbu/Qca/Qci/Qch and other amendments</i> )	Current[16]	6 July 2018
IEEE 802.1Q-Rev	Bridges and Bridged Networks	Draft 1.0[17]	1 August 2021
IEEE 802.1Qcc-2018	Stream Reservation Protocol (SRP) Enhancements and Performance Improvements	Current[18]	31 October 2018
IEEE 802.1Qcy-2019	Virtual Station Interface (VSI) Discovery and Configuration Protocol (VDP)	Current[19]	4 June 2018
IEEE 802.1Qcj	Automatic Attachment to Provider Backbone Bridging (PBB) services	Draft 1.3 [20]	12 February 2021
IEEE 802.1Qcr-2020	Asynchronous Traffic Shaping	Current[21][22]	6 November 2020
IEEE 802.1Qcz	Congestion Isolation	Draft 2.0[23]	15 January 2021
IEEE 802.1Qdd	Resource Allocation Protocol	Draft 0.5[24]	1 September 2021
IEEE 802.1Qdj	Configuration enhancements for TSN	Draft 0.1[25]	20 November 2020
IEEE 802.1AB-2016	Station and Media Access Control Connectivity Discovery ( <a href="#">Link Layer Discovery Protocol (LLDP)</a> )	Current[26]	11 March 2016
IEEE 802.1ABdh	Station and Media Access Control Connectivity Discovery - Support for Multiframe Protocol Data Units (LLDPv2)	Draft 2.0[27]	15 July 2021
IEEE 802.1AX-2020	Link Aggregation	Current[28][29]	30 January 2020
IEEE 802.1CB-2017	Frame Replication and Elimination for Reliability	Current[30]	27 October 2017
IEEE 802.1CBdb	FRER Extended Stream Identification Functions	Draft 2.0[31]	22 July 2021
IEEE 802.1CM-2018	Time-Sensitive Networking for Fronthaul	Current[32][33]	8 June 2018
IEEE 802.1CMde-2020	Enhancements to Fronthaul Profiles to Support New Fronthaul Interface, Synchronization, and Syntonization Standards	Current[34]	16 October 2020
IEEE 802.1CS-2020	Link-Local Registration Protocol	Current[35][36]	3 December 2020
IEEE 802.1CQ	Multicast and Local Address Assignment	Draft 0.7[37]	19 July 2021
IEEE 802.1DC	Quality of Service Provision by Network Systems	Draft 1.1[38]	18 October 2019
IEEE 802.1DF	TSN Profile for Service Provider Networks	Draft 0.1[39]	21 December 2020
IEEE 802.1DG	TSN Profile for Automotive In-Vehicle Ethernet Communications	Draft 1.3[40]	18 December 2020
IEEE 802.1DP	TSN for Aerospace Onboard Ethernet Communications	preparation [41]	3 December 2020
IEC/IEEE 60802	TSN Profile for Industrial Automation	Draft 1.3[42]	25 August 2020



■ <https://1.ieee802.org/tsn/>

## Time-Sensitive Networking (TSN) Profiles (Selection and Use of TSN tools)

Audio Video Bridging  
[802.1BA/Revision]

Fronthaul  
[802.1CM/de]

Industrial Automation  
[IEC/IEEE 60802]

Automotive In-Vehicle  
[P802.1DG]

Service Provider  
[P802.1DF]

Aerospace Onboard  
[IEEE P802.1DP / SAE AS6675]

### Time synchronization:

Timing and Synchronization [802.1AS-2020]  
(a profile of IEEE 1588)  
Hot Standby [P802.1ASdm]  
YANG [P802.1ASdn]  
Inclusive Terminology [P802.1ASdr]

## TSN Components

(Tools of the TSN toolset)

Synchronization

Reliability

Latency

Resource Management

Zero congestion loss =  
Bounded latency

### Bounded low latency:

Credit Based Shaper [802.1Qav]  
Frame Preemption [802.1Qbu & 802.3br]  
Scheduled Traffic [802.1Qbv]  
Cyclic Queuing and Forwarding [802.1Qch]  
Asynchronous Traffic Shaping [802.1Qcr]  
Shaper Parameter Settings [P802.1Qdq]  
QoS Provisions [P802.1DC]

### High availability / Ultra reliability:

Frame Replication and Elimination [802.1CB]  
Path Control and Reservation [802.1Qca]  
Per-Stream Filtering and Policing [802.1Qci]  
Reliability for Time Sync [802.1AS-2020]

### Dedicated resources & API:

Stream Reservation Protocol [802.1Qat]  
Link-local Registration Protocol [802.1CS]  
TSN Configuration [802.1Qcc]  
Foundational Bridge YANG [802.1Qcp]  
YANG for CFM [802.1Qcx]  
YANG for LLDP [P802.1ABcu]  
YANG for 802.1Qbv/Qbu/Qci [P802.1Qcw]  
YANG & MIB for FRER [P802.1CBcv]  
Extended Stream Identification [P802.1CBdb]  
Resource Allocation Protocol [P802.1Qdd]  
TSN Configuration Enhancements [P802.1Qdj]  
LLDPv2 for Multiframe Data Units [P802.1ABdh]  
Multicast and Local Address Assignment [P802.1CQ]

Note: A 'P' in front of '802.1' indicates an ongoing Project.

More on [TSN standards](#) and [ongoing projects](#) at: <https://www.ieee802.org/1/tsn>

11/16/2021

...



## 2.3.3.5 SOAFEE

■ <https://soafee.io/>

■ <https://www.arm.com/en/solutions/automotive/software-defined-vehicles>

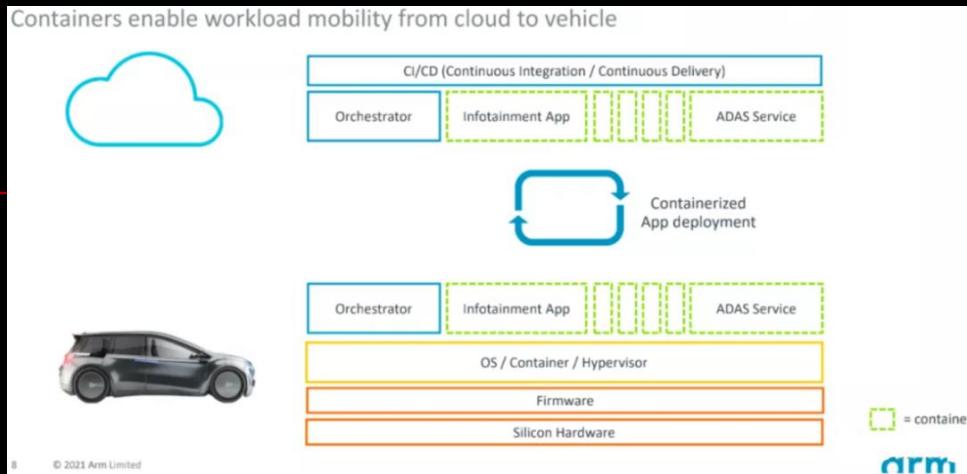
SOAFEE Summary		
	Key Information	Other Information
What is SOAFEE?	<ul style="list-style-type: none"><li>• Software framework for cloud &amp; auto software</li><li>• Open-source reference software framework</li><li>• Reference hardware platform</li><li>• Builds on Arm's Project Cassini</li><li>• Leverages Arm SystemReady compliance</li></ul>	<ul style="list-style-type: none"><li>• Cloud-native: develop in cloud, deploy in car</li><li>• To improve software-defined car</li><li>• For early software development</li><li>• Cloud-native ecosystem for edge apps</li><li>• Hardware-firmware certification standards</li></ul>
SOAFEE advantages	<ul style="list-style-type: none"><li>• Accelerates cloud-native tech to auto software</li><li>• Increased software portability across platforms</li><li>• Improved software quality and quantity</li><li>• Software container advantages</li></ul>	<ul style="list-style-type: none"><li>• Expands current trends and capabilities</li><li>• For Arm-based hardware and systems</li><li>• Faster development and lower cost</li><li>• Key to portable software modules</li></ul>
SOAFEE cloud-native technology	<ul style="list-style-type: none"><li>• Software containers: Portable &amp; reusable apps</li><li>• Microservice architecture: Lowers complexity</li><li>• Orchestrator: Manages cloud microservices</li><li>• DevOps: Cloud-native workflow aspects</li></ul>	<ul style="list-style-type: none"><li>• Virtualized app; tied to a specific OS</li><li>• A service-oriented architecture (SOA)</li><li>• Kubernetes is most common</li><li>• Development &amp; operational workflow</li></ul>
SOAFEE cloud-native enhancements	<ul style="list-style-type: none"><li>• Automotive orchestrator enhancements</li><li>• Automotive container runtime enhancements</li><li>• Automotive DevOps enhancements</li></ul>	<ul style="list-style-type: none"><li>• For safety and real-time functions</li><li>• Proposal for virtualized container runtime</li><li>• CI/CD testing and validations</li></ul>
SOAFEE availability	<ul style="list-style-type: none"><li>• Downloadable reference software stack</li><li>• Reference hardware platforms for purchase</li></ul>	<ul style="list-style-type: none"><li>• <a href="https://gitlab.arm.com/soafee">https://gitlab.arm.com/soafee</a></li><li>• 2 systems available from ADLINK</li></ul>
SOAFEE competition	<ul style="list-style-type: none"><li>• Will there be SOAFEE competitors?</li><li>• Intel will need equivalent for automotive software</li><li>• Nvidia can leverage SOAFEE</li></ul>	<ul style="list-style-type: none"><li>• For other processor platforms</li><li>• The opportunity window will close quickly</li><li>• Even if Arm acquisition does not happen</li></ul>
Summary	<ul style="list-style-type: none"><li>• Auto industry is moving to cloud-native software</li><li>• SOAFEE accelerates this trend</li><li>• SOAFEE is not a new Arm revenue stream</li></ul>	<ul style="list-style-type: none"><li>• Development, lifetime support &amp; SaaS</li><li>• Likely de facto standard for Arm systems</li><li>• Grows the barrier for processor competitors</li></ul>
CI/CD=Continuous Integration/Continuous Development; OCI= Open Container Initiative		
Source: Egil Juliussen, September 2021		

Source: <https://www.eetimes.com/arm-soafee-brings-automotive-to-the-cloud/>



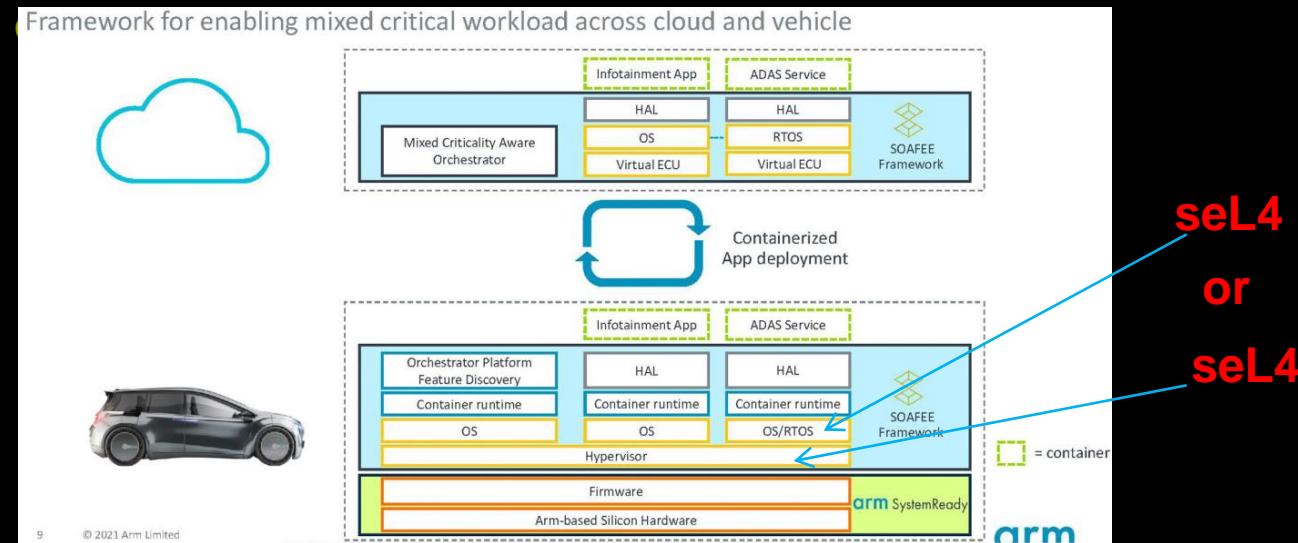
# A Software-Defined Future Requires a Cloud-Native Approach

- Containers enable workload mobility from cloud to vehicle



## Accelerating Cloud Native Development with SOAFEE

- Framework for enabling mixed criticality workload across cloud and vehicle





## 2.3.3.6 AUTOSAR

### ■ <https://en.wikipedia.org/wiki/AUTOSAR>

**AUT**omotive **O**pen **S**ystem **A**Rchitecture (**AUTOSAR**) is a development partnership of **automotive** interested parties founded in 2003. It pursues the objective to create and establish an open and standardized **software architecture** for automotive **electronic control units** (ECUs). Goals include the **scalability** to different vehicle and platform variants, transferability of software, the consideration of availability and safety requirements, a collaboration between various partners, sustainable use of natural resources, and maintainability during the **product lifecycle**.<sup>[1][2][3]</sup>

#### History [edit]

AUTOSAR was formed in July 2003 by Bavarian Motor Works (BMW), Robert Bosch GmbH, Continental AG, Daimler AG (formerly Daimler-Benz, then DaimlerChrysler), Siemens VDO, and Volkswagen to promote an open industry standard for automotive electrical-electronic (E/E) architecture. In November 2003, Ford Motor Company joined as a core partner, and in December, Groupe PSA (formerly PSA Peugeot Citroën) and Toyota Motor Corporation joined. The following November, General Motors also became a core partner. After Siemens VDO was acquired by Continental in February 2008, it ceased being an independent core partner.

Since 2003, AUTOSAR provided four major releases of the automotive software architecture for its classic platform and one release of acceptance tests. The work can be divided into three phases:

- Phase I (2004–2006): Basic development of the standard (releases 1.0, 2.0, 2.1)<sup>[4]</sup>
- Phase II (2007–2009): Extension of the standard in architecture and methodology (releases 3.0, 3.1, 4.0)<sup>[5]</sup>
- Phase III (2010–2013): Maintenance and selected improvements (releases 3.2, 4.1, 4.2)<sup>[6]</sup>

In 2013, AUTOSAR entered a continuous working mode for its classic Platform to maintain the standard and provide selected improvements, including releases R4.2, and 1.0 of acceptance tests.

In 2016, work on the Adaptive Platform began. A first release (17-03) was published in early 2017, followed by release 17-10 in October 2017<sup>[7]</sup> and release 18-03 in March 2018.<sup>[8]</sup> With release 18-10 in October 2018, major development activities were published.<sup>[9]</sup>

In December 2020, AUTOSAR R20-11 was virtually released.<sup>[10]</sup>

<https://www.autosar.org/about/history/>

...

### ■ <https://www.autosar.org/>

#### **AUTOSAR (AUT**omotive **O**pen **S**ystem **A**Rchitecture)

is a worldwide development partnership of vehicle manufacturers, suppliers, service providers and companies from the automotive electronics, semiconductor and software industry.

 [read more](#)

BMW  
GROUP



BOSCH

Continental

DAIMLER



PSA  
GROUPE

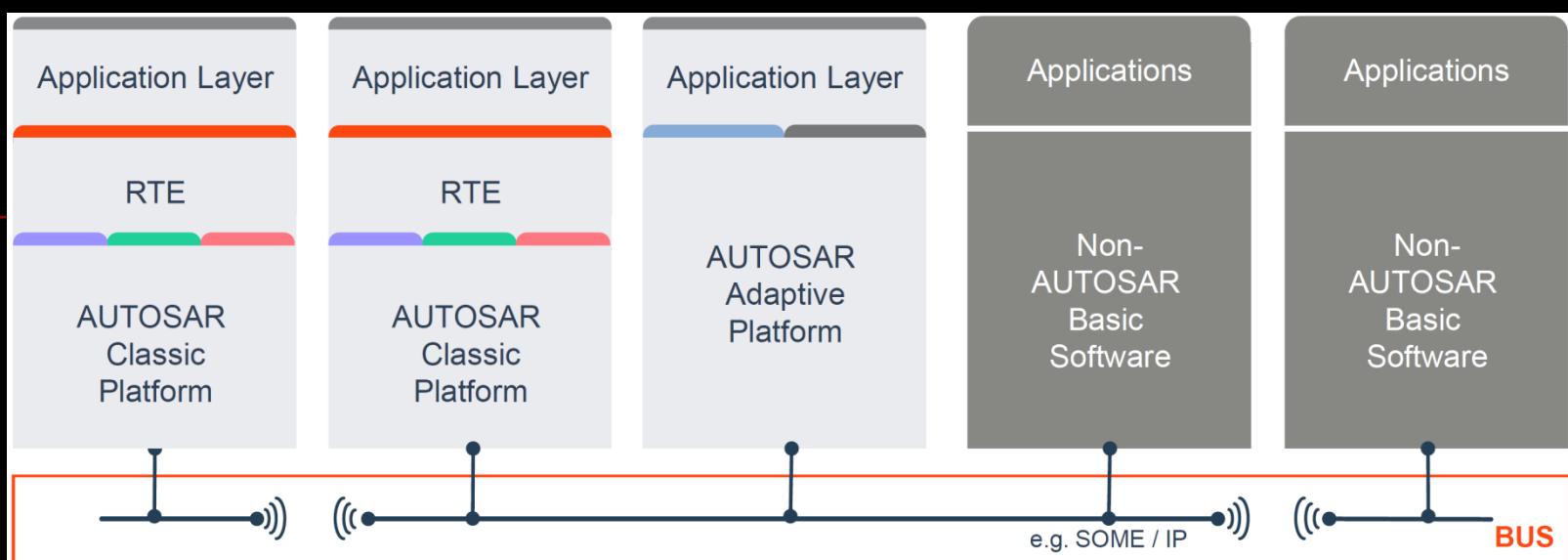
TOYOTA

VOLKSWAGEN  
AGTEIGENESSELSCHAFT

> all partners



## AUTOSAR in a Vehicle Network

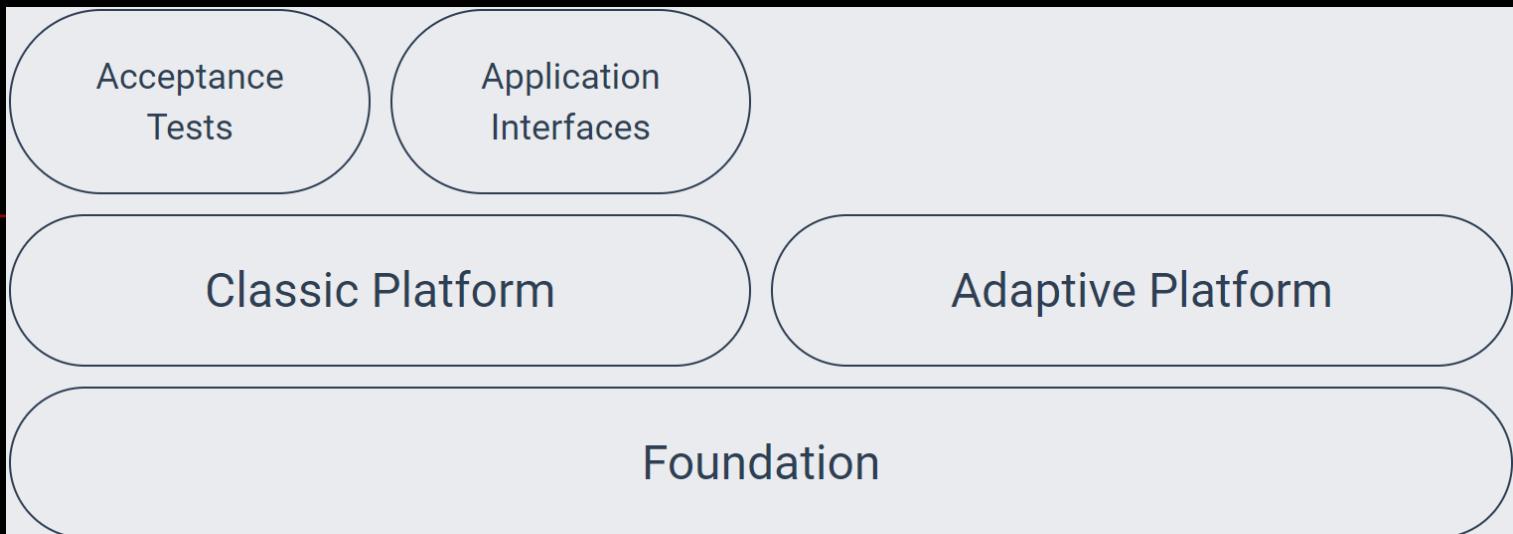


Common Bus Interface Specification

Source: [https://www.autosar.org/fileadmin/user\\_upload/AUTOSAR\\_EXP\\_Introduction\\_Part2.pdf](https://www.autosar.org/fileadmin/user_upload/AUTOSAR_EXP_Introduction_Part2.pdf)

## Architecture & Design

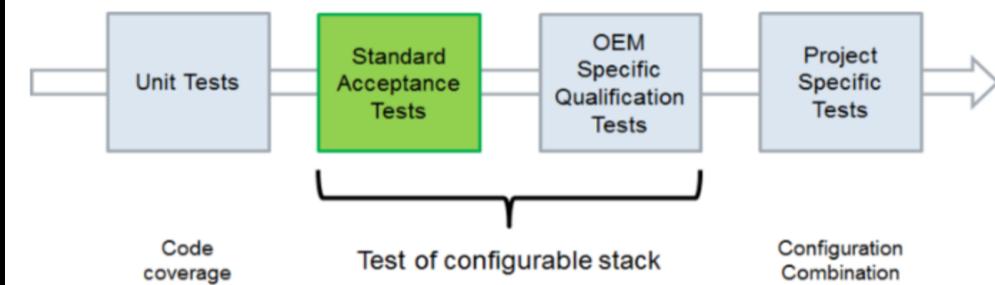
- 



### Acceptance Tests

Thus, acceptance tests can be used for the validation of the interoperability of different AUTOSAR stacks within a network. The test cases provided by AUTOSAR cover RTE requirements (like the generation from artifacts, existence of APIs, RTE behavior), basic software services, libraries, as well as bus behavior (e.g. transmission behavior, bus off handling, network management) and bus protocols (e.g. transport protocol, network management, diagnostic communication). The number of test cases specified by AUTOSAR increases continuously.

In addition AUTOSAR provides a methodology, which can be used by users to extend the standard test suite, for instance on standard features not covered in the standard set, or on user specific features.





## Application Interfaces

### FUNCTIONAL CLUSTER

Body and Comfort

Transmission

Powertrain

Chassis

Occupant and Pedestrian Safety

HMI, Multimedia and Telematics

General

The focus is on interface specifications of well-established applications in order to emphasize software reuse and exchange, which is considered as one of the main requirements of AUTOSAR. The deployment of standardized application interfaces is a key factor for the reuse of applications.

The application interface descriptions contain a richness of data standardized by experts of all partners. These standardized interfaces allow software designers and implementers to use them in case of expanding or reusing software components independent of a specific hardware and/or Electronic Control Unit (ECU).

In general, applications are the competitive edge of an ECU. AUTOSAR is not going to standardize the functional internal behavior of an application, e.g. algorithms, but the content exchanged between applications.

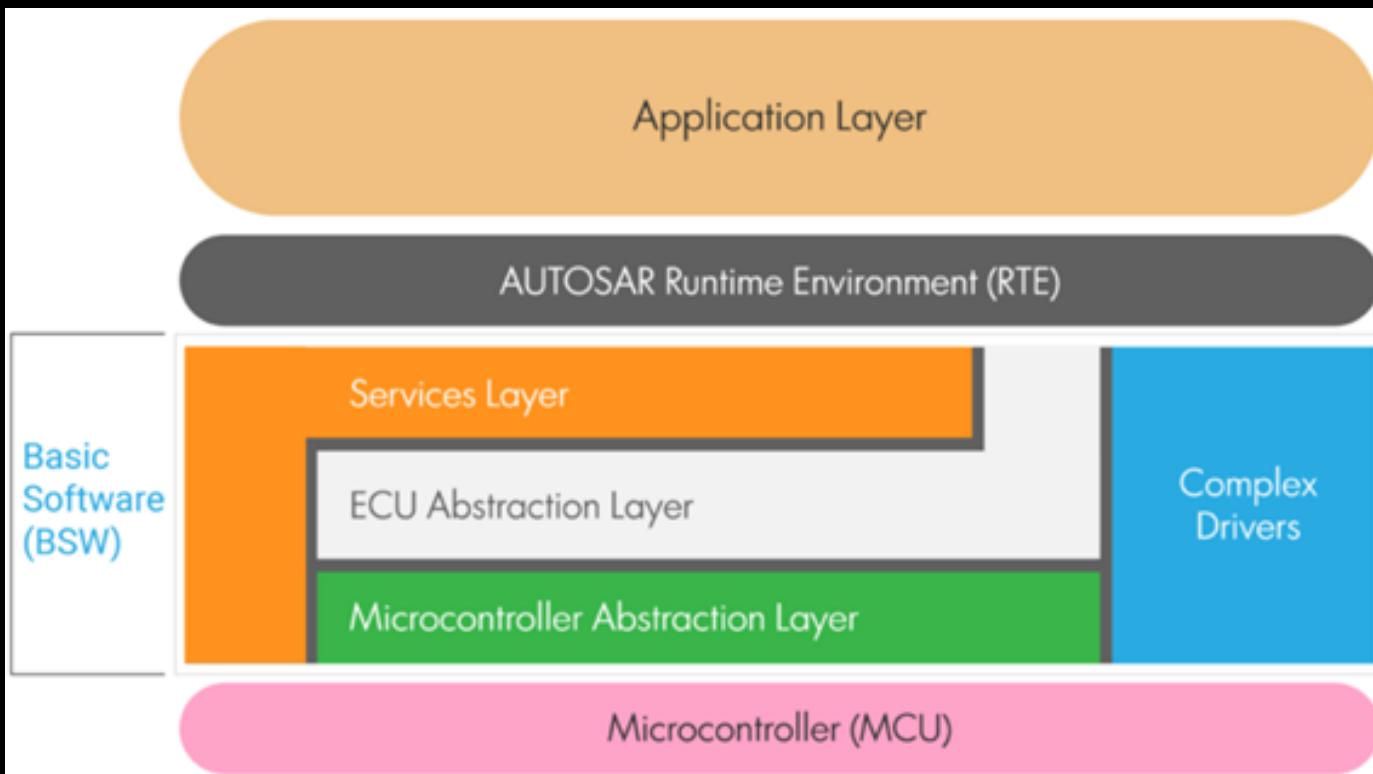
Typical examples of applications are electronic stability control (ESC), steering, electric parking brake, park distance control, exterior light, anti-theft systems, remote keyless entry and so on.



## Classic Platform

The AUTOSAR Classic Platform architecture distinguishes on the highest abstraction level between three software layers which run on a microcontroller: application, runtime environment (RTE) and basic software (BSW).

- The application software layer is mostly hardware independent.
- Communication between software components and access to BSW via RTE.
- The RTE represents the full interface for applications.
- The BSW is divided in three major layers and complex drivers:
  - Services, ECU (Electronic Control Unit) abstraction and microcontroller abstraction.
- Services are divided furthermore into functional groups representing the infrastructure for system, memory and communication services.

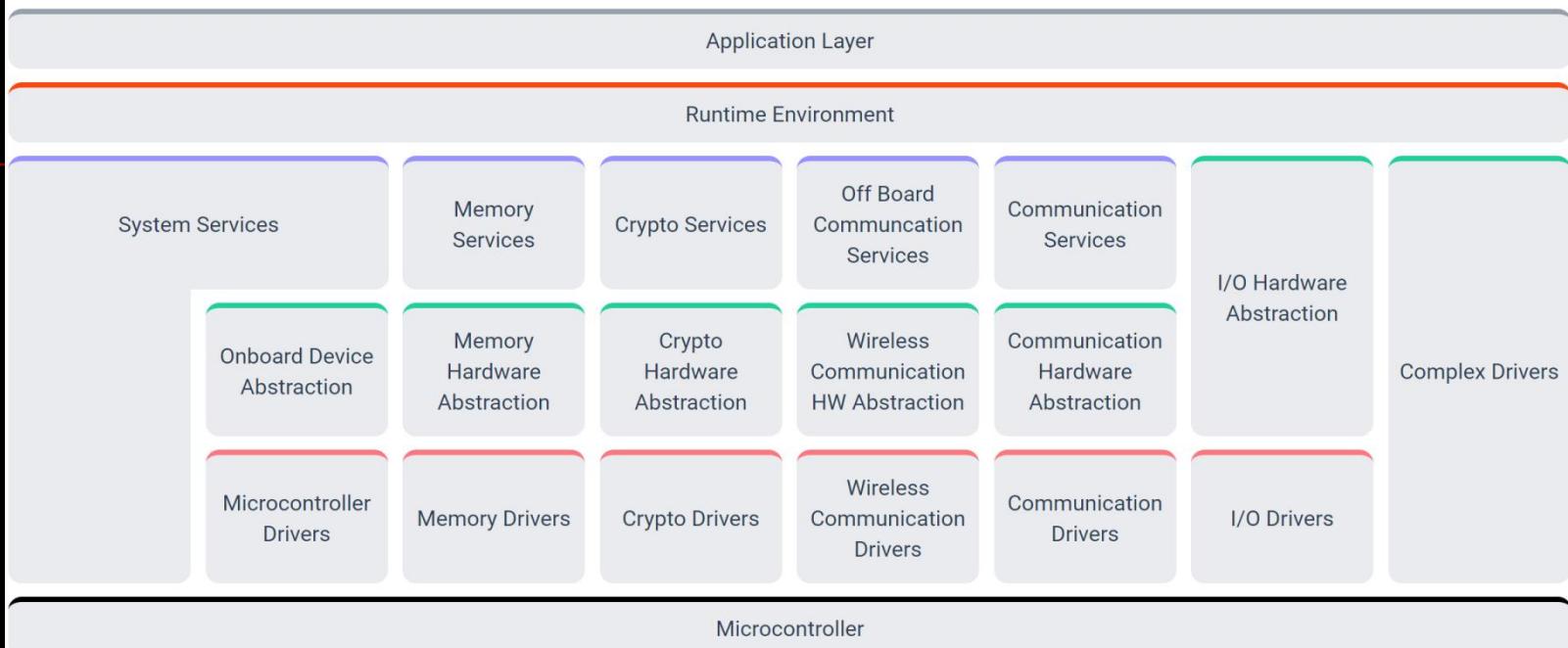


Source: <https://www.faststreamtech.com/industries/smart-automotive-solution/autosar/>



## Current Release

AUTOSAR Classic Release R21-11





# Adaptive Platform

The AUTOSAR Adaptive Platform implements the AUTOSAR Runtime for Adaptive Applications (ARA). Two types of interfaces are available, services and APIs. The platform consists of functional clusters which are grouped in services and the Adaptive AUTOSAR Basis.

Functional clusters...

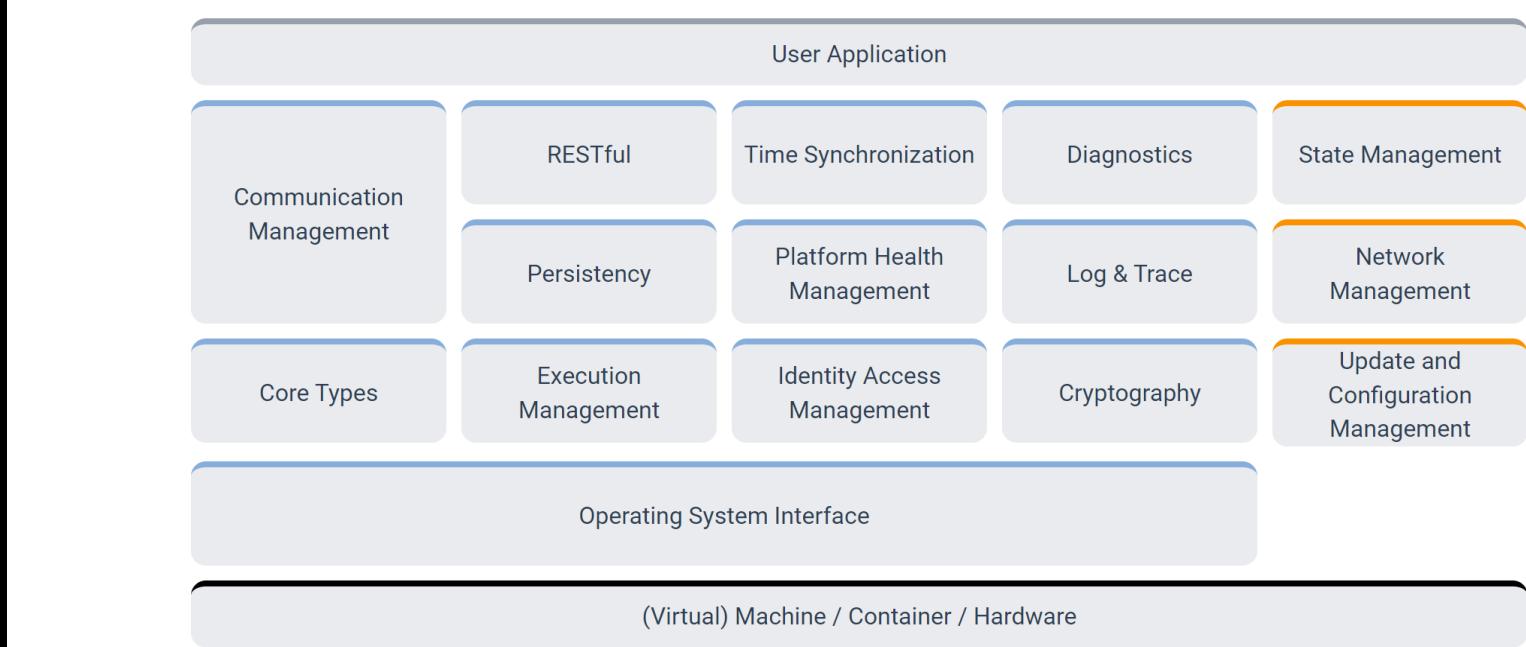
- assemble functionalities of the Adaptive Platform
- define clustering of requirements specification
- describe behavior of software platform from application and network perspective
- but, do not constrain the final SW design of the architecture implementing the Adaptive Platform.

Functional clusters in AUTOSAR Adaptive Platform Basis have to have at least one instance per (virtual) machine while services may be distributed in the in-car network.

In comparison to the AUTOSAR Classic Platform the AUTOSAR Runtime Environment for the Adaptive Platform dynamically links services and clients during runtime.

## Current Release

AUTOSAR Adaptive Release R21-11





## Foundation

The purpose of the Foundation standard is to enforce interoperability between the AUTOSAR platforms.

Foundation contains common requirements and technical specifications (for example protocols) shared between the AUTOSAR platforms.

### CURRENT RELEASE

[AUTOSAR Foundation Release R21-11](#)

Objectives and Main Requirements

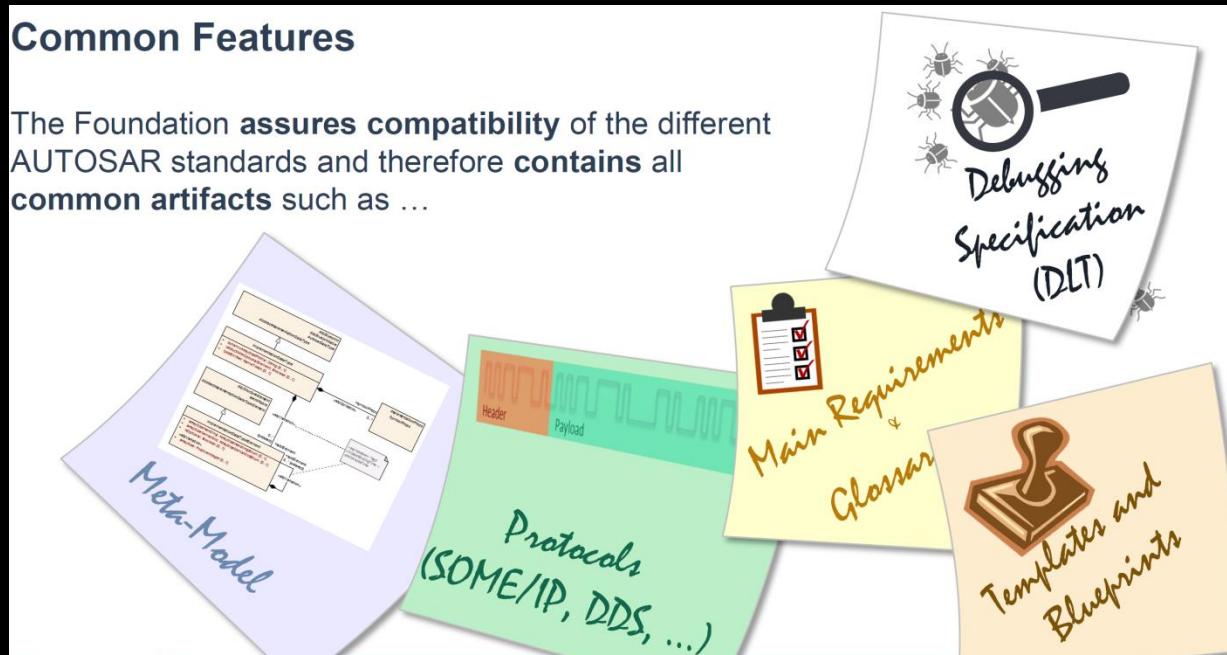
Software Architecture

Protocols

General

## Common Features

The Foundation **assures compatibility** of the different AUTOSAR standards and therefore **contains all common artifacts** such as ...



Source: [https://www.autosar.org/fileadmin/user\\_upload/AUTOSAR\\_EXP\\_Introduction\\_Part2.pdf](https://www.autosar.org/fileadmin/user_upload/AUTOSAR_EXP_Introduction_Part2.pdf)

## ***Rust in AUTOSAR***



- **<https://www.autosar.org/news-events/details/autosar-announces-new-working-group-for-programming-language-rust-in-automotive-software-context-202/>**

Rust is a multi-paradigm, general-purpose programming language designed for performance and safety, especially safe concurrency. Rust is syntactically similar to C++, but can guarantee memory safety without garbage collection. Rust has been called a systems programming language, and in addition to high-level features such as functional programming it also offers mechanisms for low-level memory management.

First appearing in 2010, Rust designers refined the language while writing the Firefox browser engine. It has gained popularity and investment from the industry, including Amazon, Discord, Dropbox, Facebook (Meta), Google (Alphabet), and Microsoft. Repeatedly, Rust has been voted the "most loved programming language" in the Stack Overflow Developer Survey.

As Rust is built by its community, each major decision in Rust starts as a Request for Comments (RFC). Everyone is invited to discuss the proposal, to work toward a shared understanding of the tradeoffs. Though sometimes arduous, this community deliberation is Rust's secret sauce for quality. Therefore, it is important to bridge among communities. The future AUTOSAR Working Group Speaker is a lucky catch for this task.

Christof Petig, well known in the Rust community, has agreed to take over the subgroup speaker role. He has 25 years of C++ experience and has become a Rust enthusiast in the meantime. "The code written in Rust is checked to be memory safe and free from data races. At the same time, since all possible checks are checked at compile time, there is negligible runtime overhead. This means that the performance of Rust is comparable to C++", Christof summarizes during the initial talks. As other standardization bodies such as Khronos or SAE in the automotive field are in line with such assessment, the Embedded Software focus is to combine efforts for efficient standardization.

All this is not new to The AUTOSAR (AUTomotive Open System ARchitecture) development partnership and its community. Experienced in ramp up of C++14 Coding Guidelines AUTOSAR want to keep alive its tradition of innovation and being a Functional Safety and Automotive Cybersecurity focused standardization body. The decision to form a subgroup within the Working Group for Functional Safety (WG-SAF) and to investigate the utilization of Rust in AUTOSAR Adaptive Platform is a consequence. The subgroup will officially get started on April 2022 and plans to produce two documents. One of the documents will be providing guidance on how Rust can be utilized in the context of AUTOSAR Adaptive Platform projects. The other document will propose Coding Guidelines on Rust.

Additionally, the AUTOSAR Adaptive Demonstrator could be used as code base for hands-on implementations. If your company wishes to be a part of this activity, join as an AUTOSAR-Partner or stay tuned for the first release on [www.autosar.org](http://www.autosar.org).



## AUTOSAR Adaptive Platform Development Approach

### Specification

#### Identify needs & use-cases:

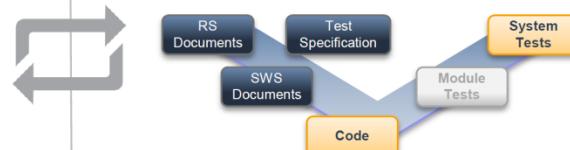
- 1) Concepts
- 2) Features
- 3) Requirements



### Implementation

#### Gain speed:

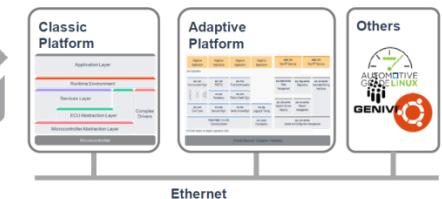
- 1) Spec validation
- 2) Reduce room for spec interpretation
- 3) Training / dissemination of AP



### Demonstration

#### Gain trust:

- 1) Advertises the progress
- 2) Highlights some specific features



#### Quality:

- TF-ARC approval
- Cross team review
- Lifecycle : preliminary → draft → valid

#### Attracting environment for coders:

- Appealing technology (C++, Yocto, Git, ...)
- Modern use case (ADAS EBA)
- Handy documentation (Wiki)
- Peer programming sessions

**Best tradeoff between commercial cooperation & compatibility between different vendors**

Source: [https://www.autosar.org/fileadmin/user\\_upload/AUTOSAR\\_EXP\\_Introduction\\_Part2.pdf](https://www.autosar.org/fileadmin/user_upload/AUTOSAR_EXP_Introduction_Part2.pdf)



## 2.3.3.7 MIH EV

- <https://www.mih-ev.org/>

**Creating an open EV ecosystem that promotes collaboration in the mobility industry.**



### Open Ecosystem

Open ecosystem results in accelerating innovation. MIH welcomes all brands, developers, and auto supply chain participants to become members of the Open EV Alliance to build the Open EV Platform with collective wisdom. The platform provides an environment where all the major functions of an EV such as chassis, batteries, ADAS (Advanced Driver Assistance Systems), cyber security, cloud connectivity, BMS (Battery Management System), and many more can be developed onto.



### Lower Barriers Of Entry

MIH Open EV Platform develops reference designs and standards that can help reduce cost and development cycles to drive more business opportunities for the overall electric vehicle market.



### Resources for Growth

Participate in Interest/Working Groups to exchange ideas and influence strategic direction in the EV development journey. MIH empowers our members with the latest market insights, certification training, IP consultation, and maturity assessment to guide your business decisions and put you on the path to success.

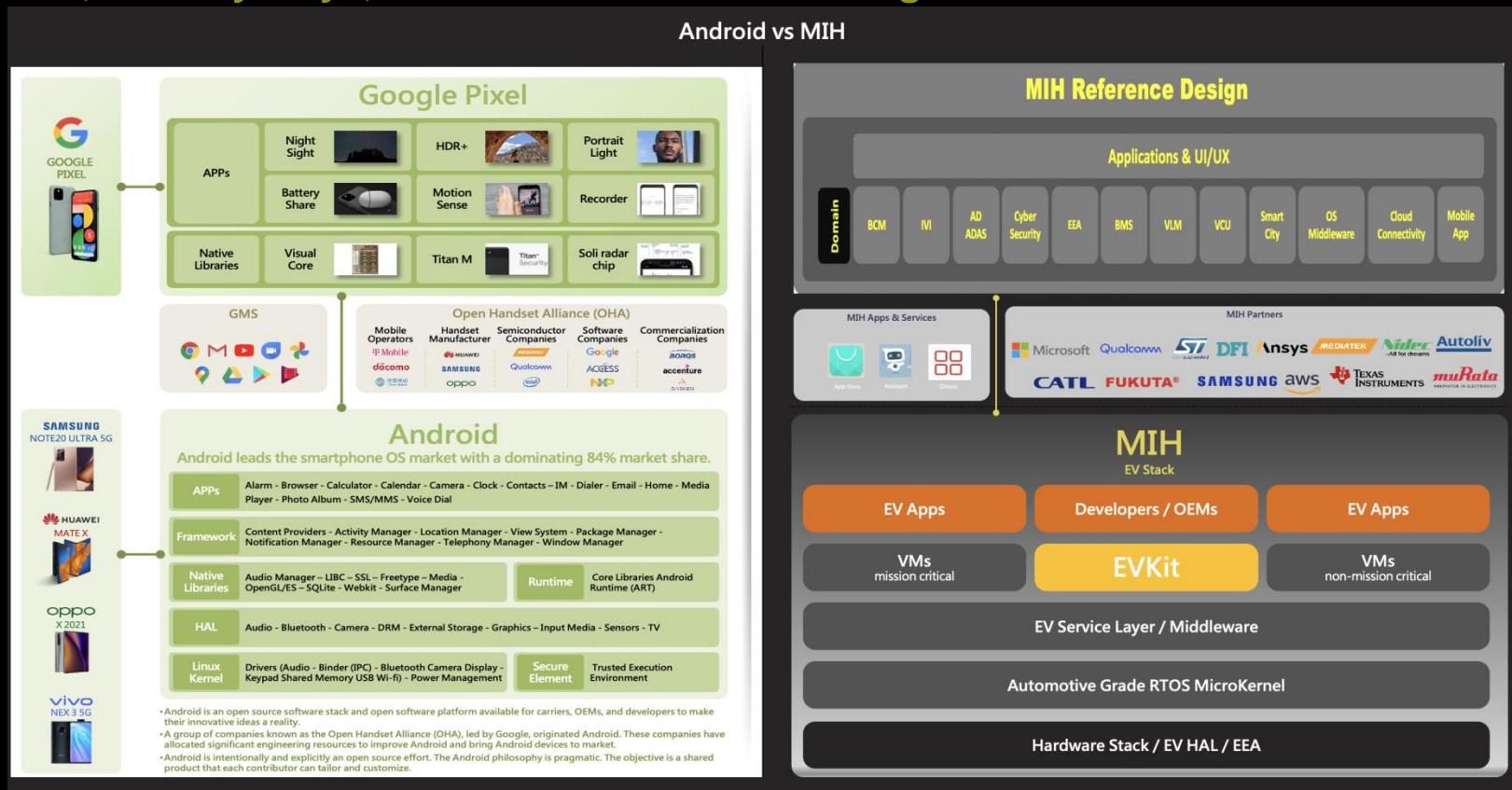


## Development

■ <https://developer.mih-ev.org/>

### What is MIH?

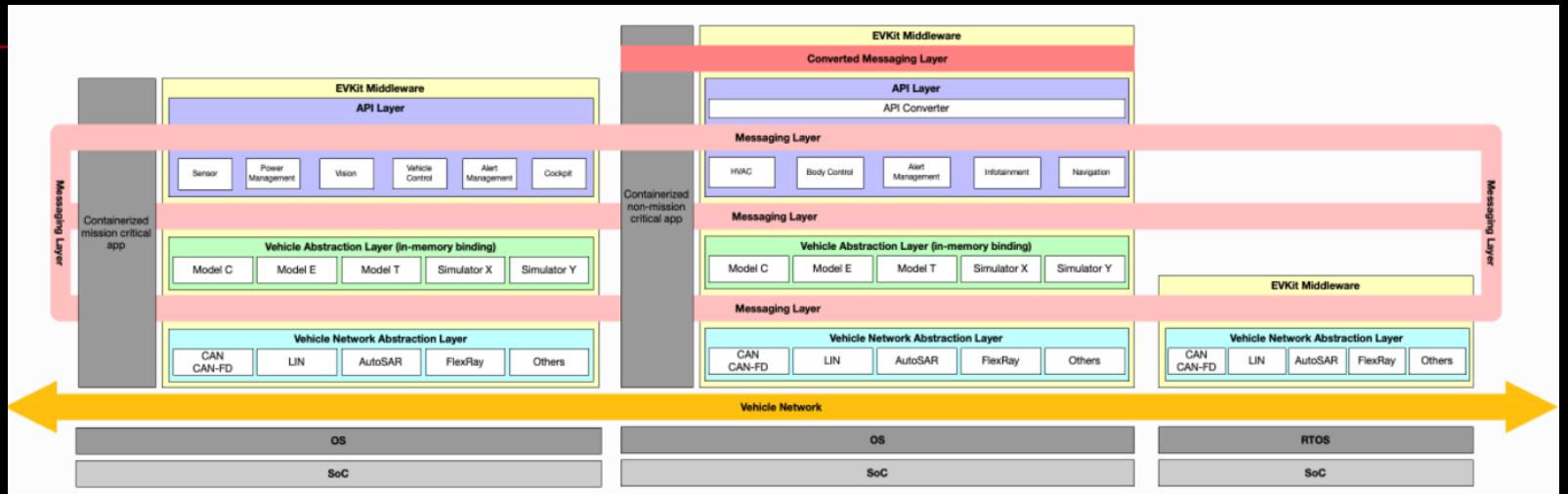
**MIH is building an integrated/multi-layered EV Stake (Platform and Ecosystem). It is, in many ways, similar to Android and Google.**





## What is MIH EVKit?

MIH EVKit is the software stack for software defined vehicle (SDV), it provides an universal interface for application developers to develop applications that will work on every EVKit compatible vehicles.



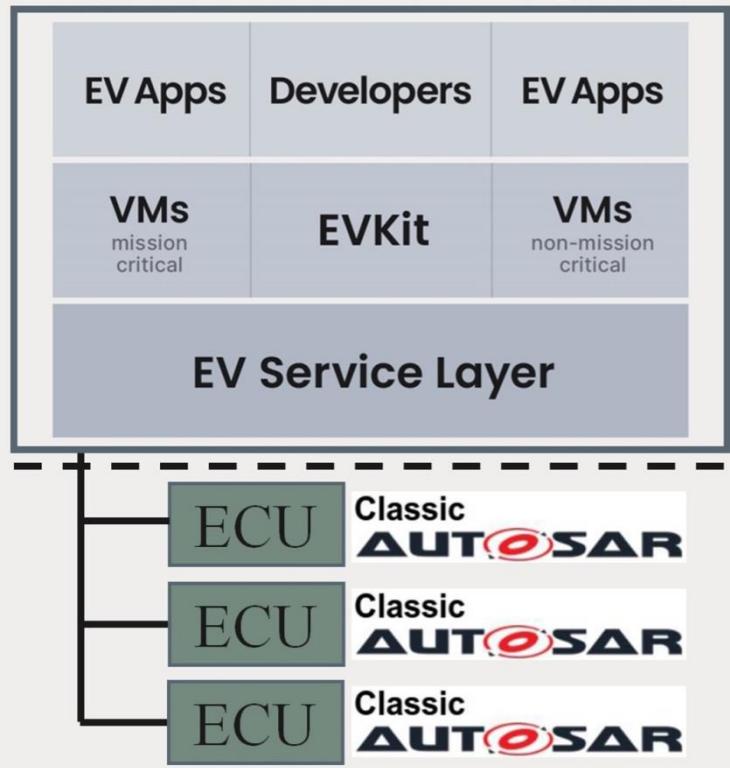


## EVKit and AutoSAR

EVKit is compatible with AUTOSAR, not only does EVKit meet the requirements of traditional OEM, but EVKit also offers options other than AUTOSAR.

Developers can use AUTOSAR tools provided by third parties for software design and development in a familiar model based method. With the automatically generated code, EVKit lowers entry barriers of software development, and complies with functional safety certification.

### High Performance Computing





## **Good Resources**

- <https://www.mih-ev.org/en/news/>
- <https://techtaiwan.com/20211029/foxconn-initiated-mih-unveils-open-ev-software-platform-with-arm/>
- ...



## 2.4 Edge Computing

### 2.4.1 MEC

- [https://en.wikipedia.org/wiki/Multi-access\\_edge\\_computing](https://en.wikipedia.org/wiki/Multi-access_edge_computing)

**Multi-access edge computing (MEC)**, formerly **mobile edge computing**, is an ETSI-defined<sup>[1]</sup> network architecture concept that enables **cloud computing** capabilities and an IT service environment at the **edge of the cellular network**<sup>[2][3]</sup> and, more in general at the edge of any network. The basic idea behind MEC is that by running applications and performing related processing tasks closer to the cellular customer, network congestion is reduced and applications perform better. MEC technology is designed to be implemented at the **cellular base stations** or other edge nodes, and enables flexible and rapid deployment of new applications and services for customers. Combining elements of information technology and telecommunications networking, MEC also allows cellular operators to open their **radio access network** (RAN) to authorized third parties, such as application developers and content providers.

Technical standards for MEC are being developed by the **European Telecommunications Standards Institute**, which has produced a technical white paper about the concept.<sup>[4]</sup>

#### Distributed computing in the RAN [edit]

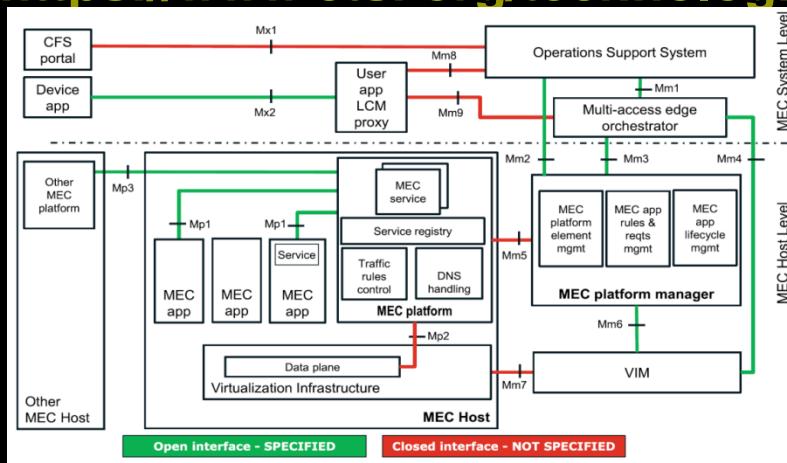
MEC provides a **distributed computing** environment for application and service hosting. It also has the ability to store and process content close to cellular subscribers, for faster response time.<sup>[5]</sup> Applications can also be exposed to real-time **radio access network** (RAN) information.<sup>[6]</sup>

The key element is the MEC application server, which is integrated at the RAN element. This server provides computing resources, storage capacity, connectivity and access to RAN information. It supports a **multitenancy** run-time and hosting environment for applications. The **virtual appliance** applications are delivered as packaged operating system **virtual machine** (VM) images or containers incorporating operating systems and applications. The platform also provides a set of **middleware** application and infrastructure services. **Application software** can be provided from equipment vendors, service providers and third-parties.

#### Key points:

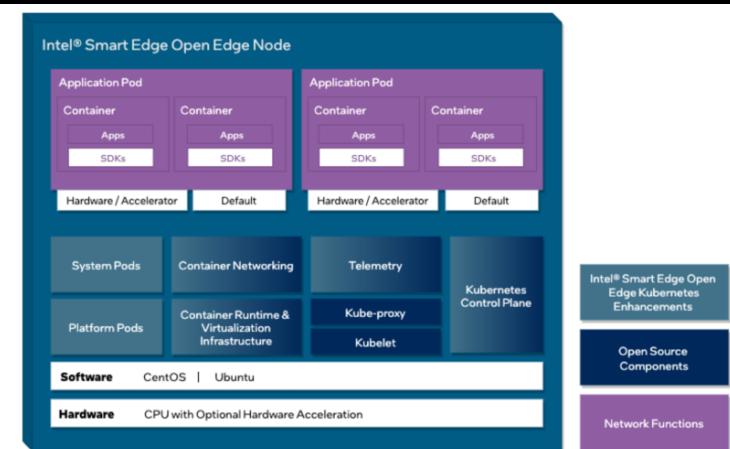
- 1) distributed computing
- 2) security
- 3) virtualization

- <https://www.etsi.org/technologies/multi-access-edge-computing>



# OpenNESS

- <https://www.openness.org/>  
<https://smart-edge-open.github.io/docs/product-overview/>



The Intel® Smart Edge Open node architecture is specialized for each experience kit, to enable developers to create solutions for specific use cases at a given edge location.

The edge node for the Developer Experience Kit:

## Edge Node Component



Harbor registry  
Container registry  
Helm Charts repo



K8s node  
kube-proxy  
kubenet



K8s control plane

collectd plugins: telemetry

collectd node exporters: CPU, memory  
telemetry

cAdvisor: CPU, memory telemetry

Prometheus: telemetry backend

StatsD exporter: StatsD metrics instrumented  
application

Grafana: dashboard

NFD: node labelling support

Multus: multiple network interface support

Calico: default CNI

SR-IOV Network Operator: SR-IOV device plugin

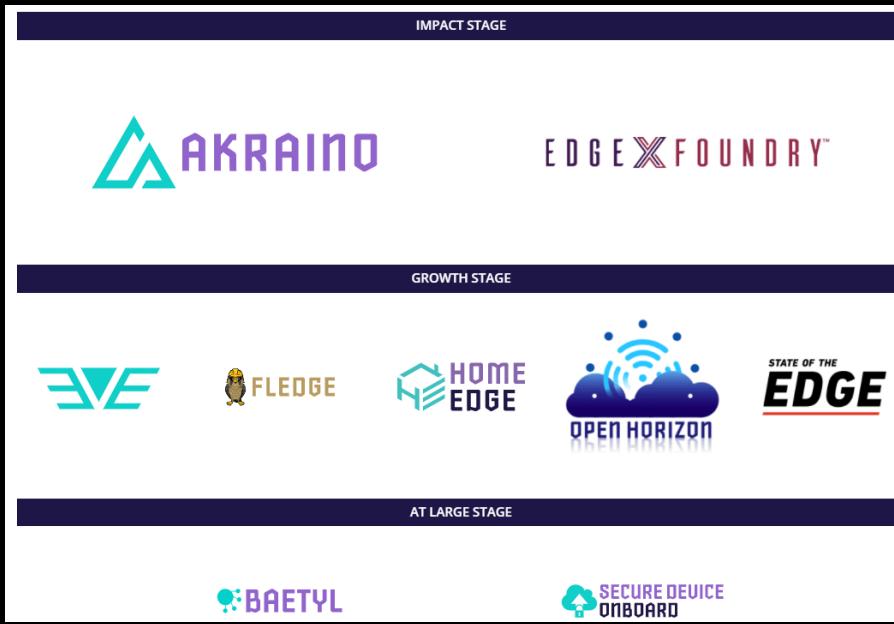
SR-IOV CNI: CNI for SR-IOV VF kernel interface

Ubuntu 20.04 LTS: Default the latest kernel, Ansible, Python, system services to manage attestation, systemd, docker,



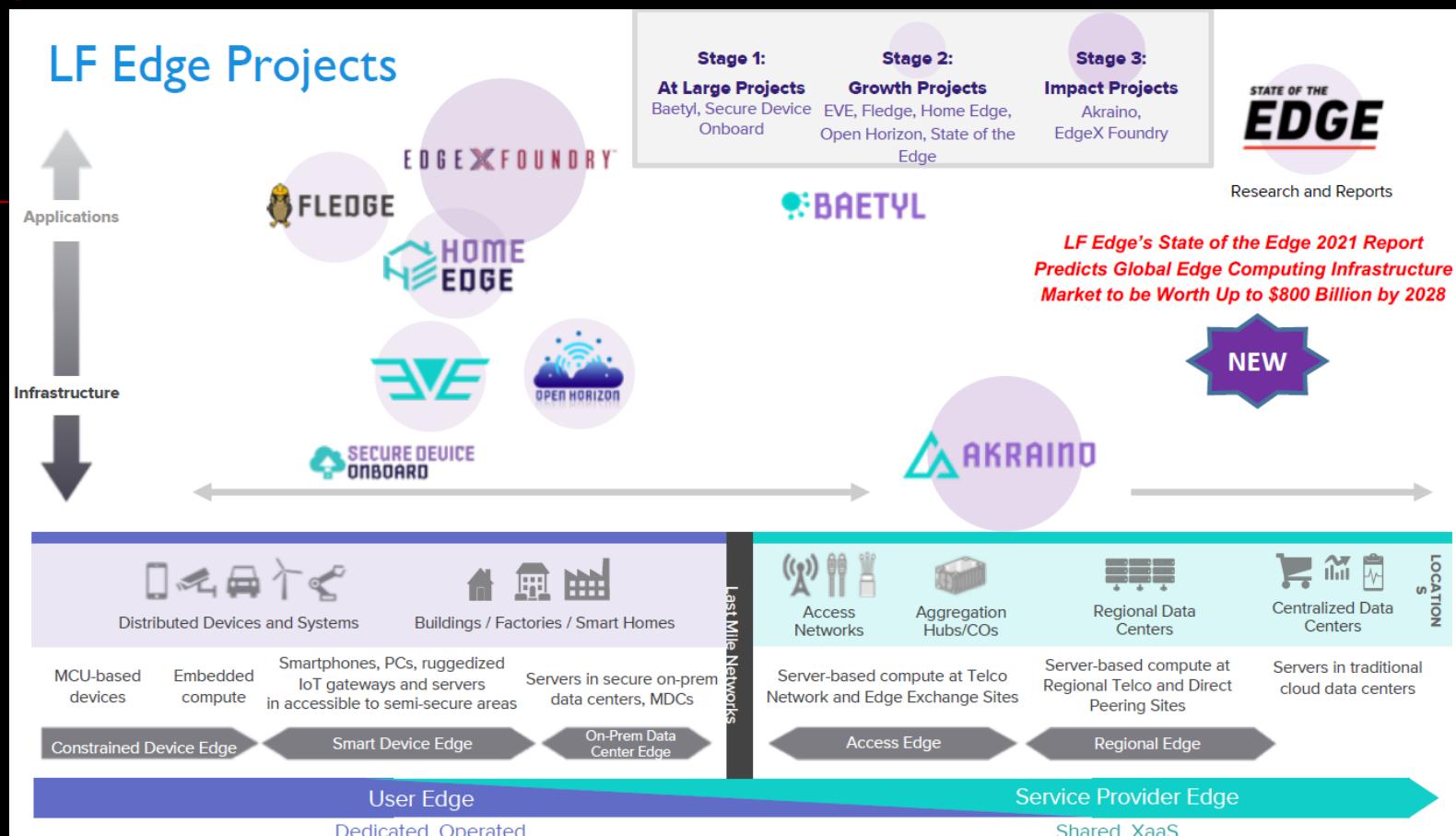
## 2.4.2 LF Edge

- <https://www.lfedge.org/>
- **IOT + Telecom + Cloud + Enterprise + Industrial**
- <https://www.linuxfoundation.org/en/press-release/the-linux-foundation-launches-new-lf-edge-to-establish-a-unified-open-source-framework-for-the-edge/>
- <https://www.lfedge.org/resources/publications/>
- <https://landscape.lfedge.org/>
- <https://www.lfedge.org/projects/>



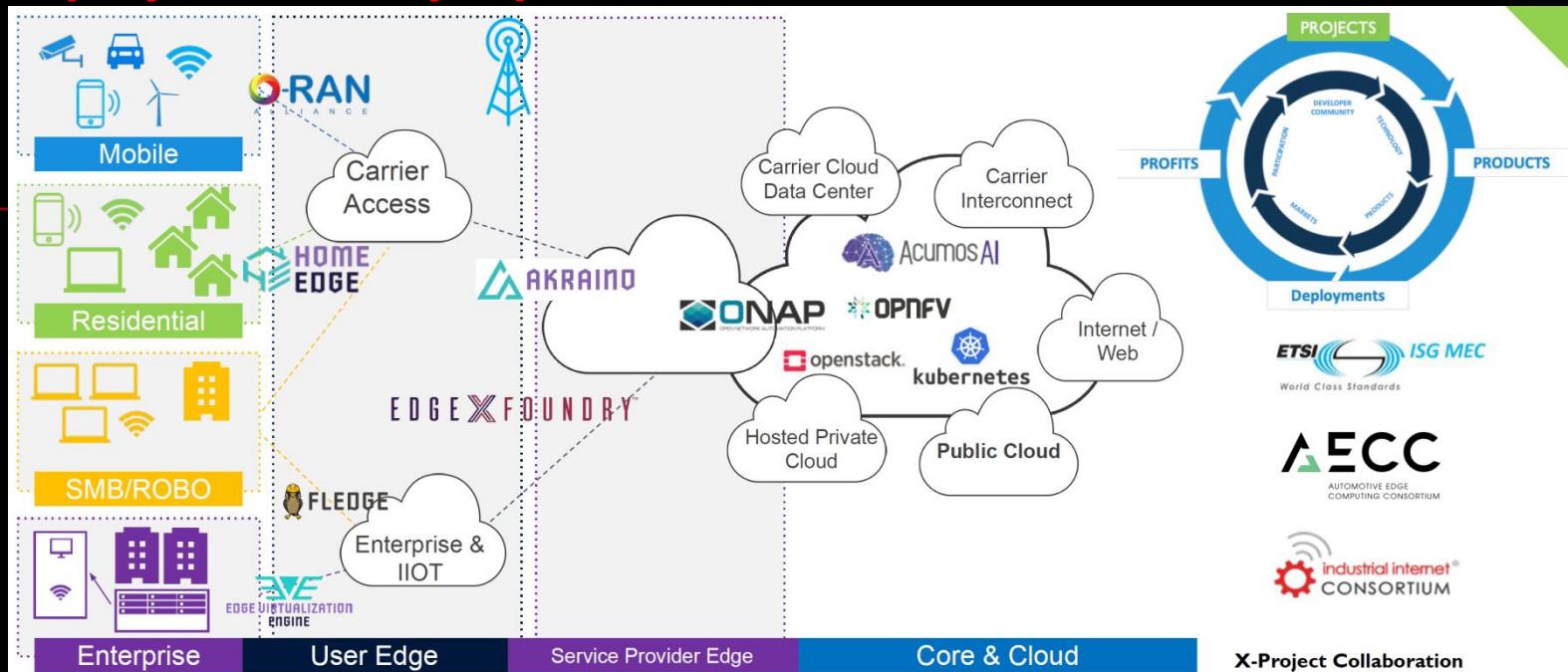


## ■ Overview



Source: <https://www.lfedge.org/wp-content/uploads/2021/04/LF-Edge-web-apr2021.pdf>

## ■ Deployment ready Open Source - use cases



Source: <https://wiki.lfedge.org/display/EVE/Slide+presentations>

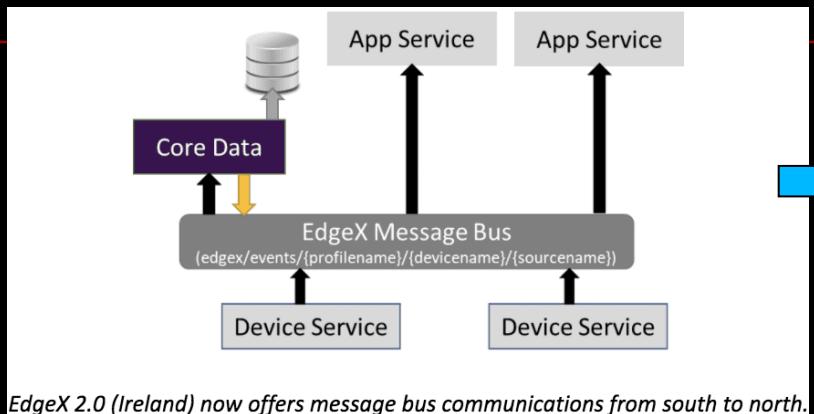
- For details, you may refer to my previous talk "**Kata Containers for Edge Computing**" at Kata Containers Meetup 2021(Shanghai) and the upcoming follow-ups.

# EdgeX 3.0

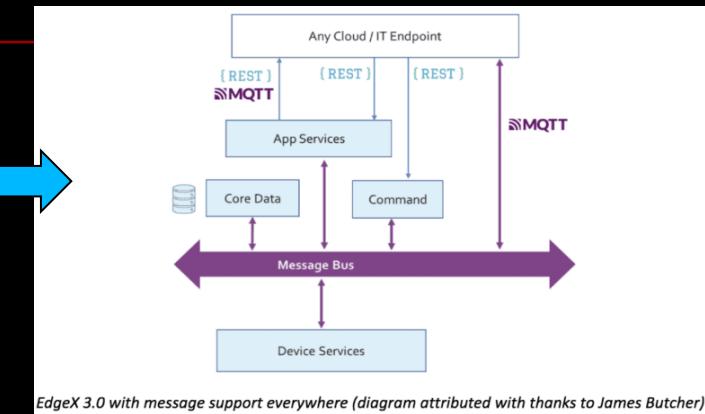
- <https://www.lfedge.org/2021/12/15/edgex-3-0-the-future-of-edge-computing/>



## Take the Bus but Allow Walking



*EdgeX 2.0 (Ireland) now offers message bus communications from south to north.*



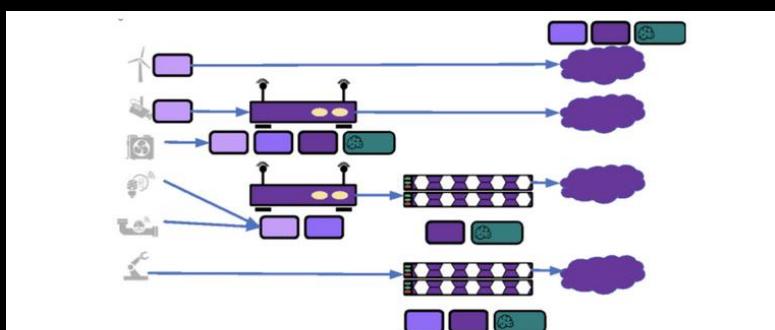
*EdgeX 3.0 with message support everywhere (diagram attributed with thanks to James Butcher)*

## Size Matters and Time is Relative

## Adopt Versus Build – Finding Edge Worthy Components

**Not Cloud Native – Aim for Edge Native**

**Distributable**



*EdgeX services should be allowed to run anywhere in the continuum of compute from thin edge to cloud.*



**Resiliency and Rapid Recovery**

**The Other Data – EdgeX Control Plane Telemetry and Health / Monitoring**

**Alternate Language App Functions SDK**

**Distributed Ledger Support**

**K8s Is Coming**

---

**UoM**

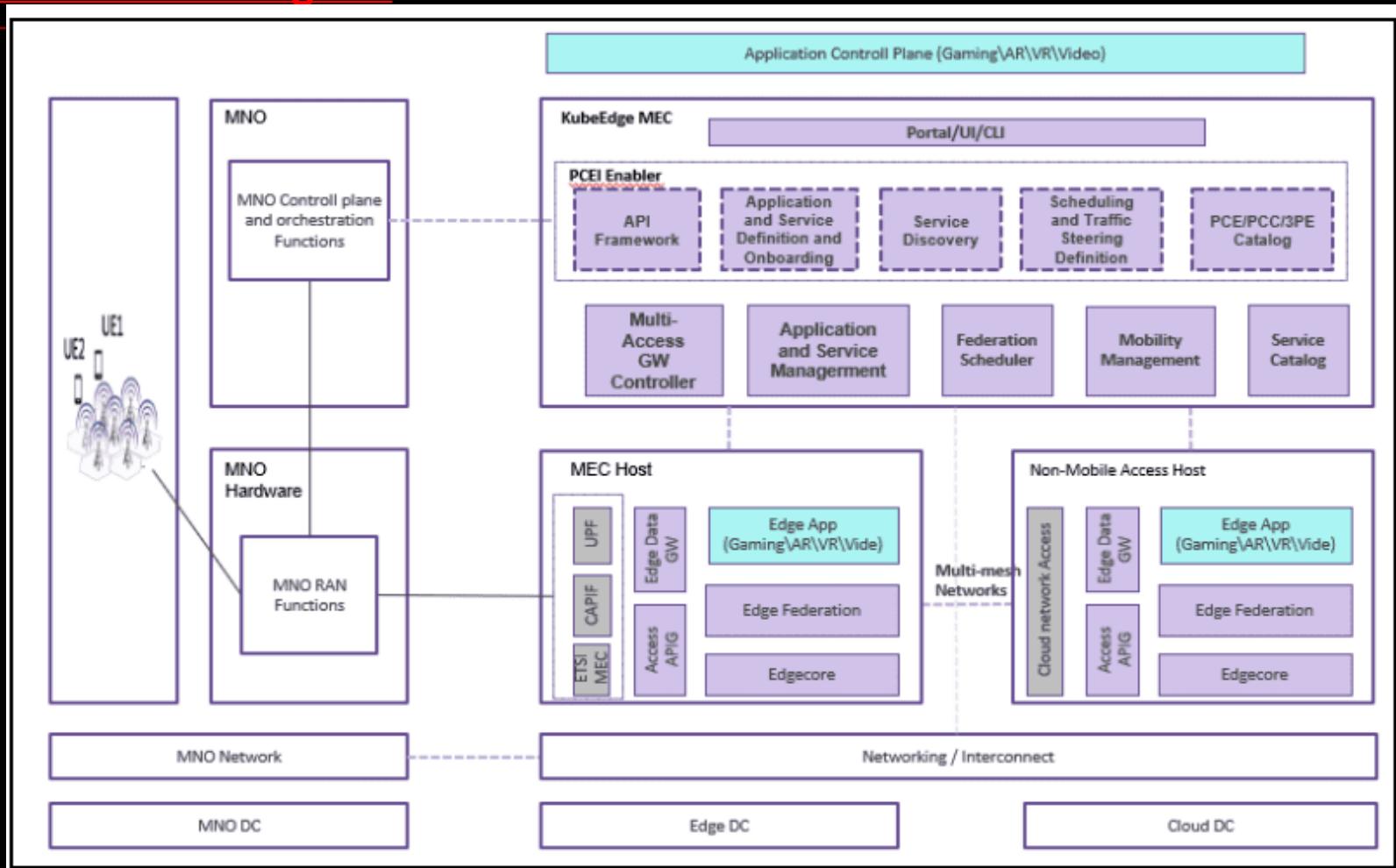
**Automate Thing Provisioning**

**...**



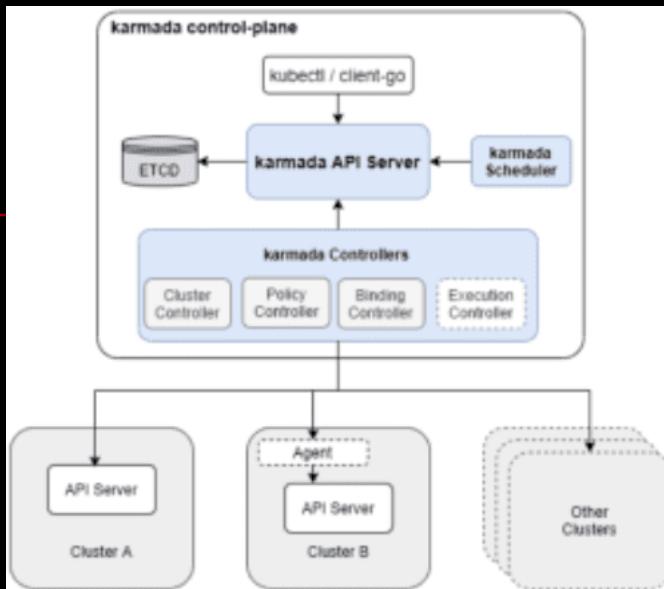
## Akraino's Federated MEC

- <https://www.lfedge.org/2022/01/05/introduction-akrainos-federated-multi-access-edge-cloud-platform-blueprint-in-r5/>
- **Functional Diagram**





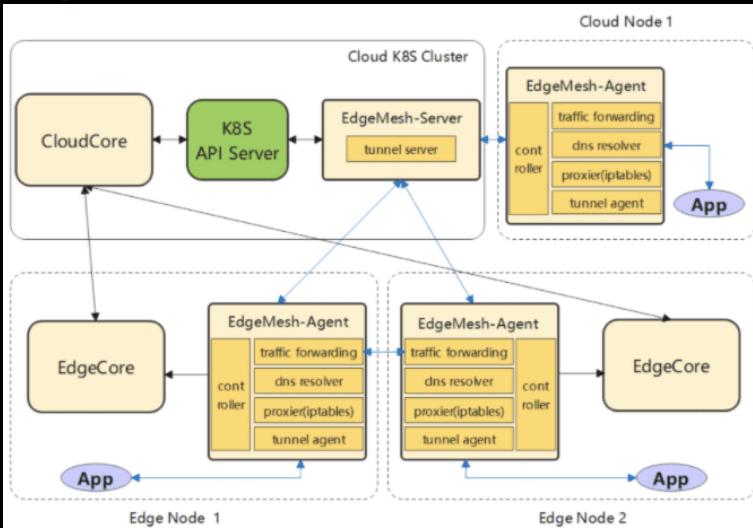
## Federation Scheduler



As a “Global Scheduler”, responsible for application QoS oriented global scheduling in accordance to the placement policies. Essentially, it refers to a decision-making capability that can decide how workloads should be spread across different clusters similar to how a human operator would. It maintains the resource utilization information for all the MEC edge cloud sites. Cloud federation functionality in our blueprint is enabled using open source Karmada project. The following is an architecture diagram for Karmada.

Karmada (Kubernetes® Armada) is a Kubernetes® management system that enables cloud-native applications to run across multiple Kubernetes® clusters and clouds with no changes to the underlying applications. By using Kubernetes®-native APIs and providing advanced scheduling capabilities, Karmada truly enables multi-cloud Kubernetes® environment. It aims to provide turnkey automation for multi-cluster application management in multi-cloud and hybrid cloud scenarios with key features such as centralized multi-cloud management, high availability, failure recovery, and traffic scheduling. More details related to Karmada project can be found [here](#).

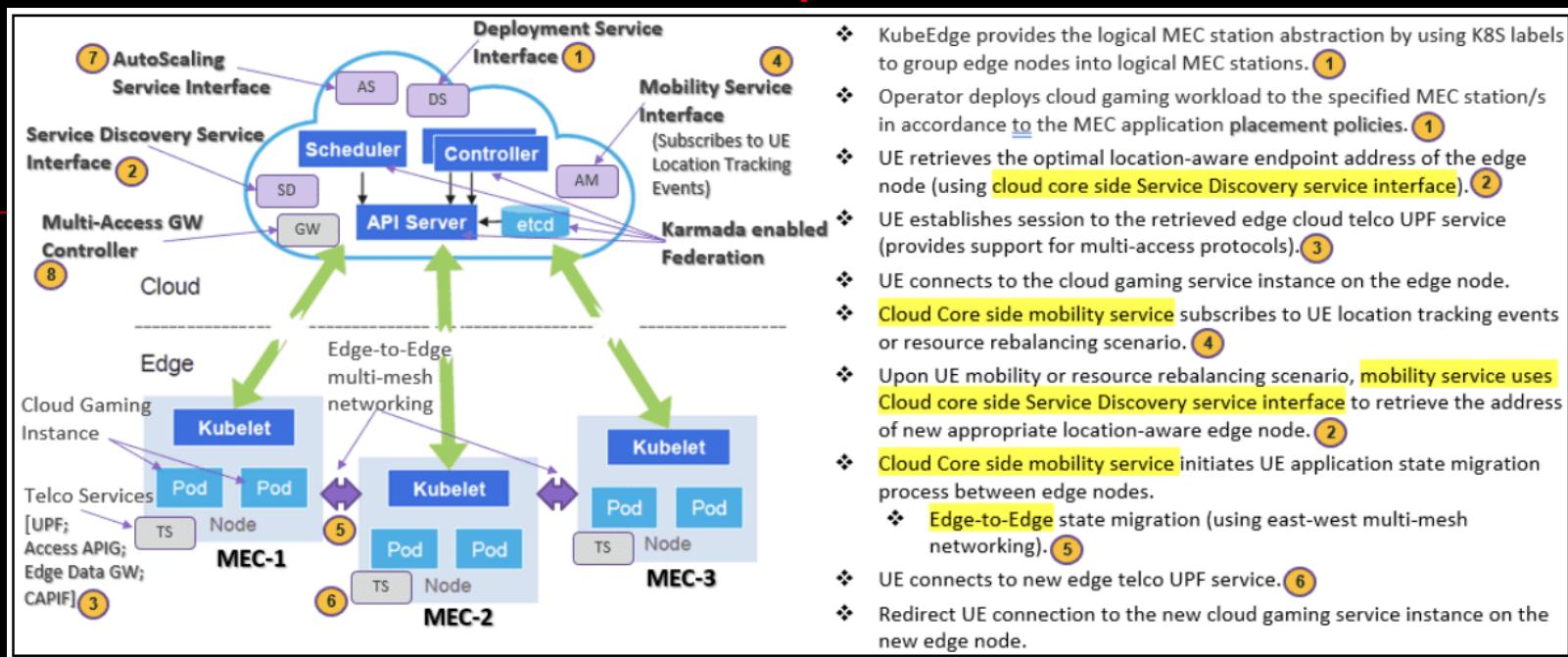
## EdgeMesh



EdgeMesh provides support for service mesh capabilities for the edge clouds in support of microservice communication cross cloud and edges. EdgeMesh provides a simple network solution for the inter-communications between services at edge scenarios (east-west communication).

The network topology for edge cloud computing scenario is quite complex. Various Edge nodes are often not interconnected and the direct inter-communication of traffic between applications on these edge nodes is highly desirable requirement for businesses. EdgeMesh addresses these challenges by shielding the complex network topology at the edge applications scenario. More details related to EdgeMesh project can be found [here](#).

## Detail Flow of various Architectural Components





## Good Resources

- <https://wiki.akraino.org/display/AK/Federated+Multi-Access+Edge+Cloud+Platform>
  - <https://www.digitaltwinconsortium.org/press-room/03-30-21.htm>
  - <https://www.lfedge.org/author/lfedge/>
  - ...
-



## 2.5 Serverless Architecture

- [https://en.wikipedia.org/wiki/Serverless\\_computing](https://en.wikipedia.org/wiki/Serverless_computing)

**Serverless computing** is a [cloud computing](#) execution model in which the cloud provider allocates machine resources on demand, taking care of the [servers](#) on behalf of their customers. Serverless computing does not hold resources in volatile memory; computing is rather done in short bursts with the results persisted to storage. When an app is not in use, there are no computing resources allocated to the app. Pricing is based on the actual amount of resources consumed by an application.<sup>[1]</sup> It can be a form of [utility computing](#). "Serverless" is a [misnomer](#) in the sense that servers are still used by cloud service providers to execute code for developers. However, developers of serverless applications are not concerned with [capacity planning](#), configuration, management, maintenance, fault tolerance, or scaling of containers, [VMs](#), or physical servers.

Serverless computing can simplify the process of [deploying code](#) into production. Serverless code can be used in conjunction with code deployed in traditional styles, such as [microservices](#) or [monoliths](#). Alternatively, applications can be written to be purely serverless and use no provisioned servers at all.<sup>[2]</sup> This should not be confused with computing or networking models that do not require an actual server to function, such as [peer-to-peer](#) (P2P).

### Serverless runtimes

Serverless vendors offer compute runtimes, also known as **Function as a Service (FaaS) platforms**, which execute application logic but do not store data. Common languages supported by serverless runtimes are Java, Python and PHP. Generally, the functions run under isolation boundaries, such as, Linux containers.

- [https://en.wikipedia.org/wiki/Function\\_as\\_a\\_service](https://en.wikipedia.org/wiki/Function_as_a_service)

### Pros

#### Cost [edit]

Serverless can be more cost-effective than renting or purchasing a fixed quantity of servers,<sup>[18]</sup> which generally involves significant periods of underutilization or idle time.<sup>[1]</sup> It can even be more cost-efficient than provisioning an [autoscaling group](#), due to more efficient [bin-packing](#) of the underlying machine resources.

This can be described as pay-as-you-go computing<sup>[18]</sup> or bare-code<sup>[18]</sup> as you are charged based solely upon the time and memory allocated to run your code; without associated fees for idle time.<sup>[18]</sup>

Immediate cost benefits are related to the lack of operating costs, including: licenses, installation, dependencies, and personnel cost for maintenance, support, or patching.<sup>[18]</sup> The lack of personnel cost is an advantage that applies broadly to cloud computing.

#### Elasticity versus scalability [edit]

*See also: [Scalability](#) and [Elasticity \(cloud computing\)](#)*

In addition, a serverless architecture means that developers and operators do not need to spend time setting up and tuning autoscaling policies or systems; the cloud provider is responsible for scaling the capacity to the demand.<sup>[1][12][18]</sup> As Google puts it: "from prototype to production to planet-scale."<sup>[18]</sup>

As cloud native systems inherently scale down as well as up, these systems are known as elastic rather than scalable.

Small teams of developers are able to run code themselves without the dependence upon teams of infrastructure and support engineers; more developers are becoming [DevOps](#) skilled and distinctions between being a software developer or hardware engineer are blurring.<sup>[18]</sup>

#### Productivity [edit]

With [function as a service](#), the units of code exposed to the outside world are simple event driven [functions](#). This means that typically, the programmer does not have to worry about [multithreading](#) or directly handling [HTTP](#) requests in their code, simplifying the task of back-end software development.



## Cons

### Performance [edit]

Infrequently-used serverless code may suffer from greater response latency than code that is continuously running on a dedicated server, virtual machine, or container. This is because, unlike with autoscaling, the cloud provider typically "spins down" the serverless code completely when not in use. This means that if the runtime (for example, the Java runtime) requires a significant amount of time to start up, it will create additional latency.<sup>[19]</sup>

### Resource limits [edit]

Serverless computing is not suited to some computing workloads, such as [high-performance computing](#), because of the resource limits imposed by cloud providers, and also because it would likely be cheaper to bulk-provision the number of servers believed to be required at any given point in time.<sup>[20]</sup>

### Monitoring and debugging [edit]

Diagnosing performance or excessive resource usage problems with serverless code may be more difficult than with traditional server code, because although entire functions can be timed,<sup>[2]</sup> there is typically no ability to dig into more detail by attaching profilers, debuggers or APM tools.<sup>[21]</sup> Furthermore, the environment in which the code runs is typically not [open source](#), so its performance characteristics cannot be precisely replicated in a local environment.

### Security [edit]

Serverless is sometimes mistakenly considered as more secure than traditional architectures. While this is true to some extent because OS vulnerabilities are taken care of by the cloud provider, the total attack surface is significantly larger as there are many more components to the application compared to traditional architectures and each component is an entry point to the serverless application. Moreover, the security solutions customers used to have to protect their cloud workloads become irrelevant as customers cannot control and install anything on the [endpoint](#) and [network](#) level such as an [intrusion detection/prevention system \(IDS/IPS\)](#).<sup>[22]</sup>

This is intensified by the mono-culture properties of the entire server network. (A single flaw can be applied globally.) According to Protego, the "solution to secure serverless apps is close partnership between developers, DevOps, and AppSec, also known as DevSecOps. Find the balance where developers don't own security, but they aren't absolved from responsibility either. Take steps to make it everyone's problem. Create cross-functional teams and work towards tight integration between security specialists and development teams. Collaborate so your organization can resolve security risks at the speed of serverless."<sup>[23]</sup>

### Privacy [edit]

Many serverless function environments are based on [proprietary](#) public cloud environments. Here, some [privacy](#) implications have to be considered, such as [shared resources](#) and access by external employees. However, serverless computing can also be done on private cloud environment or even on-premises, using for example the Kubernetes platform. This gives companies full control over privacy mechanisms, just as with hosting in traditional server setups.

### Standards [edit]

Serverless computing is covered by [International Data Center Authority](#) (IDCA) in their Framework AE360.<sup>[24]</sup> However, the part related to portability can be an issue when moving business logic from one public cloud to another for which the [Docker](#) solution was created. [Cloud Native Computing Foundation](#) (CNCF) is also working on developing a specification with Oracle.<sup>[25]</sup>

### Vendor lock-in [edit]

Serverless computing is provided as a third-party service. Applications and software that run in the serverless environment are by default locked to a specific cloud vendor.<sup>[26]</sup> Therefore, serverless can cause multiple issues during migration.<sup>[27]</sup>

## ■ Serverless for Edge

**For details, you may refer to my previous talk "AOT compilation based Wasm compiler and runtime for Serverless Edge computing" at OpenInfra Days China 2021(Beijing) and upcoming follow-ups.**

...



## 2.6 Programming Languages

### 2.6.1 Rust

- [https://en.wikipedia.org/wiki/Rust\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Rust_(programming_language))

Rust is a multi-paradigm, high-level, general-purpose programming language designed for performance and safety, especially safe concurrency.<sup>[12][13]</sup> Rust is syntactically similar to C++,<sup>[14]</sup> but can guarantee memory safety by using a *borrow checker* to validate references.<sup>[15]</sup> Rust achieves memory safety without garbage collection, and reference counting is optional.<sup>[16][17]</sup>

Rust was originally designed by Graydon Hoare at Mozilla Research, with contributions from Dave Herman, Brendan Eich, and others.<sup>[18][19]</sup> The designers refined the language while writing the Servo layout or browser engine,<sup>[20]</sup> and the Rust compiler. It has gained increasing use in industry, and Microsoft has been experimenting with the language for secure and safety-critical software components.<sup>[21][22]</sup>

Rust has been voted the "most loved programming language" in the Stack Overflow Developer Survey every year since 2016.<sup>[23]</sup>

- <https://www.rust-lang.org/>

#### Why Rust?

##### Performance

Rust is blazingly fast and memory-efficient: with no runtime or garbage collector, it can power performance-critical services, run on embedded devices, and easily integrate with other languages.

##### Reliability

Rust's rich type system and ownership model guarantee memory-safety and thread-safety — enabling you to eliminate many classes of bugs at compile-time.

##### Productivity

Rust has great documentation, a friendly compiler with useful error messages, and top-notch tooling — an integrated package manager and build tool, smart multi-editor support with auto-completion and type inspections, an auto-formatter, and more.

#### Build it in Rust

In 2018, the Rust community decided to improve programming experience for a few distinct domains (see the 2018 roadmap). For these, you can find many high-quality crates and some awesome guides on how to get started.



##### Command Line

Whip up a CLI tool quickly with Rust's robust ecosystem. Rust helps you maintain your app with confidence and distribute it with ease.



##### WebAssembly

Use Rust to supercharge your JavaScript, one module at a time. Publish to npm, bundle with webpack, and you're off to the races.



##### Networking

Predictable performance. Tiny resource footprint. Rock-solid reliability. Rust is great for network services.



##### Embedded

Targeting low-resource devices? Need low-level control without giving up high-level conveniences? Rust has you covered.

BUILDING TOOLS

WRITING WEB APPS

WORKING ON SERVERS

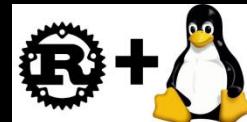
STARTING WITH  
EMBEDDED

- <https://www.memoriesafety.org/>

...

## Rust heads into Linux Kernel

### ■ What's happening!



<https://lwn.net/Articles/853423/> //Rust heads into the kernel?

<https://lwn.net/Articles/852704/> //Rust in the Linux kernel

<https://lwn.net/Articles/849849/> //Rust support hits linux-next

<https://lwn.net/Articles/829858/> //Supporting Linux kernel development in Rust

...

<https://www.zdnet.com/article/rust-in-the-linux-kernel-why-it-matters-and-whats-happening-next/>

<https://blogs.gartner.com/manjunath-bhat/2021/01/03/why-2021-will-be-a-rusty-year-for-system-programmers/>

<https://developers.slashdot.org/story/21/04/17/009241/linus-torvalds-says-rust-closer-for-linux-kernel-development-calls-c-a-crap-language>

<https://www.infoq.com/news/2021/04/rust-linux-kernel-development/>

<https://itwire.com/open-source/rust-support-in-linux-may-be-possible-by-5-14-release-torvalds.html>

<https://thenewstack.io/rust-in-the-linux-kernel-good-enough/>

...



## ■ <https://github.com/Rust-for-Linux>

The goal of this project is to add support for the Rust language to the Linux kernel. This repository contains the work that will be eventually submitted for review to the LKML.

Feel free to [contribute!](#) To start, take a look at [Documentation/rust](#).

## ■ [Rust for Linux patch 1](#)

<https://lore.kernel.org/lkml/20210704202756.29107-1-ojeda@kernel.org/>

141 files changed, 33003 insertions(+), 45 deletions(-)

## ■ [Rust for Linux patch 2](#)

<https://lore.kernel.org/lkml/20211206140313.5653-1-ojeda@kernel.org/>

156 files changed, 32369 insertions(+), 50 deletions(-)

<https://www.infoq.com/news/2021/12/rust-for-linux-patch-2/>

With the new submission, Rust for Linux moved to using the latest stable release of the Rust compiler, version 1.57.0. The objective here is not relying eventually on any [unstable language features](#) and being able to declare a minimum Rust version required for kernel development. The new infrastructure also includes a number of new diagnostics and clippy lints to ensure stricter checks.

## ■ [Rust for Linux patch 3](#)

<https://lore.kernel.org/lkml/20220117053349.6804-1-ojeda@kernel.org/>

162 files changed, 33506 insertions(+), 58 deletions(-)

## ■ [Rust for Linux patch 4](#)

<https://lore.kernel.org/lkml/20220212130410.6901-1-ojeda@kernel.org/>

162 files changed, 34341 insertions(+), 57 deletions(-)



## ■ Rust for Linux patch 5

<https://lore.kernel.org/lkml/20220317181032.15436-1-ojeda@kernel.org/>

168 files changed, 35290 insertions(+), 63 deletions(-)

With the v5 patches there has been more infrastructure improvements and continuing to refine the integration and sample Rust code for the kernel. Some of the Rust for Linux v5 changes include:

- The toolchain and alloc are upgraded against Rust 1.59.
- Added support for host programs written in Rust.
- A new "HAVE\_RUST" kernel option that is to be set by architectures supporting Rust.
- The Rust for Linux kernel abstractions have added a new abstraction interface for covering the kernel's Hardware Random Number Generator (HWRNG).



## ***GCC for Rust***

- <https://rust-gcc.github.io/>

### **GCC Front-End For Rust**

This is a full alternative implementation of the Rust language on top of GCC with the goal to become fully upstream with the GNU toolchain.

As this is a front-end project, the compiler will gain full access to all of GCC's internal middle-end optimization passes which are distinct from LLVM. For example, users of this compiler can expect to use the familiar -O2 flags to tune GCC's optimizer. Going forward, we will be happy to see more LLVM vs GCC graphs in respect to compilation speed, resulting code size and performance.

The project is still in an early phase with the goal to compile the official Rust test suite. There are no immediate plans for a borrow checker as this is not required to compile rust code and is the last pass in the RustC compiler.

This can be handled as a separate project when we get to that point.

<https://github.com/Rust-GCC>

- [https://github.com/rust-lang/rustc\\_codegen\\_gcc](https://github.com/rust-lang/rustc_codegen_gcc)

This is a GCC codegen for rustc, which means it can be loaded by the existing rustc frontend, but benefits from GCC: more architectures are supported and GCC's optimizations are used.

Despite its name, libgccjit can be used for ahead-of-time compilation, as is used here.

**was merged into upstream Rust**

[https://blog.antoyo.xyz/rustc\\_codegen\\_gcc-progress-report-10](https://blog.antoyo.xyz/rustc_codegen_gcc-progress-report-10)

■ ...



# Rust for Cloud Native

■ <https://rust-cloud-native.github.io/>

## Applications and Services

- [apache/incubator-teaclave](#): open source universal secure computing platform, making computation on privacy-sensitive data safe and simple
- [bottlerocket-os/bottlerocket](#): an operating system designed for hosting containers
- [containers/krunvmm](#): manage lightweight VMs created from OCI images
- [containers/youki](#): a container runtime written in Rust
- [datafuselabs/datafuse](#): A Modern Real-Time Data Processing & Analytics DBMS with Cloud-Native Architecture, built to make the Data Cloud easy
- [firecracker-microvm/firecracker](#): secure and fast microVMs for serverless computing
- [infinyon/fluvio](#): Cloud-native real-time data streaming platform with in-line computation capabilities
- [krustlet/krustlet](#): Kubernetes Rust Kubelet
- [kube-rs/controller-rs](#): a Kubernetes example controller
- [kube-rs/version-rs](#): example Kubernetes reflector and web server
- [kubewarden/policy-server](#): webhook server that evaluates WebAssembly policies to validate Kubernetes requests
- [linkerd/linkerd2-proxy](#): a purpose-built proxy for the Linkerd service mesh
- [openebs/mayastor](#): A cloud native declarative data plane in containers for containers
- [rancher-sandbox/lockc](#): eBPF-based MAC security audit for container workloads
- [tikv/tikv](#): distributed transactional key-value database
- [tremor-rs/tremor-runtime](#): an event processing system that supports complex workflows such as aggregation, rollups, an ETL language, and a query language
- [valeriansaliou/sonic](#): fast, lightweight & schema-less search backend
- [WasmEdge/WasmEdge](#): WasmEdge is a high-performance WebAssembly (Wasm) Virtual Machine (VM) runtime, which enables serverless functions to be embedded into any software platform; from cloud's edge to SaaS to automobiles

## Libraries

- [CNI Plugins](#): crate/framework to write CNI (container networking) plugins in Rust (includes a few custom plugins as well)
- [containers/libkrun](#): a dynamic library providing Virtualization-based process isolation capabilities
- [kube-rs/kube-rs](#): Kubernetes Rust client and async controller runtime
- [qovery/engine](#): Qovery Engine is an open-source abstraction layer library that turns easy app deployment on AWS, GCP, Azure, and other Cloud providers in just a few minutes
- [open-telemetry/opentelemetry-rust](#): OpenTelemetry is a set of APIs, SDKs, tooling and integrations that are designed for the creation and management of telemetry data such as traces, metrics, and logs.

...



## Rust for Embedded

- <https://blog.rust-embedded.org>this-year-in-embedded-rust-2021/>
- <https://github.com/rust-embedded/awesome-embedded-rust/>
- <https://docs.rust-embedded.org/book/>
- <https://github.com/rust-embedded/rust-raspberrypi-OS-tutorials/>
- ...



## ***Rust Compiler Ambitions for 2022***

- <https://blog.rust-lang.org/inside-rust/2022/02/22/compiler-team-ambitions-2022.html>

### **Work Items**

<b>Category</b>	<u>Concrete Initiatives</u>	<u>Aspirations</u>
I-unsound (🦀)	<a href="#">Initiatives</a>	
Async Rust (🦀, 🚧)	<a href="#">Initiatives</a>	
Debugging (🦀, 📈)	<a href="#">Initiatives</a>	<a href="#">Aspirations</a>
Faster Builds (cargo, 🚧)	<a href="#">Initiatives</a>	<a href="#">Aspirations</a>
Expressiveness (cargo, 🦀)	<a href="#">Initiatives</a>	<a href="#">Aspirations</a>
Librarification (🔗)	<a href="#">Initiatives</a>	<a href="#">Aspirations</a>
P-high Backlog (🦀)		<a href="#">Aspirations</a>
Team Operations (🔗)		<a href="#">Aspirations</a>
Backend (🔗, 🚧)		<a href="#">Aspirations</a>
Diagnostics (cargo)		<a href="#">Aspirations</a>



## 2.6.1.1 Rust for Edge Computing

- For details, please look forward to our new talk "Rust for Edge Computing"...
-



## 2.7 Toolchain

### 2.7.1 LLVM

#### LLVM 14

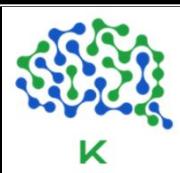
- LLVM and Clang have support for the Armv9-A architecture, including Armv9.1-A and Armv9.2-A and [Armv9.3-A](#).  
Clang also adds support for the Arm Cortex X2, A710, and A510 processors.
- Another Arm change is Clang now honors the "-mtune" flag on AArch64 to tune code generation to a particular CPU without setting any specific architectural features.
- Support for [AVX512-FP16](#) instructions is added to LLVM for new Intel server CPUs.
- Clang supports a wide range of additional SiFive RISC-V processors from the SiFive E20 through the SiFive S76 with new "-mcpu=" targets.
- Clang has more additions in preparation for C23.
- Clang now supports NVIDIA CUDA versions up to v11.5 and also bumped the default GPU architecture target to sm-35.
- Clang now uses [DWARFv5](#) as the default debug format where supported rather than DWARFv4.
- When building Clang it can now be configured that -fPIE and -pie are used by default on Linux to match the behavior of GCC.
- Clangd now provides inlay hints by default as textual hints interleaved with the code for its Language Server Protocol handling for integration with integrated development environments, etc. There are also numerous other Clangd improvements, including better code completion and more.
- Libc++ now has support for the C++20 co-routines, the C++20 format header, and other C++20 and early C++2b work.
- Facebook's BOLT was merged for optimizing the layout of generated binaries.
- [HIPSPV](#) for AMD HIP to SPIR-V landed and continues to be worked on.
- Continued performance optimizations.

Source: [https://www.phoronix.com/scan.php?page=news\\_item&px=LLVM-14.0-Released](https://www.phoronix.com/scan.php?page=news_item&px=LLVM-14.0-Released)

## 2.7.2 GCC

### GCC 12

- ...
- 





## 2.7.3 Linker

### 2.7.3.1 mold

- <https://github.com/rui314/mold>

#### A Modern Linker.

mold is a faster drop-in replacement for existing Unix linkers. It is several times faster than the LLVM lld linker, the second-fastest open-source linker which I originally created a few years ago. mold is designed to increase developer productivity by reducing build time, especially in rapid debug-edit-rebuild cycles.

Here is a performance comparison of GNU gold, LLVM lld, and mold for linking final debuginfo-enabled executables of major large programs on a simulated 8-core 16-threads machine.

Program (linker output size)	GNU gold	LLVM lld	mold
Chrome 96 (1.89 GiB)	53.86s	11.74s	2.21s
Clang 13 (3.18 GiB)	64.12s	5.82s	2.90s
Firefox 89 libxul (1.64 GiB)	32.95s	6.80s	1.42s

- Currently supports x86-64, i386, ARM64 and 64-bit RISC-V.
- Languages



- <https://github.com/rui314/mold/blob/main/docs/design.md>
- ...



## 2.7.4 Utilities

### 2.7.4.1 uutils

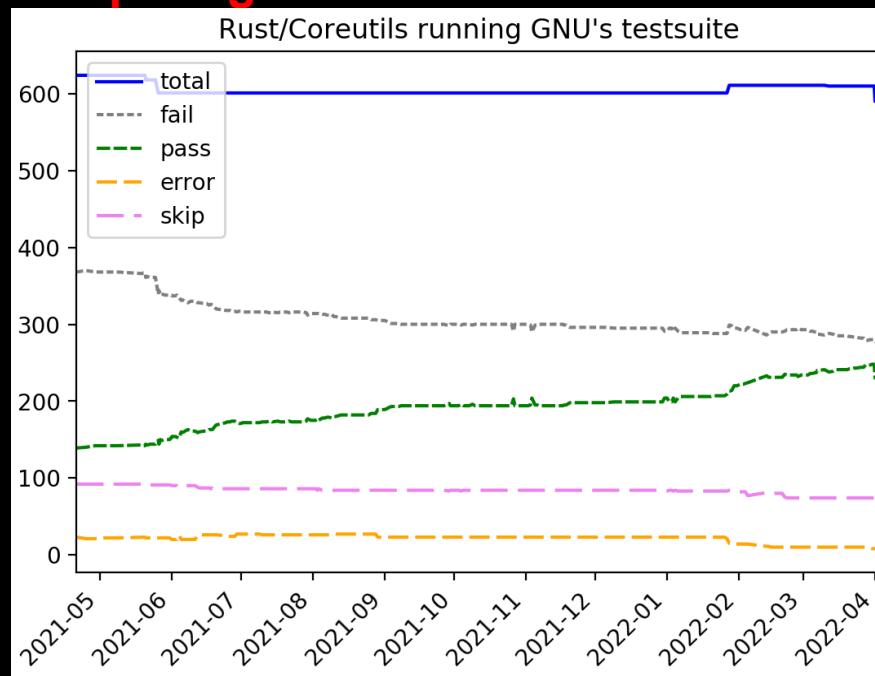
- <https://github.com/uutils/coreutils>

**Cross-platform Rust rewrite of the GNU coreutils.**

- **Why is it?**

uutils aims to work on as many platforms as possible, to be able to use the same utils on Linux, Mac, Windows and other platforms. This ensures, for example, that scripts can be easily transferred between platforms. Rust was chosen not only because it is fast and safe, but is also excellent for writing cross-platform code.

- **Comparing with GNU**

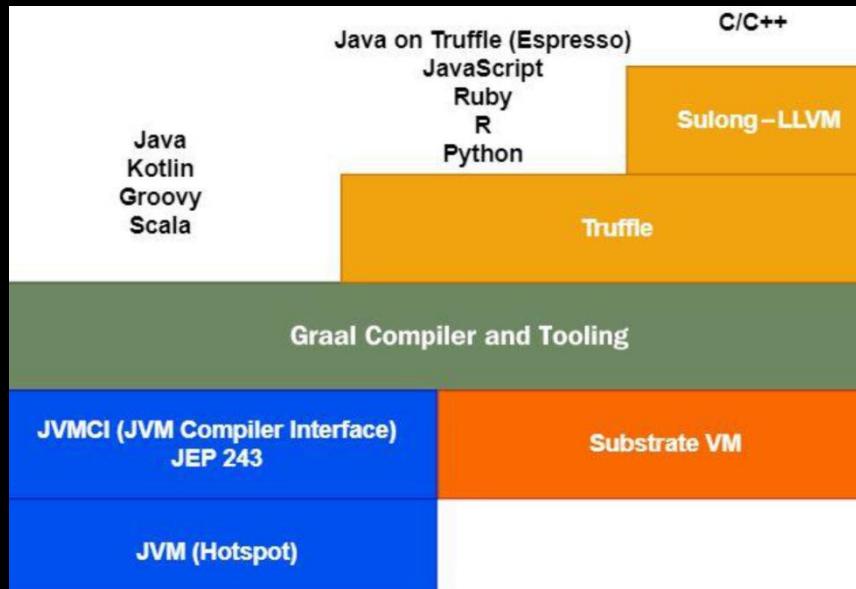




## 2.8 Polyglot Runtimes

### 2.8.1 GraalVM

- <https://en.wikipedia.org/wiki/GraalVM>
- <https://www.graalvm.org/>



Source: [https://static.packt-cdn.com/downloads/9781800564909\\_ColorImages.pdf](https://static.packt-cdn.com/downloads/9781800564909_ColorImages.pdf)

- **A Universal High-Performance Polyglot VM**
- **A meta-runtime for Language-Level Virtualization**
- **Base on OpenJDK 8, 11, 16, and 17 with JVMCI support.**
- <https://github.com/graalvm>
- <https://github.com/oracle/graal>

# Editions

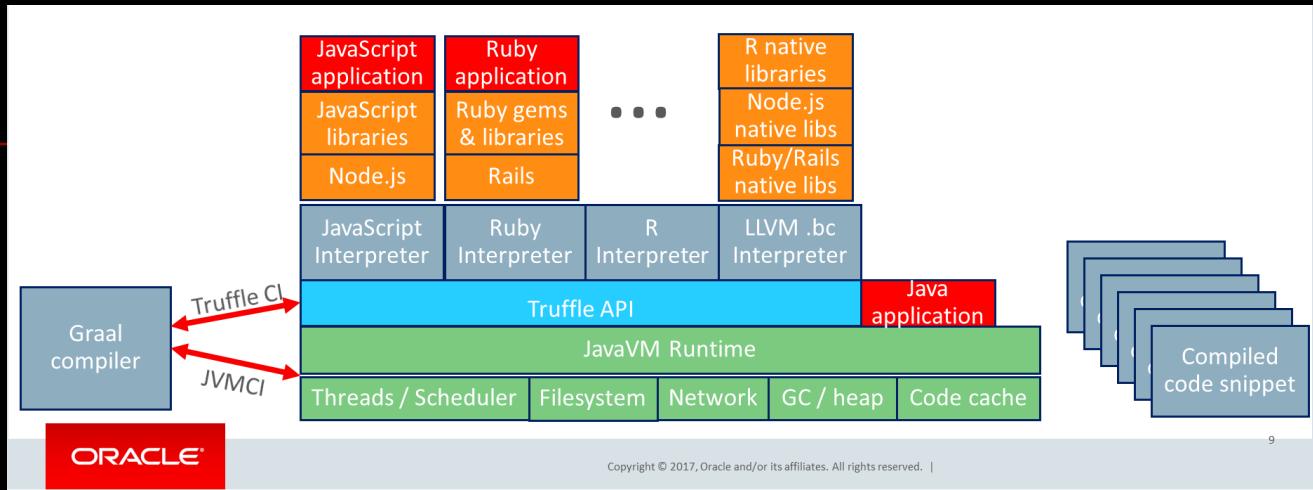


	GraalVM Community	GraalVM Enterprise
License	GNU General Public License V2 with the "Classpath" Exception	Oracle Technology Network (OTN) license for dev/test; Commercial license for production deployments
Base JDKs	OpenJDK 11.0.13 and 17.0.1	Oracle JDK 8u311, 11.0.13 and 17.0.1
Support	Community support via <a href="#">public channels</a>	Global 24x7 Enterprise support from <a href="#">Oracle</a>
Speedup vs. OpenJDK on <a href="#">Renaissance Suite</a>	1.04x	1.3x
Native Image performance vs. OpenJDK (OPs per GB/sec)	0.82x	1.34x
Docker Container Images	✓	✓
Patented advanced compiler optimizations		✓
Compressed pointers for low memory usage (Native Image)		✓
Profile guided optimization for improved performance (Native Image)		✓
G1 garbage collection for low latency (Native Image)		✓

Source: <https://www.graalvm.org/downloads/>

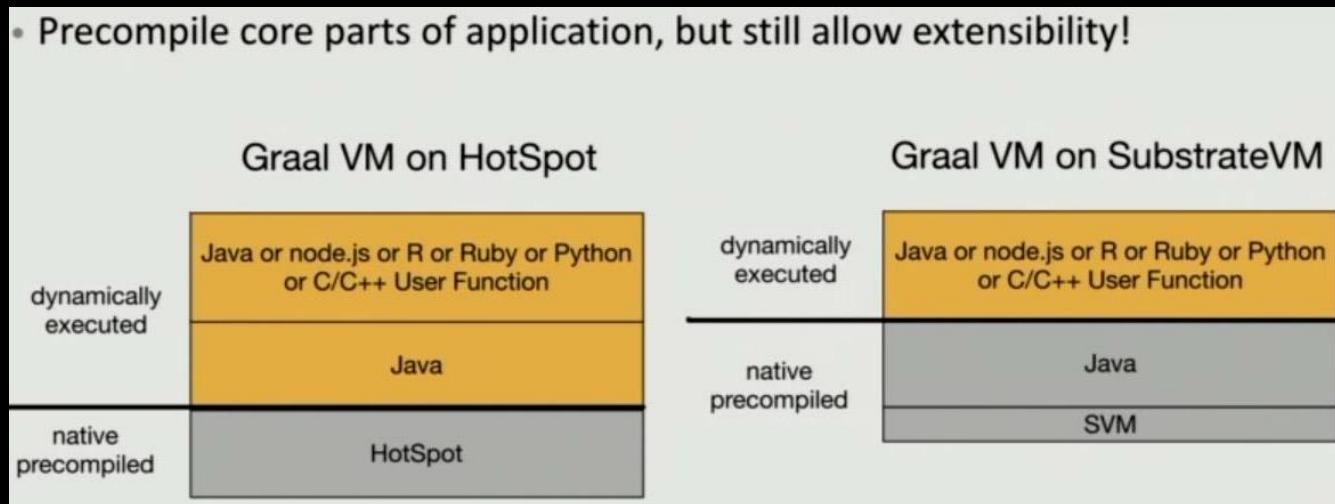
# Architecture

## ■ A hybrid of static & dynamic runtimes



Source: <https://ics.psu.edu/wp-content/uploads/2017/02/GraalVM-PSU.pptx>

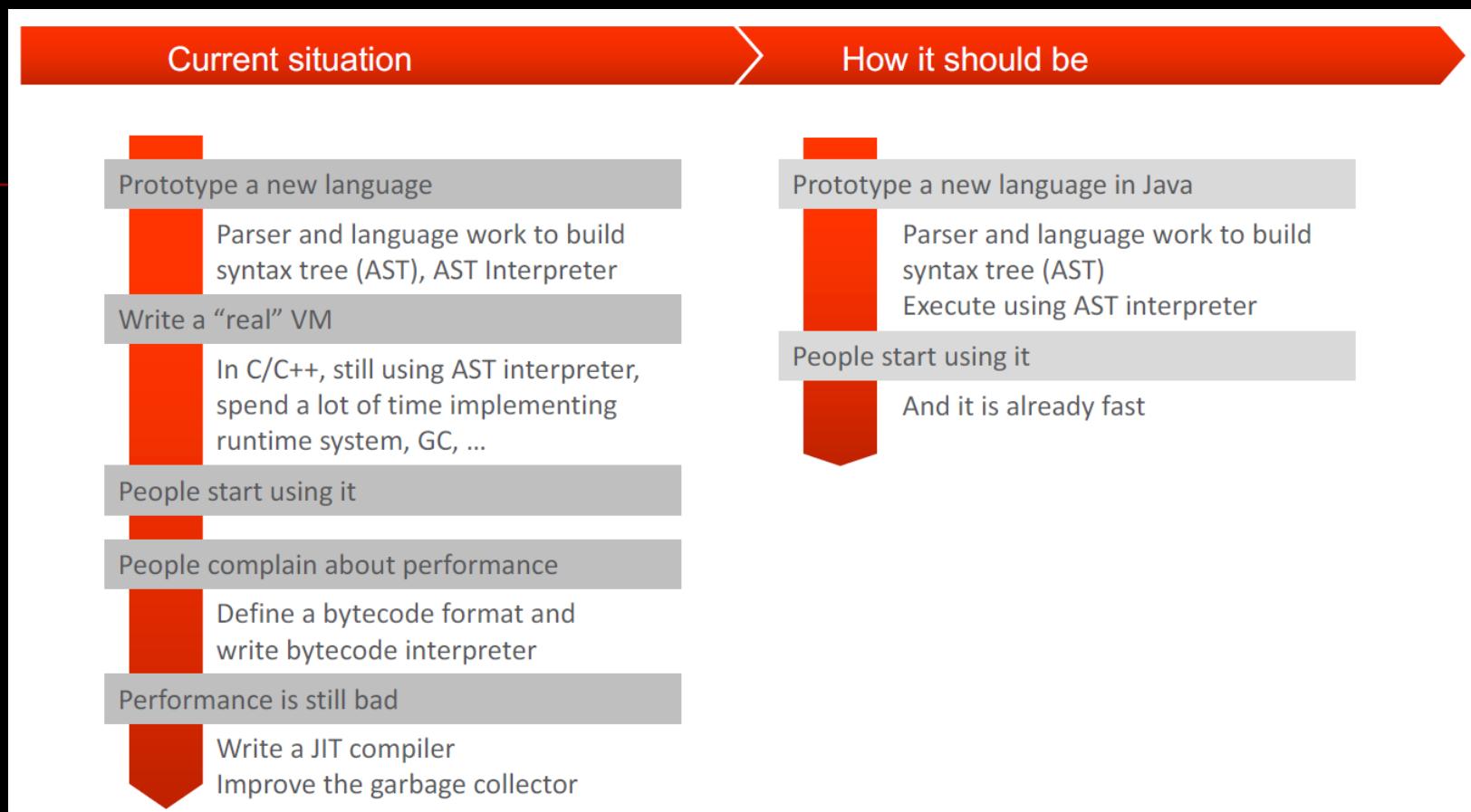
- Precompile core parts of application, but still allow extensibility!



Source: “Adopting Java for the Serverless world”, Vadym Kazulkin, JUG London 2020.



## Implement a new language runtime



Source: “Turning the JVM into a Polyglot VM with Graal”, Chris Seaton, Oracle Labs



# Languages

- <https://github.com/oracle/graal/blob/master/truffle/docs/Languages.md>

## Language Implementations

This page is intended to keep track of the growing number of language implementations and experiments on top of Truffle. The following language implementations exist already:

- Espresso, a meta-circular Java bytecode interpreter. \*
- FastR, an implementation of GNU R. \*
- Graal.js, an ECMAScript 2020 compliant JavaScript implementation. \*
- Graal.Python, an early-stage implementation of Python. \*
- grICUDA, a polyglot CUDA integration.
- SimpleLanguage, a toy language implementation to demonstrate Truffle features.
- SOMNs, a Newspeak implementation for Concurrency Research.
- Sulong, an LLVM bitcode interpreter. \*
- TRegex, a generic regular expression engine (internal, for use by other languages only). \*
- TruffleRuby, an implementation of Ruby. \*
- TruffleSOM, a SOM Smalltalk implementation.
- TruffleSqueak, a Squeak/Smalltalk VM implementation and polyglot programming environment.
- Yona, the reference implementation of a minimalistic, strongly and dynamically-typed, parallel and non-blocking, polyglot, strict, functional programming language.
- Enso, an open source, visual language for data science that lets you design, prototype and develop any application by connecting visual elements together.

\* Shipped as part of GraalVM.

## Experiments

- bf, an experimental Brainfuck programming language implementation.
- brainfuck-jvm, another Brainfuck language implementation.
- Cover, a Safe Subset of C++.
- DynSem, a DSL for declarative specification of dynamic semantics of languages.
- Heap Language, a tutorial showing the embedding of Truffle languages via interoperability.
- hextruffe, an implementation of Hex.
- LuaTruffle, an implementation of the Lua language.
- Mozart-Graal, an implementation of the Oz programming language.
- Mumbler, an experimental Lisp programming language.
- PorcE, an Orc language implementation.
- ProloGraal a Prolog language implementation supporting interoperability.
- PureScript, a small, strongly-typed programming language.
- Reactive Ruby, TruffleRuby meets Reactive Programming.
- shen-truffle, a port of the Shen programming language.
- TruffleMATE, a Smalltalk with a completely reified runtime system.
- TrufflePascal, a Pascal interpreter.
- ZipPy, a Python implementation.

...



## Good Resources

- <https://www.oracle.com/sg/java/graalvm/>
- <https://medium.com/graalvm>
- <https://blog.frankel.ch/start-rust/7/>
- ...



## 2.8.1.1 GraalVM for Cloud Native

- <https://quarkus.io/>

A Kubernetes Native Java stack tailored for OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards.

■ Amazingly fast boot time, incredibly low RSS memory!

```
$ ./my-native-java-rest-app  
Quarks started in 0.008s
```



- <https://github.com/quarkusio/quarkus>

Quarkus is a Cloud Native, (Linux) Container First framework for writing Java applications.

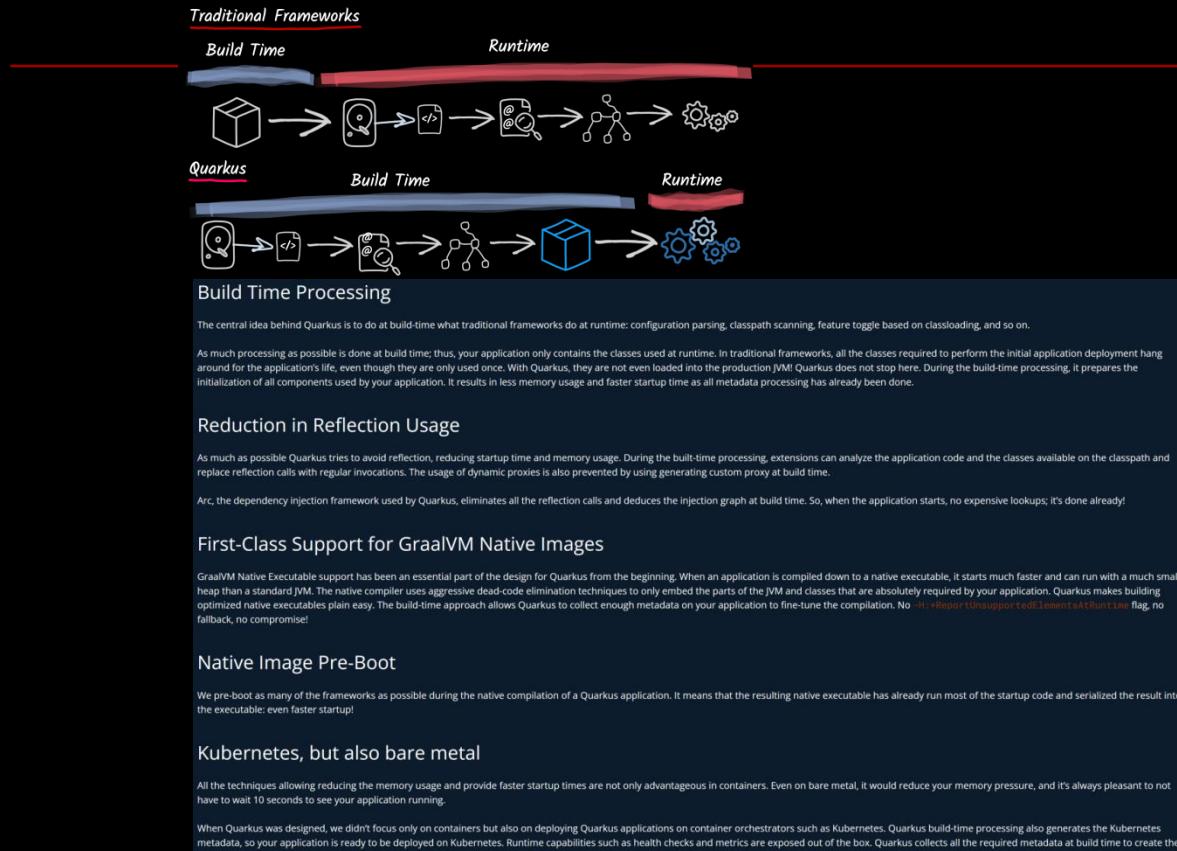
- Container First: Minimal footprint Java applications optimal for running in containers.
- Cloud Native: Embraces [12 factor architecture](#) in environments like Kubernetes.
- Unify imperative and reactive: Brings under one programming model non-blocking and imperative styles of development.
- Standards-based: Based on the standards and frameworks you love and use (RESTEasy and JAX-RS, Hibernate ORM and JPA, Netty, Eclipse Vert.x, Eclipse MicroProfile, Apache Camel...).
- Microservice First: Brings lightning fast startup time and code turn around to Java apps.
- Developer Joy: Development centric experience without compromise to bring your amazing apps to life in no time.

All under ONE framework.



## ■ <https://quarkus.io/vision/container-first>

**From the outset, Quarkus has been designed around a container-first philosophy. What this means in concrete terms is that Quarkus applications are optimised for low memory usage and fast startup times in the following ways:**



## ■ Spring Framework & Boot, Micronaut, and more.

<https://www.graalvm-on-lambda.com/>

...



## Mandrel

### ■ <https://github.com/graalvm/mandrel>

Mandrel is a downstream distribution of the GraalVM community edition. Mandrel's main goal is to provide a native-image release specifically to support Quarkus. The aim is to align the native-image capabilities from GraalVM with OpenJDK and Red Hat Enterprise Linux libraries to improve maintainability for native Quarkus applications. Mandrel can best be described as a distribution of a regular OpenJDK with a specially packaged GraalVM Native Image builder (native-image).

#### How Does Mandrel Differ From Graal

Mandrel releases are built from a code base derived from the upstream GraalVM code base, with only minor changes but some significant exclusions. A full distribution of GraalVM is much more than native-image: it has polyglot support, the Truffle framework which allows for efficient implementation of interpreters, an LLVM compiler back end for native image, the libgraal JIT compiler as a replacement for Hotspot's C2 server compiler and much more. Mandrel is the small subset of that functionality we support for the native-image use-case.

Mandrel's native-image also doesn't include the following features:

- The experimental image-build server, i.e., the --experimental-build-server option.
- The LLVM backend, i.e., the -H:CompilerBackend=llvm option.
- The musl libc implementation, i.e., the --libc=musl option.
- Support for generating static native images, i.e., the --static option.
- Support for non JVM-based languages and polyglot, i.e., the --language:<languageId> option.

Mandrel is also built slightly differently to GraalVM, using the standard OpenJDK project release of jdk11u. This means it does not profit from a few small enhancements that Oracle have added to the version of OpenJDK used to build their own GraalVM downloads. Most of these enhancements are to the JVMI module that allows the Graal compiler to be run inside OpenJDK. The others are small cosmetic changes to behaviour. These enhancements may in some cases cause minor differences in the progress of native image generation. They should not cause the resulting images themselves to execute in a noticeably different manner.

...



## 2.8.1.2 GraalVM 22.x

### 22.0

- <https://medium.com/graalvm/graalvm-22-0-is-here-c7acc82a8c2e>
- [https://www.phoronix.com/scan.php?page=news\\_item&px=GraalVM-22.0-Released](https://www.phoronix.com/scan.php?page=news_item&px=GraalVM-22.0-Released)

### Highlights

- Java 8 support has been removed. GraalVM 22.0 is only targeting JDK 11 and JDK 17, with JDK 12/13/14/15/16 support also being removed.
- A new optimization to improve the performance in Native Image of a type switch. There are also Native Image updates to reduce the image size.
- Native Image has improved support for the Java Platform Module System.
- Various new Java compiler optimizations albeit limited to GraalVM Enterprise.
- ECMAScript 2022 features are now enabled by default for GraalVM's JavaScript support.
- GraalVM's LLVM Runtime has switched to the Truffle Fame API, has new optimizations, and other fixes.
- Various improvements to GraalVM's WebAssembly implementation.

- ...



## 2.8.1.3 Java Roadmap

18

- <https://openjdk.java.net/projects/jdk/18/>

### Features

- 400: UTF-8 by Default
- 408: Simple Web Server
- 413: Code Snippets in Java API Documentation
- 416: Reimplement Core Reflection with Method Handles
- 417: Vector API (Third Incubator)
- 418: Internet-Address Resolution SPI
- 419: Foreign Function & Memory API (Second Incubator)
- 420: Pattern Matching for switch (Second Preview)
- 421: Deprecate Finalization for Removal

19

- <https://openjdk.java.net/projects/jdk/19/>

### Features

#### JEPs proposed to target JDK 19

- 422: Linux/RISC-V Port

review ends

2022/03/17



## Concurrency

- <https://openjdk.java.net/projects/loom/>

### **Loom - Fibers, Continuations and Tail-Calls for the JVM**

**PLEASE NOTE!** Go to the [Wiki](#) for additional and up-to-date information.

The goal of this [Project](#) is to explore and incubate Java VM features and APIs built on top of them for the implementation of lightweight user-mode threads ([fibers](#)), delimited continuations (of some form), and related features, such as explicit [tail-call](#).

- <https://openjdk.java.net/jeps/8277131>

### **JEP draft: Virtual Threads (Preview)**

- <https://openjdk.java.net/projects/tsan/>

The goal of this [Project](#) project is to provide a venue to explore and incubate a Thread SANitizing (TSAN) feature that could be integrated into the Hotspot JVM and the JVM Tool Interface. This includes working, evaluating, and incubating a Thread-Sanitizer implementation for Java as described in the [corresponding JEP](#).

...



## 2.8.2 Wasm

- <https://en.wikipedia.org/wiki/WebAssembly>

C source code and corresponding WebAssembly		
C source code	WebAssembly .wat text format	WebAssembly .wasm binary format
<pre>int factorial(int n) {     if (n == 0)         return 1;     else         return n * factorial(n-1); }</pre>	<pre>(func (param i64) (result i64)   local.get 0   i64.eqz   if (result i64)     i64.const 1   else     local.get 0     local.get 0     i64.const 1     i64.sub     call 0     i64.mul   end)</pre>	<pre>00 61 73 6D 01 00 00 00 01 00 01 60 01 73 01 73 06 03 00 01 00 02 04 00 01 00 00 20 00 50 04 7E 42 01 05 20 00 20 00 42 01 7D 10 00 7E 0B 0B 15 17</pre>

- <https://webassembly.org/>

**WebAssembly(abbr. *Wasm*) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications.**

- <https://github.com/WebAssembly/design>



WebAssembly 1.0 has shipped in 4 major browser engines.

- <https://webassembly.org/specs/>

### Limitations

- - 1. In general, WebAssembly does not allow direct interaction with the DOM. All interaction must flow through JavaScript interop.
  - 2. [Multithreading](#) (although there are plans to address this.)
  - 3. [Garbage collection](#) (although there are plans to address this.)
  - 4. Security considerations (discussed below)

WebAssembly is supported on desktops, and mobile, but on the latter, in practice (for non-small memory allocations, such as with [Unity](#) game engine) there are "grave limitations that make many applications infeasible to be *reliably* deployed on mobile browsers [...]. Currently allocating more than ~300MB of memory is not reliable on Chrome on Android without resorting to Chrome-specific workarounds, nor in Safari on iOS."<sup>[62]</sup>

All major web browsers allow WebAssembly if Content-Security-Policy is not specified, or if "unsafe-eval" is used, but otherwise the major web browsers behave differently.<sup>[63]</sup> In practice WebAssembly can't be used on Chrome without "unsafe-eval",<sup>[64][65]</sup> while a worker thread workaround is available.<sup>[66]</sup>



# Implementations

While WebAssembly was initially designed to enable near-native code execution speed in the web browser, it has been considered valuable outside of such, in more generalized contexts.<sup>[37][38]</sup> Since WebAssembly's runtime environments (RE) are low-level virtual stack machines (akin to [JVM](#) or [Flash VM](#)) that can be embedded into host applications, some of them have found a way to standalone runtime environments like [Wasmtime](#) and [Wasmer](#).<sup>[8][13]</sup>

## Web browsers [edit]

In November 2017, Mozilla declared support "in all major browsers"<sup>[39]</sup> after WebAssembly was enabled by default in Edge 16.<sup>[40]</sup> The support includes mobile web browsers for iOS and Android. As of July 2021, 94% of installed browsers support WebAssembly.<sup>[41]</sup> But for older browsers, Wasm can be compiled into asm.js by a JavaScript [polyfill](#).<sup>[42]</sup>

## Compilers:

Because WebAssembly [executables](#) are precompiled, it is possible to use a variety of programming languages to make them.<sup>[43]</sup> This is achieved either through direct compilation to Wasm, or through implementation of the corresponding [virtual machines](#) in Wasm. There have been around 40 programming languages reported to support Wasm as a compilation target.<sup>[44]</sup>

Emscripten compiles C and C++ to Wasm<sup>[32]</sup> using the Binaryen and LLVM as backend.<sup>[45]</sup>

As of version 8, a standalone Clang can compile C and C++ to Wasm.<sup>[46]</sup>

Its initial aim is to support compilation from C and C++,<sup>[47]</sup> though support for other source [languages](#) such as [Rust](#), [.NET languages](#)<sup>[48][49][44]</sup> and [AssemblyScript](#)<sup>[50]</sup> (TypeScript-like) is also emerging. After the MVP release, there are plans to support [multithreading](#) and [garbage collection](#)<sup>[51][52]</sup> which would make WebAssembly a compilation target for garbage-collected programming languages like C# (supported via Blazor), F# (supported via Bolero<sup>[53]</sup> with help of Blazor), Python, and even JavaScript where the browser's just-in-time compilation speed is considered too slow. A number of other languages have some support including [Python](#),<sup>[54]</sup> [Java](#),<sup>[55]</sup> [Julia](#),<sup>[56][57][58]</sup> [Zig](#),<sup>[59]</sup> and [Ruby](#),<sup>[60]</sup> as well as [Go](#).<sup>[61]</sup>

- <https://github.com/appcypher/awesome-wasm-langs>
- <https://github.com/appcypher/awesome-wasm-runtimes>

## Wasm & Rust

- <https://www.rust-lang.org/what/wasm>
- <https://rustwasm.github.io/book>
- <https://github.com/rustwasm>



## Beyond the browser

- <https://webassembly.org/docs/non-web/>
- <https://www.zdnet.com/article/microsoft-google-back-bytocode-alliance-to-move-webassembly-beyond-the-browser/>

## ■ WASI (WebAssembly System Interface)

WebAssembly System Interface (WASI) is a simple interface (ABI and API) designed by Mozilla intended to be portable to any platform.<sup>[77]</sup> It provides POSIX-like features like file I/O constrained by capability-based security.<sup>[78][79]</sup> There are also a few other proposed ABI/APIs.<sup>[80][81]</sup>

WASI is influenced by CloudABI and Capsicum.

Solomon Hykes, a co-founder of Docker, wrote in 2019, "If WASM+WASI existed in 2008, we wouldn't have needed to create Docker. That's how important it is. WebAssembly on the server is the future of computing."<sup>[82]</sup> Wasmer, out in version 1.0, provides "software containerization, we create universal binaries that work anywhere without modification, including operating systems like Linux, macOS, Windows, and web browsers. Wasm automatically sandboxes applications by default for secure execution".<sup>[82]</sup>

<https://wasi.dev/>

<https://github.com/WebAssembly/WASI>

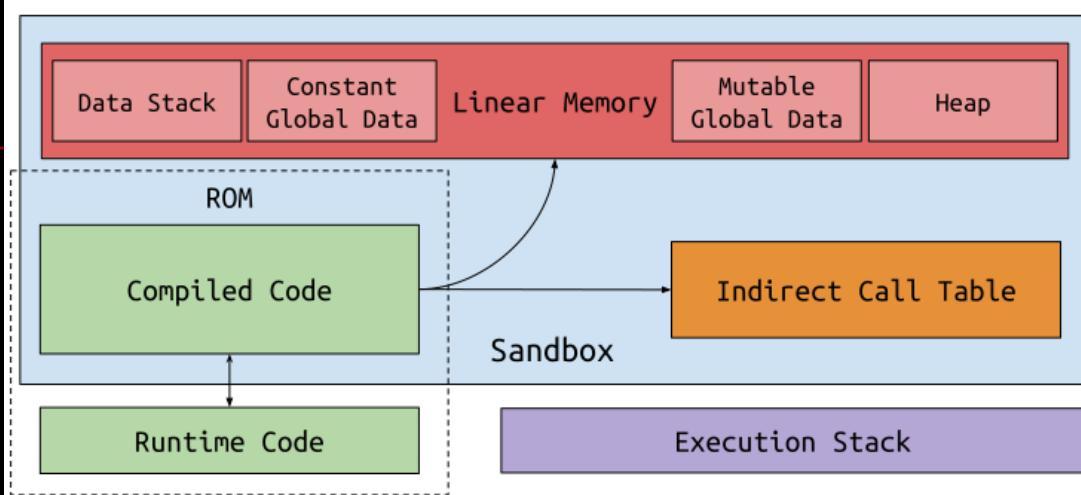
<https://github.com/bytocodealliance/wasmtime/blob/main/docs/WASI-documents.md>

<https://hacks.mozilla.org/2019/03/standardizing-wasi-a-webassembly-system-interface/>

<https://training.linuxfoundation.org/announcements/wasi-bringing-webassembly-way-beyond-browsers>

# Runtime

## Sandboxing



Wasm uses a co-design between the compiler, and the dynamic checks of the runtime system to provide the sandbox that isolates the surrounding system from the logic of the contained code. The figure depicts the main aspects of the sandbox. These include:

- *Linear memory* that holds all memory accessed by the sandbox. The compiler emits code that checks that all loads and stores remain within the linear memory, thus preventing errant accesses outside the sandbox. Linear memory is expandable much like a traditional heap.
- The *indirect function call table* that facilitates function pointer calls. To ensure that function pointer invocations are safe (to code generated by the compiler), function pointers reference an *offset* into the table. Each entry includes the type of the function, and ensures that function invocations are well-typed.
- The separation of the *execution stack* -- used to track function calls -- and the *data stack* -- used to contain stack-allocated data that can be referenced, thus must be in linear memory.

The first of these ensures the proper memory isolation of the sandbox, while the latter two provide control-flow integrity of the sandbox.

Source: <https://github.com/gwsysystems/aWsm/blob/master/doc/design.md>



## Good Resources

- <https://webassembly.org/roadmap/>
  - <https://platform.uno/blog/the-state-of-webassembly-2021-and-2022/>
  - <https://www.w3.org/groups/wg/wasm>
  - ...
-



## 2.8.2.1 Deno

### ■ <https://deno.land/>

**A modern runtime for JavaScript and TypeScript.**

Deno is a simple, modern and secure runtime for JavaScript and TypeScript that uses V8 and is built in [Rust](#).

- Secure by default. No file, network, or environment access, unless explicitly enabled.
- Supports TypeScript out of the box.
- Ships only a single executable file.
- Has built-in utilities like a dependency inspector (`deno info`) and a code formatter (`deno fmt`).
- Has a set of reviewed (audited) standard modules that are guaranteed to work with Deno: [deno.land/std](#)
- Has a number of [companies interested in using and exploring Deno](#)

### ■ <https://github.com/denoland/deno>





## 2.8.3 .Net

- <https://en.wikipedia.org/wiki/.NET>
- <https://dotnet.microsoft.com/>
- [https://en.wikipedia.org/wiki/Windows\\_Runtime](https://en.wikipedia.org/wiki/Windows_Runtime)
- ~~[https://en.wikipedia.org/wiki/.NET\\_Framework](https://en.wikipedia.org/wiki/.NET_Framework)~~
- <https://github.com/quozd/awesome-dotnet>
- <https://github.com/thangchung/awesome-dotnet-core>

### History

- 

Version	Release date	Released with	Latest update	Latest update date	Support ends <sup>[19]</sup>
.NET Core 1.0	2016-06-27 <sup>[20]</sup>	Visual Studio 2015 Update 3	1.0.16	May 14, 2019	June 27, 2019
.NET Core 1.1	2016-11-16 <sup>[21]</sup>	Visual Studio 2017 Version 15.0	1.1.13	May 14, 2019	June 27, 2019
.NET Core 2.0	2017-08-14 <sup>[13]</sup>	Visual Studio 2017 Version 15.3	2.0.9	July 10, 2018	October 1, 2018
.NET Core 2.1	2018-05-30 <sup>[14]</sup>	Visual Studio 2017 Version 15.7	2.1.30 (LTS)	August 19, 2021	August 21, 2021
.NET Core 2.2	2018-12-04 <sup>[15]</sup>	Visual Studio 2019 Version 16.0	2.2.8	November 19, 2019	December 23, 2019
.NET Core 3.0	2019-09-23 <sup>[22]</sup>	Visual Studio 2019 Version 16.3	3.0.3	February 18, 2020	March 3, 2020
.NET Core 3.1	2019-12-03 <sup>[23]</sup>	Visual Studio 2019 Version 16.4	3.1.22 (LTS)	December 14, 2021	December 3, 2022
.NET 5	2020-11-10 <sup>[24]</sup>	Visual Studio 2019 Version 16.8	5.0.14	February 8, 2022	May 8, 2022
.NET 6	2021-11-08 <sup>[25]</sup>	Visual Studio 2022 Version 17.0	6.0.2 (LTS)	February 8, 2022	November 8, 2024
.NET 7	2022-11 (projected)				May 2024 (projected)
.NET 8	2023-11 (projected)		(will be LTS)		November 2026 (projected)

- <https://devblogs.microsoft.com/dotnet/happy-20th-anniversary-net/>



## .Net on Web

### ■ Blazor

<https://en.wikipedia.org/wiki/Blazor>

a free and open-source web framework that enables developers to create web apps using C# and HTML.

<https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor>

<https://github.com/dotnet/aspnetcore>

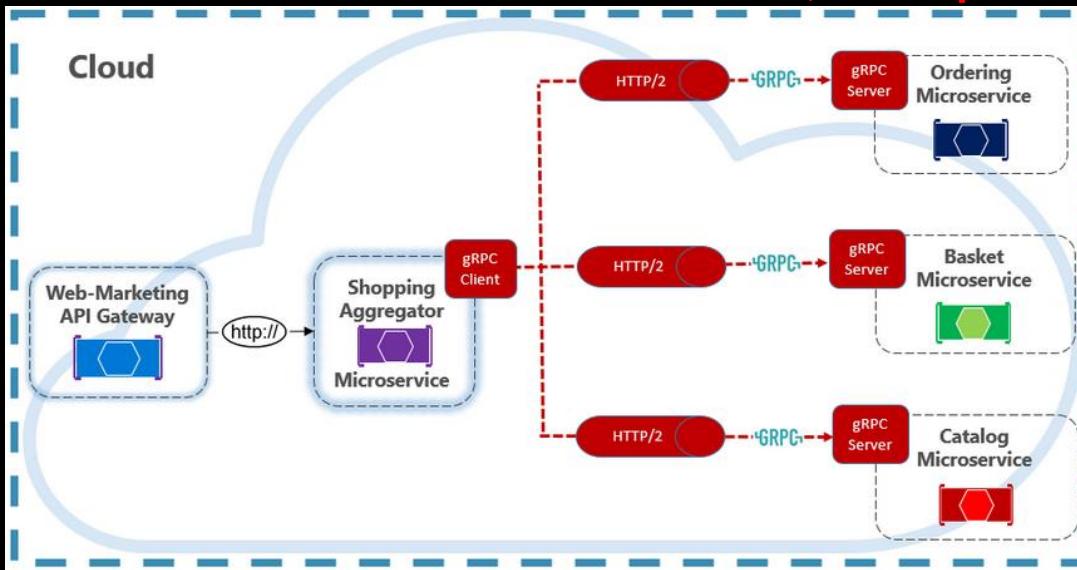
<https://github.com/AdrienTorris/awesome-blazor>



## gRPC support in .Net

- <https://docs.microsoft.com/en-us/aspnet/core/grpc/>
- <https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/grpc>

The microservice reference architecture, “eShop on Containers”:



- <https://grpc.io/docs/languages/csharp/dotnet/>
- ...



## .Net 6

- <https://rubikscode.net/2021/08/30/net-6-top-6-new-features-in-the-upcoming-net-6-version/>



- <https://devblogs.microsoft.com/dotnet/announcing-net-6/>
- <https://docs.microsoft.com/en-us/dotnet/core/compatibility/6.0>
- ...



## .Net 7

- <https://devblogs.microsoft.com/dotnet/announcing-net-7-preview-1/>
- <https://devblogs.microsoft.com/dotnet/asp-net-core-updates-in-net-7-preview-1/>
- <https://visualstudiomagazine.com/articles/2022/02/18/net-7-webassembly.aspx>
- ...



## Good Resources

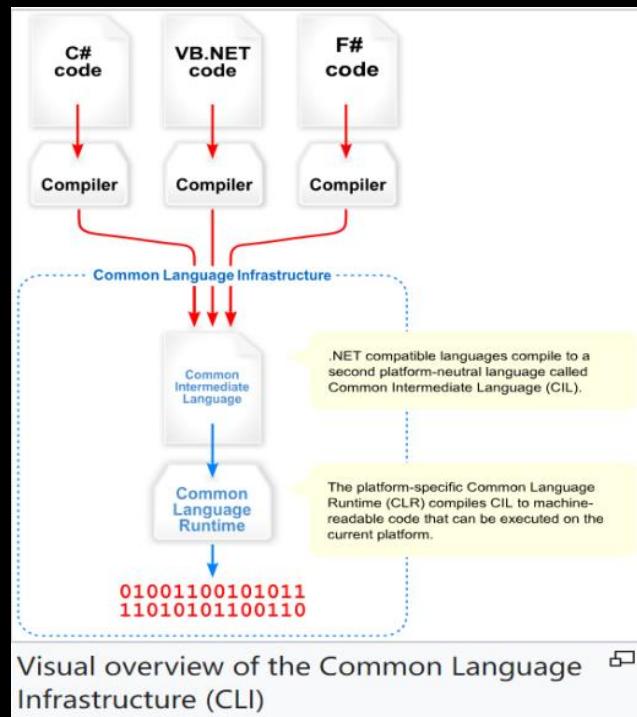
- [https://www.theregister.com/2022/02/15/20\\_years\\_of\\_dotnet/](https://www.theregister.com/2022/02/15/20_years_of_dotnet/)
  - ...
-



## 2.8.3.1 Runtime CLI

- [https://en.wikipedia.org/wiki/Common\\_Language\\_Infrastructure](https://en.wikipedia.org/wiki/Common_Language_Infrastructure)

The Common Language Infrastructure (CLI) is an open specification and technical standard originally developed by Microsoft and standardized by ISO (ISO/IEC 23271) and Ecma International (ECMA 335)<sup>[1][2]</sup> that describes executable code and a runtime environment that allows multiple high-level languages to be used on different computer platforms without being rewritten for specific architectures. This implies it is platform agnostic. The .NET Framework, .NET and Mono are implementations of the CLI. The metadata format is also used to specify the API definitions exposed by the Windows Runtime.<sup>[3][4]</sup>

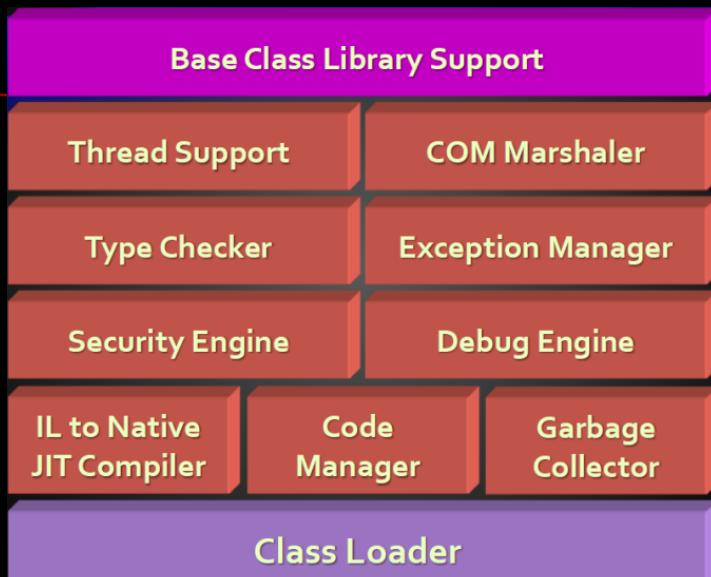


- [https://en.wikipedia.org/wiki/Common\\_Intermediate\\_Language](https://en.wikipedia.org/wiki/Common_Intermediate_Language)



## CLR

- [https://en.wikipedia.org/wiki/Common\\_Language\\_Runtime\\_Architecture](https://en.wikipedia.org/wiki/Common_Language_Runtime_Architecture)



Source: <https://www.slideshare.net/MohamadKrm/net-framework-250644809>

- <https://docs.microsoft.com/en-us/dotnet/standard/clr>



## Src

### ■ <https://github.com/dotnet/runtime>

```
[mydev@fedora Bak]$ tree -L 1 runtime-main  
runtime-main  
├── build.cmd  
├── Build.proj  
└── build.sh  
    ├── CODE-OF-CONDUCT.md  
    ├── CONTRIBUTING.md  
    ├── Directory.Build.props  
    ├── Directory.Build.targets  
    ├── Directory.Solution.props  
    └── docs  
        ├── dotnet.cmd  
        └── dotnet.sh  
    ├── eng  
    ├── global.json  
    ├── LICENSE.TXT  
    ├── NuGet.config  
    ├── PATENTS.TXT  
    ├── README.md  
    ├── SECURITY.md  
    └── src  
        └── THIRD-PARTY-NOTICES.TXT
```

```
[mydev@fedora Bak]$ tree -L 1 runtime-main/src  
runtime-main/src  
├── coreclr  
├── installer  
├── libraries  
├── mono  
├── native  
├── samples  
├── tasks  
└── tests  
    └── workloads
```

```
[mydev@fedora Bak]$ tree -L 1 runtime-main/src/coreclr/  
coreclr/  
├── binder  
├── _build-commons.sh  
├── build-runtime.cmd  
└── build-runtime.sh  
    ├── classlibnative  
    ├── clrdefinitions.cmake  
    ├── clr.featuredefines.props  
    ├── clrfeatures.cmake  
    ├── CMakeLists.txt  
    ├── components.cmake  
    ├── cpp_hint  
    ├── crosscomponents.cmake  
    ├── crossgen-corelib.proj  
    ├── debug  
    ├── Directory.Build.props  
    ├── Directory.Build.targets  
    └── dlls  
        ├── EmptyProps.props  
        ├── enablesanitizers.sh  
        ├── gc  
        ├── gcdump  
        ├── gcinfo  
        ├── generatedefinesfile.sh  
        ├── hosts  
        ├── ilasm  
        ├── ildasm  
        ├── inc  
        ├── interop  
        ├── jit  
        ├── md  
        ├── minipal  
        ├── nativeaot  
        ├── nativeresources  
        ├── pal  
        ├── palrt  
        ├── pgosupport.cmake  
        └── run-cppcheck.sh  
    ├── runtime-prereqs.proj  
    ├── runtime.proj  
    ├── scripts  
    ├── System.Private.CoreLib  
    ├── tools  
    └── unwinder  
        └── utilcode  
    └── vm
```

### ■ <https://github.com/dotnet/llvm-project>



## 2.9 Cloud Native

### 2.9.1 Replace Docker with Wasm?

- 

 Solomon Hykes  
@solomonstre

If WASM+WASI existed in 2008, we wouldn't have needed to created Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task!

Lin Clark @linclark

WebAssembly running outside the web has a huge future. And that future gets one giant leap closer today with...

📢 Announcing WASI: A system interface for running WebAssembly outside the web (and inside it too)

<https://t.co/HdEAZAyqYu>

March 27th 2019

- 

**Kubernetes is deprecating Docker as a container runtime after v1.20**

- 

**Containers vs WASI on K8s**

Containers	WASI
High startup cost.	Low startup cost.
Bigger binary files and highest memory consumption.	Smaller .wasm files and lower memory consumption.
Security requires high starting cost.	High security by default.
Active open-source community.	Active open-source community.
Over 10 years of active development for Docker.	WASI was only created 2 years ago.
Access to multithreading.	Only able to run in a single thread.
Access to Network.	Limited network access.
Any library can be compiled to.	Not every library can be compiled to.

Source: “Living on the Edge: Using IoT Devices on Kubernetes WebAssembly Applications”, Kate Goldenring (Microsoft) and Rodrigo Rodrigues Lemos (Facebook), Cloud Native Wasm Day North America 2021



## 2.9.2 Rust-based new solutions

### 2.9.2.1 Krustlet

- <https://krustlet.dev/>
- <https://github.com/krustlet/krustlet>

**Kubernetes Kubelet in Rust for running WASM.**

- **Upcoming v1.0**

 Krustlet 1.0 coming soon!

Krustlet acts as a Kubelet by listening on the event stream for new pods that the scheduler assigns to it based on specific Kubernetes [tolerations](#).

The default implementation of Krustlet listens for the architecture `wasm32-wasi` and schedules those workloads to run in a `wasmtime`-based runtime instead of a container runtime.

#### Networking

After many discussions, we have decided that [CNI](#) and 100% native Kubernetes network support will not be part of the 1.0 release. Networking implementations vary wildly between different Krustlet providers and it would be difficult and unwise to try and define a common abstraction at this time. For example, wasmCloud connects using a system called a "lattice" which can consist of an arbitrary graph of connected hosts. Whereas the CRI provider relies on the [CNI](#) spec implementations built in to many Kubernetes distros. Not only are we at a point where we need to understand how more complex networking should work, but the same discussions involve things like sidecars, service meshes, and so on. As a result, we decided to let you handle your own networking here, which you can do, while the community understands what's the best path forward with these more complex interactions.

Our hope is that once we have several different networking examples from various providers, it will be easier to add in a common abstraction at this time. Waiting on that information would only introduce additional delay for releasing Krustlet as an otherwise stable and production-ready (though still bleeding edge) project.

- <https://cloudblogs.microsoft.com/opensource/2020/04/07/announcing-krustlet-kubernetes-rust-kubelet-webassembly-wasm/>
- **For Krustlet on RPi4, you may refer to my previous talk "Cloud-Hypervisor on ARM" at the Rust China Meetup 2021(Hangzhou)**



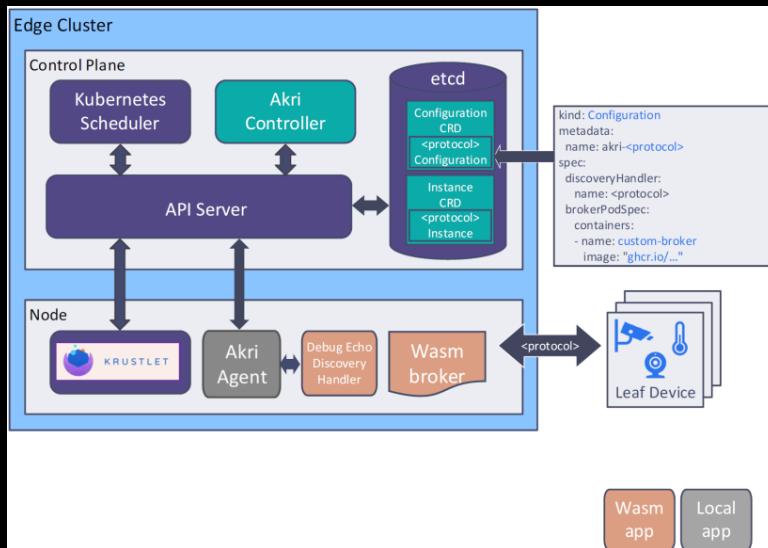
## 2.9.2.2 Akri & Hippo

- <https://docs.akri.sh/>
- <https://github.com/deislabs/akri>

### A Kubernetes Resource Interface for the Edge.

Akri lets you easily expose heterogeneous leaf devices (such as IP cameras and USB devices) as resources in a Kubernetes cluster, while also supporting the exposure of embedded hardware resources such as GPUs and FPGAs. Akri continually detects nodes that have access to these devices and schedules workloads based on them.

- <https://github.com/deislabs/akri-on-krustlet>
- Convert Akri from a containerized application to Wasm modules



Source: “Living on the Edge: Using IoT Devices on Kubernetes WebAssembly Applications”, Kate Goldenring(Microsoft) and Rodrigo Rodrigues Lemos (Facebook), Cloud Native Wasm Day North America 2021.

- <https://docs.akri.sh/demos/usb-camera-demo-rpi4>
- <https://github.com/deislabs/hippo> //The WebAssembly PaaS



## 2.9.3 Kubernetes

### 2.9.3.1 k0s

- <https://k0sproject.io/>
- <https://github.com/k0sproject/k0s>

k0s is an all-inclusive Kubernetes distribution, which is configured with all of the features needed to build a Kubernetes cluster and packaged as a single binary for ease of use.

k0s fits well in any cloud environment, but can also be used in IoT gateways, Edge and Bare metal deployments due to its simple design, flexible deployment options and modest system requirements.

#### ■ Key Features

- Different installation methods: [single-node](#), [multi-node](#), [airgap](#) and [Docker](#)
- Automatic lifecycle management with k0sctl: [upgrade](#), [backup](#) and [restore](#)
- Modest [system requirements](#) (1 vCPU, 1 GB RAM)
- Vanilla upstream Kubernetes (with no changes)
- Available as a single binary with no [OS dependencies](#) besides the kernel
- Flexible deployment options with [control plane isolation](#) as default
- Scalable from a single node to large, [high-available](#) clusters
- Supports custom [Container Network Interface \(CNI\)](#) plugins (Kube-Router is the default, Calico is offered as preconfigured alternative)
- Supports custom [Container Runtime Interface \(CRI\)](#) plugins (containerd is the default)
- Supports all Kubernetes storage options with [Container Storage Interface \(CSI\)](#)
- Supports a variety of [datastore backends](#): etcd (default for multi-node clusters), SQLite (default for single node clusters), MySQL, and PostgreSQL
- Supports x86-64, ARM64 and ARMv7
- [Konnectivity service](#), CoreDNS, Metrics Server



## Zero Friction

`k8s` drastically reduces the complexity of installing and running a fully conformant Kubernetes distribution. New kube clusters can be bootstrapped in minutes. Developer friction is reduced to zero, allowing anyone, with no special skills or expertise in Kubernetes to easily get started.

## Zero Deps

`k8s` is distributed as a single binary with zero host OS dependencies besides the host OS kernel. It works with any operating system without additional software packages or configuration. Any security vulnerabilities or performance issues can be fixed directly in the `k8s` distribution.

## Zero Cost

`k8s` is completely free for personal or commercial use, and it always will be. The source code is available on [Github](#) under Apache 2 license. It's a no brainer foundation for any Kubernetes projects and easy build upon.



## 2.9.3.2 Arkade

- <https://github.com/alexellis/arkade>  
**The Open Source Kubernetes Marketplace.**
- <https://kccncna20.sched.com/event/ekAF/the-past-present-and-future-of-kubernetes-on-raspberry-pi-alex-ellis-openfaas-ltd>



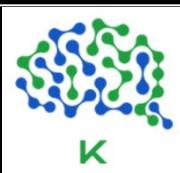
- **Complete list of CLI tools**

argocd	doctl	k0s	kops	linkerd2	polaris
argocd-autopilot	faas-cli	k0sctl	krew	mc	popeye
arkade	flux	k10multicluster	kube-bench	metal	porter
autok3s	gh	k10tools	kubebuilder	minikube	rekor-cli
buildx	goreleaser	k3d	kubectl	nats	stern
civo	helm	k3s	kubectx	nerdctl	terraform
clusterctl	helmfile	k3sup	kubens	nova	tfsec
cosign	hugo	k9s	kubescape	opa	tilt
devspace	influx	kail	kubeseal	osm	tkn
dive	inlets-pro	kanctl	kubestr	pack	trivy
docker-compose	inletsctl	kgctl	kubetail	packer	vagrant
	istioctl	kim	kustomize	packer	waypoint
	jq	kind		yq	

## 2.10 Infrastructure

### 2.10.1 Virtualization

- ...
- 





## 2.10.1.1 IO SIOV

- <https://www.intel.com/content/www/us/en/newsroom/opinion/new-specification-hyperscale-virtualization.html#gs.ukd78i>
- [https://www.phoronix.com/scan.php?page=news\\_item&px=Intel-Microsoft-SIOV](https://www.phoronix.com/scan.php?page=news_item&px=Intel-Microsoft-SIOV)

**The Scalable I/O Virtualization specification is designed to be a "scalable and flexible approach to hardware-assisted I/O virtualization" and builds atop PCI Express and Compute Express Link (CXL).**

Intel's [announcement](#) from Tuesday sums it up as:

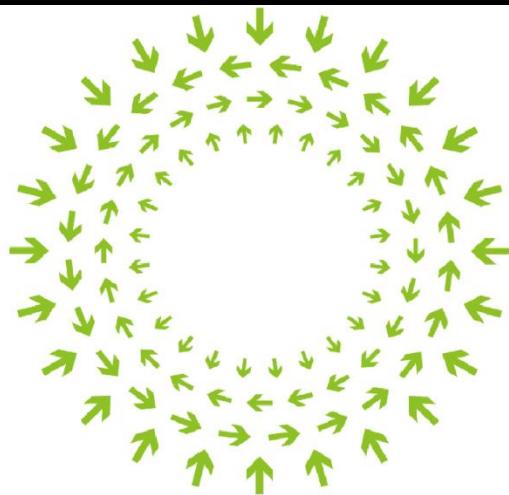
*SIOV architecture will enable data center operators to deliver more cost-effective access to high-performance accelerators and other key I/O devices for their customers, as well as relieve I/O device manufacturers of cost and programming burdens imposed under previous standards.*

*The new SIOV specification is a modernized hardware and software architecture that enables efficient, mass-scale virtualization of I/O devices, and overcomes the scaling limitations of prior I/O virtualization technologies. Under the terms of the OCP contribution, any company can adopt SIOV technology and incorporate it into their products under an open, zero-cost license.*

*In cloud environments, I/O devices including network adaptors, GPUs and storage controllers are shared among many virtualized workloads requiring their services. Hardware-assisted I/O virtualization technologies enable efficient routing of I/O traffic from the workloads through the virtualization software stack to the devices. It helps keep overhead low and performance close to "bare-metal" speeds.*



## ■ Specification



# OPEN

Compute Project

Scalable I/O Virtualization Revision 1.0

Version 1.2

February 2022

Authors:

Intel Corporation (Contact Point: Tom Stachura)

Microsoft Corporation (Contact Point: Renee L'Heureux)

Source: <https://www.opencompute.org/documents/ocp-scalable-io-virtualization-technical-specification-revision-1-v1-2-pdf>



### 3) An eBPF overview

- [https://en.wikipedia.org/wiki/Berkeley\\_Packet\\_Filter](https://en.wikipedia.org/wiki/Berkeley_Packet_Filter)

#### BPF (Berkeley Packet Filter, aka cBPF)

- introduced in kernel 2.1.75 (1997)
- <https://blog.cloudflare.com/bpf-the-forgotten-bytecode/>

#### eBPF (extended BPF)

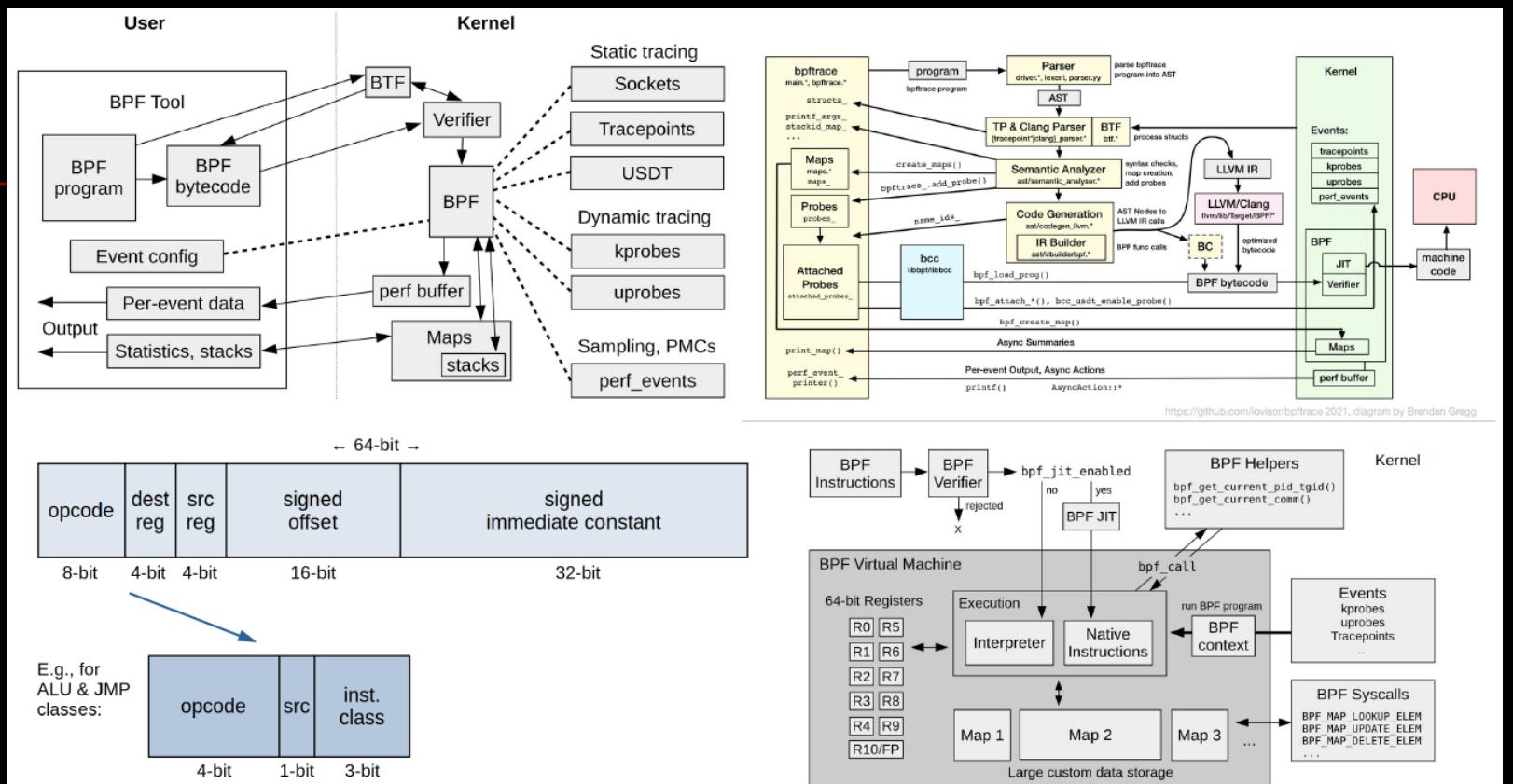
- since Linux Kernel v3.15 and ongoing
- aims at being a universal in-kernel virtual machine
- a simple way to extend the functionality of Kernel at runtime
- “dtrace for Linux”

#### uBPF (userspace BPF)

- <https://stackoverflow.com/questions/65904948/why-is-having-an-userspace-version-of-ebpf-interesting>
- mainly applied to **Networking & Blockchain...**



# How it works

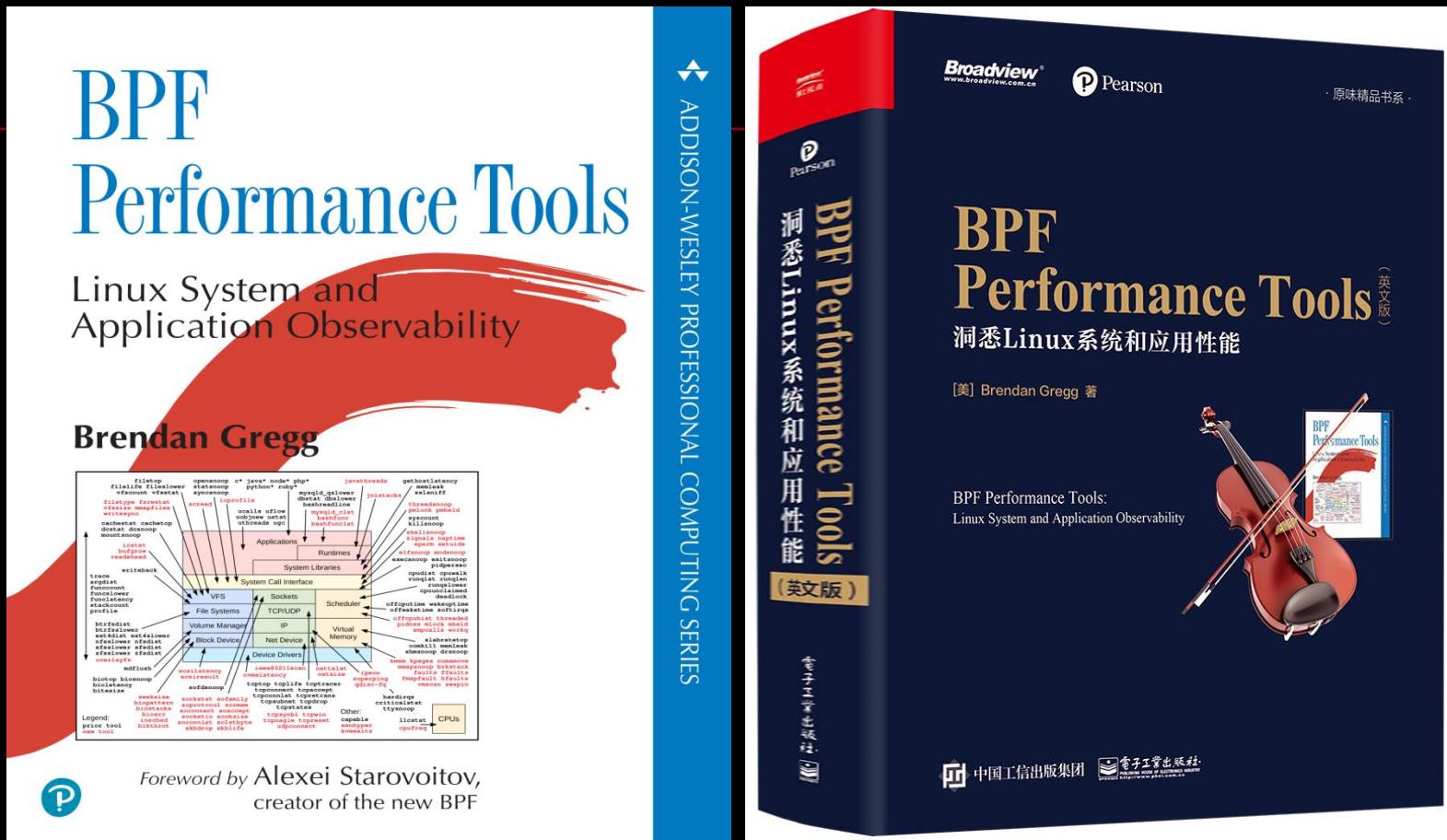


Source: [https://www.brendangregg.com/Slides/LISA2021\\_BPF\\_Internals.pdf](https://www.brendangregg.com/Slides/LISA2021_BPF_Internals.pdf)



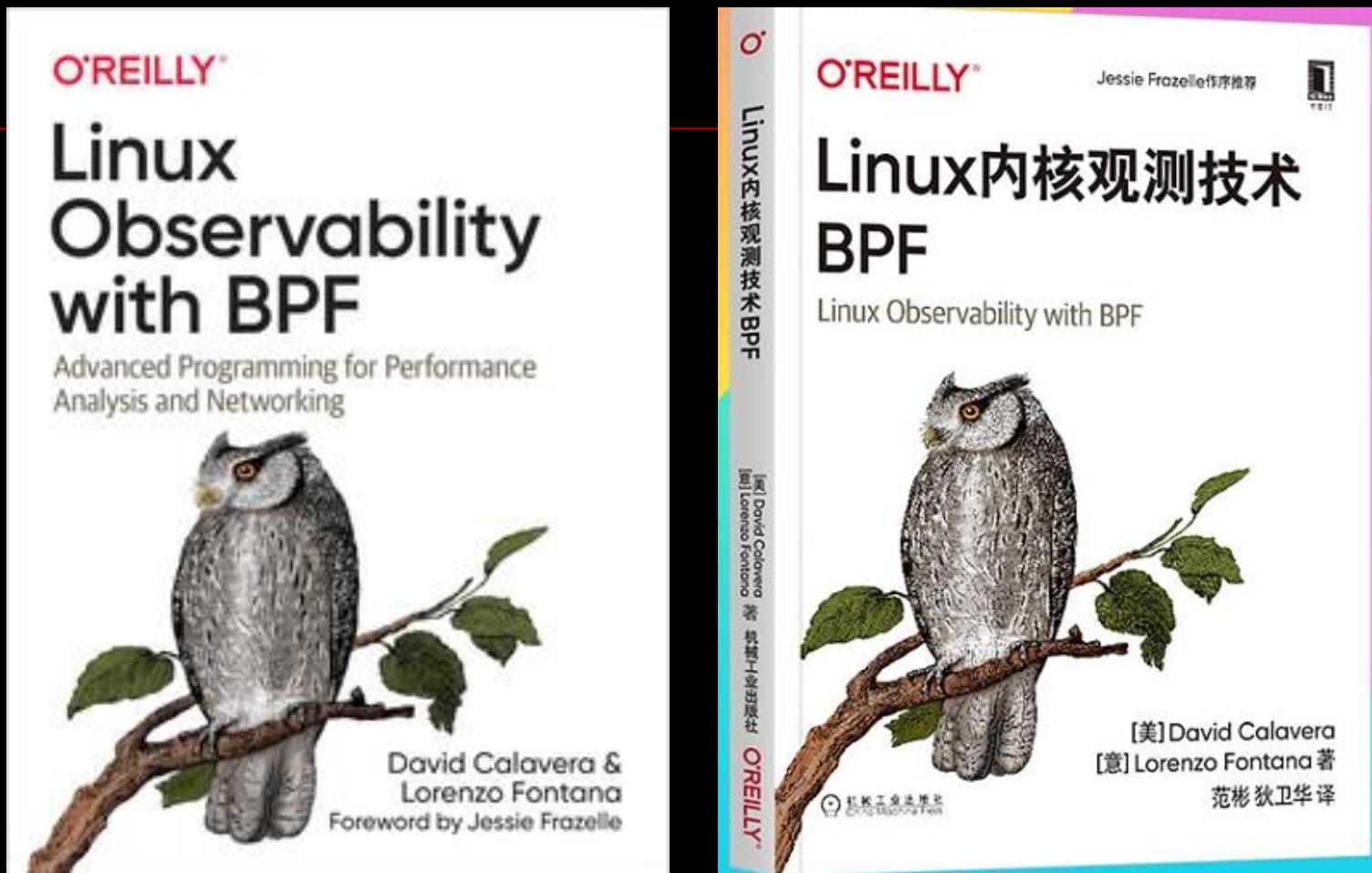
## Good Resources

- <https://www.brendangregg.com/bpf-performance-tools-book.html>





- <https://www.oreilly.com/library/view/linux-observability-with/9781492050193/>





## Details

- For details, you may refer to my previous talks at  
<https://github.com/XianBeiTuoBaFeng2015/MySlides/tree/master/Conf>  
and the upcoming “eBPF 101” at  
<https://github.com/XianBeiTuoBaFeng2015/MySlides/tree/master/LTS>



## 4) HW/SW Co-design and Co-development

University of Victoria

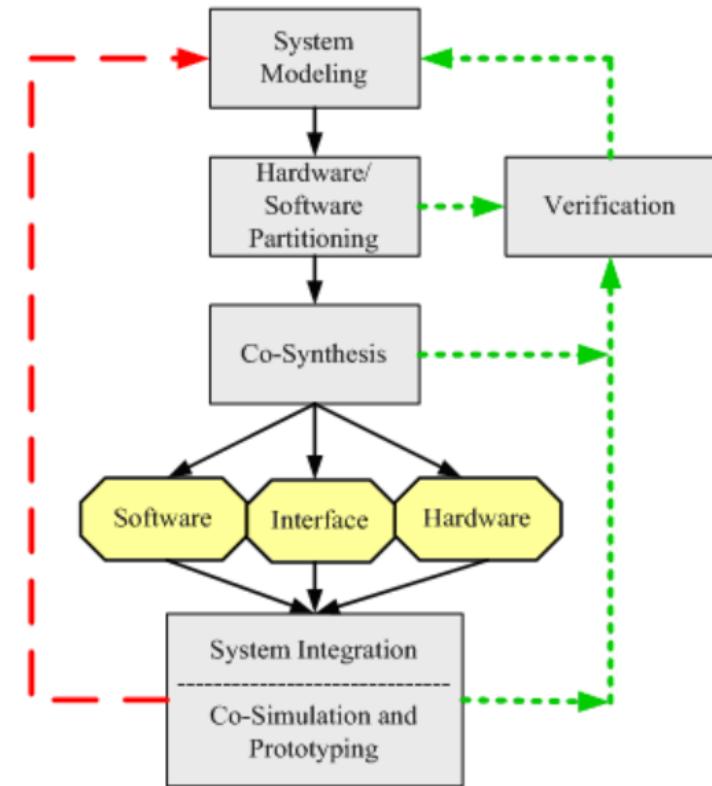
- <https://webhome.cs.uvic.ca/~mserra/HScodesign.html>

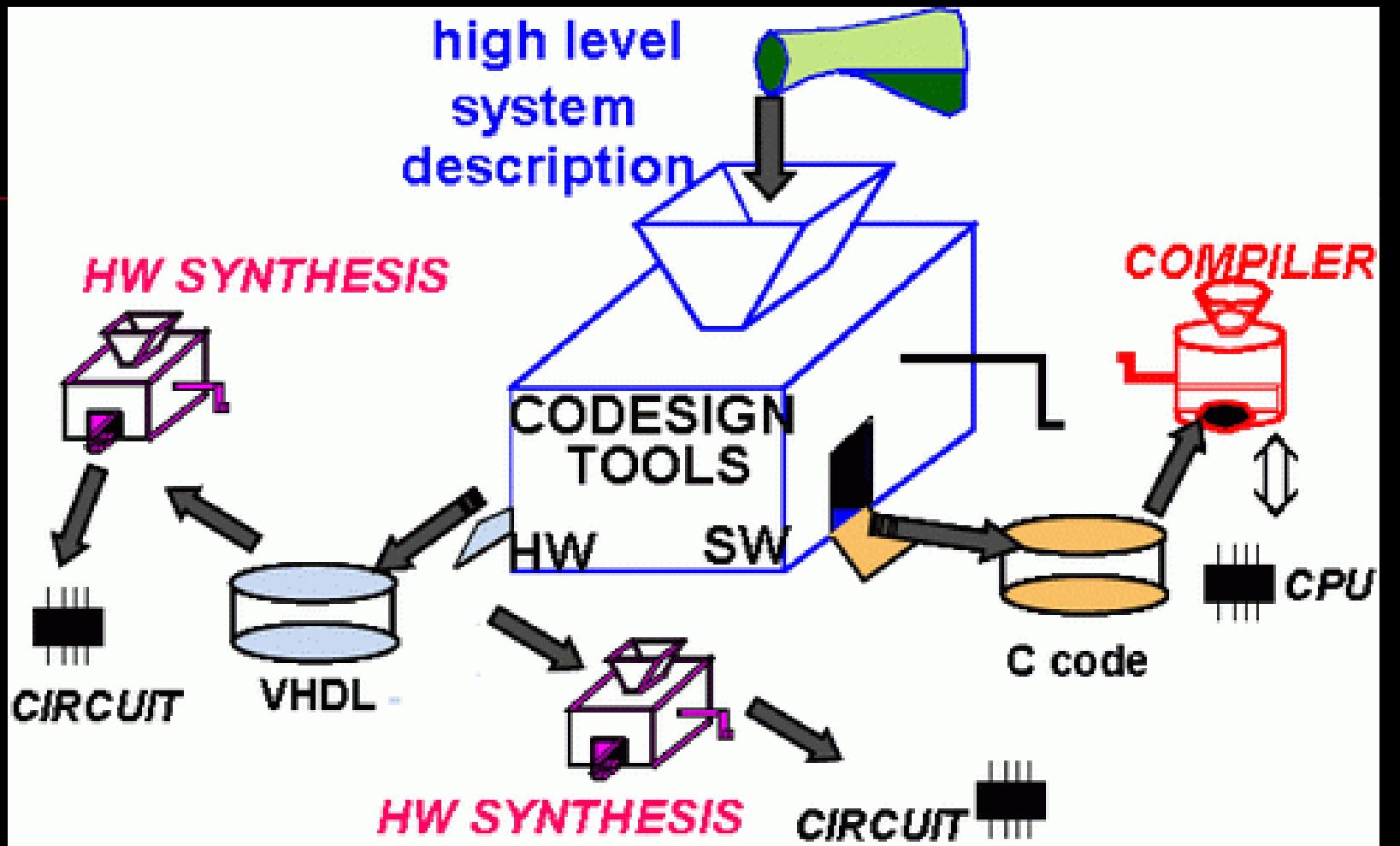
The co-design of HW/SW systems may be viewed as composed of four main phases as illustrated in the side diagram:

1. Modeling
2. Partitioning
3. Co-Synthesis
4. C-Simulation

**Modeling** involves specifying the concepts and the constraints of the system to obtain a refined specification. This phase of the design also specifies a software and hardware model. The first problem is to find a suitable specification methodology for a target system. Some researchers favour a formal language which can yield provably-correct code. But most prefer a hardware-type language (e.g., VHDL, Verilog), a software-type language (C, C++, Handel-C, SystemC), or other formalism lacking a hardware or software bias (such as Codesign Finite State Machines). There are three different paths the modeling process can take, considering its starting point:

- An existing software implementation of the problem.
- An existing hardware, for example a chip, is present.
- None of the above is given, only specifications leaving an open choice for a model.







## 4.1 RISC-V

### 4.1.1 The most promising

#### A New Golden Age for Computer Architecture

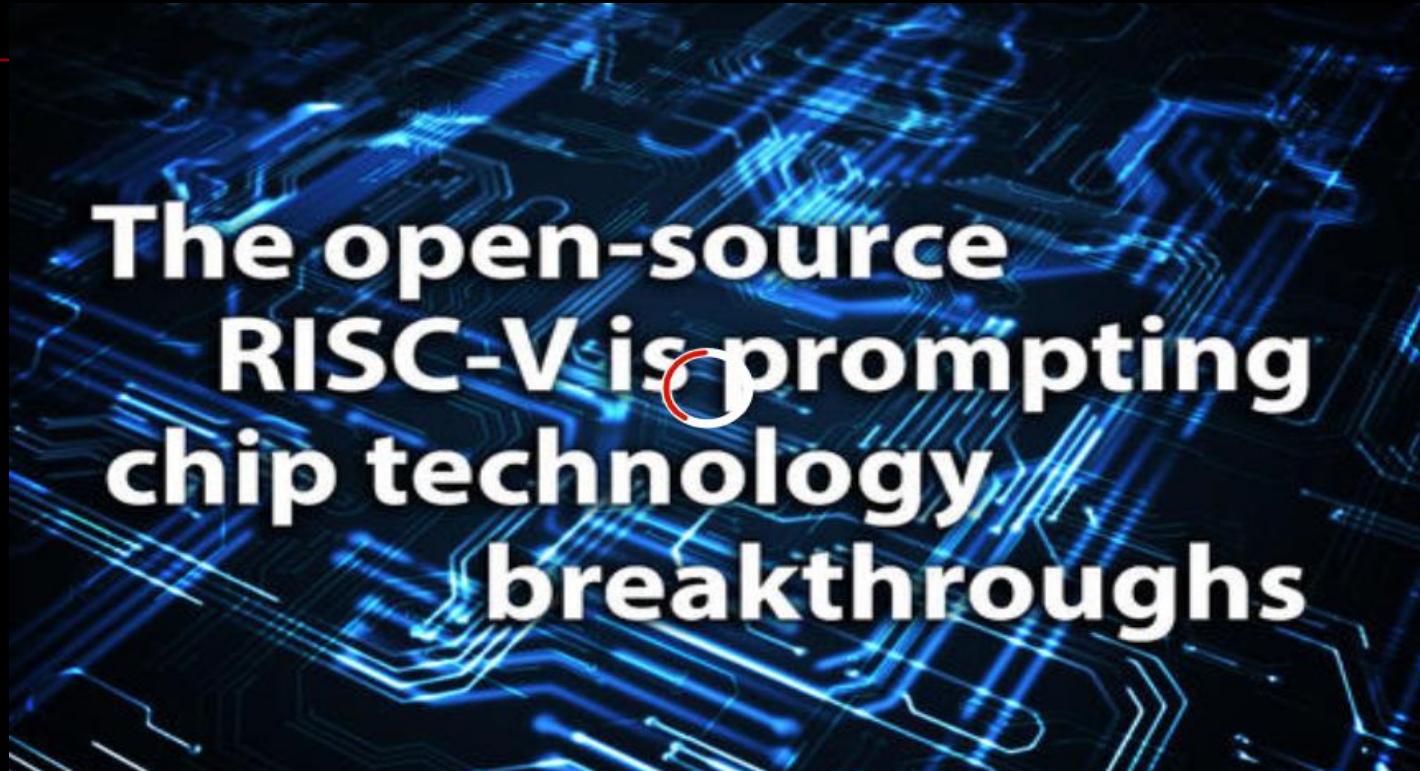
- ~~<https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext>~~





**"The Linux of the chip world"**

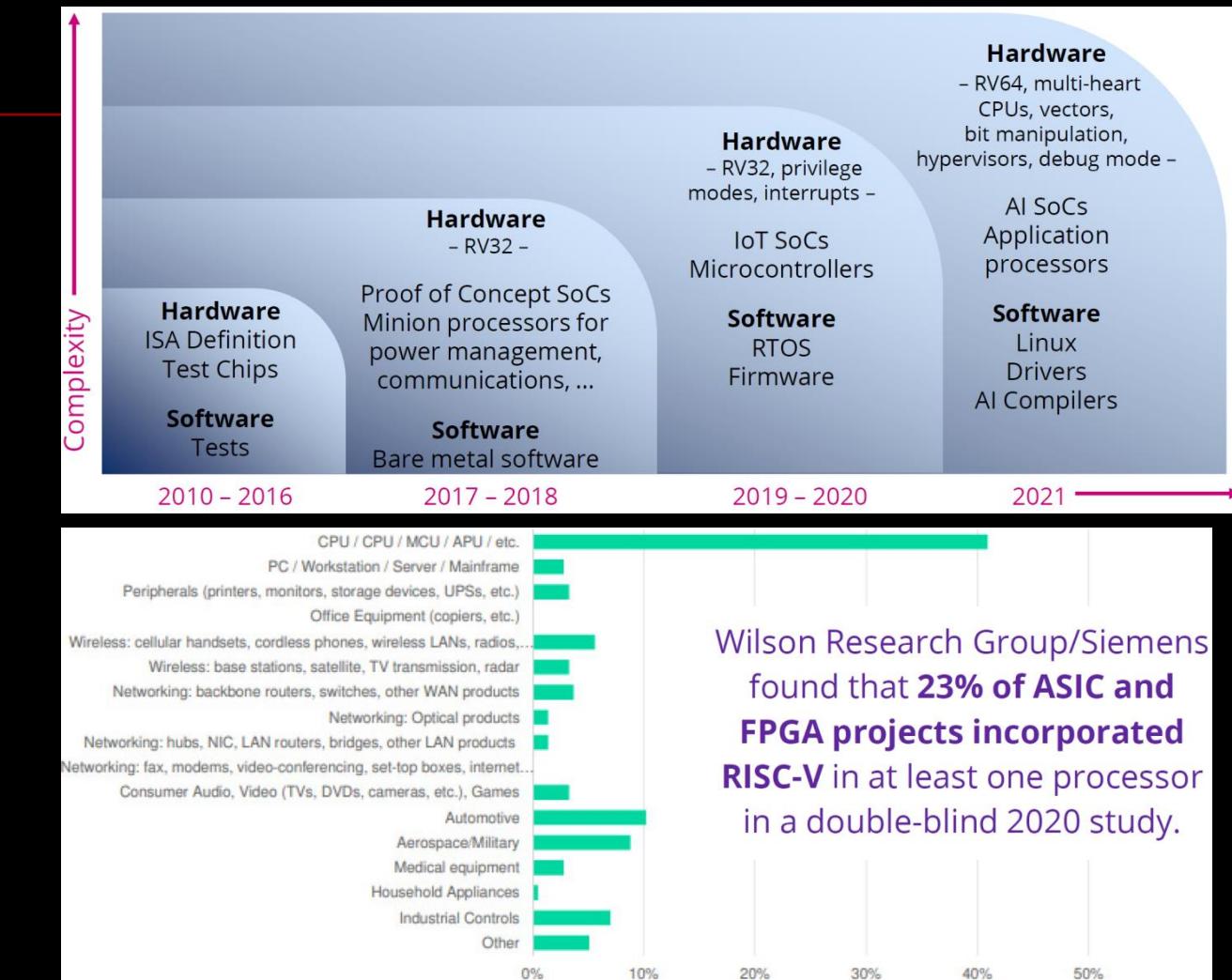
- <https://www.zdnet.com/article/risc-v-the-linux-of-the-chip-world-is-starting-to-produce-technological-breakthroughs/>





## 4.1.2 The growing ecosystem

### Industry innovation on RISC-V



Wilson Research Group/Siemens found that **23% of ASIC and FPGA projects incorporated RISC-V** in at least one processor in a double-blind 2020 study.

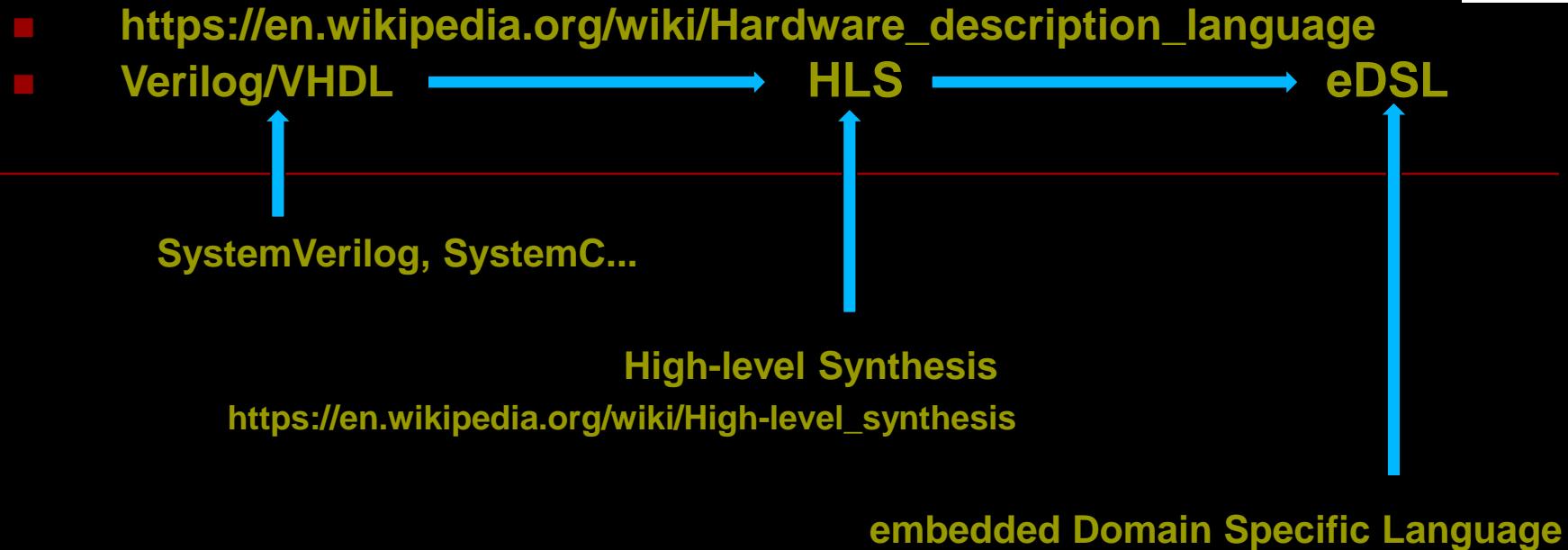


## 4.2 Open Source EDA

- [https://en.wikipedia.org/wiki/Comparison\\_of\\_EDA\\_software](https://en.wikipedia.org/wiki/Comparison_of_EDA_software)
- <https://sem wiki.com/wikis/industry-wikis/eda-open-source-tools-wiki/>
- <https://ieeexplore.ieee.org/document/9398963>
- <https://ieeexplore.ieee.org/document/9398960>
- <https://ieeexplore.ieee.org/document/9336682>
- <https://ieeexplore.ieee.org/document/9105619>
- ...



## 4.2.1 Evolution of HDL

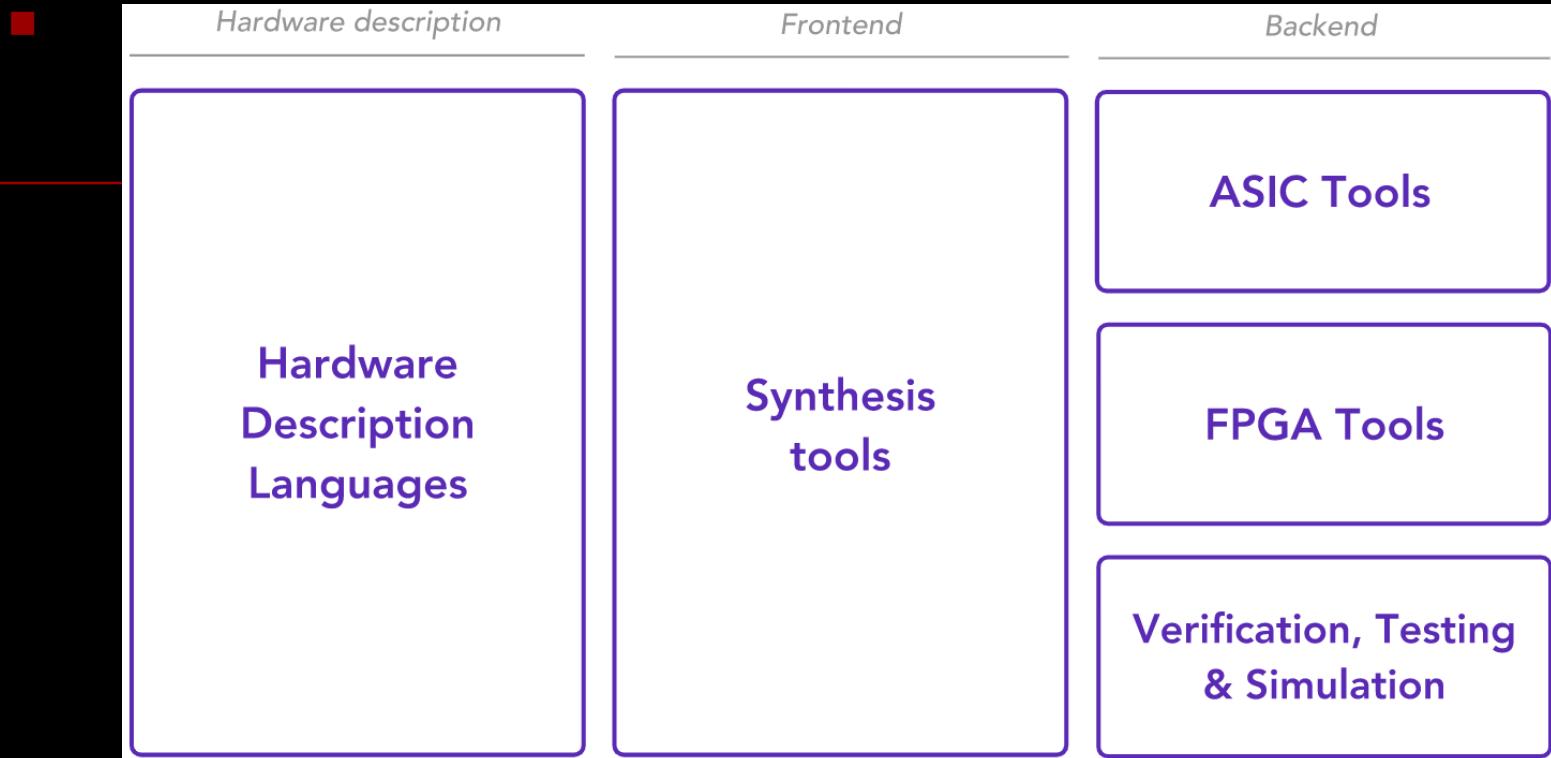


### Typical eDSLs

- Haskell as host  
Bluespec, Clash...
- Scala as host  
Chisel, SpinalHDL...
- Python as host  
FHDL...
- Finally convert to Verilog or VHDL  
[https://en.wikipedia.org/wiki/Source-to-source\\_compiler](https://en.wikipedia.org/wiki/Source-to-source_compiler)



## 4.2.2 Open FPGA



Source: <https://symbiflow.github.io/>

#### 4.2.2.1 OSFPGA

- <https://osfpga.org/>
- <https://github.com/os-fpga>
- 



## Open-Source FPGA Foundation

The Open-Source FPGA Foundation will bring together companies, universities and individuals working on or interested in advancing open-source FPGA capabilities, establish the necessary cooperation channels, promote outreach and education, and coordinate joint efforts to enable easier collaboration around an open source FPGA ecosystem.

[Join Us](#)

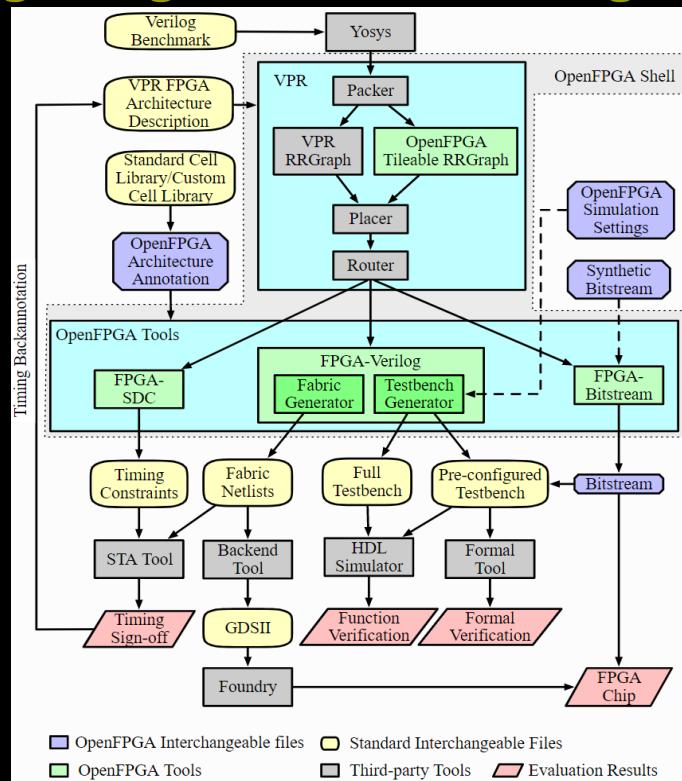


## 4.2.2.2 OpenFPGA

- <https://github.com/Inis-uofu/OpenFPGA>
- <https://openfpga.readthedocs.io/en/master/>

### Tool Suites & Design Flows

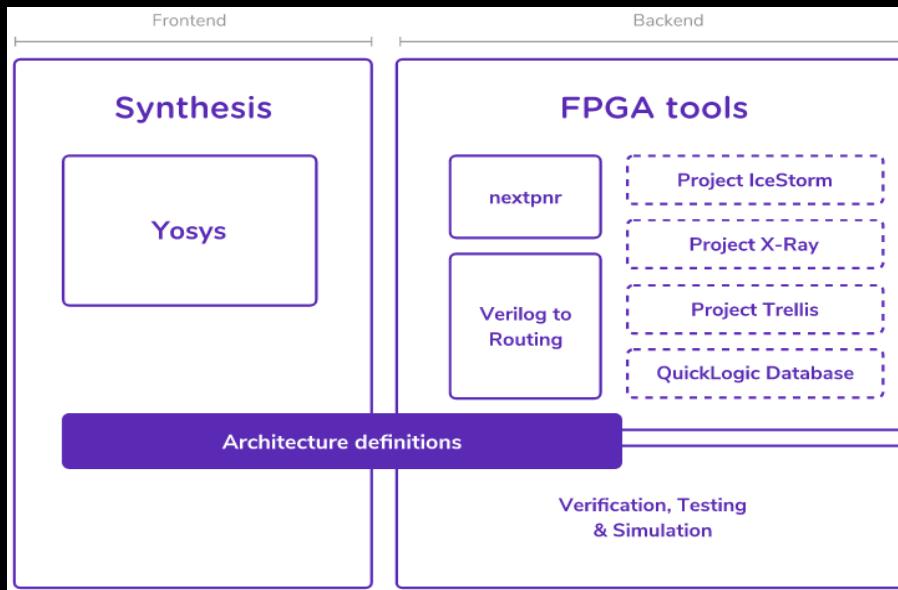
- [https://openfpga.readthedocs.io/en/master/tutorials/getting\\_started/tools/#fig-openfpga-tools](https://openfpga.readthedocs.io/en/master/tutorials/getting_started/tools/#fig-openfpga-tools)





### 4.2.2.3 SymbiFlow

- <https://symbiflow.github.io/>  
**the GCC of FPGAs**
- **Supported the Xilinx 7-Series, Lattice iCE40/ECP5, QuickLogic EOS S3...**
- **Src**
- **Python-based...**
- **FASM(FPGA Assembly)**
- <https://symbiflow.readthedocs.io/en/latest/fasm/docs/specification.html>
- **Structure**



## 4.2.3 Modeling and Simulation

- ...
- 





### 4.2.3.1 Renode

- <https://renode.io/>

**It lets you run, debug and test unmodified embedded software on your PC -- from bare System-on-Chips, through complete devices to multi-node systems.**

- **Supported Archs**



- **Features**

Open source hardware-simulation framework for:

- Development of complex software for embedded and IoT systems
- Architectural exploration and research
- Pre-silicon prototyping and HW-SW co-development

Features in brief:

- Plug-and-play building blocks
- Simulates system on many levels - CPUs, SoCs, peripherals, sensors, wired/wireless connection
- Flexible, deterministic and software-agnostic
- Continuous Integration-oriented
- “Batteries included” - lots of demos and binaries

- **Linux-Capable Platforms & Demos**

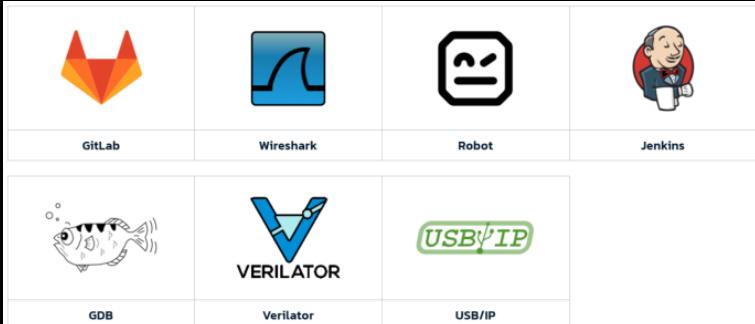
- Kendryte (Linux-nommu)
- LiteX/VexRiscv
- HiFive Unleashed (buildroot)
- PolarFire SoC Icicle Kit (Yocto/buildroot)
- Zedboard
- Vybrid
- Versatile
- Versatile Express
- Nvidia Tegra 3



## ■ Tested SWs

- Linux
- FreeRTOS
- Zephyr
- RIOT
- TensorFlow Lite
- Mbed
- NuttX
- Tock
- micropython
- Android
- Bare Metal software
- Contiki
- Wolfboot
- eCos
- Redboot
- and more!

## ■ Various integrations



■ **RENODE™**  **RISC-V®**

<https://risc-v-getting-started-guide.readthedocs.io/en/latest/>

## ■ Porting Renode to ARM

You may refer to my previous talks "**Python-based Open Source Toolchain for RISC-V Development**" at the 1st RISC-V World Conference China and the upcoming follow-ups like "**Renode: a Swiss Army Knife for RISC-V development**".

## IoT Friendly



- <https://github.com/renode/renode>  
a virtual development tool for multinode embedded networks (both wired and wireless) and is intended to enable a scalable workflow for creating effective, tested and secure IoT systems.
- **Protocol/Stack testing**  
OPC-UA, TSN, 6lowpan, Thread etc
- **Vedliot**  
<https://vedliot.eu/>  
<https://antmicro.com/blog/2021/01/vedliot-launch/>
- **Kenning**  
<https://antmicro.github.io/kenning/>  
<https://antmicro.com/blog/2021/06/kenning-edge-ai-framework/>
- ...

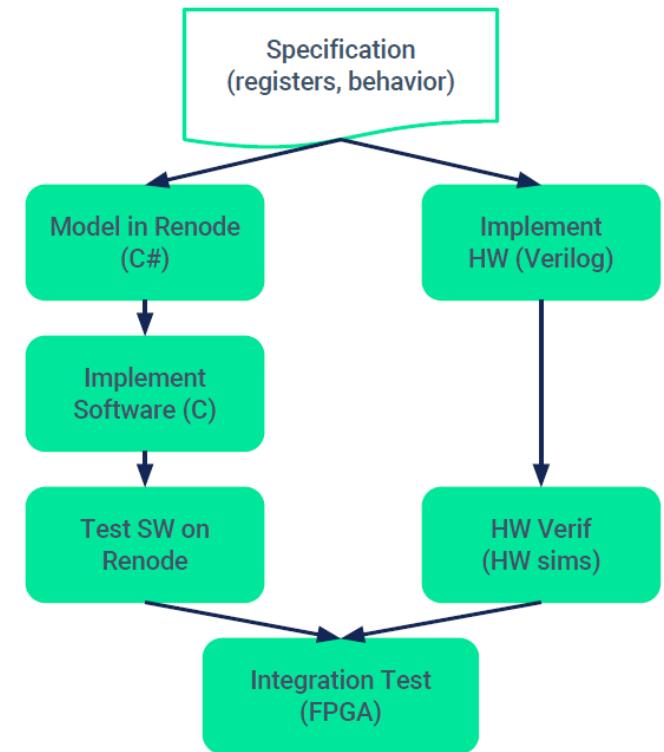


## Parallel HW/SW Development

- HW spec developed between HW and SW teams
- SW team implements spec in Renode and writes firmware against spec, testing on Renode
- In parallel, HW is implementing and testing HW design
- Integration test via FPGA and/or HW simulator

### EXAMPLE

HDMI device. We had SW working against spec, under Renode, in advance of HW.



Source: "Hardware/Software Co-Design with the Open Source Renode Framework and RISC-V", Michael Gieda, Getting Started With RISC-V NA Tour 2019.

- <https://renode.readthedocs.io/en/latest/tutorials/verilator-cosimulation.html>
- <https://renode.io/news/co-simulation-for-Zynq-with-Renode-and-Verilator/>
- ...



## 4.2.4 EDA in the Cloud

- <https://www.arm.com/company/news/2020/12/arm-moves-production-level-electronic-design-automation-to-the-cloud-with-the-help-of-aws>
- ...
- **Serverless Architecture**
- ...



## 4.2.5 Summary

- For details, you may refer to my presentations as below:
  - "Scala-based FOSS EDA on ARM" at the 5th China Functional Programming Meetup(Shanghai 2021)
  - "Python-based Open Source Toolchain for RISC-V Development" at the 1st RISC-V World Conference China(Shanghai 2021)
  - "Renode--a Swiss Army Knife for RISC-V development" at .Net Conf China 2021(Online)
  - "Rust for FOSS EDA" at OSDT China 2021(Online)
- And the upcoming follow-ups:
  - "Revisiting Python-based FOSS EDA", "Revisiting Scala-based FOSS EDA", "Revisiting Rust for FOSS EDA"...



## 4.3 ASIP, DSA, and CFU

### Overview

- [https://en.wikipedia.org/wiki/Hardware\\_acceleration](https://en.wikipedia.org/wiki/Hardware_acceleration)
  - <https://>
  - ...
-

## 4.3.1 ASIP

### Overview

- **Application-Specific Instruction Set Processor**
  - [https://en.wikipedia.org/wiki/Application-specific\\_instruction-set\\_processor](https://en.wikipedia.org/wiki/Application-specific_instruction-set_processor)
-

## 4.3.1.1 OpenASIP

### Overview

- <http://openasip.org/>

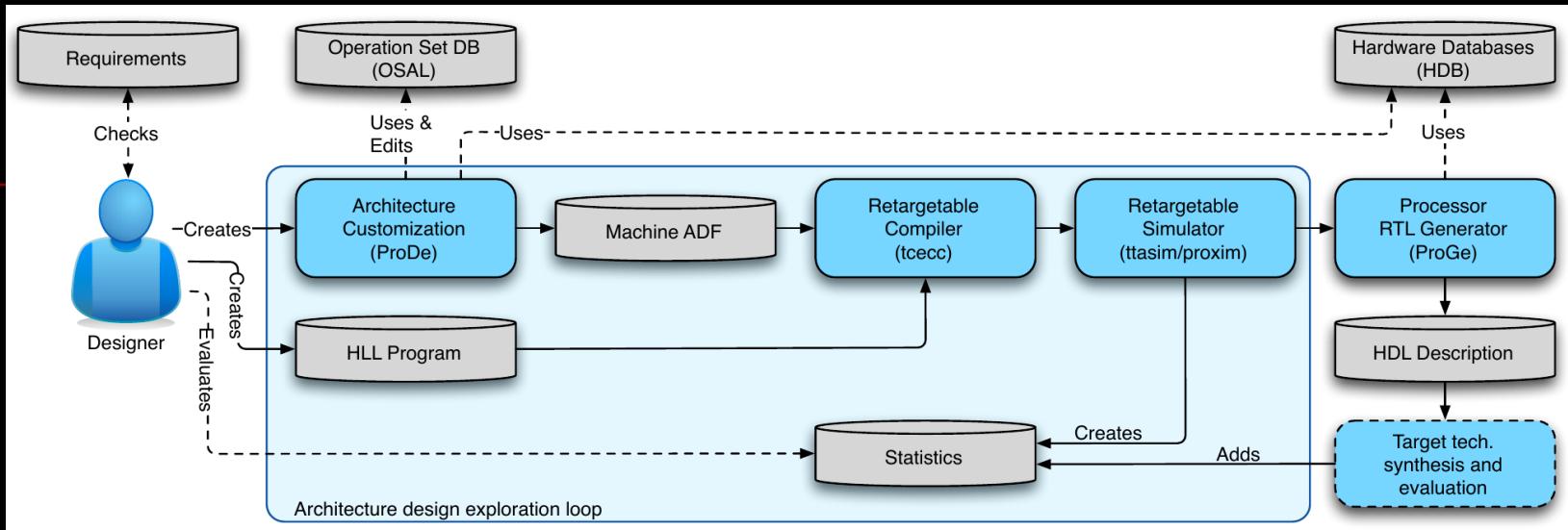
### TTA-based Co-design Environment (TCE) tools.

**TTA-based Co-Design Environment** (TCE) is an *open application-specific instruction-set toolset*. It can be used to design and program customized processors based on the energy efficient **Transport Triggered Architecture** (TTA). The toolset provides a complete retargetable co-design flow from high-level language programs down to synthesizable processor RTL (VHDL and Verilog backends supported) and parallel program binaries. Processor customization points include the register files, function units, supported operations, and the interconnection network.

### ■ Features

- Compiler:
  - LLVM based, Clang as the default frontend
  - OpenCL support via the [pcl](#) project
  - Basic block instruction scheduler (top-down and bottom-up)
  - Delay slot filling
  - Software bypassing
  - (experimental) Operand sharing
  - Custom operation support
  - Parallel TTA assembler
  - Software and hardware floating point support
  - Basic debugging info support
  - Multiple address space support
  - Support for native computation on half precision floats (fp16)
- Simulator:
  - Graphical and command line user interfaces
  - Interpretive debugging engine for cycle stepping
  - Static compiled engine for fast simulation with basic block granularity (but cycle count accuracy)
  - Dynamic compiled engine for improved startup time with fast simulation
  - SystemC integration API
- Processor and Program Image Generation:
  - Support for generating implementation for the designed processor as VHDL. Experimental support for Verilog.
  - Generates bit image of the program (supported formats include the Altera MIF)
  - Dictionary-based instruction compression
  - Automated generation of the files needed to integrate the core to different FPGA platforms.
  - IP-XACT 1.5 support
- Design space exploration:
  - Automated, manual and semi-automatic algorithm implementations
  - Tools that allow easy modification of the target architecture
  - Automated search of the connectivity design space
- Integrated Development Environment tools:
  - Graphical user interface (GUI) for editing architecture resources
  - GUI for editing operation set definitions

## Architecture and Design



## 4.3.2 DSA

### Overview

- Domain Specific Architecture
  - ...
-

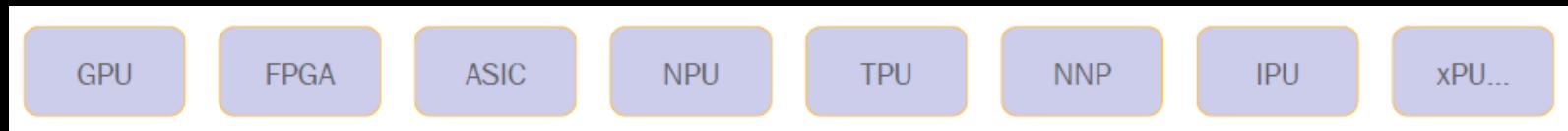
#### 4.3.2.1 ODSA

##### Overview

- **Open Domain-Specific Architecture**
- <https://www.opencompute.org/wiki/Server/ODSA>
- **Motivation**

**started from OCP Accelerator Module (OAM)**

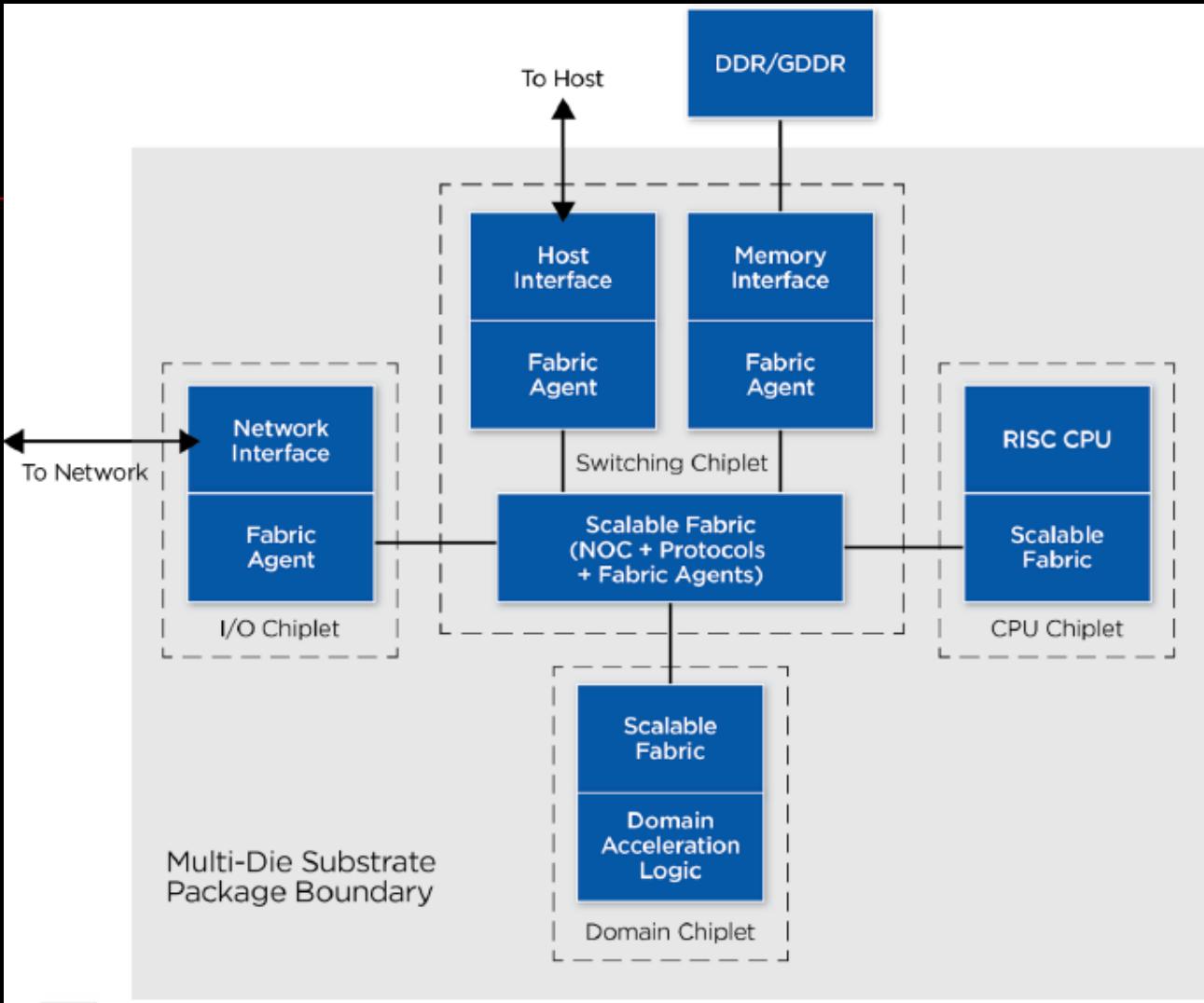
<https://www.opencompute.org/wiki/Server/OAI>



- Extending Moore's Law
  - Domain-Specific Architectures: Programmable ASICs to accelerate high-intensity workloads (e.g. Tensorflow, Network Flow Processor, Antminer...)
  - Chiplets: Build complex ASICs from multiple die, instead of as monolithic devices, to reduce development time/costs and manufacturing costs.
- Open Domain-Specific Architecture: An architecture to build domain-specific products
  - Today: All multi-chiplet products are based on proprietary interfaces
  - Tomorrow: Select best-of-breed chiplets from multiple vendors
  - Incubating a new group, to define a new open interface, build a PoC

Source: “ODSA: Technical Introduction”, Bapi Vinnakota, ODSA Project Workshop June, 2019.

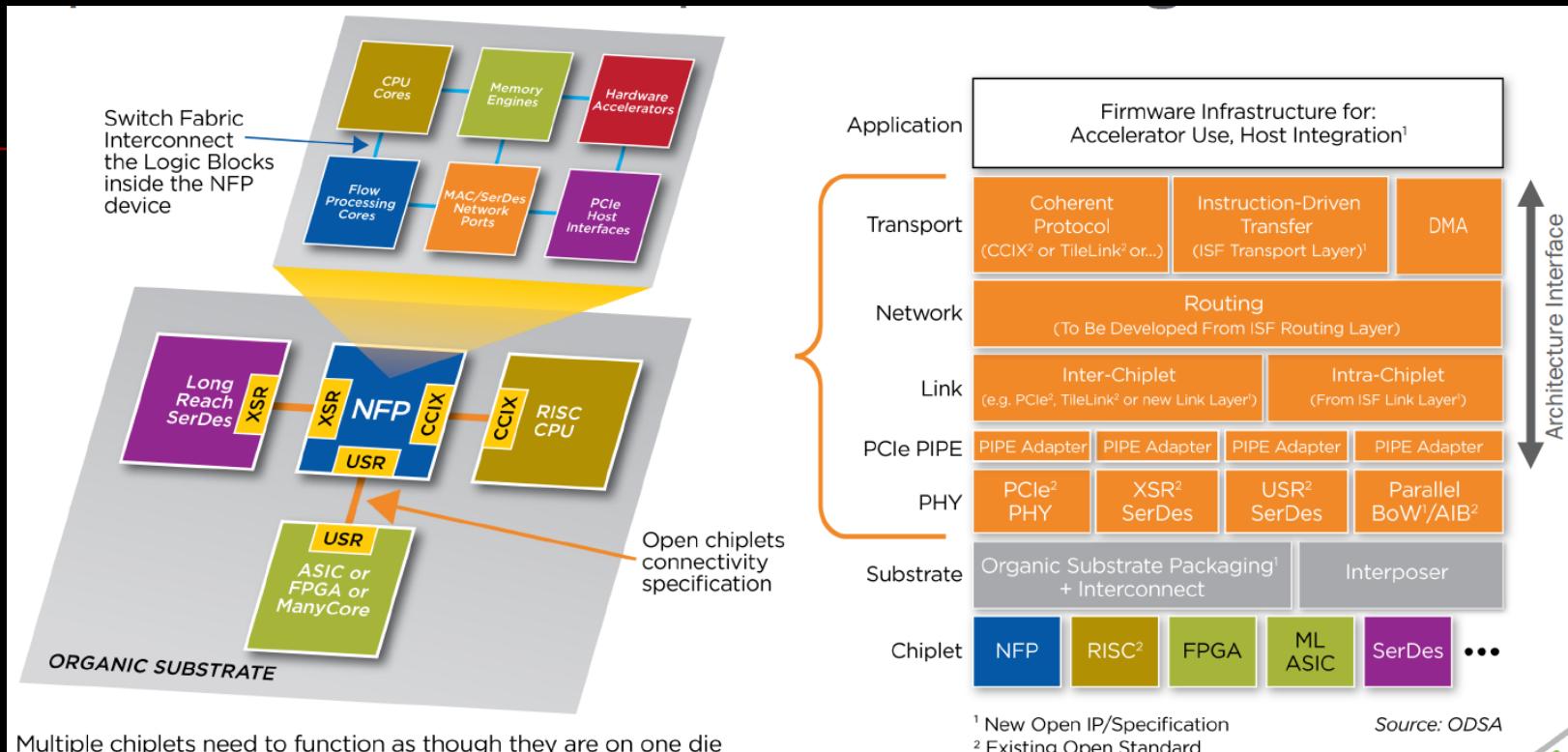
## Multi-chiplet Reference Architecture for DSA



Source: <http://files.opencompute.org/oc/public.php?service=files&t=c7d8ae0fc2f67ed540b27d4839f4740c&download>

## The ODSA Stack

### ■ Open Interface for Chiplet-Based Design



# Domain-Specific Architectures

## Tailor architecture to a domain\*

- Server-attached devices — programmable, not hardwired
- Integrated application and deployment-aware development of devices, firmware, systems, software
- 5-10X power performance improvement
- Big - more of a processor to I/O mismatch => more memory
- Each serves a smaller market

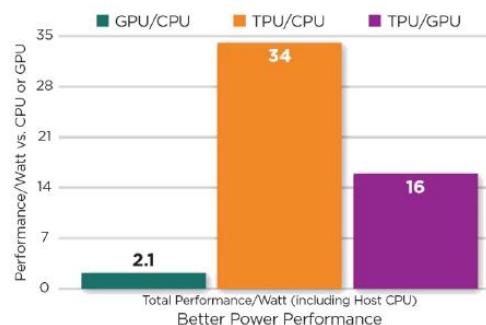
\*A New Golden Age for Computer Architecture

John L. Hennessy, David A. Patterson

Communications of the ACM, February 2019, Vol. 62 No. 2, Pages 48-60

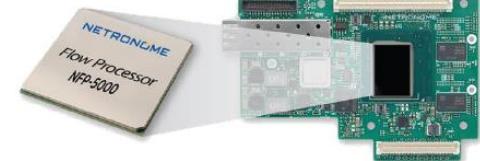


Domain-Specific for Machine Learning and AI

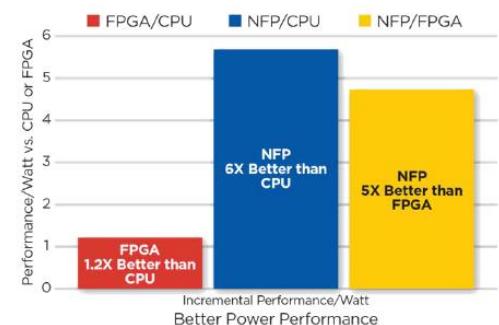


Google TPU vs. CPU and GPU

Source: "An in-depth look at Google's first Tensor Processing Unit (TPU)," Google Cloud, May 2017



Domain-Specific for Networking and Security



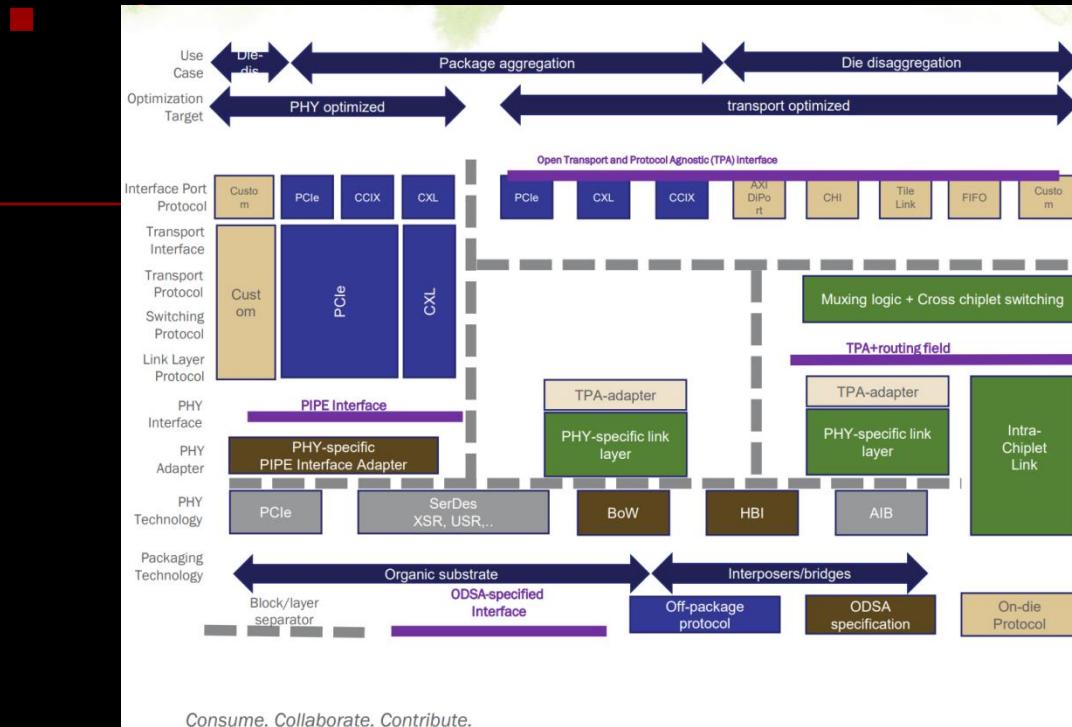
Netronome NFP vs. CPU and FPGA

Source: Netronome, based on internal benchmarks and industry reports related to Xeon CPUs and Arria FPGAs



Source: "ODSA: Technical Introduction", Bapi Vinnakota, ODSA Project Workshop June, 2019.

## ODSA D2D Interface

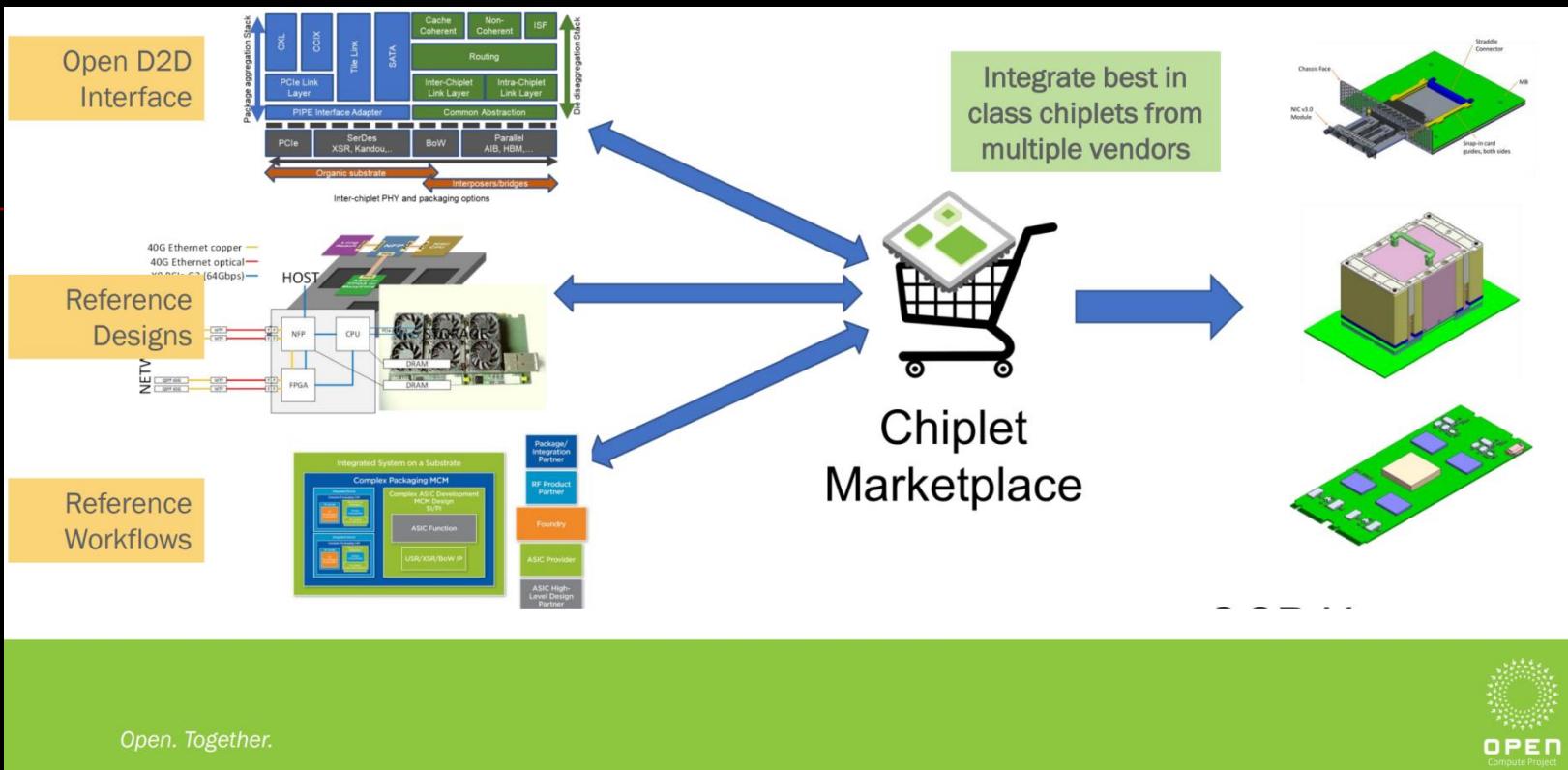


Open with abstraction layers

- Open logic, Open PHY evolve separately
- Reuse protocols, minimize “new”
- Multiple options to make chiplets, market will choose



# ODSA Charter



Source: [https://eps.ieee.org/images/files/Roadmap/The\\_Rise\\_of\\_Chiplets\\_Bapi\\_ODSA.pdf](https://eps.ieee.org/images/files/Roadmap/The_Rise_of_Chiplets_Bapi_ODSA.pdf)

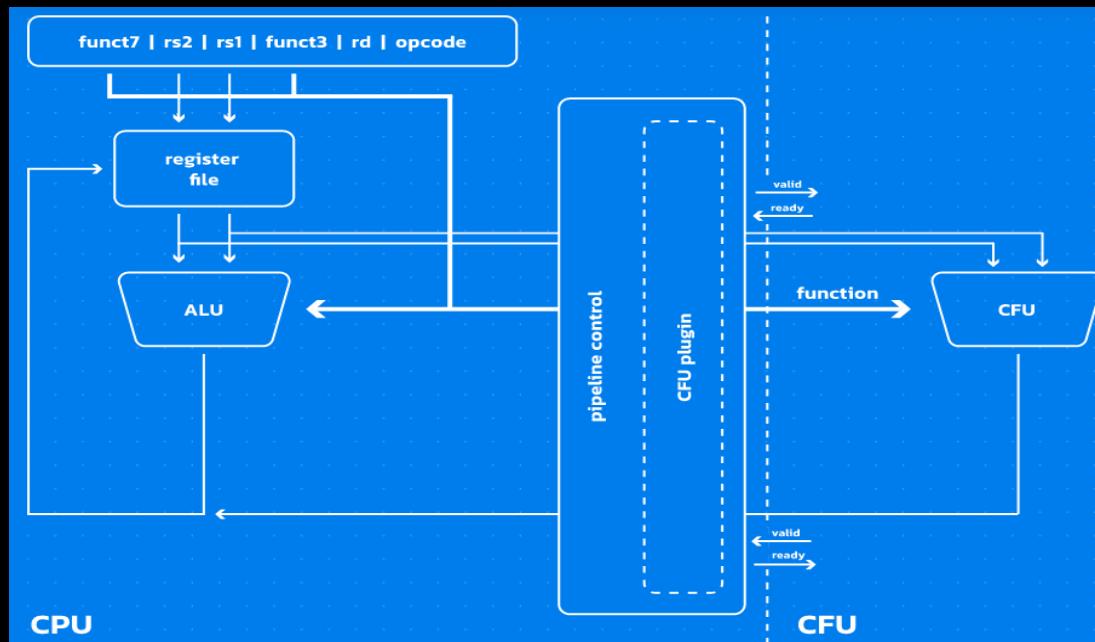


### 4.3.3 CFU

#### ■ Custom Function Units

<https://antmicro.com/blog/2021/09/cfu-support-in-renode/>

RISC-V is also excellent for FPGA-based ML development, offering a multitude of FPGA-friendly softcore options such as VexRiscv and specialized ML-oriented extensions called CFU - which you can experiment with both in cheap, easily accessible hardware as well as, you guessed it, with Renode, using Verilator co-simulation capabilities that we have described a few times already.



### 3.3.1 CFU Playground

- <https://cfu-playground.readthedocs.io/en/latest/>  
**Accelerate ML models on FPGAs.**

"CFU" stands for Custom Function Unit: accelerator hardware that is tightly coupled into the pipeline of a CPU core, to add new custom function instructions that complement the CPU's standard functions (such as arithmetic/logic operations).

The CFU Playground is a collection of software, gateware and hardware configured to make it easy to:

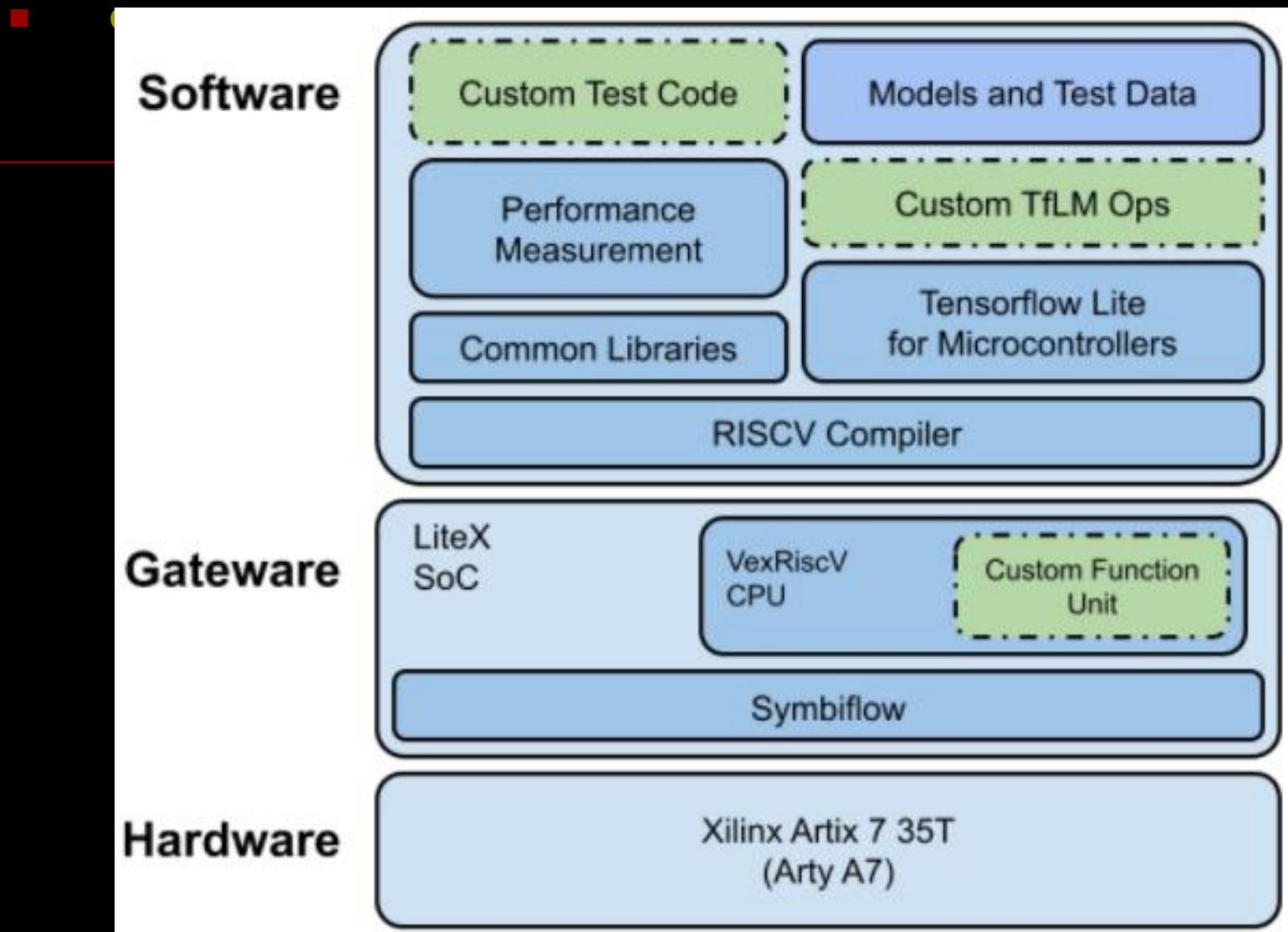
- Run ML models
- Benchmark and profile performance
- Make incremental improvements
  - In software by modifying source code
  - In gateware with a CFU
- Measure the results of changes

ML acceleration on microcontroller-class hardware is a new area, and one that, due to the expense of building ASICs, is currently dominated by hardware engineers. In order to encourage software engineers to join in the innovation, the CFU-Playground aims to make experimentation as simple, fast and fun as possible.

As well as being a useful tool for accelerating ML inferencing, the CFU Playground is a relatively gentle introduction to using FPGAs for computation.

...

## Architecture & Design





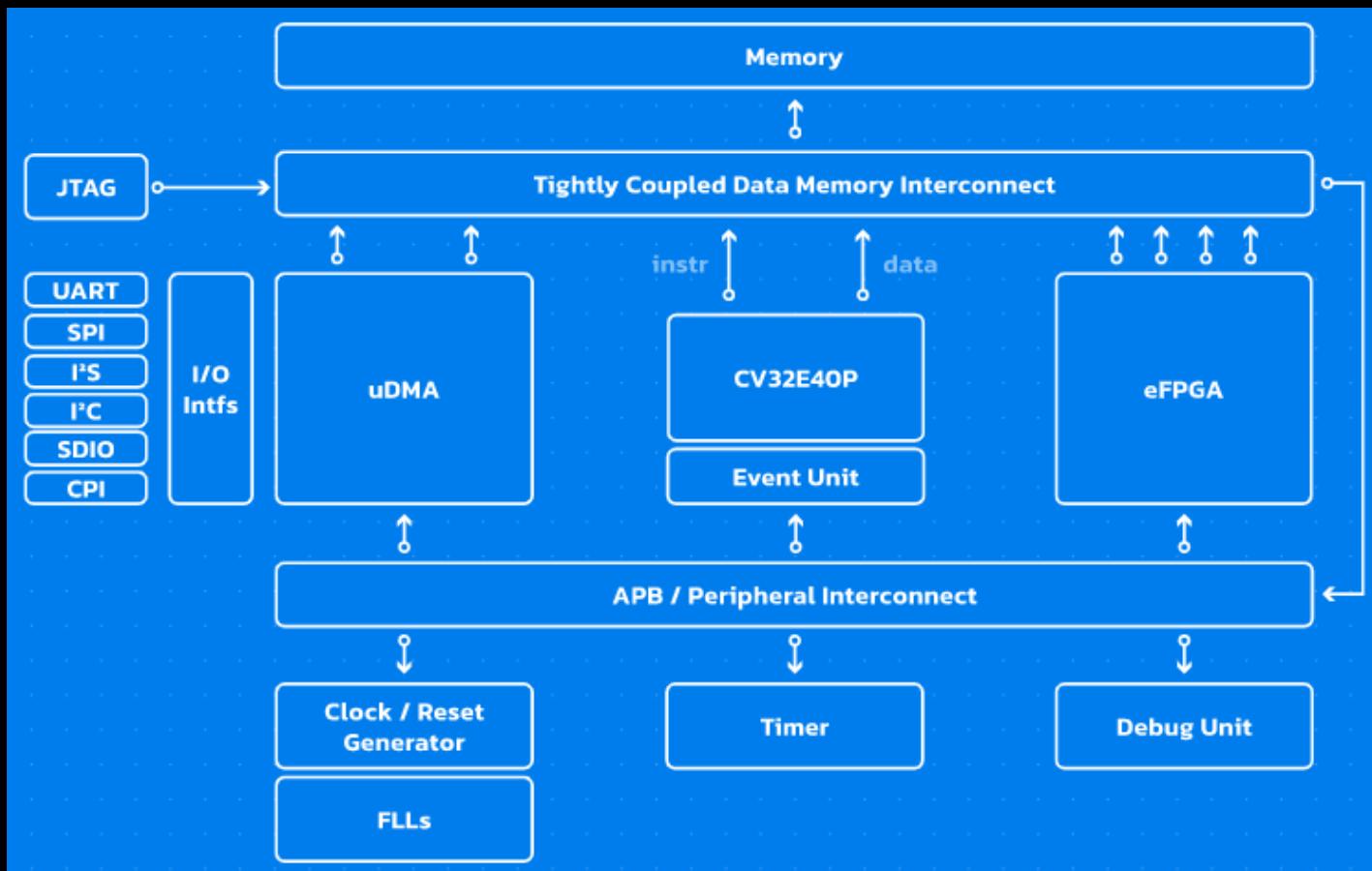
## 4.4 eFPGA

### Overview

- **embedded Field Programmable Grid Arrays**
- <https://www.edn.com/embedded-fpga-efpga-technology-past-present-and-future/>
- ...

## 4.4.1 Core-V

- <https://docs.openhwgroup.org/projects/core-v-mcu/>
- <https://github.com/openhwgroup/core-v-mcu>
- <https://antmicro.com/blog/2020/12/open-source-fpga-tools-and-renode-for-core-v/>



## Good Resources

- <https://semiwiki.com/efpga/>
- <https://www.menta-efpga.com/>
- <https://www.quicklogic.com/>
- <https://arxiv.org/abs/2112.09691>
- ...



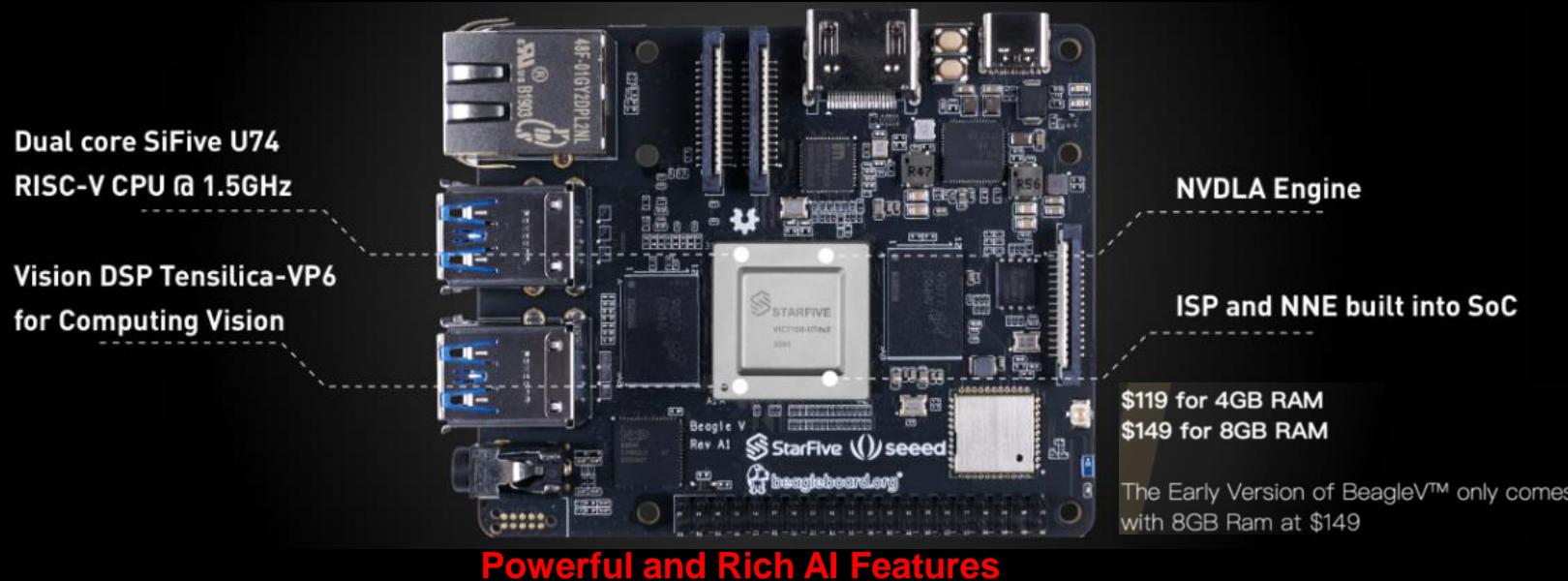


## 4.5 HW platform

### 4.5.1 Development Boards

#### 4.5.1.1 BeagleV(abandoned)

- <https://beaglev.org/>
- The First Affordable RISC-V Computer Designed to Run Linux



TRULY OPEN SOURCE



Open Source Software



Based on RISC-V Open Architecture

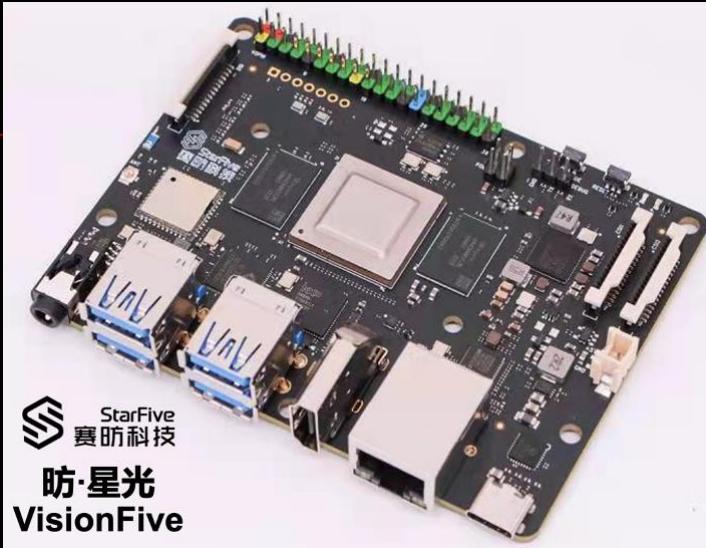


Open Hardware Design

- The successors are coming!

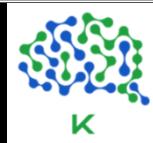
# VisionFive

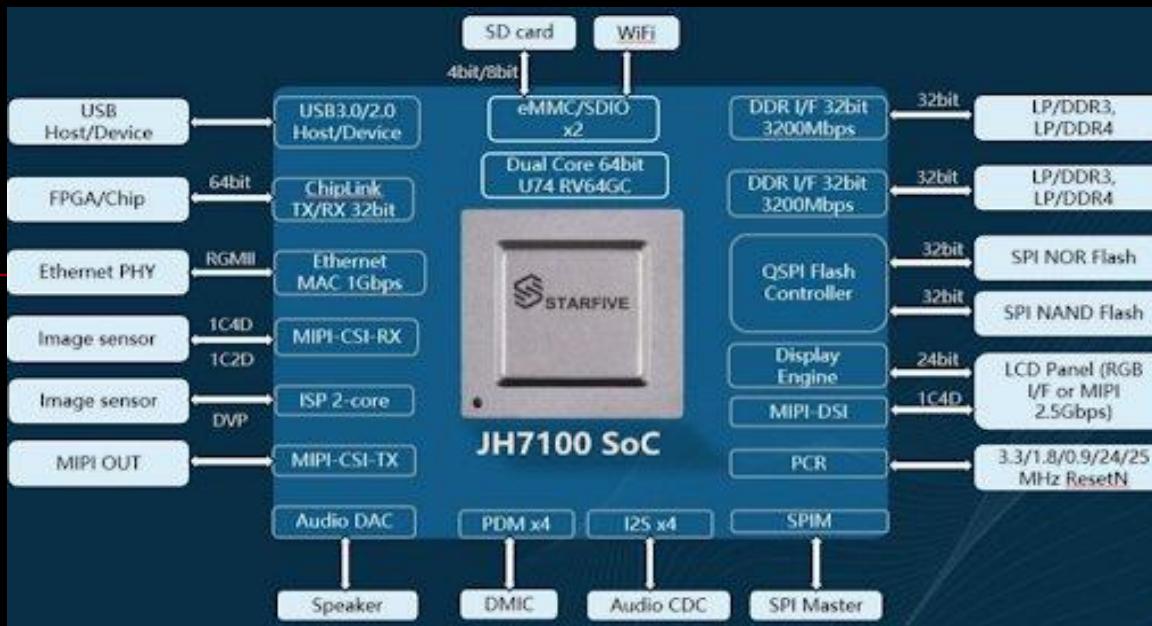
- base on StarFive's JH7100 vision processing SoC



## VisionFive V1 specifications:

- SoC – StarFive JH7100 Vision SoC with:
  - Dual-core Sifive U74 RISC-V processor @ 1.5 GHz with 2MB L2 cache
  - Vision DSP Tensilica-VP6 for computing vision
  - [NVDLA](#) Engine 1 core (configuration 2048 MACs @ 800MHz – 3.5 TOPS)
  - Neural Network Engine (1024MACs @ 500MHz – 1 TOPS)
  - VPU – H.264/H.265 decoder up to 4Kp60, dual-stream decoding up to 4Kp30
  - JPEG encoder/decoder
  - Audio Processing DSP and sub-system
- System Memory – 8GB LPDDR4
- Storage – MicroSD card slot
- Video output
  - 1x HDMI 1.4 port up to 1080p60
  - 1x MIPI DSI interface up to 4Kp30
  - MIPI-CSI TX for video output after ISP and AI processing
- Camera
  - Dual channels of ISP, each channel support up to 4K @ 30FPS
  - 2x MIPI-CSI Rx
- Audio – 3.5mm audio jack
- Connectivity – 1x Gigabit Ethernet, 2.4 GHz 802.11b/g/n WiFi 4, and Bluetooth 4.2
- USB – 4x USB 3.0 Ports
- Expansion – 40-pin color-coded GPIO header with 28x GPIO, I2C, I2S, SPI, UART
- Security – Support TRNG and OTP
- Misc – Reset and power buttons
- Power Supply – USB Type-C port **with USB PD and QC support**
- Dimensions – 100 x 72 mm





- <https://github.com/starfive-tech/VisionFive>
- [https://github.com/starfive-tech/JH7100\\_Docs](https://github.com/starfive-tech/JH7100_Docs)
- [https://rvspace.org/en/Product/VisionFive/Technical\\_Documents/VisionFive\\_Single\\_Board\\_Computer\\_Quick\\_Start\\_Guide](https://rvspace.org/en/Product/VisionFive/Technical_Documents/VisionFive_Single_Board_Computer_Quick_Start_Guide)
- <https://rvspace.org/>
- <https://forum.rvspace.org/t/how-to-purchase-visionfive/37/5>
- [https://www.phoronix.com/scan.php?page=news\\_item&px=Linux-5.17-StarFive](https://www.phoronix.com/scan.php?page=news_item&px=Linux-5.17-StarFive)
- ...

## II. eBPF-centric Lightweight Solution for HCI/EC

### 1) Design Goals and Principles

---

#### Goals

- An **eBPF-centric lightweight** Edge computing solution with high cost–performance ratio
- **Flexible and modular** architecture
- **Scale out**, not scale up
- Embracing the new technology trends for **GraalVM**, **Wasm**, **.Net**, **Rust**, **LLVM**, **RISC-V** etc
- Replacing **C++** and **Go** with **Rust** and a better system programming language?
- ...





## Principles

- **Fully open-sourced**
- **Reduce the dependencies for Go-based project to least**
- **Make project code reusable and self-contained as much as possible**
- **Gradually move to a HW-SW co-designed system**
- ...



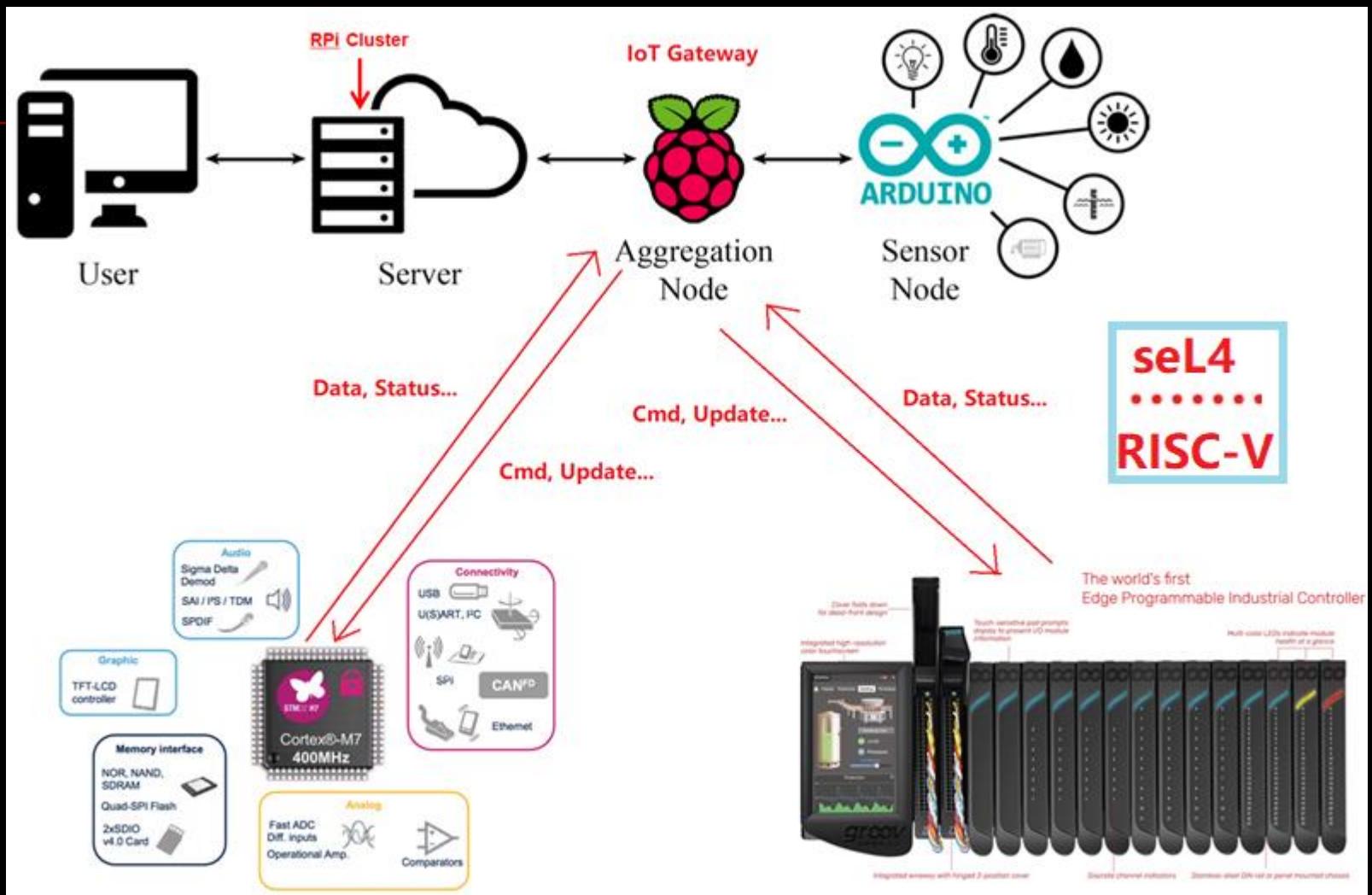
## Hardware Platform

- **The first priority is still ARM in current stage:**  
high cost–performance ratio development boards  
an increasingly mature ecosystem  
a lot of vendors to choose from  
lower power consumption  
FPGA vendors like Xilinx are using ARM as hardware cores  
in their Reconfigurable Computing platform  
ARM is ruling IoT and Embedded, and is making significant  
progress in the desktop and server market  
...
- **Also preparing a hybrid cluster with ARM, X86, and RISC-V.**
- **Working on the RISC-V based HW-SW co-designed system.**



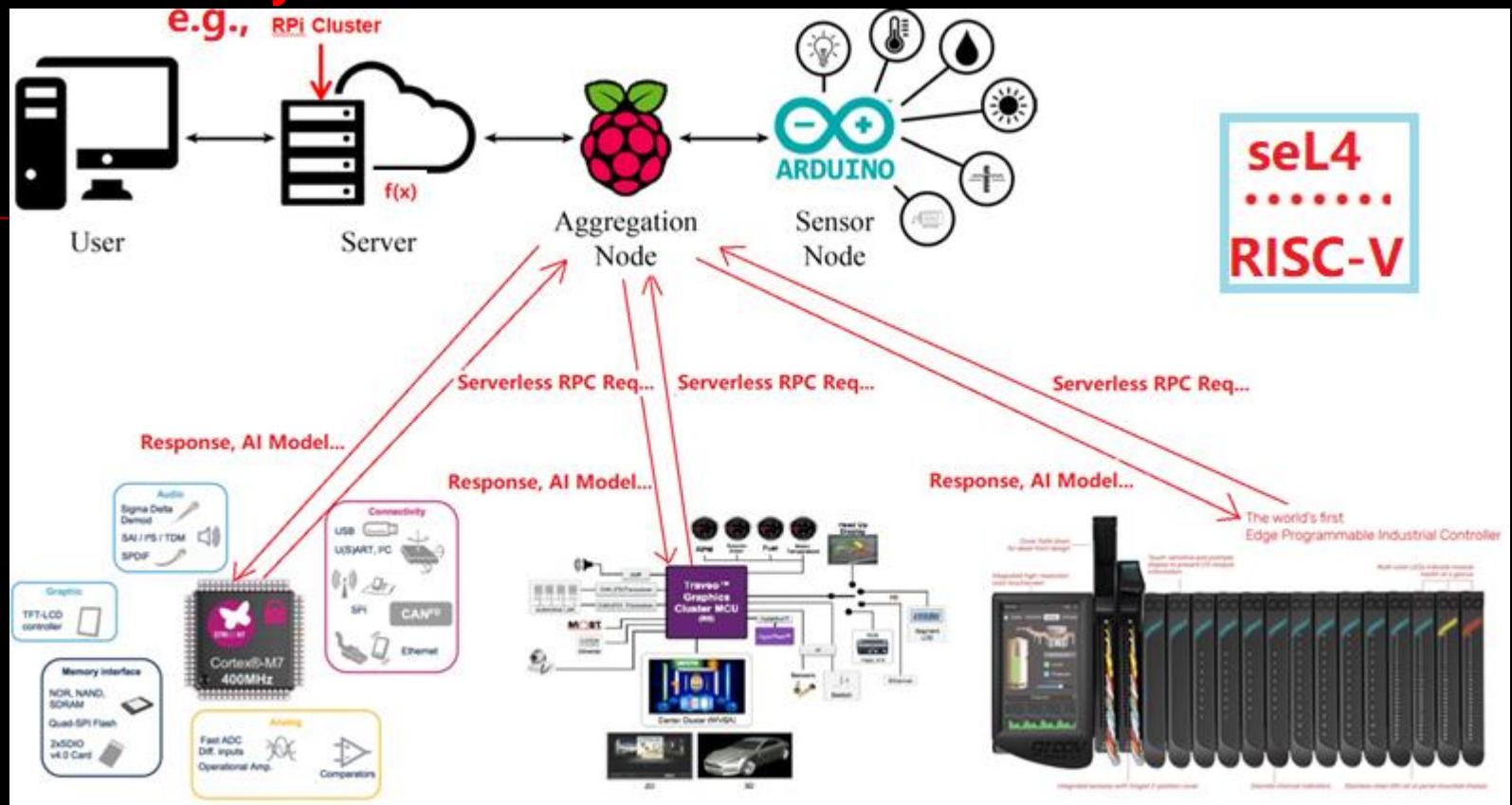
## Usage scenarios

### ■ General case





## Case of TinyML





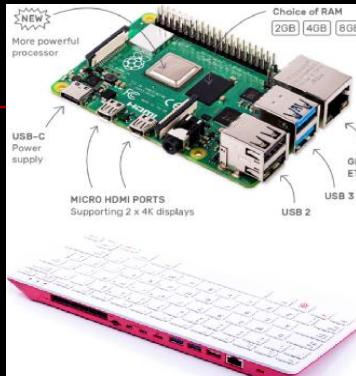
## 2) Overall Design

- Still emending and perfecting perpetually...
  - The new design should be coming in 2022 when the third discussion of this topic is ready.
-

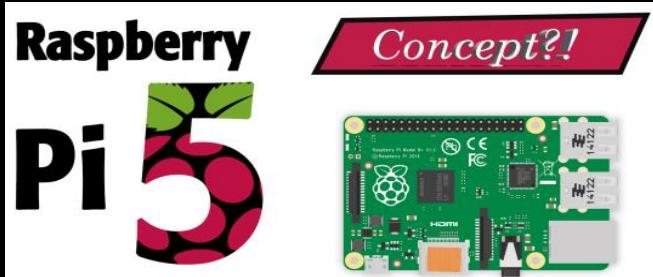
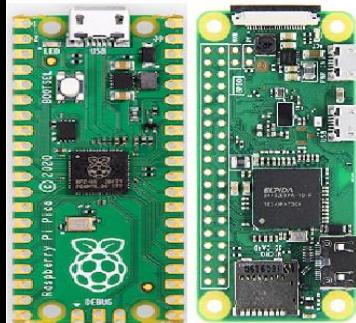


### 3) Testbed

#### 3.1 Raspberry Pi



Features/Specs	Raspberry Pi 4B	Raspberry Pi 3 B+
Release date	24th June 2019	14th March 2018
SoC	Broadcom BCM2711 quad-core Cortex-A72 @ 1.5 GHz	Broadcom BCM2837B0 quad-core Cortex-A53 @ 1.4 GHz
GPU	VideoCore VI with OpenGL ES 1.1, 2.0, 3.0	VideoCore IV with OpenGL ES 1.1, 2.0
Video Decode	H.265 4Kp60, H.264 1080p60	H.264 & MPEG-4 1080p30
Video Encode		H.264 1080p30
Memory	1GB, 2GB, or 4GB LPDDR4	1GB LPDDR2
Storage		microSD card
Video & Audio Output	2x micro HDMI ports up to 4Kp60 3.5mm AV port (composite + audio) MIPI DSI connector	1x HDMI 1.4 port up to 1080p60 3.5mm AV port (composite + audio) MIPI DSI connector
Camera		MIPI CSI connector
Ethernet	Native Gigabit Ethernet	Gigabit Ethernet over USB (300 Mbps max.)
WiFi		Dual band 802.11 b/g/n/ac
Bluetooth	Bluetooth 5.0 + BLE	Bluetooth 4.2 + BLE
USB	2x USB 3.0 + 2x USB 2.0	4x USB 2.0
Expansion		40-pin GPIO header
Power Supply	5V via USB type-C up to 3A 5V via GPIO header up to 3A Power over Ethernet via PoE HAT	5V via micro USB up to 2.5A 5V via GPIO header up to 3A Power over Ethernet via PoE HAT
Dimensions		85x56 mm
Default OS	Raspbian (after June 24, 2019)	Raspbian (after March 2018)
Price	\$35 (1GB RAM), \$45 (2GB RAM), \$55 (4GB RAM)	\$35 (1GB RAM)





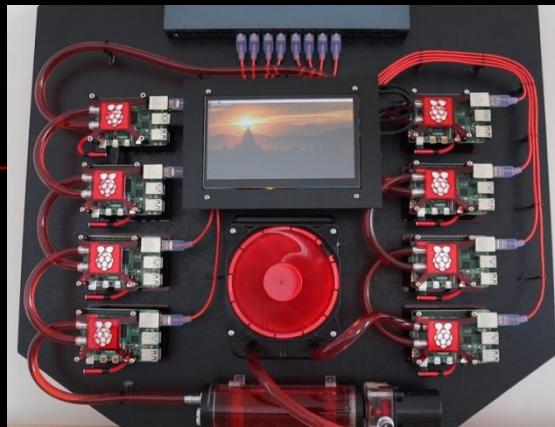
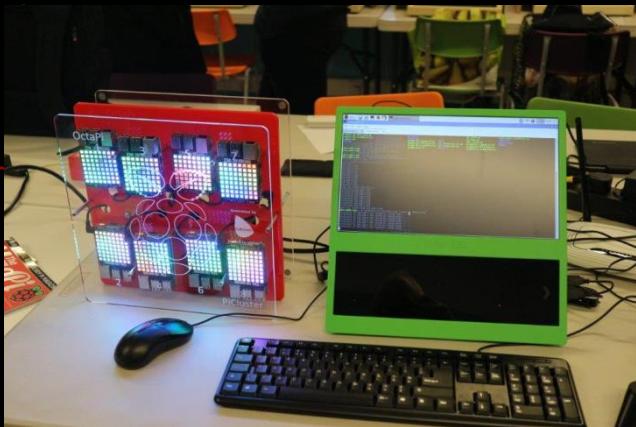
## Vulkan

- <https://www.raspberrypi.com/news/vulkan-update-version-1-1-conformance-for-raspberry-pi-4/>
- <https://qengineering.eu/install-vulkan-on-raspberry-pi.html>
- <https://github.com/karnkaul/rpi4-install-vulkan>
- [https://archive.fosdem.org/2021/schedule/event/rpi4\\_vulkan/](https://archive.fosdem.org/2021/schedule/event/rpi4_vulkan/)
- ...

## OpenCL

- ...

### 3.1.1 Clustering



Pico 20 Raspberry PI4 8GB  
\$ 758.00 \$ 689.00 Sale





## Turing Pi 2

- <https://turingpi.com/v2/>
- <https://turingpi.com/turing-pi-v2-is-here/>



The image displays the Turing Pi 2 mainboard on the left, followed by a central graphic with the product name and key specifications, and two different black cases on the right.

**TURING PI 2**

32 GB of RAM    SATA III interface    CM4 support

Layer 2 Managed Switch  
2 x Mini PCI Express  
2 x SATA III 6 Gbps

12 Gbps Backplane Bandwidth  
Cluster Management Bus (I2C)  
VLAN Support

### Self-hosted

Host cloud applications locally or at the edge



### Learning

Learn Kubernetes, Docker, Serverless



### Development

Build cloud-native and CI/CD for ARM edge infrastructure



### Network-Attached Storage

Connect up to 2 x 2.5" SSD storage



- The Turing Pi V2 now supports Raspberry Pi CM4, and Nvidia Jetson compute modules.



## 3.1.2 Applications

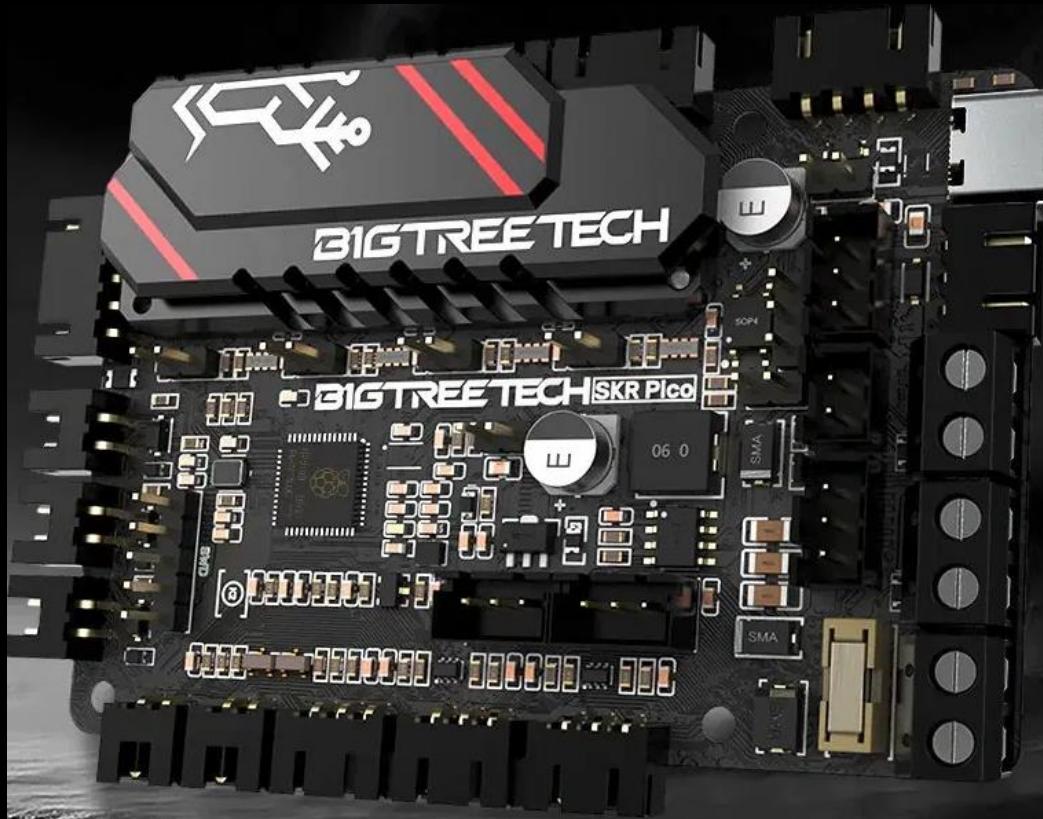
- o
-



## BTT SKR Pico

- <https://www.vorondesign.com/voron0.1>
- <https://github.com/VoronDesign>
- <https://www.cnx-software.com/2022/01/27/btt-skr-pico-a-raspberry-pi-rp2040-based-3d-printer-control-board/>

A Raspberry Pi RP2040 based 3D printer control board.





■ <https://vorondesign.com/voron2.4>



## 3.1.3 System

### 3.1.3.1 Fedora

- A Linux distribution developed by the community-supported Fedora Project which is sponsored primarily by Red Hat.

- [https://en.wikipedia.org/wiki/Fedora\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Fedora_(operating_system))
- <https://getfedora.org/>
- <https://alt.fedoraproject.org/alt/>
- <https://spins.fedoraproject.org/>
- <https://fedoraproject.org/wiki/Architectures/ARM>
- <https://fedoramagazine.org/>
- <https://silverblue.fedoraproject.org/>
- Developer friendly!

### Fedora 36

- [https://www.phoronix.com/scan.php?page=news\\_item&px=Fedora-36-Beta](https://www.phoronix.com/scan.php?page=news_item&px=Fedora-36-Beta)



After a slight delay the **Fedora 36** beta images are officially available today.

Beyond the usual assortment of bleeding-edge package updates, Fedora 36 beta brings changes such as defaulting to Noto fonts, a 128-bit IEEE long double ABI for IBM 64-bit POWER LE, replacing of FBDEV drivers with SimpleDRM, Wayland by default when using the NVIDIA proprietary driver, improvements to Cockpit, and much more.

On the package update front for Fedora 36, there is a near-final state of GCC 12 plus shipping Glibc 2.35, Autoconf 2.71, Golang 1.18, OpenJDK 17, LLVM 14, OpenSSL 3.0, PHP 8.1, Postman 4.0, PostgreSQL 14, Ruby on Rails 7.0, and more. On the desktop front is KDE Plasma 5.24, LXQt 1.0, and GNOME 42 as the default desktop experience of Fedora Workstation.

- <https://fedoraproject.org/wiki/Releases/36/ChangeSet>



## Fedora IoT

- <https://getfedora.org/iot/>

The screenshot shows the Fedora IoT landing page with several key features highlighted:

- Containerized applications**: Build, deploy, and manage your own applications with built-in Open Container Initiative (OCI) image support using podman or deploy containerized applications from popular public registries.
- Reliable operating system**: Fedora IoT uses [OSTree technology](#) to provide an immutable operating system with atomic updates. With the greenboot health check framework for systemd, administrators can ensure the system boots into the expected state. A timeline shows versions v1.0, v1.1, v1.2, and v1.3.
- Security in mind**: Security is important for IoT devices especially since they likely don't have the security of a data center. With support for TPM2, SecureBoot, and automated storage decryption with Clevis, Fedora IoT is built with a focus on security.
- Web-based provisioning**: With the Ignition provisioning utility and Zezere web service, administrators can deploy and configure Fedora IoT in a scalable manner without needing a physical console.
- Multiple architecture support**: IoT devices use a wide range of hardware and with Fedora IoT your software runs on any device without major changes. Fedora IoT is built for x86\_64, aarch64, and armhf processors in the same way with the same versions across all architectures.

- <https://docs.fedoraproject.org/en-US/iot/FIDO>

[https://www.phoronix.com/scan.php?page=news\\_item&px=Fedora-IoT-Better-Onboarding](https://www.phoronix.com/scan.php?page=news_item&px=Fedora-IoT-Better-Onboarding)



For Fedora 37 later this year the Linux distribution is looking at providing support for zero touch onboarding for IoT / edge devices.

Fedora IoT 37 is looking at making use of the FIDO Alliance's FIDO Device Onboard (FDO) protocol as an open standard for simply and securely onboarding devices to cloud and on-premise management platforms. The FIDO Alliance announced the FDO Protocol last year for easily onboarding IoT devices for simplifying device setup, allowing more flexible setups, and doing so securely.

This "zero touch onboarding" just relies on a device credential and a root and chain of trust to ensure onboarding of devices without stored credentials. There is a Rust language based implementation of the FIDO device onboarding stack that Fedora is looking to use and enable by default for the Fedora IoT Edition.

<https://fedoraproject.org/wiki/Changes/FIDODeviceOnboarding>

<https://github.com/fedora-iot/fido-device-onboard-rs>



### 3.1.3.2 Manjaro

- An open-source Linux distribution based on the Arch Linux operating system.
- [https://en.wikipedia.org/wiki/Manjaro\\_Linux](https://en.wikipedia.org/wiki/Manjaro_Linux)
- <https://distrowatch.com/>

Page Hit Ranking		
Rank	Distribution	HPD*
1	MX Linux	3369▼
2	Manjaro	2490▼
3	Mint	2172▼
4	Pop!_OS	1967▼
5	EndeavourOS	1643▲
6	Ubuntu	1378▼
7	Debian	1281▬
8	elementary	1149▼
9	Fedora	1018▼
10	openSUSE	890▲

■ Official  
■ Community  
■ ARM  
■ Development

Raspberry Pi 4

- Pinebook Pro
- Pinebook
- Raspberry Pi 4 Gnome 21.06
- Raspberry Pi 4 KDE Plasma 21.06
- Raspberry Pi 4 MATE 21.06
- Raspberry Pi 4 Minimal 21.06
- Raspberry Pi 4 Sway 21.06
- Raspberry Pi 4 XFCE 21.06
- Rock Pi 4B
- Rock Pi 4C
- RockPro64
- Rock64
- Khadas Vim 2
- Khadas Vim 3
- Odroid C4
- Odroid N2
- Odroid N2+
- Pine64 LTS
- Pine H64



- <https://manjaro.org>
- More and more developer friendly!



# III. Programming language and runtime

## 1) Rust for System Programming

### 1.1 Rust for eBPF

---

#### 1.1.1 Overview

- <https://kbknapp.dev/ebpf-part-i/>
- <https://kbknapp.dev/ebpf-part-ii/>
- <https://kbknapp.dev/ebpf-part-iii/>
- <https://kbknapp.dev/ebpf-part-iv/>
- ...



## 1.1.2 Aya

- <https://github.com/aya-rs/aya>

**An eBPF library for the Rust programming language, built with a focus on developer experience and operability.**

### ■ Features

Aya is an eBPF library built with a focus on operability and developer experience. It does not rely on `libbpf` nor `bcc` - it's built from the ground up purely in Rust, using only the `libc` crate to execute syscalls. With BTF support and when linked with musl, it offers a true [compile once, run everywhere solution](#), where a single self-contained binary can be deployed on many linux distributions and kernel versions.

Some of the major features provided include:

- Support for the **BPF Type Format (BTF)**, which is transparently enabled when supported by the target kernel. This allows eBPF programs compiled against one kernel version to run on different kernel versions without the need to recompile.
- Support for function call relocation and global data maps, which allows eBPF programs to make **function calls** and use **global variables and initializers**.
- **Async support** with both `tokio` and `async-std`.
- Easy to deploy and fast to build: aya doesn't require a kernel build or compiled headers, and not even a C toolchain; a release build completes in a matter of seconds.

- [https://lwn.net/Articles/859784/ //Aya: writing BPF in Rust](https://lwn.net/Articles/859784/)
- <https://www.linuxadictos.com/en/aya-the-first-library-to-create-ebpf-controllers-in-rust.html>
- ...



## Example

- Aya supports a large chunk of the eBPF API. The following example shows how to use a `BPF_PROG_TYPE_CGROUP_SKB` program with aya:

```
use std::fs::File;
use std::convert::TryInto;
use aya::Bpf;
use aya::programs::{CgroupSkb, CgroupSkbAttachType};

// load the BPF code
let mut bpf = Bpf::load_file("bpf.o")?;

// get the `ingress_filter` program compiled into `bpf.o`.
let ingress: &mut CgroupSkb = bpf.program_mut("ingress_filter")?.try_into()?;

// load the program into the kernel
ingress.load()?;

// attach the program to the root cgroup. `ingress_filter` will be called for all
// incoming packets.
let cgroup = File::open("/sys/fs/cgroup/unified")?;
ingress.attach(cgroup, CgroupSkbAttachType::Ingress)?;
```



## Good Resources

- <https://github.com/aya-rs/awesome-aya>
-



### 1.1.3 RedBPF

- <https://github.com/foniod/reedbpf>  
**Rust library for building and running BPF/eBPF modules.**

#### Overview

- The reedbpf project is a collection of tools and libraries to build eBPF programs using Rust. It includes:
  - `redbpf` - a user space library that can be used to load eBPF programs or access eBPF maps.
  - `redbpf-probes` - an idiomatic Rust API to write eBPF programs that can be loaded by the linux kernel
  - `redbpf-macros` - companion crate to `redbpf-probes` which provides convenient procedural macros useful when writing eBPF programs. For example, `#[map]` for defining a map, `# [kprobe]` for defining a BPF program that can be attached to kernel functions.
  - `cargo-bpf` - a cargo subcommand for creating, building and debugging eBPF programs



# Features

- - Allows users to write both BPF programs and userspace programs in Rust
  - Offers many BPF map types
    - i. `HashMap`, `PerCpuHashMap`, `LruHashMap`, `LruPerCpuHashMap`, `Array`, `PerCpuArray`, `PerfMap`, `TcHashMap`, `StackTrace`, `ProgramArray`, `SockMap`
  - Offers several BPF program types
    - i. `KProbe`, `KRetProbe`, `UProbe`, `URetProbe`, `SocketFilter`, `XDP`, `StreamParser`, `StreamVerdict`, `TaskIter`, `SkLookup`
  - Provides attribute macros that define various kind of BPF programs and BPF maps in a declarative way.
    - i. `#[kprobe]`, `#[kretprobe]`, `#[uprobe]`, `#[uretprobe]`, `#[xdp]`, `#[tc_action]`, `#[socket_filter]`, `# [stream_parser]`, `# [stream_verdict]`, `# [task_iter]`
    - ii. `#[map]`
  - Can generate Rust bindings from the Linux kernel headers or from the BTF of `vmlinux`
  - Provides API for both BPF programs and userspace programs to help users write Rust idiomatic code
  - Supports BTF for maps
  - Supports pinning maps and loading maps from pins
  - Supports BPF iterator for `task`
  - Enables users to write BPF programs for `tc` action and RedBPF compiles the programs into the ELF object file that is compatible with `tc` command
  - Provides wrappers of BPF helper functions
  - Offers asynchronous stream of `perf events` for userspace programs
  - Supports multiple versions of LLVM
  - Shows BPF verifier logs when loading BPF programs, BPF maps or BTF fails
  - Has several example programs that are separated into two parts: BPF programs and userspace programs



## 1.1.4 oxidebpf

- <https://github.com/redcanaryco/oxidebpf>

A permissive licensed Rust library for managing eBPF programs.

### Motivation

- The motivation behind `oxidebpf` is to create a permissive licensed Rust library for managing long-running eBPF programs that operate in as many environments as possible. Doing this required breaking some pre-set patterns on how eBPF applications are developed and deployed. We wanted to be able to easily deploy an eBPF solution that worked on as many distributions as possible; without forcing the user to have a tool-chain present. Users typically just want a product to do the thing - without a bunch of additional setup or maintenance. This library helped us realize that goal - and we are sharing it openly.

Initially this library meets our current eBPF requirements, so its not a fully flushed out eBPF implementation. Contributions are very much welcome, and we will slowly be adding to the feature list over time.

### Goals

- - Permissive licensed with no GPL dependencies.
  - Support custom CO-RE eBPF
  - Run eBPF programs on Linux 4.4+
  - Written in pure Rust, or as close to pure Rust as possible.
  - Minimal dependencies, pull in the bare minimum set of dependencies required to achieve our desired functionality.



## Demo

- <https://redcanary.com/blog/oxidebpf/>

## redcanary-ebpf-sensor

---

- <https://redcanary.com/blog/oxidebpf/>

## Good Resources

- <https://redcanary.com/blog/ebpf-for-security/>



## 1.1.5 rust-bpf

- <https://github.com/rust-bpf>

### rust-bcc

- <https://github.com/rust-bpf/rust-bcc>

Idiomatic Rust bindings for the BCC. The goal is to mimic the Python BCC bindings in <https://github.com/iovisor/bcc> in a Rusty way. The C BCC API (as exposed in bcc-sys) is very powerful, but it's fairly nontrivial to try to use it by itself and manage all the resources it allocates safely.

### rust-sys

- <https://github.com/rust-bpf/rust-sys>  
Rust binding for BCC.

### rust-tools

- <https://github.com/rust-bpf/rust-sys>  
A collection of BPF tools implemented in Rust.  
The goal is to provide tools like those found in the BCC repository.



## 1.1.6 lockc

### Overview

- [https://www.suse.com/c/rancher\\_blog/announcing-lockc-improving-container-security/](https://www.suse.com/c/rancher_blog/announcing-lockc-improving-container-security/)
- <https://rancher-sandbox.github.io/lockc/>

**lockc** is open source software for providing MAC (Mandatory Access Control) type of security audit for container workloads.

The main reason why **lockc** exists is that **containers do not contain**. Containers are not as secure and isolated as VMs. By default, they expose a lot of information about host OS and provide ways to "break out" from the container. **lockc** aims to provide more isolation to containers and make them more secure.

The [Containers do not contain](#) documentation section explains why we mean by that phrase and what kind of behavior we want to restrict with **lockc**.

The main technology behind lockc is [eBPF](#) - to be more precise, its ability to attach to [LSM hooks](#)

Please note that currently lockc is an experimental project, not meant for production environments. Currently we don't publish any official binaries or packages to use, except of a Rust crate. Currently the most convenient way to use it is to use the source code and follow the guide.

- <https://github.com/rancher-sandbox/lockc>



- <https://securityonline.info/lockc-ebpf-based-mac-security-audit-for-container-workloads/>



## 2) Unified runtime for eBPF and Wasm

### 2.1 Status of GraalVM-based Userland

Please refer to my previous talks and upcoming follow-ups.

#### 1. "GraalVM-based unified runtime for eBPF & Wasm" at GOTC 2021

(Shenzhen), on GraalVM CE Java11-21.2.0:

- Successfully ported project **BPF-Graal-Truffle**(<https://github.com/mattmurante/bpf-graal-truffle>) except for AOT part and some test cases.
- Successfully built **uBPF**(<https://github.com/iovisor/ubpf>) to LLVM bitcode and run on GraalVM.
- Successfully run some basic tests with the official solution **GraalWasm**(<https://github.com/oracle/graal/tree/master/wasm>), but failed with the Polyglot case...

#### Issues

- Most of the uBPF implementations do not support ARM.
- The GraalVM LLVM toolchain does not work as expected for some cases.



## 2. "Revisiting GraalVM-based unified runtime for eBPF & Wasm" at OpenInfra Days China 2021(Beijing), on GraalVM CE Java11-21.3.0-dev:

- Failed to run project rBPF(<https://github.com/qmonnet/rbpf>) together with Solana rBPF(<https://github.com/solana-labs/rbpf>) on GraalVM.  
May need to hack **rustc/cargo** as the blocking issue is that Rust projects are not built in a **finely-grained** way for generating the required LLVM bitcode file against each individual source code file.
- Successfully make **wasm3**(<https://github.com/wasm3/wasm3>) to run on GraalVM.

### Issues

- Rust projects are not GraalVM friendly.
- Confirmed that there is something wrong with the **GraalVM LLVM toolchain** from the official GraalVM CE releases, especially the linker.



2. The upcoming "The 3rd round discussion on GraalVM-based unified runtime for eBPF & Wasm" will contain the following sub-topics:

- Try to make the official P4 runtime(<https://github.com/p4lang>) run on GraalVM.
- Try to make WAMR(<https://github.com/bytecodealliance/wasm-micro-runtime>) run on GraalVM.
- ...



## 2.2 Ideas for Unified eBPF & Wasm runtime

### SuperVM (Register-based)

- A "superset" of eBPF & compatible with Wasm at bytecode level (for both user and kernel space)
  - 1. backward compatible with eBPF bytecodes
  - 2. Wasm bytecodes could be converted to that of SuperVM (just similar to convert Java bytecode for the Dalvik VM on Android)
  - 3. hardware-friendly
  - ...
- A superset of WASI (for user space)
  - 1. extended system interface for both uBPF & Wasm
  - 2. better support heterogeneous parallel computing
  - ...
- Three schemes are being considered for the in-kernel SuperVM
  - 1. C based
  - 2. DSL based
  - 3. Rust based
  - ...
- The related discussions for the above topic will come in 2022.



### 3) Smart Runtime

#### 3.1 What is it (from our point of view)

**Traditional Runtime + AI + eBPF?**

- XPU-aware compilation
- Auto-scheduling
- Auto-tuning
- Auto-debugging
- ...



## 3.2 Robot Programmer

### 3.2.1 Overview

- ...
-



## 3.2.1.1 CodeX

### ■ [https://en.wikipedia.org/wiki/OpenAI\\_Codex](https://en.wikipedia.org/wiki/OpenAI_Codex)

OpenAI Codex is an artificial intelligence model developed by OpenAI. It parses natural language and generates code in response. It is used to power GitHub Copilot, a programming autocomplete tool developed for Visual Studio Code.<sup>[1]</sup> Codex is a descendant of OpenAI's GPT-3 model, fine-tuned for use in programming applications.

OpenAI has released an API for Codex in closed beta.<sup>[1]</sup>

#### Capabilities [edit]

Based on GPT-3, a neural network trained on text, Codex has additionally been trained on 159 gigabytes of Python code from 54 million GitHub repositories.<sup>[2][3]</sup> A typical use case of Codex is typing a comment, such as "`//compute the moving average of an array for a given window size`", then using the AI to suggest a block of code satisfying that prompt.<sup>[4]</sup> OpenAI has stated that Codex can complete approximately 37% of requests and is meant to make human programming faster rather than replace it; according to OpenAI's blog, Codex excels most at "mapping [...] simple problems to existing code", which they describe as "probably the least fun part of programming".<sup>[5][6]</sup> Jeremy Howard, co-founder of Fast.ai, stated that "[Codex] is a way of getting code written without having to write as much code" and that "it is not always correct, but it is just close enough".<sup>[7]</sup> According to a paper written by OpenAI researchers, when attempting each test case 100 times, 70.2% of prompts had working solutions.<sup>[8]</sup>

OpenAI claims that Codex is able to function in over a dozen programming languages, including Go, JavaScript, Perl, PHP, Ruby, Shell, Swift, and TypeScript, though it is most effective in Python.<sup>[1]</sup> According to *VentureBeat*, demonstrations uploaded by OpenAI showed impressive coreference resolution capabilities. The demonstrators were able to create a browser game in JavaScript and generate data science charts using matplotlib.<sup>[6]</sup>

OpenAI has shown that Codex is able to interface with services and apps such as Mailchimp, Microsoft Word, Spotify, and Google Calendar.<sup>[6][9]</sup> Microsoft is reportedly interested in exploring Codex's capabilities.<sup>[9]</sup>

#### Issues [edit]

OpenAI demonstrations showcased flaws such as inefficient code and one-off quirks in code samples.<sup>[6]</sup> In an interview with *The Verge*, OpenAI chief technology officer Greg Brockman said that "sometimes [Codex] doesn't quite know exactly what you're asking" and that it can require some trial and error.<sup>[9]</sup> OpenAI researchers found that Codex struggles with multi-step and higher-level prompts, often failing or yielding counter-intuitive behavior. Additionally, they brought up several safety issues, such as over-reliance by novice programmers, biases based on the training data, and security impacts due to vulnerable code.<sup>[8]</sup>

*VentureBeat* has stated that because Codex is trained on public data, it could be vulnerable to "data poisoning" via intentional uploads of malicious code.<sup>[6]</sup> According to a study by researchers from New York University, approximately 40% of code generated by GitHub Copilot (which uses Codex) included glitches or other exploitable design flaws.<sup>[10]</sup>

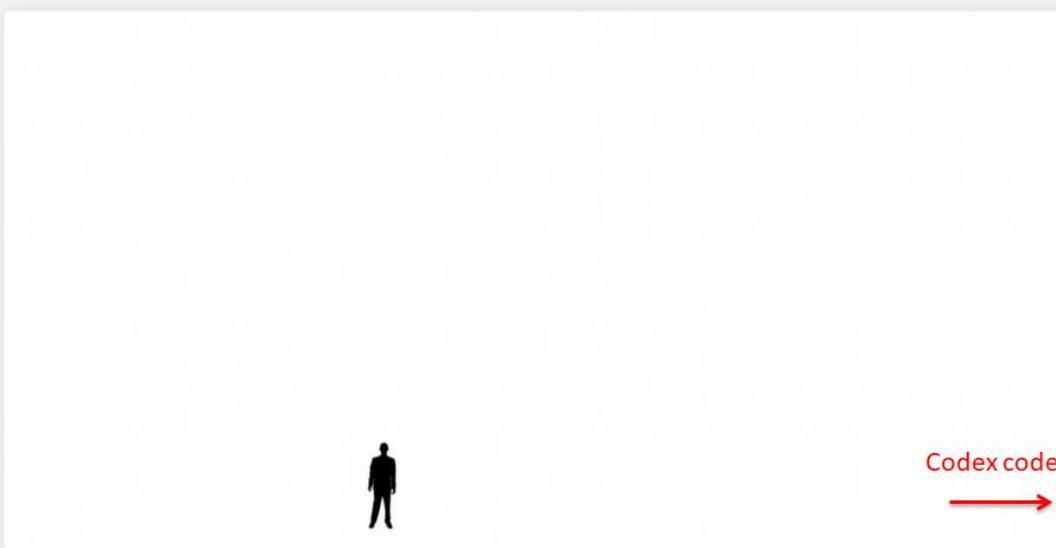
The Free Software Foundation has expressed concerns that code snippets generated by Copilot and Codex could unknowingly violate the terms of free software licenses, such as the GPL, which requires derivative works to be licensed under equivalent terms.<sup>[11]</sup> Issues they raised include whether training on public repositories falls into fair use or not, how developers could discover infringing generated code, whether trained machine learning models could be considered modifiable source code or a compilation of the training data, and if machine learning models could themselves be copyrighted and by whom.<sup>[11][12]</sup> An internal GitHub study found that approximately 0.1% of generated code contained direct copies from the training data. One specific example has been raised, in which the model outputted the original code of the fast inverse square root algorithm, including comments and an incorrect copyright notice.<sup>[4]</sup>

In response, OpenAI has stated that "legal uncertainty on the copyright implications of training AI systems imposes substantial costs on AI developers and so should be authoritatively resolved."<sup>[4]</sup> The copyright issues with Codex have been compared to the *Authors Guild, Inc. v. Google, Inc.* court case, in which judges ruled that Google Books's use of text snippets from millions of scanned books constituted fair use.<sup>[4][13]</sup>

## ■ Use Cases

- Instantly create memes
- Develop games
- Allowing contactless control for people with disabilities
- Create programs using languages other than English
- Solve math word problems
- Auto-generating captions for videos (speech-to-text)
- Building personal websites
- Creating login forms
- Analyzing data in natural language
- Change the way websites look when browsing online
- Creating web scrapers
- Making three-dimensional (3D) graphics
- Altering augmented reality (AR)

Source: <https://www.techslang.com/can-openais-codex-transform-programming/>



Codex code



Now make it controllable with the left and right arrow keys.

↑ Command



```
u=https%3A%2Fwww.publicdomainp  
ictures.net%2Fpictures%2F60000%2F  
velka%2Fbusiness-man-black-  
silhouette.jpg&f=1&nofb=1';  
document.body.appendChild(image);
```

```
/* Make the person 100 pixels. */  
image.style.width = '100px';
```

```
/* Set its position to 500 pixels  
down, and 400 pixels from the  
left. */  
image.style.position =  
'absolute';  
image.style.left = '400px';  
image.style.top = '500px';
```

```
/* Now make it controllable with  
the left and right arrow keys. */  
document.addEventListener('keydow  
n', function(e) {  
  if (e.keyCode == 37) {  
    image.style.left =  
(parseInt(image.style.left) - 10)  
+ 'px';  
  } else if (e.keyCode == 39) {  
    image.style.left =  
(parseInt(image.style.left) + 10)  
+ 'px';  
  }  
});
```

Source: <https://towardsdatascience.com/heres-how-openai-codex-will-revolutionize-programming-and-the-world-e8432aafc5f7>



# Copilot

- <https://copilot.github.com/>

Powered by  OpenAI

Trained on billions of lines of public code, GitHub Copilot puts the knowledge you need at your fingertips, saving you time and helping you stay focused.



### Extends your editor

GitHub Copilot is available as an extension for Neovim, JetBrains, and Visual Studio Code. You can use the GitHub Copilot extension on your desktop or in the cloud on [GitHub Codespaces](#). And it's fast enough to use as you type.



### Speaks all the languages you love

GitHub Copilot works with a broad set of frameworks and languages. The technical preview does especially well for Python, JavaScript, TypeScript, Ruby, Java, and Go, but it understands dozens of languages and can help you find your way around almost anything.



### You're the pilot

With GitHub Copilot, you're always in charge. You can cycle through alternative suggestions, choose which to accept or reject, and manually edit suggested code. GitHub Copilot adapts to the edits you make, matching your coding style.

## How it works





## Good Resources

- <https://openai.com/blog/openai-codex/>
- <https://spectrum.ieee.org/openai-wont-replace-coders>
- <https://gpt3demo.com/apps/openai-codex>
- ~~<https://medium.com/geekculture/creating-code-with-artificial-intelligence-good-or-bad-7ff27b5866d1>~~
- <https://www.theverge.com/2021/8/10/22618128/openai-codex-natural-language-into-code-api-beta-access>
- <https://towardsdatascience.com/heres-how-openai-codex-will-revolutionize-programming-and-the-world-e8432aafc5f7>
- <https://aiexpress.io/experimenting-with-openai-codex/>
- ...



## 3.2.1.2 AlphaCode

- <https://alphacode.deepmind.com/>
- <https://deepmind.com/blog/article/Competitive-programming-with-AlphaCode>

1

### Problem (input)

#### D.Backspace

You are given two strings  $s$  and  $t$ , both consisting of lowercase English letters. You are going to type the string  $s$  character by character, from the first character to the last one.

When typing a character, instead of pressing the button corresponding to it, you can press the "Backspace" button. It deletes the last character you have typed among those that aren't deleted yet (or does nothing if there are no characters in the current string). For example, if  $s$  is "abcbd" and you press Backspace instead of typing the first and the fourth characters, you will get the string "bd" (the first press of Backspace deletes no character, and the second press deletes the character 'c'). Another example, if  $s$  is "abcaa" and you press Backspace instead of the last two letters, then the resulting text is "a".

Your task is to determine whether you can obtain the string  $t$ , if you type the string  $s$  and press "Backspace" instead of typing several (maybe zero) characters of  $s$ .

#### Input

The first line contains a single integer  $q$  ( $1 \leq q \leq 10^5$ ) — the number of test cases.

The first line of each test case contains the string  $s$  ( $1 \leq |s| \leq 10^5$ ). Each character of  $s$  is a lowercase English letter.

The second line of each test case contains the string  $t$  ( $1 \leq |t| \leq 10^5$ ). Each character of  $t$  is a lowercase English letter.

It is guaranteed that the total number of characters in the strings over all test cases does not exceed  $2 \cdot 10^5$ .

#### Output

For each test case, print "YES" if you can obtain the string  $t$  by typing the string  $s$  and replacing some characters with presses of "Backspace" button, or "NO" if you cannot.

You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

#### Input

4
ababa
ba
ababa
bb
aaa
aaaa
aababa
ababa

#### Output

YES
NO
NO
YES

#### Note

Consider the example test from the statement.

In order to obtain "ba" from "ababa", you may press Backspace instead of typing the first and the fourth characters.

There's no way to obtain "bb" while typing "aababa".

There's no way to obtain "aaaa" while typing "aaa".

In order to obtain "ababa" while typing "aababa", you have to press Backspace instead of typing the first character, then type all the remaining characters.



②

## Solution (output)

```
t=int(input())
for i in range(t):
    s=input()
    t=input()
    a=[]
    b=[]
    for j in s:
        a.append(j)
    for j in t:
        b.append(j)
    a.reverse()
    b.reverse()
    c=[]
    while len(b)!=0 and len(a)!=0:
        if a[0]==b[0]:
            c.append(b.pop(0))
            a.pop(0)
        elif a[0]==b[0] and len(a)==1:
            a.pop(0)
            a.pop(0)
        elif a[0]==b[0] and len(a)!=1:
            a.pop(0)
            a.pop(0)
        else:
            print("NO")
    if len(b)==0:
        print("YES")
    else:
        print("NO")
```

First AlphaCode reads the two phrases.

Backspace deletes two letters. The letter you press backspace instead of, and the letter before it.

If the letters at the end of both phrases don't match, the last letter must be deleted. If they do match we can move onto the second last letter and repeat.

If we've matched every letter, it's possible and we output that.

①

AlphaCode is presented with a problem, in this case to figure out if it's possible to convert one phrase to another by pressing backspace instead of typing some letters.

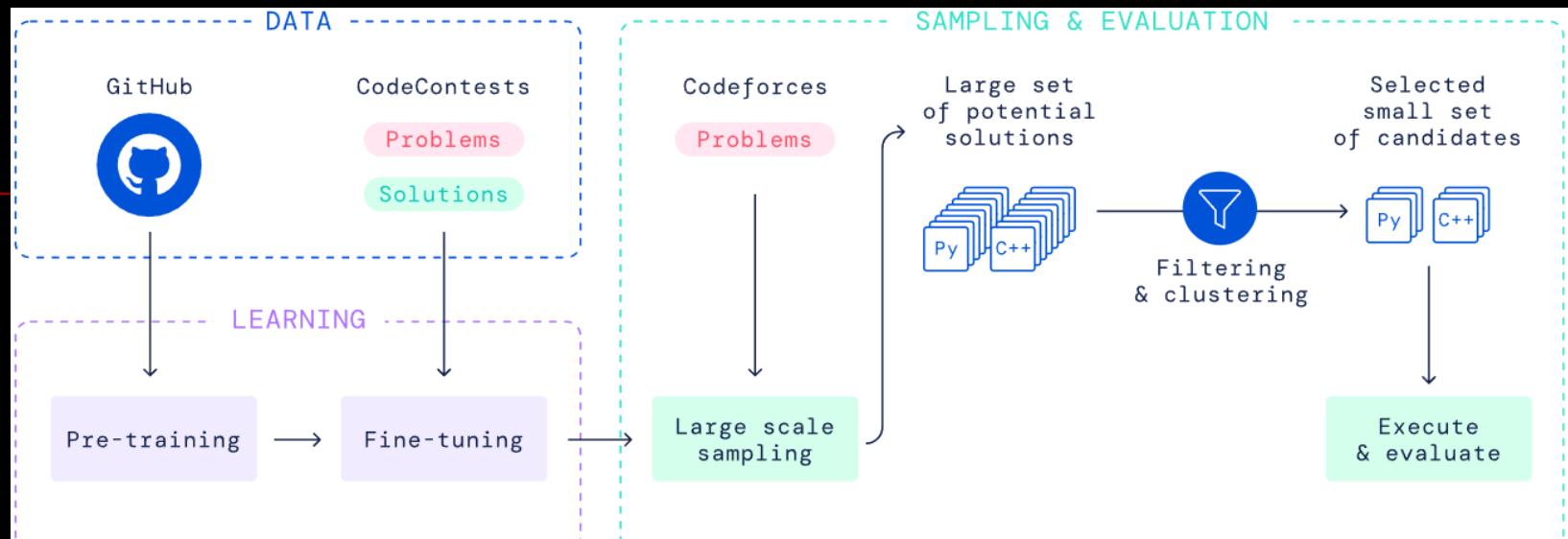
②

AlphaCode reads the whole problem statement and produces code, analogous to how a human would approach the problem by reading it, coding a solution, and submitting.



## How it works

- 



Source: <https://arxiv.org/pdf/2203.07814.pdf>



## 3.2.2 AI-assisted Development

### 3.2.2.1 PolyCoder

- <https://github.com/VHellendoorn/Code-LMs>

Guide to using pre-trained large language models of source code.

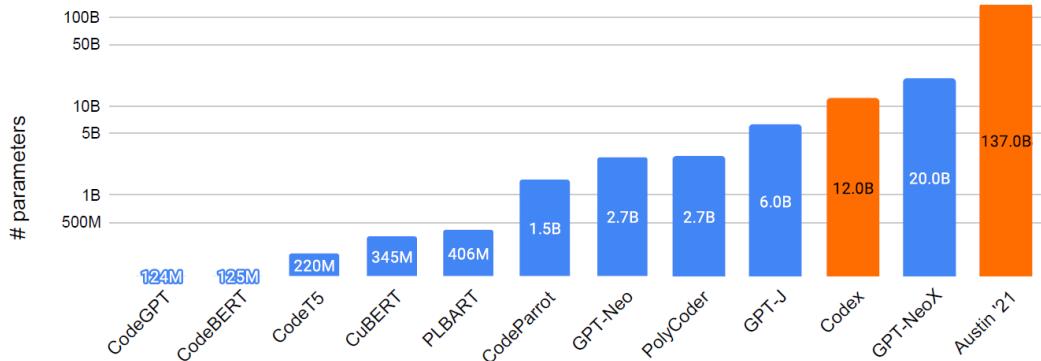


Figure 1: Existing language models of code, their sizes and availability (open source vs. not open-source).

```
# recursive binary search          # recursive binary search          # recursive MASK0
def binarySearch(arr, left, right, x): def binarySearch(arr, left, right, x): def binarySearch(arr, left, right, x):
    mid = (left + ???)           mid = (left + MASK) // 2           mid = (left + MASK1)
                                    if arr[mid] == x:             if arr[MASK2] == x:
                                        return mid           return mid
                                            ↓                               ↓
                                            right                         right
                                            ↓                               ↓
Left-to-Right Language Models      Masked Language Models       Encoder-Decoder Models
                                            ↓
                                            MASK0 binary search MASK1 right ) // 2
                                            MASK2 [ mid ]
```

Figure 2: Three types of pretrained language models.

Source: <https://arxiv.org/pdf/2202.13169.pdf>

## ■ Training corpus statistics

Language	Repositories	Files	Size Before Filtering	Size After Filtering
C	10,749	3,037,112	221G	55G
C#	9,511	2,514,494	30G	21G
C++	13,726	4,289,506	115G	52G
Go	12,371	1,416,789	70G	15G
Java	15,044	5,120,129	60G	41G
JavaScript	25,144	1,774,174	66G	22G
PHP	9,960	1,714,058	21G	13G
Python	25,446	1,550,208	24G	16G
Ruby	5,826	674,343	5.0G	4.1G
Rust	4,991	304,842	5.2G	3.5G
Scala	1,497	245,100	2.2G	1.8G
TypeScript	12,830	1,441,926	12G	9.2G
Total	147,095	24,082,681	631.4G	253.6G

Source: <https://arxiv.org/pdf/2202.13169.pdf>

■ <https://venturebeat.com/2022/03/04/researchers-open-source-code-generating-ai-they-claim-can-beat-openais-codex/>

■ ...



## Comparison

### ■ Data preprocessing strategies of different models

	PolyCoder	CodeParrot	Codex
Dedup	Exact	Exact	Unclear, mentions “unique”
Filtering	Files > 1 MB, < 100 tokens	Files > 1MB, max line length > 1000, mean line length > 100, fraction of alphanumeric characters < 0.25, containing the word “auto-generated” or similar in the first 5 lines	Files > 1MB, max line length > 1000, mean line length > 100, auto-generated (details unclear), contained small percentage of alphanumeric characters (details unclear)
Tokenization	Trained GPT-2 tokenizer on a random 5% subset (all languages)	Trained GPT-2 tokenizer on train split	GPT-3 tokenizer, add multi-whitespace tokens to reduce redundant whitespace tokens

Source: <https://arxiv.org/pdf/2202.13169.pdf>

### ■ Design decisions and hyper-parameters in training different models of code

	PolyCoder (2.7B)	CodeParrot (1.5B)	Codex (12B)
Model Initialization	From scratch	From scratch	Initialized from GPT-3
NL Knowledge	Learned from comments in the code	Learned from comments in the code	Natural language knowledge from GPT-3
Learning Rate	1.6e-4	2.0e-4	1e-4
Optimizer	AdamW	AdamW	AdamW
Adam betas	0.9, 0.999	0.9, 0.999	0.9, 0.95
Adam eps	1e-8	1e-8	1e-8
Weight Decay	-	0.1	0.1
Warmup Steps	1600	750	175
Learning Rate Decay	Cosine	Cosine	Cosine
Batch Size (#tokens)	262K	524K	2M
Training Steps	150K steps, 39B tokens	50K steps, 26B tokens	100B tokens
Context Window	2048	1024	4096

Source: <https://arxiv.org/pdf/2202.13169.pdf>



### 3.2.2.2 EleutherAI

- <https://www.eleuther.ai/>

**When OpenAI Isn't Open Enough. This group's free and open-source AI language model aims for GPT-3 power with Linux-scale collaboration and distribution.**

---

-



## GPT-NeoX

### ■ <https://github.com/EleutherAI/gpt-neox/>

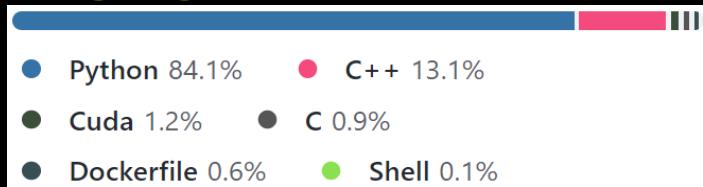
This repository records EleutherAI's work-in-progress for training large-scale language models on GPUs. Our current framework is based on NVIDIA's [Megatron Language Model](#) and has been augmented with techniques from [DeepSpeed](#) as well as some novel optimizations.

We aim to make this repo a centralized and accessible place to gather techniques for training large-scale autoregressive language models, and accelerate research into large-scale training. Additionally, we hope to train and open source a 175B parameter GPT-3 replication along the way. Please note, however, that this is a research codebase that is primarily designed for performance over ease of use. We endeavour to make it as easy to use as is feasible, but if there's anything in the readme that is unclear or you think you've found a bug, please open an issue.

If you are interested in contributing, please [join our Discord](#) and head to the `#gpt-neox` channel. We're working with cloud compute provider [CoreWeave](#) for training, and hope to release the weights of smaller models as we progress up to 175B parameters.

For those looking for a TPU-centric codebase, we recommend [Mesh Transformer JAX](#).

### ■ Languages



## ■ Pretrained Models

### GPT-NeoX-20B

GPT-NeoX-20B is a 20 billion parameter autoregressive language model trained on [the Pile](#). Technical details about GPT-NeoX-20B can be found in our [whitepaper](#). The configuration file for this model is both available at [./configs/20B.yml](#) and included in the download links below.

---

#### Download Links

[Slim weights](#) - (No optimizer states, for inference or finetuning, 39GB)

To download from the command line to a folder named `20B_checkpoints`, use the following command:

```
wget --cut-dirs=5 -nH -r --no-parent --reject "index.html*" https://mystic.the-eye.eu/public/AI/models/GPT-Ne
```

[Full weights](#) - (Including optimizer states, 268GB)

To download from the command line to a folder named `20B_checkpoints`, use the following command:

```
wget --cut-dirs=5 -nH -r --no-parent --reject "index.html*" https://mystic.the-eye.eu/public/AI/models/GPT-Ne
```

**[http://eaidata.bmk.sh/data/GPT\\_NeoX\\_20B.pdf](http://eaidata.bmk.sh/data/GPT_NeoX_20B.pdf)**

■ ...



## CARP

- <https://github.com/EleutherAI/magiCARP>

### magiCARP: Contrastive Authoring+Reviewing Pretraining

Welcome to the magiCARP API, the test bed used by EleutherAI for performing text/text bi-encoder experiments.

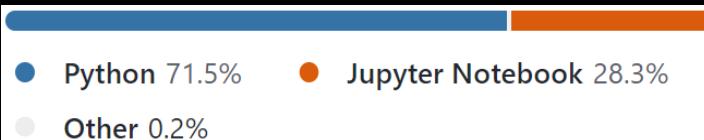
CARP, or contrastive authorship+reviewing pairings, was first outlined in [Cut the CARP: Fishing for zero-shot story evaluation](#).

CARP presents a scalable method for performing zero-shot evaluation of stories and other mediums. Current CARP efforts at EleutherAI are primarily focused around controllable code generation. This repository will be updated with more experiments over the coming months as we try varying CARP architectures.

To train a model, run `python -m carp.pytorch.training.train --data_path="carp/dataset" --config_path ./configs/base_config.yml`

Finetuning via CoOp now available. Preference learning coming soon!

## Languages



- <https://arxiv.org/pdf/2110.03111.pdf>

### Cut the CARP: Fishing for zero-shot story evaluation



## Good Resources

- <https://spectrum.ieee.org/eleutherai-openai-not-open-enough>
  - <https://www.infoq.com/news/2021/07/eleutherai-gpt-j/>
  - ...
-



### 3.2.2.3 Digma

- <https://www.digma.ai/>
- <https://github.com/digma-ai/digma>

**Digma makes observability data relevant when writing code. Digma continuously analyzes logs and traces from OpenTelemetry and other sources to automatically glean insights on how your code runs and provide them as feedback in your IDE and dev tools. Join our beta waitlist!**



#### Digma is about *Developer Observability*

We believe that understanding code real-world requirements and behavior is critical to making better software. Something that can be done only if we connect the dots between design time and runtime.

There are many observability tools out there, they all take a very monitoring centric - 'live events capture' approach to observability. We feel they have managed to miss what developers care about when writing their code.

Our goal is to create an **open platform** for interpreting and analyzing the information collected via observability. Traces, logs, metrics are great! But additional effort is needed in order to take them that last mile into the development process.



#### The Digma way:

So how do we do that? How do we make observability relevant? Digma has three main design principles that we think are key in unlocking the potential of observability data.

|| *Code Insights and analytics over Raw data*

Because if it takes time and manual work to check aggregate correlate and understand the significance of raw logs and traces you'll probably not do it a lot 🚫

|| *Proactive over Reactive*

Otherwise you won't know what to look for unless its already on 🔥

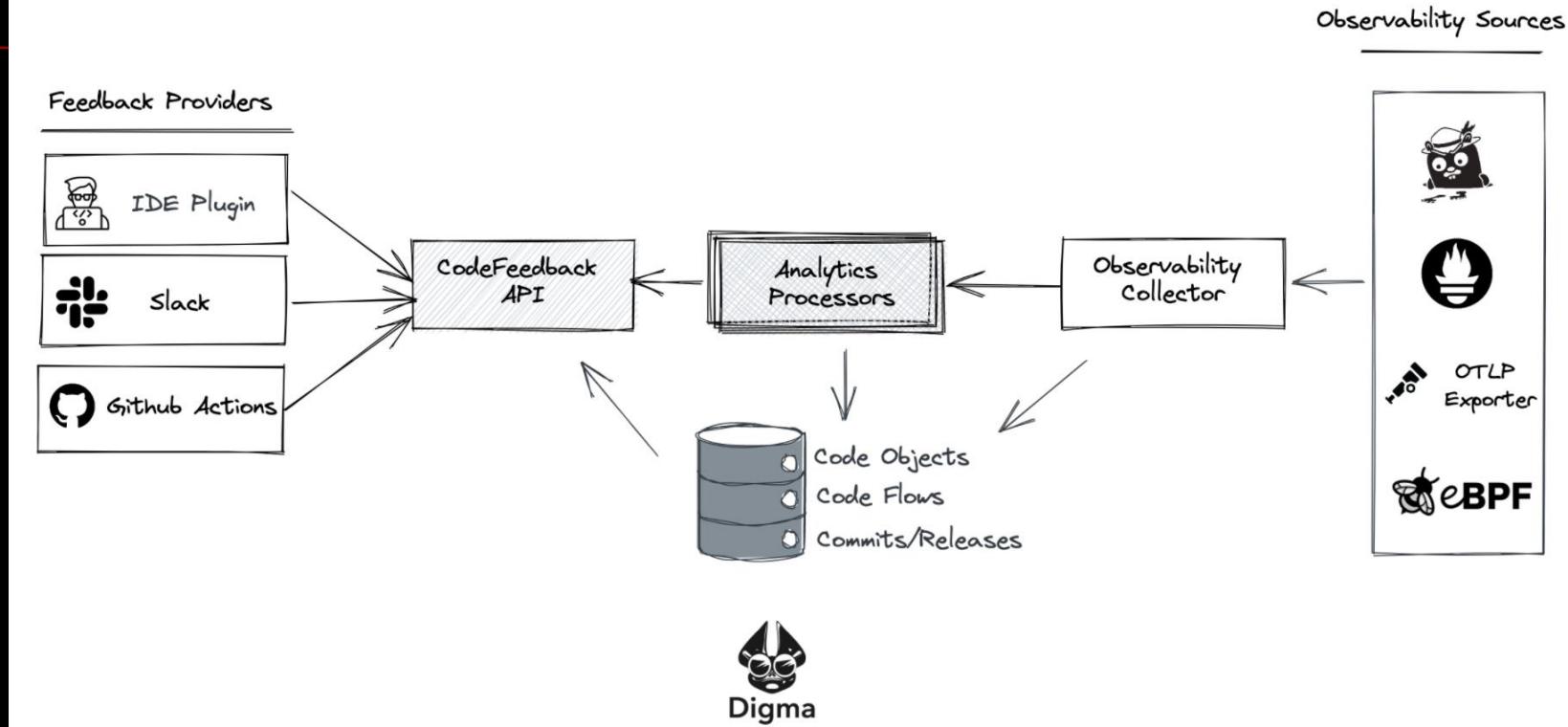
|| *Integrated over external*

Because context switching is already an issue. Going back and forth between dashboards and charts is guaranteed to slow you down.



## How it works

- It is a pipeline. A continuous feedback pipeline that injects data from your observability sources and generates feedback.





## 3.3 Implementation

### 3.3.1 Current consideration

#### 3.3.1.1 GraalVM + TornadoVM + AI

- For details, please look forward to our new talk "The first exploration of Smart Runtime"...

# IV. Networking



## 1) Emerging Networking Technologies

### 1.1 SRv6

#### 1.1.1 HIKe

- <https://netgroup.github.io/hike/>  
**Hybrid Kernel eBPF forwarding.**
- Part of the **ROSE**(<https://netgroup.github.io/rose/>) project

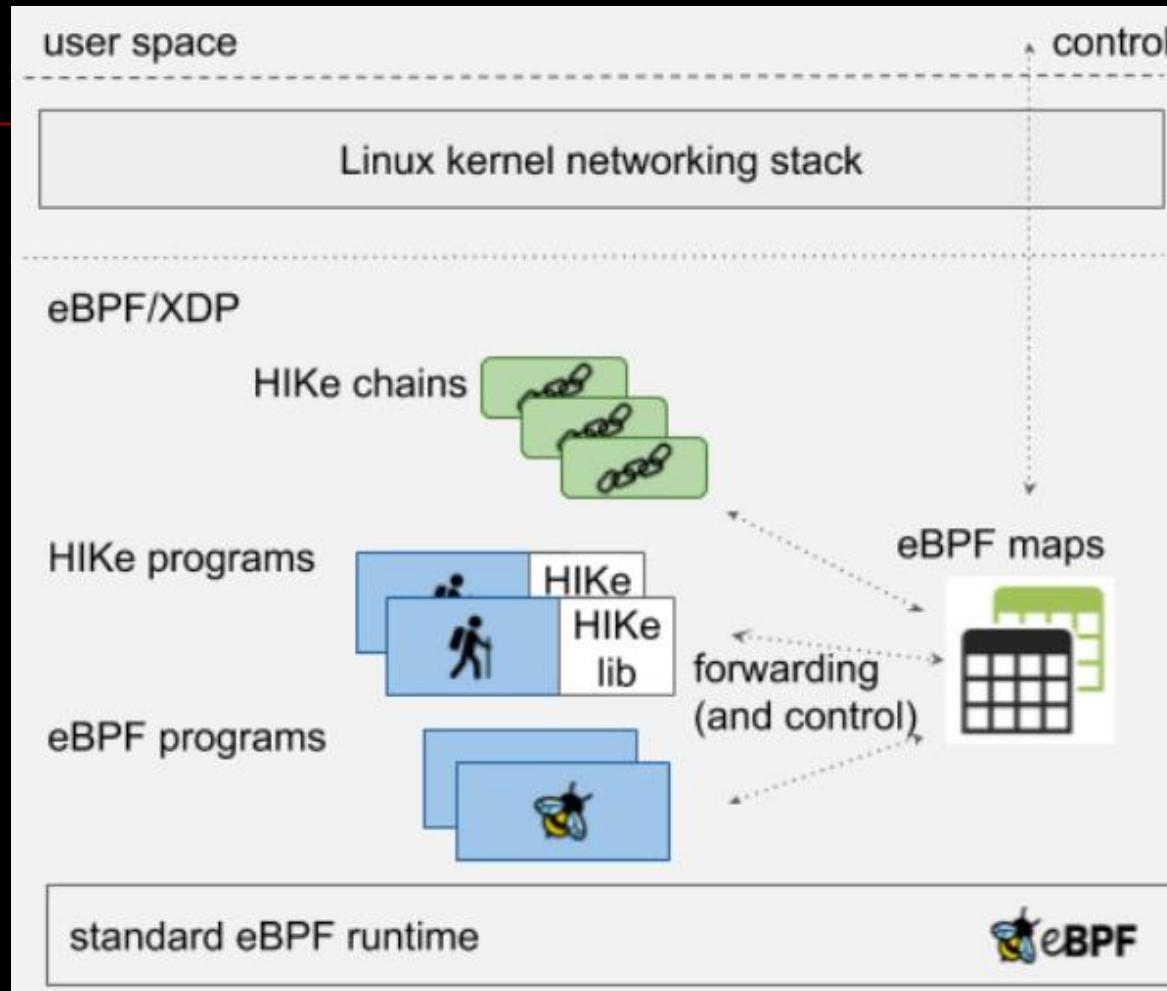
HIKe is a programmable data plane architecture that integrates the packet forwarding and processing based on the standard Linux kernel networking with the ones based on custom designed eBPF programs. HIKe does not want to trade the flexibility and generality of Linux kernel networking with eBPF performances, but wants to take the best of both worlds. In the context of the ROSE project, we use HIKe to speed up the performance of SRv6 software routers (although HIKe is not limited to SRv6!).

- <https://github.com/netgroup/hike>



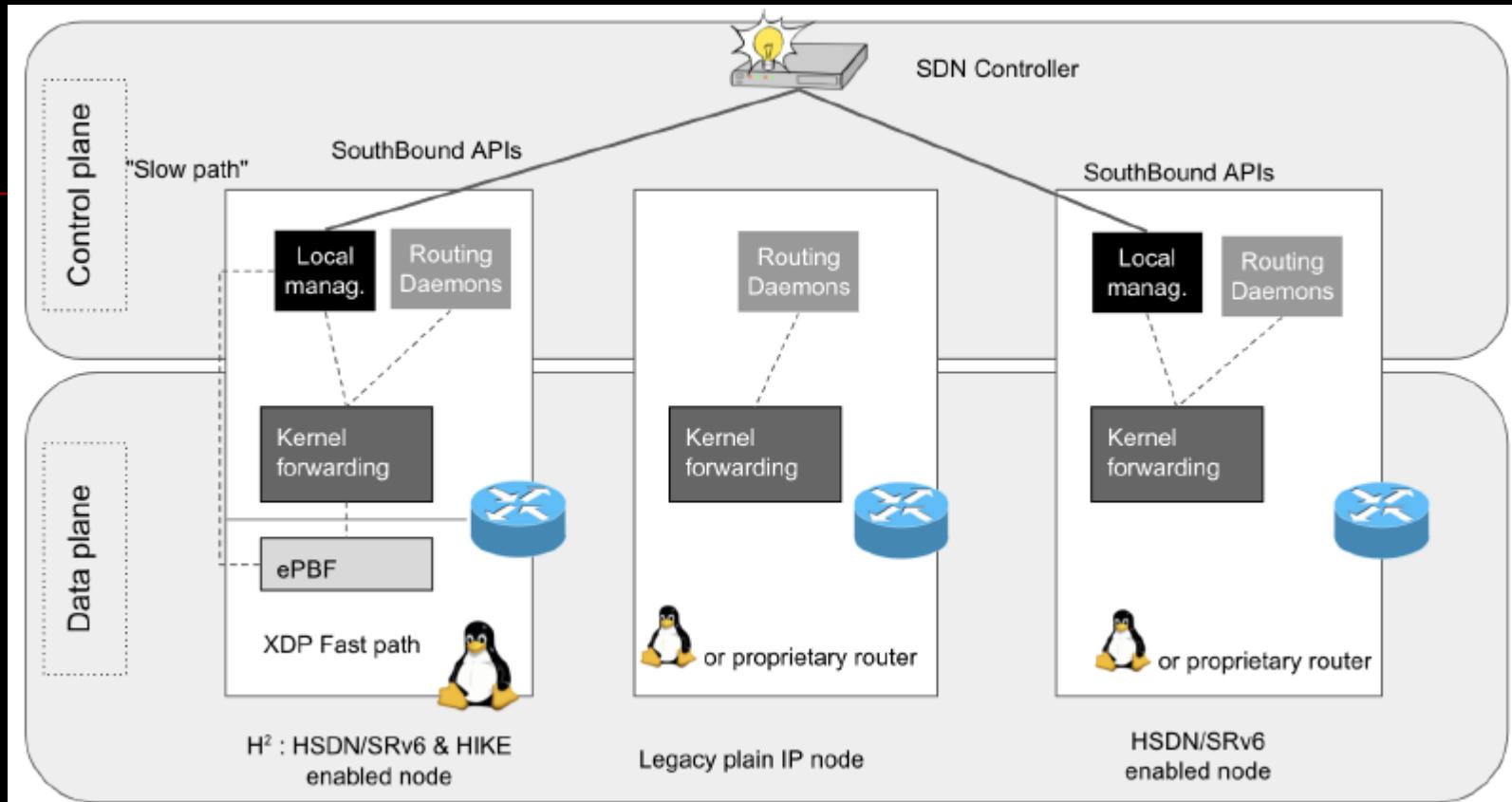
## Architecture and Design

### ■ Overview



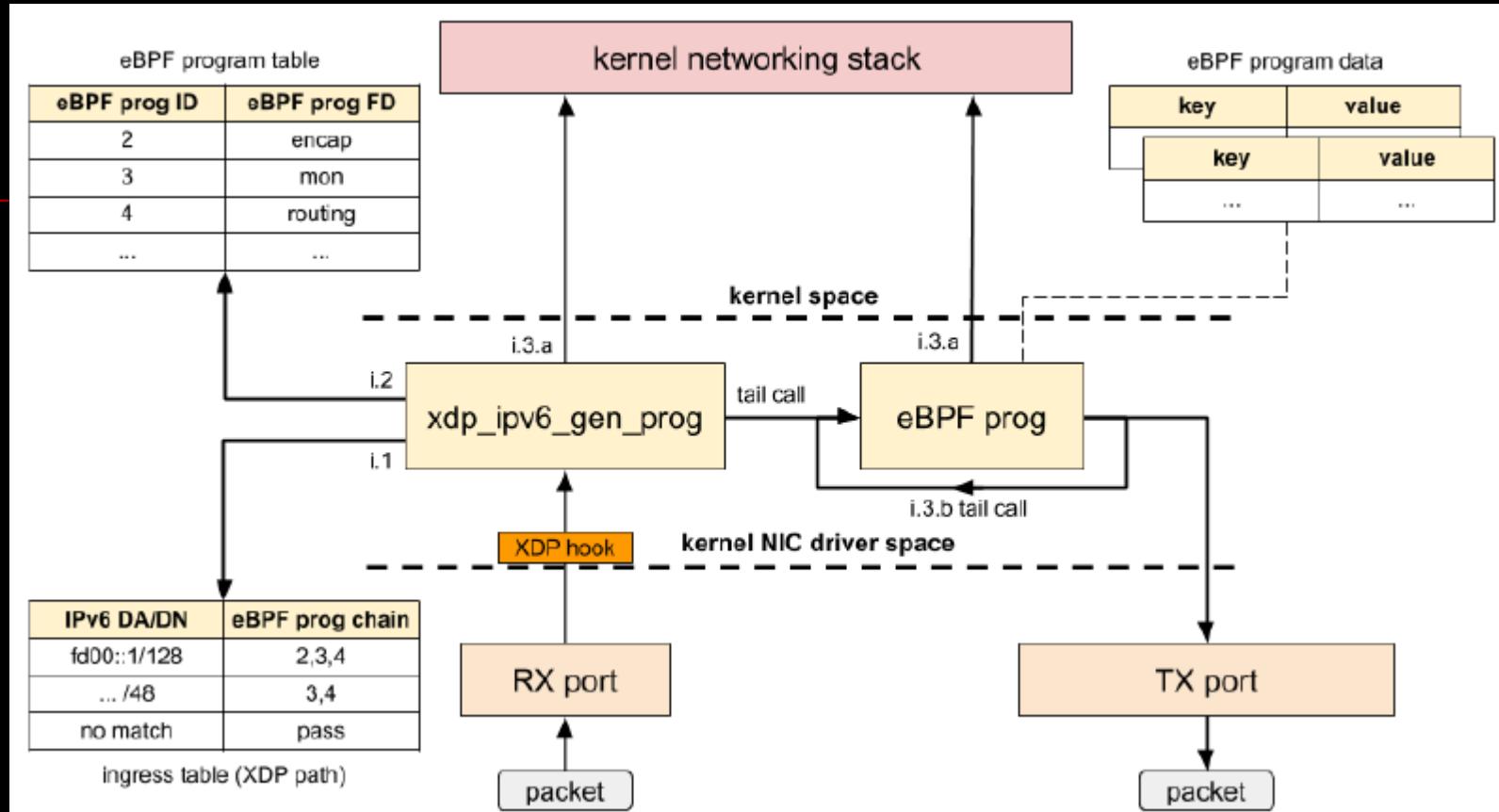


## ■ High level overview of the data plane/control plane architecture



Source: [http://netgroup.uniroma2.it/Stefano\\_Salsano/papers/20-srv6-hybrid-sdn-hike.pdf](http://netgroup.uniroma2.it/Stefano_Salsano/papers/20-srv6-hybrid-sdn-hike.pdf)

## HIKe eBPF/XDP architecture



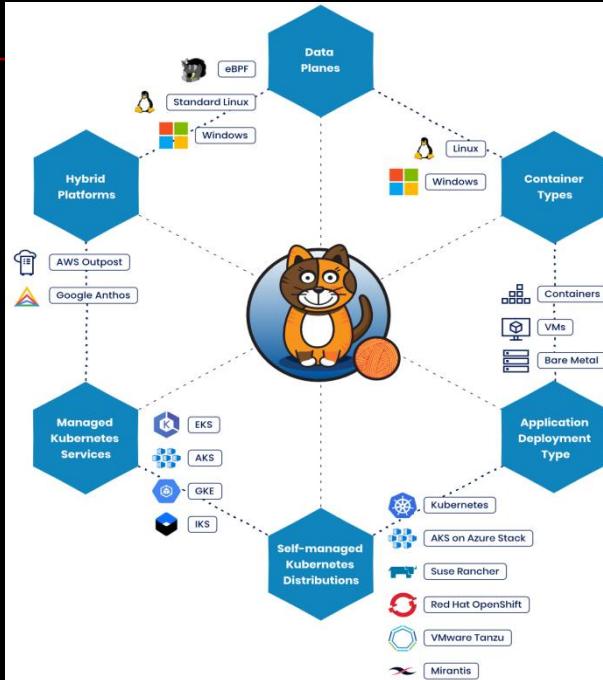
Source: [http://netgroup.uniroma2.it/Stefano\\_Salsano/papers/20-srv6-hybrid-sdn-hike.pdf](http://netgroup.uniroma2.it/Stefano_Salsano/papers/20-srv6-hybrid-sdn-hike.pdf)



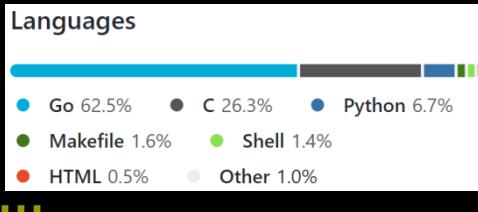
## 2) eBPF for Cloud-native Networking

### 2.1 Calico eBPF Data Plane

- <https://www.tigera.io/project-calico/>



- <https://github.com/projectcalico/calico>  
**Cloud native networking and network security.**





## Feature comparison

### ■ <https://projectcalico.docs.tigera.io/about/about-ebpf>

While the eBPF dataplane has some features that the standard Linux dataplane lacks, the reverse is also true:

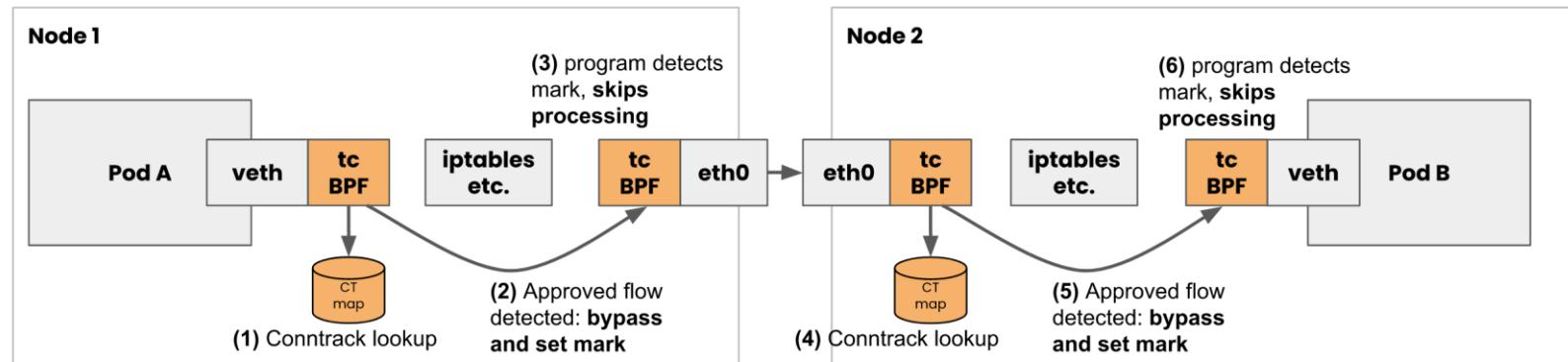
Factor	Standard Linux Dataplane	eBPF dataplane
Throughput	Designed for 10Gbit+	Designed for 40Gbit+
First packet latency	Low (kube-proxy service latency is bigger factor)	Lower
Subsequent packet latency	Low	Lower
Preserves source IP within cluster	Yes	Yes
Preserves external source IP	Only with <code>externalTrafficPolicy: Local</code>	Yes
Direct Server Return	Not supported	Supported (requires compatible underlying network)
Connection tracking	Linux kernel's conntrack table (size can be adjusted)	BPF map (fixed size)
Policy rules	Mapped to iptables rules	Mapped to BPF instructions
Policy selectors	Mapped to IP sets	Mapped to BPF maps
Kubernetes services	kube-proxy iptables or IPVS mode	BPF program and maps
IPIP	Supported	Supported (no performance advantage due to kernel limitations)
VXLAN	Supported	Supported
Wireguard	Supported	Supported
Other routing	Supported	Supported
Supports third party CNI plugins	Yes (compatible plugins only)	Yes (compatible plugins only)
Compatible with other iptables rules	Yes (can write rules above or below other rules)	Partial; iptables bypassed for workload traffic
Host endpoint policy	Supported	Supported
Enterprise version	Available	Available
XDP DoS Protection	Supported	Supported
IPv6	Supported	Not supported (yet)



## Architecture overview

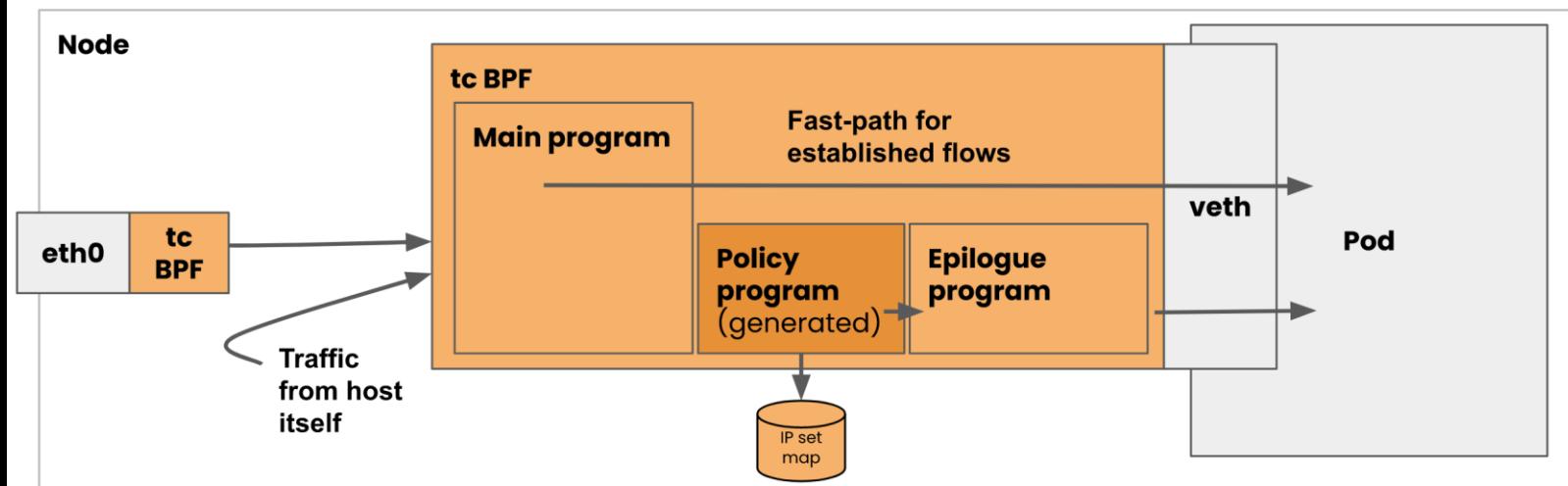
- <https://projectcalico.docs.tigera.io/about/about-ebpf>

Calico's eBPF dataplane attaches eBPF programs to the `tc` hooks on each Calico interface as well as your data and tunnel interfaces. This allows Calico to spot workload packets early and handle them through a fast-path that bypasses iptables and other packet processing that the kernel would normally do.



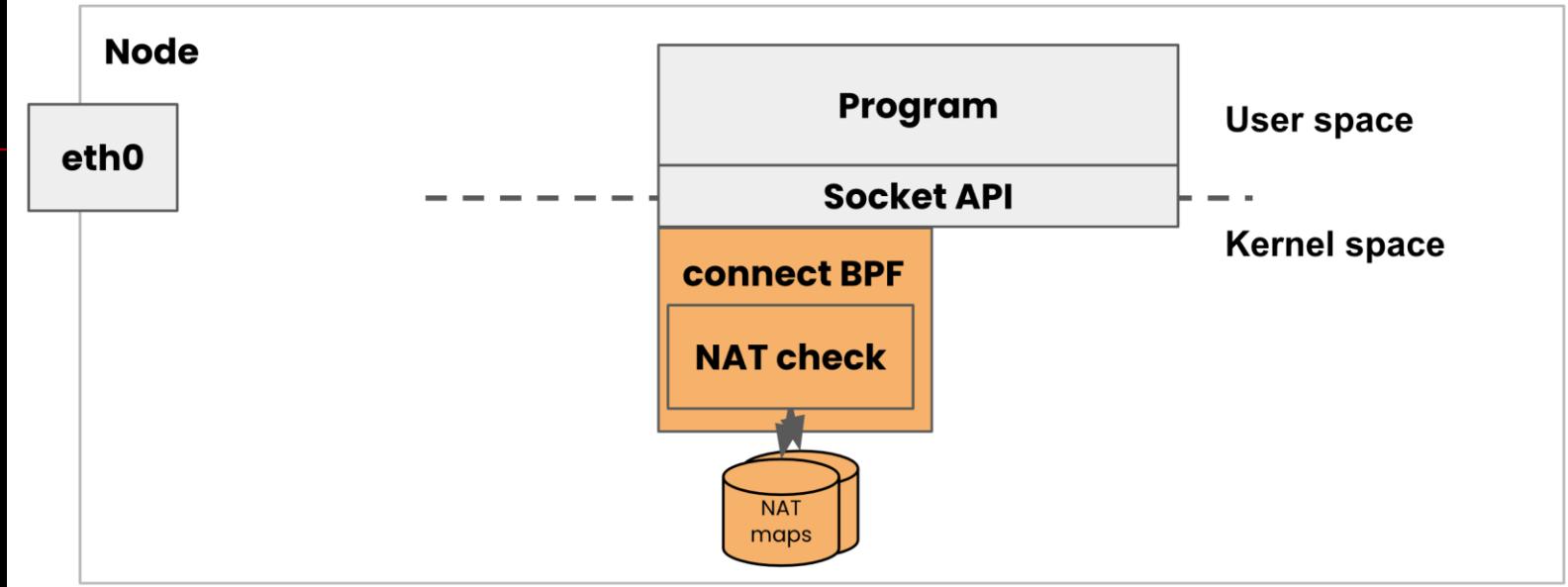
The logic to implement load balancing and packet parsing is pre-compiled ahead of time and relies on a set of BPF maps to store the NAT frontend and backend information. One map stores the metadata of the service, allowing for `externalTrafficPolicy` and "sticky" services to be honoured. A second map stores the IPs of the backing pods.

In eBPF mode, Calico converts your policy into optimised eBPF bytecode, using BPF maps to store the IP sets matched by policy selectors.





To improve performance for services, Calico also does connect-time load balancing by hooking into the socket BPF hooks. When a program tries to connect to a Kubernetes service, Calico intercepts the connection attempt and configures the socket to connect directly to the backend pod's IP instead. This removes *all* NAT overhead from service connections.





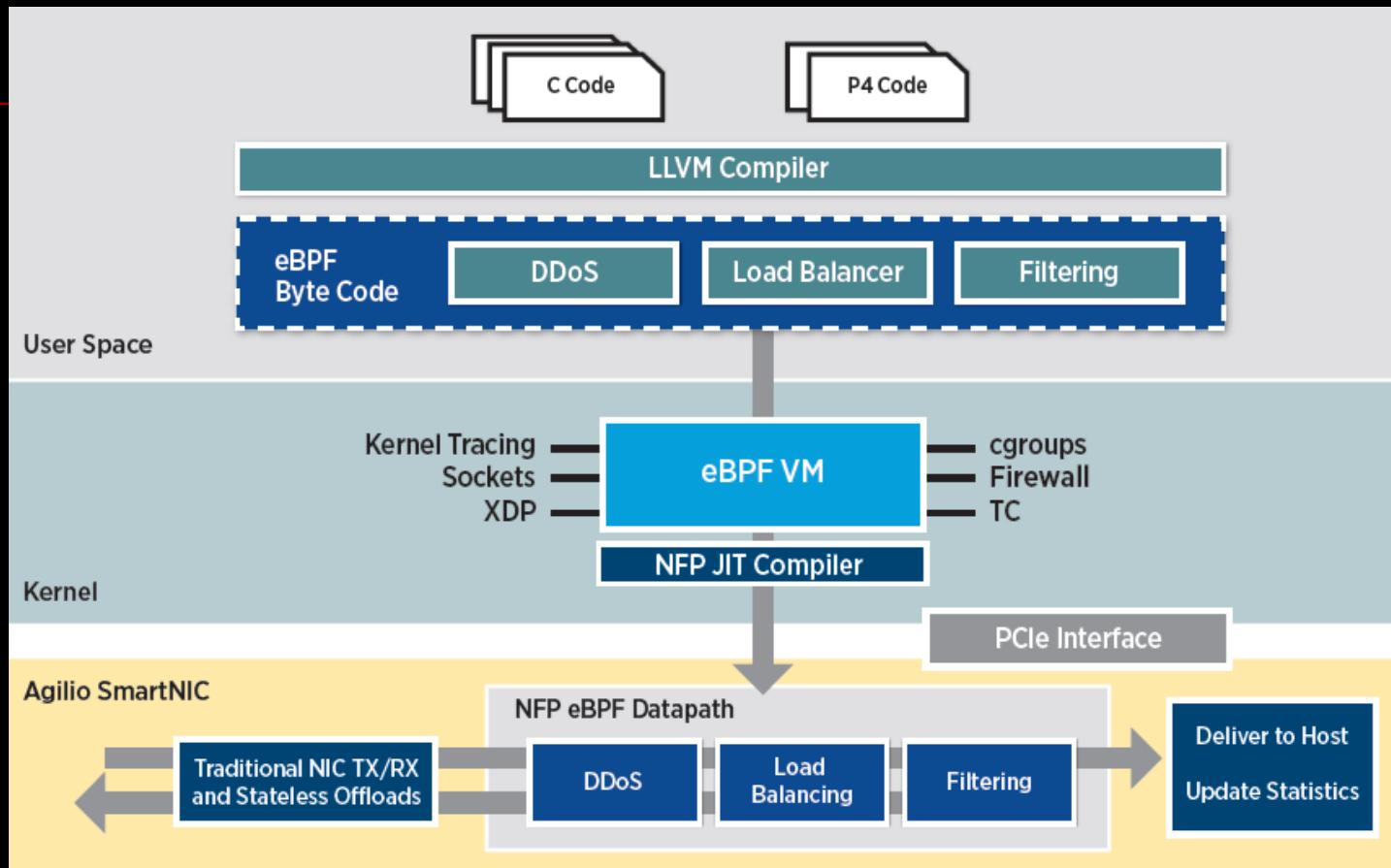
## Good Resources

- <https://projectcalico.docs.tigera.io/about/about-calico>
- <https://www.tigera.io/blog/introducing-the-calico-ebpf-dataplane/>
- <https://projectcalico.docs.tigera.io/maintenance/ebpf/enabling-bpf>
- ~~<https://www.tigera.io/blog/calico-ebpf-data-plane-deep-dive/>~~
- <https://projectcalico.docs.tigera.io/maintenance/troubleshoot/troubleshoot-ebpf>
- <https://thenewstack.io/beyond-kube-proxy-tigera-calico-harnesses-ebpf-for-a-faster-data-plane/>
- <https://www.tigera.io/learn/guides/ebpf/>
- ...



### 3) eBPF/XDP Hardware Offload to SmartNIC/DPU

#### 3.1 Netronome



Source: [https://www.netronome.com/media/documents/PB\\_Agilio-eBPF-7-20.pdf](https://www.netronome.com/media/documents/PB_Agilio-eBPF-7-20.pdf)

<https://www.netronome.com/products/agilio-software/agilio-ebpf-software/>

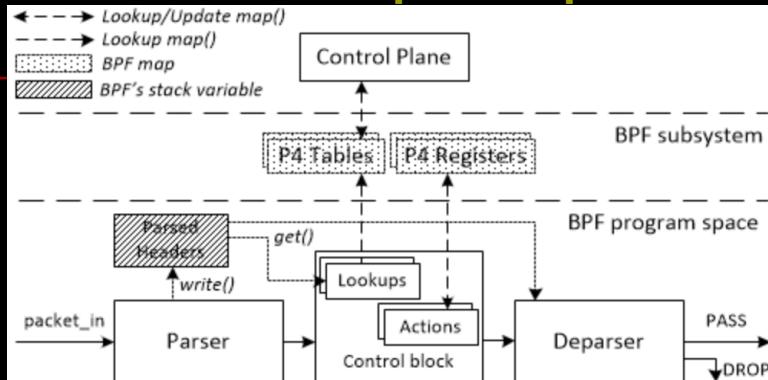
<https://qmonnet.github.io/whirl-offload/2021/09/23/bptool-features-thread/>

...



## 4) P4 and eBPF

- <https://opennetworking.org/news-and-events/blog/p4c-ubpf-a-new-back-end-for-the-p4-compiler/>



- <https://github.com/p4lang/p4c/tree/main/backends/ubpf>

The p4c-ubpf compiler allows to translate P4 programs into the uBPF programs. We use the uBPF implementation provided by [the P4rt-OVS switch](#). The uBPF VM is based on the open-source implementation provided by [IOVisor](#).

The P4-to-uBPF compiler accepts only the P4\_16 programs written for the [ubpf\\_model.p4](#) architecture model.

The backend for uBPF is mostly based on [P4-to-eBPF compiler](#). In fact, it implements the same concepts, but generates C code, which is compatible with the user space BPF implementation.

P4\_16 ---> | p4c-ubpf | ---> C -----> | clang | --> uBPF

- <https://github.com/p4lang/p4c/blob/main/backends/ebpf/README.md>

P4 ---> | P4-to-eBPF | ---> C -----> | clang/BCC | --> eBPF

...



# V. Messaging & RPC

## 1) Lightweight RPC

### 1.1 Low overhead

#### 1.1.1 ttrpc

##### Kata Container 2.x

- <https://medium.com/kata-containers/kata-containers-version-2-0-e45df4dd328> 11MB to 300K
- What's New

- Kata-agent has been rewritten in **rust** to significantly reduce memory overhead overall attack surface.
- Agent protocol has been simplified to use **ttRPC** from **grpc**.
- Component called Kata-monitor has been introduced to improve observability and manageability
- New component called agent-ctl has been introduced to help validate agent API
- We have moved to a mono-repo model, all code and document repositories consolidated into one single repo.
- Virtio-fs is now the default shared file system type which mean better POSIX compliance compared to virtio-9p
- Latest Cloud Hypervisor support on par with QEMU
- Move to support solely shimv2 api to simplify code and reduce attack surface further. This also means kata-shim and kata-proxy used in 1.x are no longer required.
- Guest kernel updated to v5.4.71
- QEMU updated to v5.0.0

- <https://github.com/containerd/ttrpc>  
**GRPC for low-memory environments**



## 2) Hardware-accelerated RPC

### 2.1 Programmable NIC oriented System Design

#### 2.1.1 nanoPU

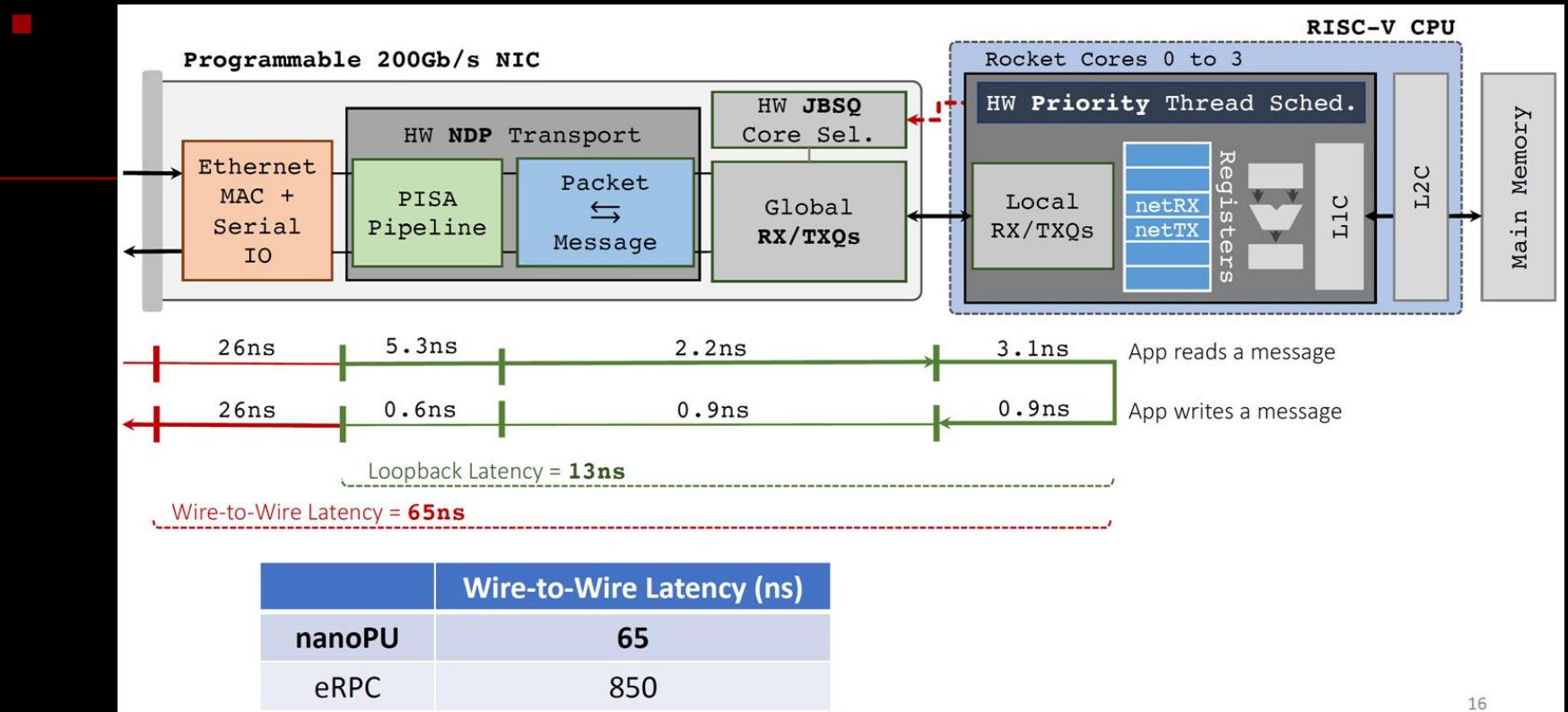
- A Nanosecond Network Stack for Datacenters
- Previous approaches to minimize RPC latency and software

Approach	Limitation	Wire-to-Wire Latency	RPC Throughput
Dataplane operating systems (e.g. Shinjuku, Shenango)	Too coarse grained	Median: ~2-5µs Tail: 10-100µs	<10Mrps
Efficient RPC software libraries (e.g. eRPC)	Neglect tail latency optimizations	Median: 850ns Tail: 10-100µs	~10Mrps / core
Transport protocol offload (e.g. Tonic)	Only part of the solution	N/A	>100Mrps
RDMA NICs	Need low latency to remote <i>compute</i> , not <i>memory</i>	Median: ~700ns	N/A
Integrated NICs (e.g. NeBuLa)	Still room for improvement of latency and throughput	Median: ~100ns Tail: ~2-5µs	~20Mrps / core

Source: “The nanoPU: A Nanosecond Network Stack for Datacenters”, Stephen Ibanez etc, OSDI 2021



## Microbenchmarks



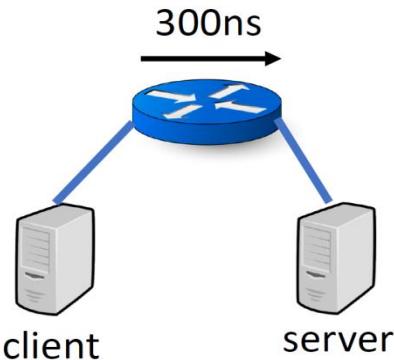
Source: "The nanoPU: A Nanosecond Network Stack for Datacenters", Stephen Ibanez etc, OSDI 2021



## Comparison

### ■ State-of-the-art RDMA NIC

- Implemented in NIC HW
- End-to-end latency:  $\sim 2\mu\text{s}$



Source: “The nanoPU: A Nanosecond Network Stack for Datacenters”, Stephen Ibanez etc, OSDI 2021

### The nanoPU

- Implemented in SW
- Can support arbitrary one-sided operations

One-sided RDMA	Latency (ns)	
	Median	90th %ile
Read	678	680
Write	679	686
Compare-and-Swap	687	690
Fetch-and-Add	688	692
Indirect Read	691	715



## 3) eBPF-inspired Messaging and RPC

### 3.1 ...

#### 3.1.1 ...

- ...
  -
-



# VI. HPC

## ■ **Heterogeneous Parallel Computing**

### ~~1) Standards and Frameworks~~

#### 1.1 OpenCL

##### 1.1.1 OpenCL 3.0

- <https://www.khronos.org/opencl/>

### OpenCL 3.0 Final is Here!

The OpenCL 3.0 Finalized Specification was released on September 30th 2020

[Read the Blog about the final release of OpenCL 3.0](#)

[Provisional Press Release](#)

[Provisional Launch Presentation](#)

OpenCL 3.0 realigns the OpenCL roadmap to enable developer-requested functionality to be broadly deployed by hardware vendors, and it significantly increases deployment flexibility by empowering conformant OpenCL implementations to focus on functionality relevant to their target markets. OpenCL 3.0 also integrates subgroup functionality into the core specification, ships with a new unified API and OpenCL C 3.0 language specifications and introduces extensions for asynchronous data copies to enable a new class of embedded processors.

### OpenCL 3.0 Materials

[Specification](#)

[SDK](#)

[OpenCL Guide](#)

[OpenCL Blogs](#)

[Issues](#)

[Discussions](#)

[Resources](#)

[Reference Guide](#)



# OpenCL 3.0

## Increased Ecosystem Flexibility

All functionality beyond OpenCL 1.2 queryable  
Macros for optional OpenCL C language features  
Widely adopted extensions to be integrated into core

## OpenCL C++ for OpenCL

Open-source [C++ for OpenCL](#) front end compiler combines  
OpenCL C and C++17 replacing OpenCL C++ language spec

## Unified Specification

All versions of OpenCL in one specification for easier  
maintenance, evolution and accessibility

[Source](#) on Khronos GitHub for community feedback,  
functionality requests and bug fixes

## New Functionality

Subgroups with SPIR-V 1.3 in core (optional)  
Asynchronous DMA extension for embedded processors

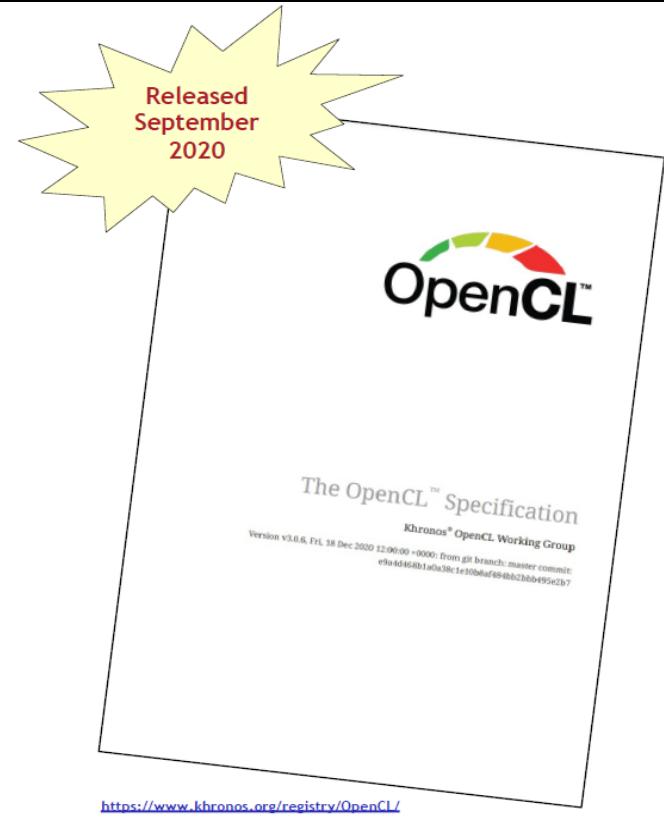
## Easy OpenCL 3.0 migration for applications

OpenCL 1.2 applications - no change  
OpenCL 2.X applications - no code changes if all used  
functionality present  
Queries recommended for future portability

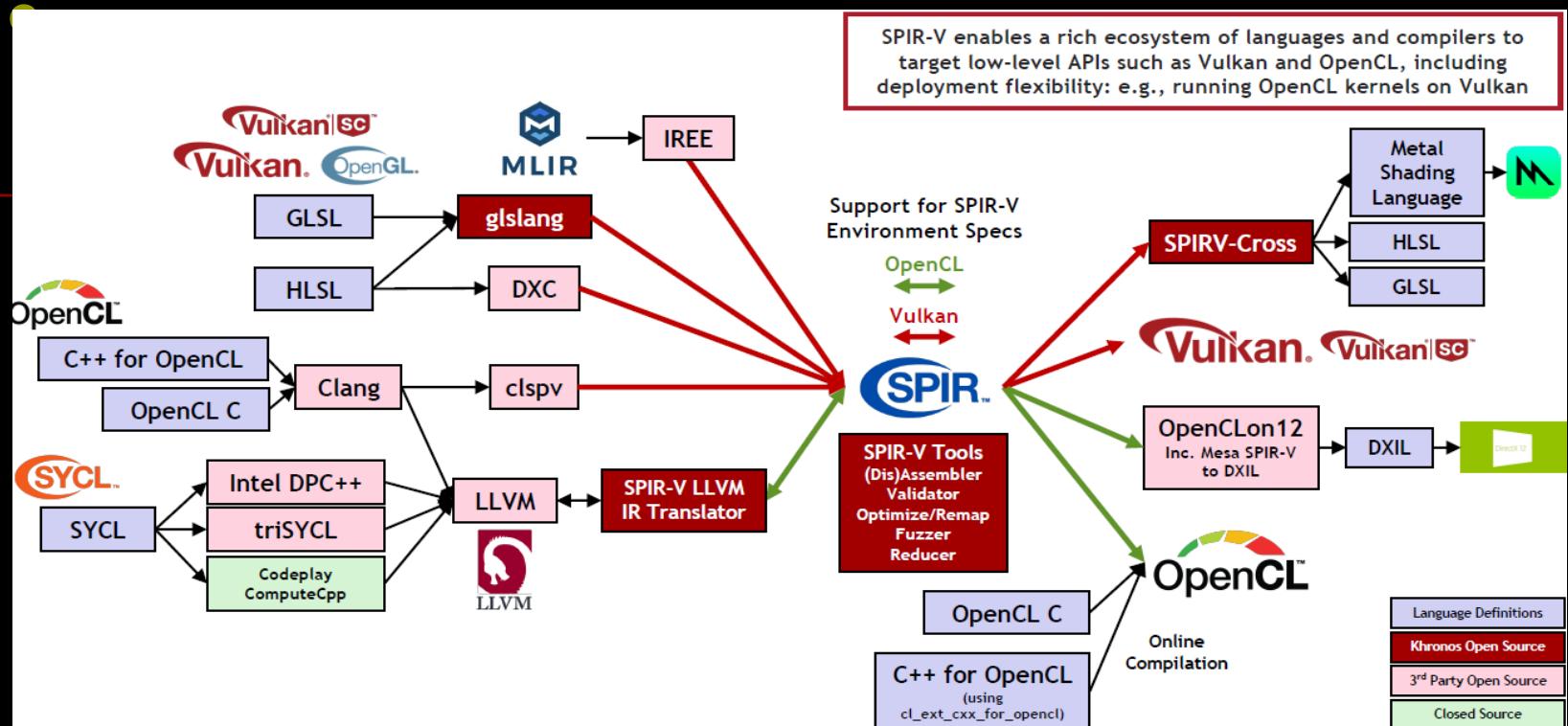
Source: "State of the Union OpenCL Working Group", Neil Trevett, IWOCL 2021.

<https://github.com/KhronosGroup/OpenCL-SDK>

...



# SPIR-V Language Ecosystem



Source: "State of the Union OpenCL Working Group", Neil Trevett, IWOCL 2021.

# 1.2 CXL

## ■ [https://en.wikipedia.org/wiki/Compute\\_Express\\_Link](https://en.wikipedia.org/wiki/Compute_Express_Link)



**Compute Express Link (CXL)** is an open standard for high-speed central processing unit (CPU)-to-device and CPU-to-memory connections, designed for high performance data center computers.<sup>[5][6]</sup> CXL is built on the PCI Express (PCIe) physical and electrical interface and includes PCIe-based block input/output protocol (CXL.io) and new cache-coherent protocols for accessing system memory (CXL.cache) and device memory (CXL.mem).

### History [edit]

The standard was primarily developed by Intel. The CXL Consortium was formed in March 2019 by founding members Alibaba Group, Cisco, Dell EMC, Facebook, Google, Hewlett Packard Enterprise (HPE), Huawei, Intel and Microsoft,<sup>[5][6]</sup> and officially incorporated in September 2019.<sup>[7]</sup> As of January 2022, AMD, NVidia, Samsung and Xilinx joined the founders on the board of directors, while ARM, Broadcom, Ericsson, IBM, Keysight, Kioxia, Marvell, Mellanox, Microchip, Micron, Oracle, Qualcomm, Rambus, Renesas, Seagate, SK Hynix, Synopsys, and Western Digital, among others, joined as contributing members.<sup>[8][9]</sup> Industry partners include the PCI-SIG,<sup>[10]</sup> Gen-Z,<sup>[11]</sup> SNIA,<sup>[12]</sup> and DMTF.<sup>[13]</sup>

On April 2, 2020, the Compute Express Link and Gen-Z Consortiums announced plans to implement interoperability between the two technologies.<sup>[14][15]</sup> with initial results presented in January 2021.<sup>[16]</sup> On November 10, 2021, Gen-Z specifications and assets were transferred to CXL, to focus on developing a single industry standard moving forward.<sup>[17]</sup> At the time of this announcement, 70% of Gen-Z members already joined the CXL Consortium, which now includes companies behind memory coherent interconnect technologies such as OpenCAPI (IBM), CCIX (Xilinx), and Gen-Z (HPE) open standards, and proprietary InfiniBand / RoCE (Mellanox), Infinity Fabric (AMD), Omni-Path and QuickPath/Ultra Path (Intel), and NVLink/NVSwitch (Nvidia) protocols.<sup>[18][19]</sup>

### Specifications [edit]

On March 11, 2019, the CXL Specification 1.0 based on PCIe 5.0 was released.<sup>[6]</sup> It allows host CPU to access shared memory on accelerator devices with a cache coherent protocol. The CXL Specification 1.1 was released in June, 2019.

On November 10, 2020, the CXL Specification 2.0 was released. The new version adds support for CXL switching, to allow connecting multiple CXL 1.x and 2.0 devices to a CXL 2.0 host processor, and/or pooling each device to multiple host processors, in distributed shared memory and disaggregated storage configurations; it also implements device integrity and data encryption.<sup>[20]</sup> There is no bandwidth increase from CXL 1.x, because CXL 2.0 still utilizes PCIe 5.0 PHY.

Next version of CXL specifications is expected in H1 2022, to be based on PCIe 6.0 PHY.<sup>[19][21]</sup>

### Implementations [edit]

On April 2, 2019, Intel announced their family of Agilex FPGAs featuring CXL.<sup>[22]</sup>

On May 11, 2021, Samsung announced a DDR5 based memory expansion module that allows for terabyte level memory expansion along with high performance for use in data centres and potentially next generation PCs.<sup>[23]</sup>

In 2021, CXL 1.1 support was announced for Intel Sapphire Rapids processors<sup>[24]</sup> and AMD Zen 4 EPYC "Genoa" and "Bergamo" processors.<sup>[25]</sup>

CXL devices were shown at the SC21 conference by Intel,<sup>[26]</sup> Astra, Rambus, Synopsys, Samsung, and Teledyne LeCroy, among others.<sup>[27][28][29]</sup>

### Protocols [edit]

The CXL standard defines three separate protocols:<sup>[30][20]</sup>

- **CXL.io** - based on PCIe 5.0 with a few enhancements, it provides configuration, link initialization and management, device discovery and enumeration, interrupts, DMA, and register I/O access using non-coherent loads/stores.
- **CXL.cache** - allows peripheral devices to coherently access and cache host CPU memory with a low latency request/response interface.
- **CXL.mem** - allows host CPU to coherently access cached device memory with load/store commands for both volatile (RAM) and persistent non-volatile (flash memory) storage.

CXL.cache and CXL.mem protocols operate with a common link/transaction layer, which is separate from the CXLio protocol link and transaction layer. These protocols/layers are multiplexed together by an Arbitration and Multiplexing (ARB/MUX) block before being transported over standard PCIe 5.0 PHY using fixed-width 528 bit (66 byte) Flow Control Unit (FLIT) block consisting of four 16-byte data 'slots' and a two-byte cyclic redundancy check (CRC) value.<sup>[30]</sup> CXL FLITs encapsulate PCIe standard Transaction Layer Packet (TLP) and Data Link Layer Packet (DLLP) data with a variable frame size format.<sup>[31][32]</sup>

### Device types [edit]

CXL is designed to support three primary device types:<sup>[20]</sup>

- Type 1 (CXL.io and CXL.cache) – specialised accelerators (such as smart NIC) with no local memory. Devices rely on coherent access to host CPU memory.
- Type 2 (CXL.io, CXL.cache and CXL.mem) – general-purpose accelerators (GPU, ASIC or FPGA) with high-performance GDDR or HBM local memory. Devices can coherently access host CPU's memory and/or provide coherent or non-coherent access to device local memory from the host CPU.
- Type 3 (CXL.io and CXL.mem) – memory expansion boards and storage-class memory. Devices provide host CPU with low-latency access to local DRAM or non-volatile storage.

Type 2 devices implement two memory coherence modes, managed by device driver. In device bias mode, device directly accesses local memory and no caching is performed by the CPU; in host bias mode, the host CPU's cache controller handles all access to device memory. Coherence mode can be set individually for each 4 KB page, stored in a translation table in local memory of Type 2 devices. Unlike other CPU-to-CPU memory coherency protocols, this arrangement only requires the host CPU memory controller to implement the cache agent; such asymmetric approach reduces implementation complexity and reduces latency.<sup>[30]</sup>

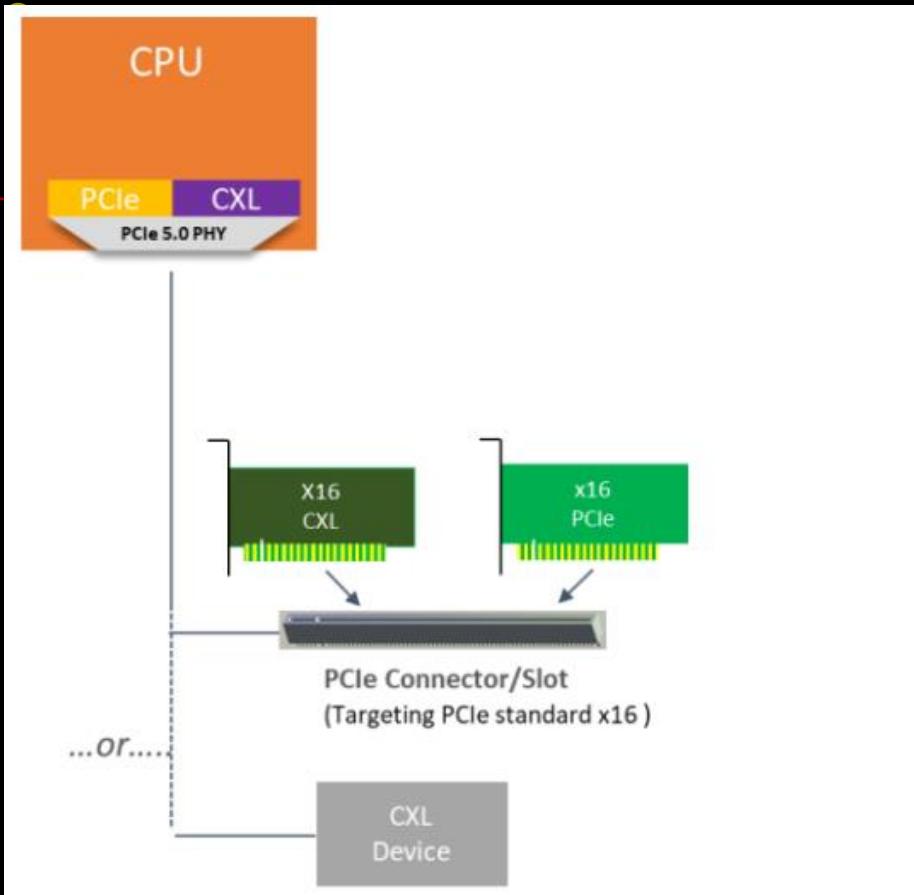
...



Source: <https://www.computeexpresslink.org/post/cxl-consortium-makes-a-splash-at-supercomputing-2021-sc21>

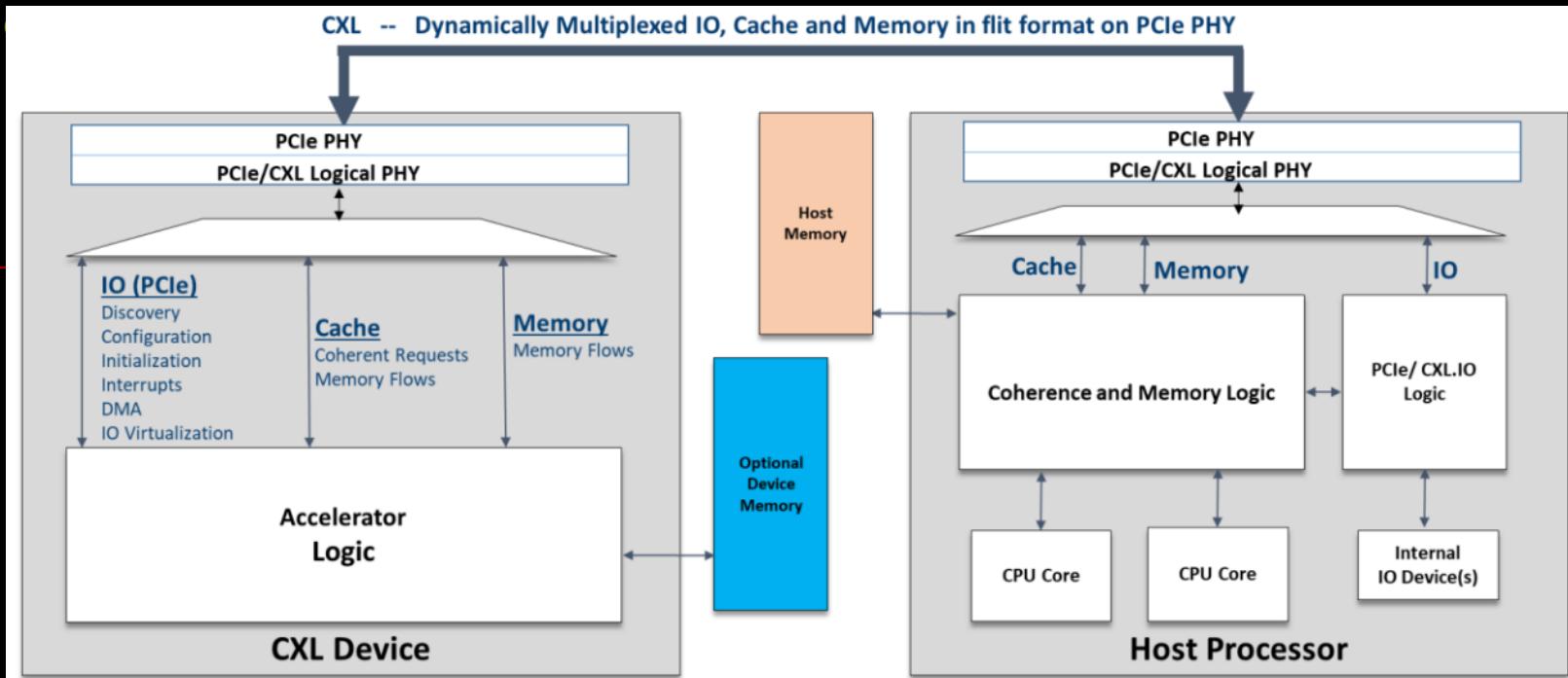


## Design



*Compute Express Link has the benefit of supporting both standard PCIe devices as well as CXL devices – all on the same Link*

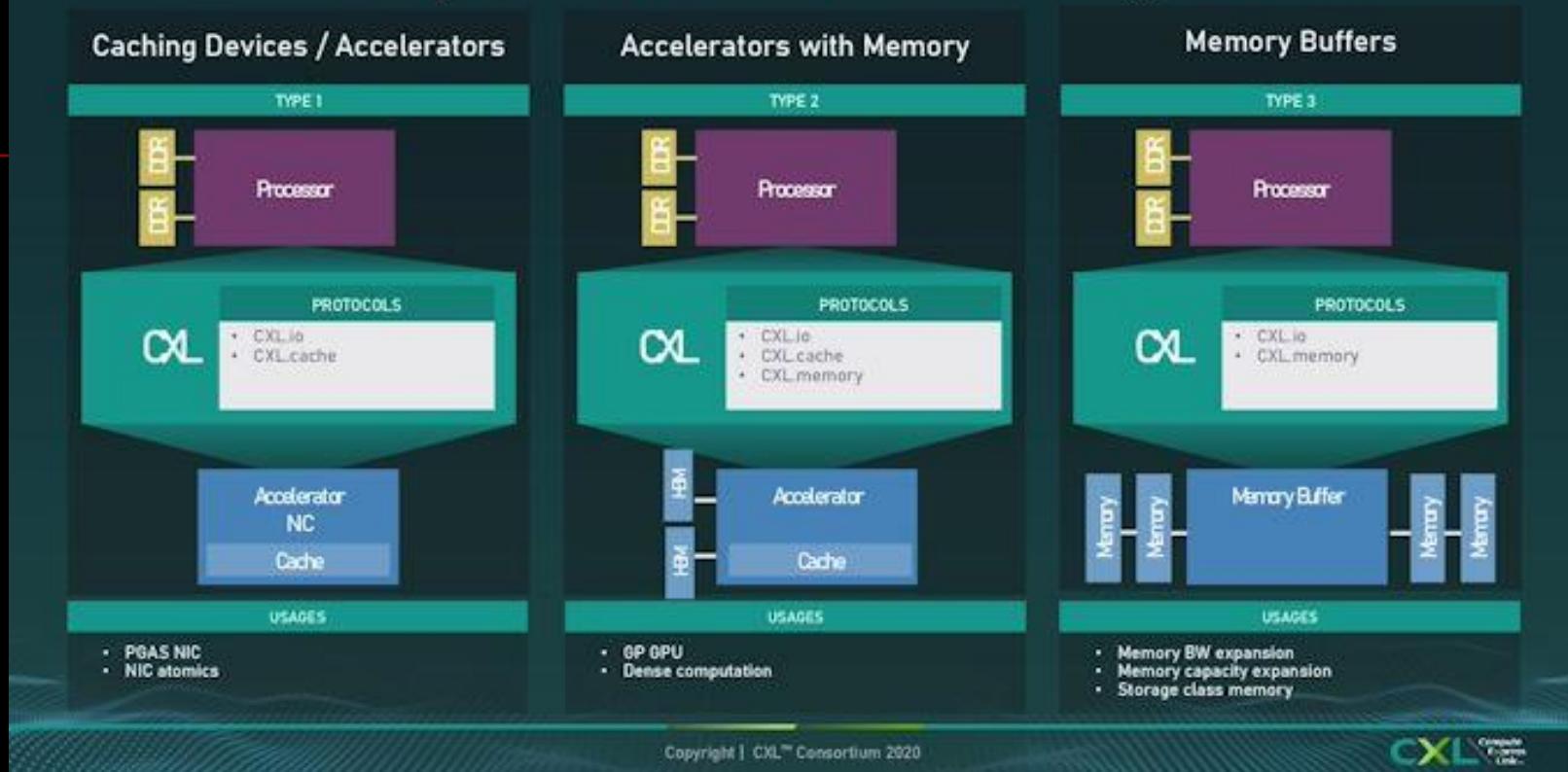
Source: [https://docs.wixstatic.com/ugd/0c1418\\_d9878707bbb7427786b70c3c91d5fbd1.pdf](https://docs.wixstatic.com/ugd/0c1418_d9878707bbb7427786b70c3c91d5fbd1.pdf)



Source: [https://docs.wixstatic.com/ugd/0c1418\\_d9878707bbb7427786b70c3c91d5fb1.pdf](https://docs.wixstatic.com/ugd/0c1418_d9878707bbb7427786b70c3c91d5fb1.pdf)



# Representative CXL Usages

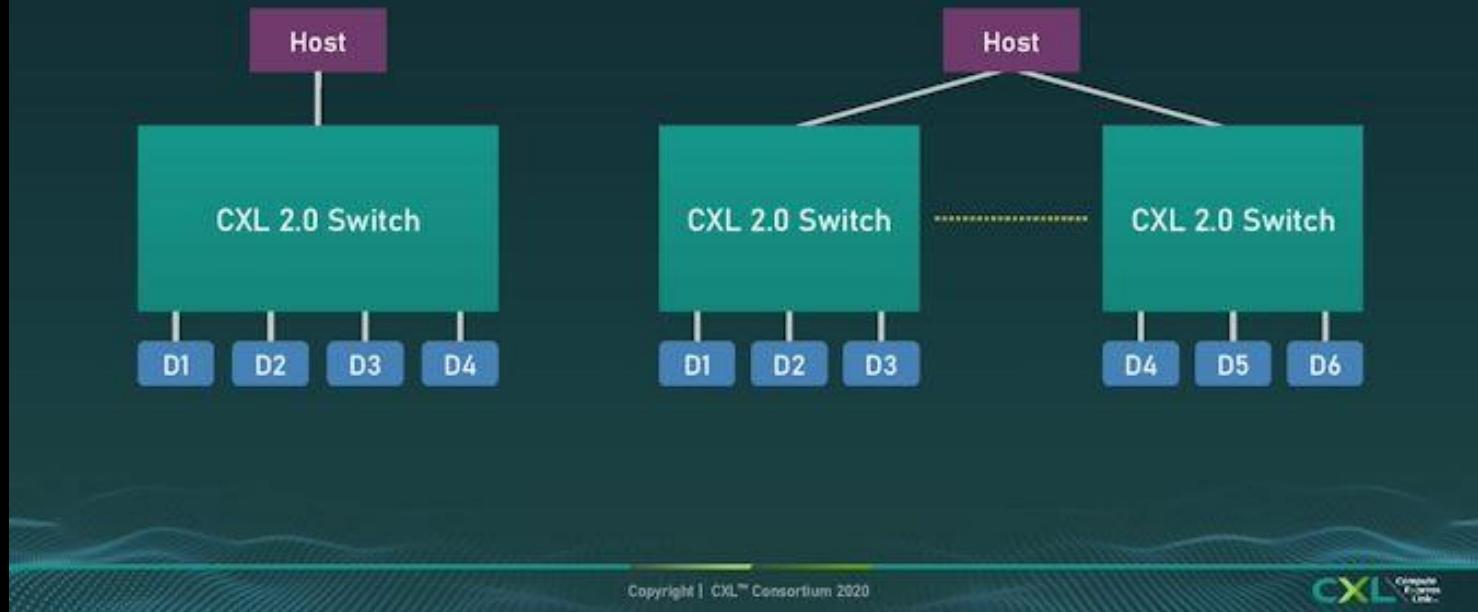




## Spec v2.0

- <https://www.computeexpresslink.org/post/compute-express-link-2-0-specification-now-available>
- **What's New?**

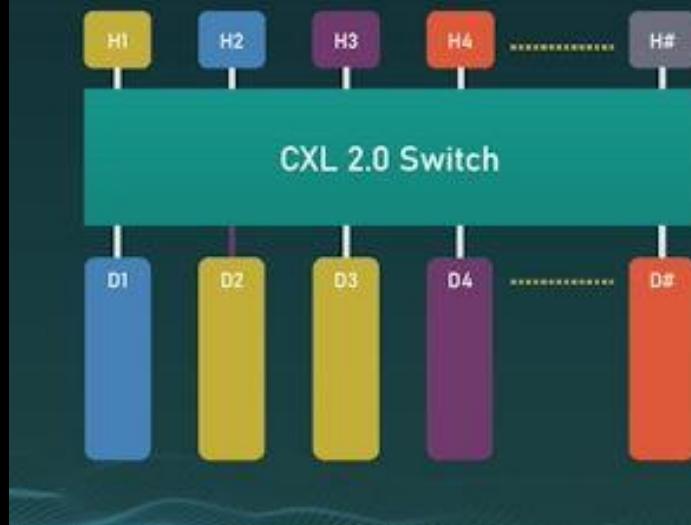
### Benefit of CXL 2.0 Switching Expansion



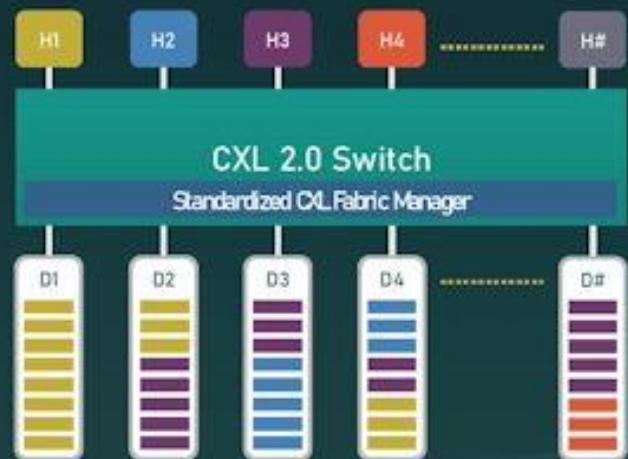


# Benefit of CXL 2.0 Switching Pooling

Memory/Accelerator Pooling with Single Logical Devices



Memory Pooling with Multiple Logical Devices





# Benefits of CXL 2.0 and Persistent Memory

Moves Persistent Memory  
from Controller to CXL

Enables Standardized Management  
of the Memory and Interface

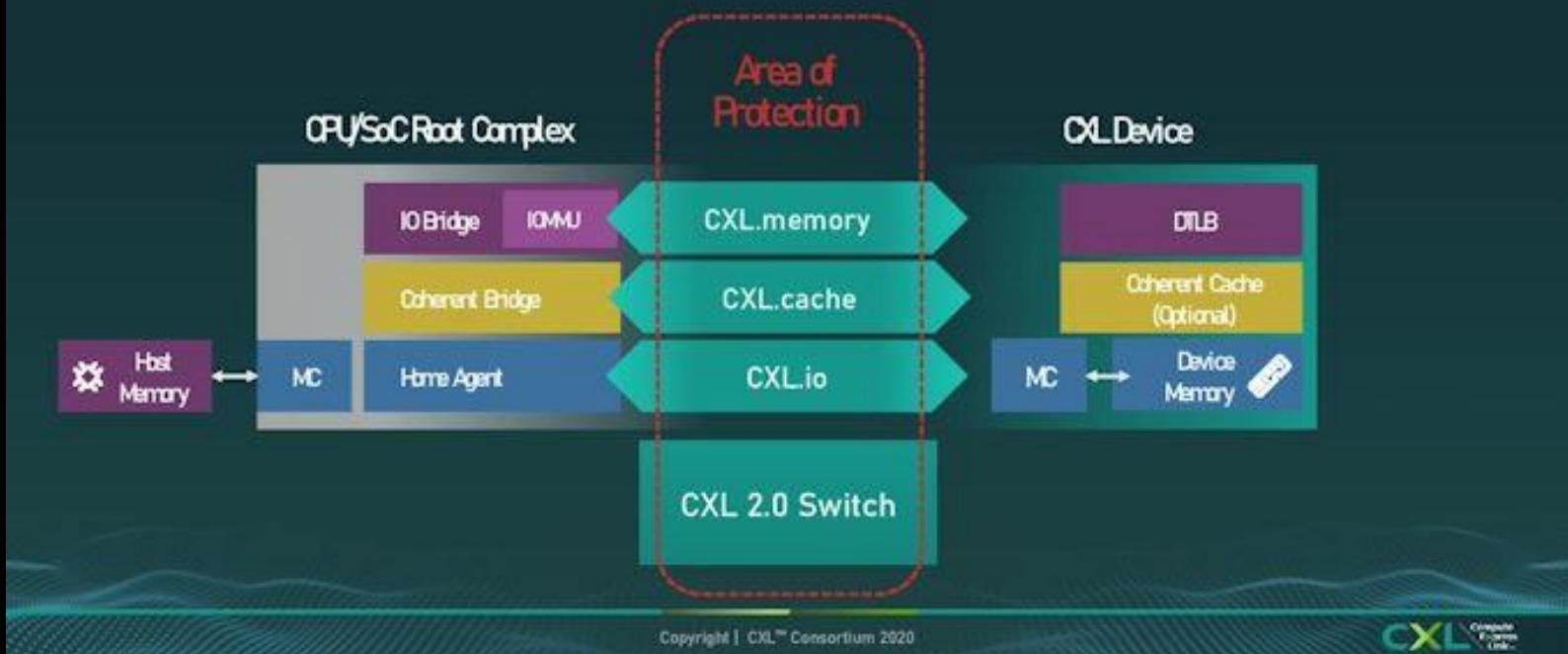
Supports a Wide Variety  
of Industry Form Factors





# CXL 2.0 Security Benefits

CXL 2.0 provides Integrity and Data Encryption of traffic across all entities (Root Complex, Switch, Device)



## Good Resources

- <https://www.computeexpresslink.org/blog>
- <https://www.rambus.com/blogs/compute-express-link/>
- <https://www.anandtech.com/show/16227/compute-express-link-cxl-2-0-switching-pmem-security>
- <https://semiengineering.com/cxl-and-omi-competing-or-complementary/>
- ...





## 1.2.1 CXL in Linux

- `$KERNEL_SRC/Documentation/driver-api/cxl`
  - `$KERNEL_SRC/drivers/cxl`
  - `$KERNEL_SRC/drivers/misc/cxl`
  - `$KERNEL_SRC/tools/testing/cxl`
- 

### Linux 5.18

- [https://www.phoronix.com/scan.php?page=news\\_item&px=Linux-5.18-CXL](https://www.phoronix.com/scan.php?page=news_item&px=Linux-5.18-CXL)
- <https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=b9132c32e01976686efa26252cc246944a0d2cab>



## 2) New Runtime for HPC

### 2.1 Java-based

#### 2.1.1 TornadoVM

- <https://www.tornadovm.org/>
- <https://github.com/beehive-lab/TornadoVM>

**A practical and efficient heterogeneous programming framework for managed languages.**

TornadoVM is a plug-in to OpenJDK and GraalVM that allows programmers to automatically run Java programs on heterogeneous hardware. TornadoVM currently targets OpenCL-compatible devices and it runs on multi-core CPUs, dedicated GPUs (NVIDIA, AMD), integrated GPUs (Intel HD Graphics and ARM Mali), and FPGAs (Intel and Xilinx).

- **Src**

Languages



Java 89.5%    C 5.9%    C++ 3.5%

Python 0.7%    Shell 0.4%

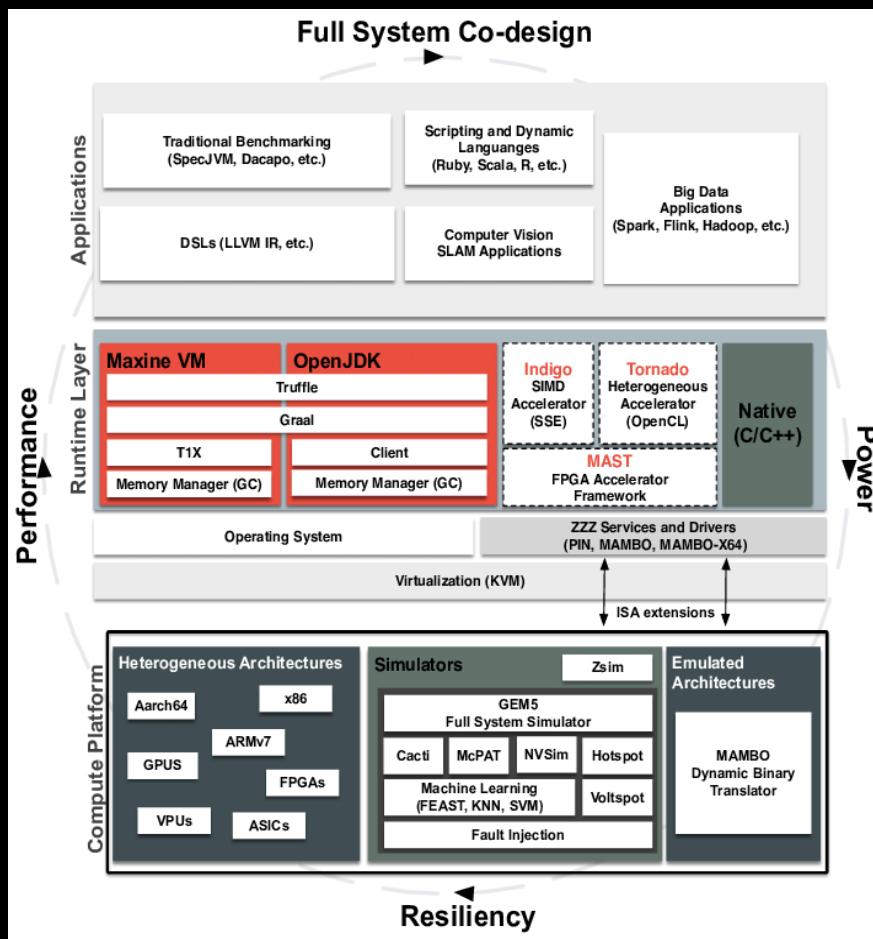
Makefile 0.0%

...

## History

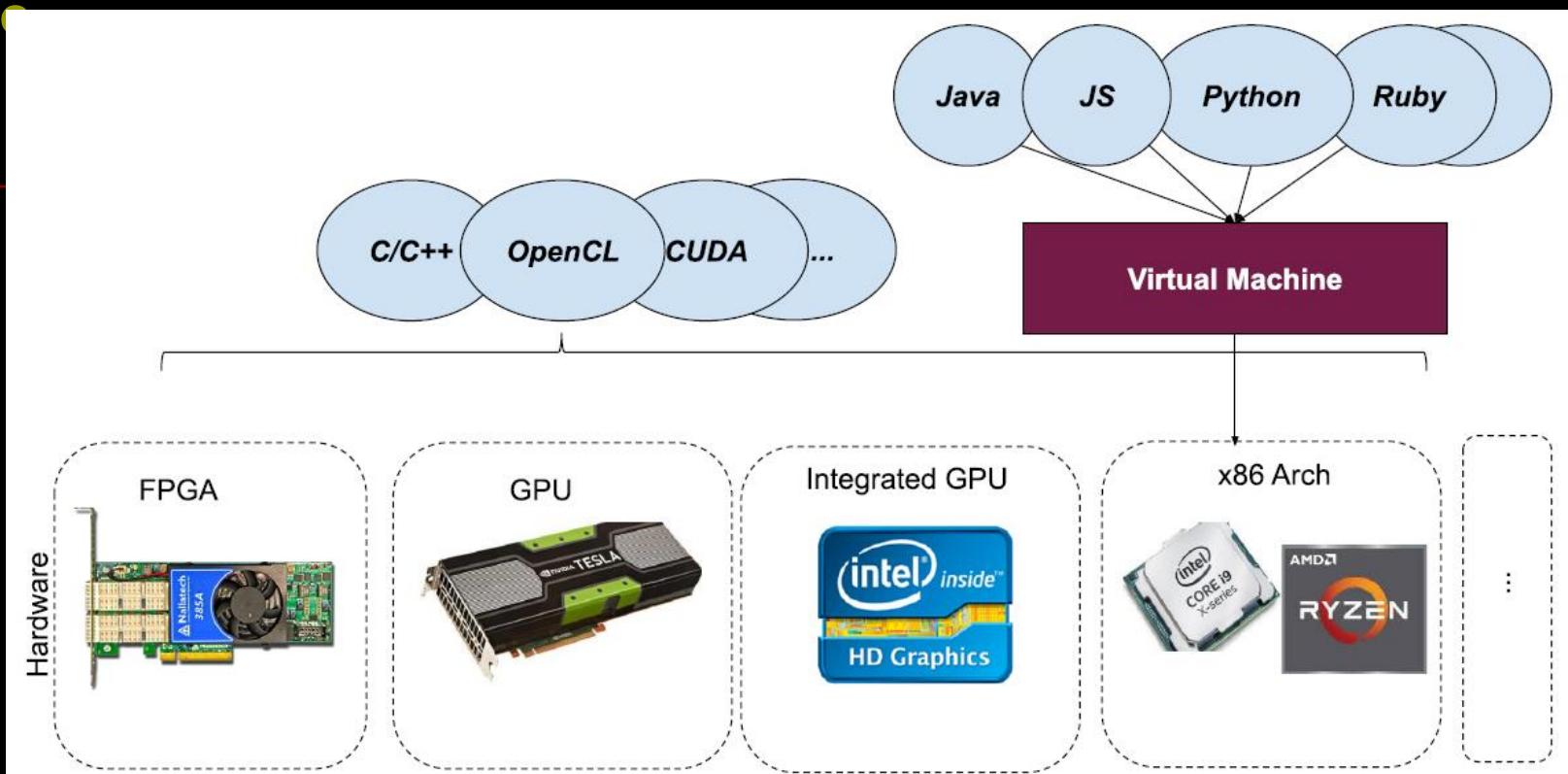
- <https://research.ac.upc.edu/multiprog/multiprog2016/papers/multiprog-15.pdf>

### Project Beehive: A Hardware/Software Co-designed Stack for Runtime and Architectural Research.





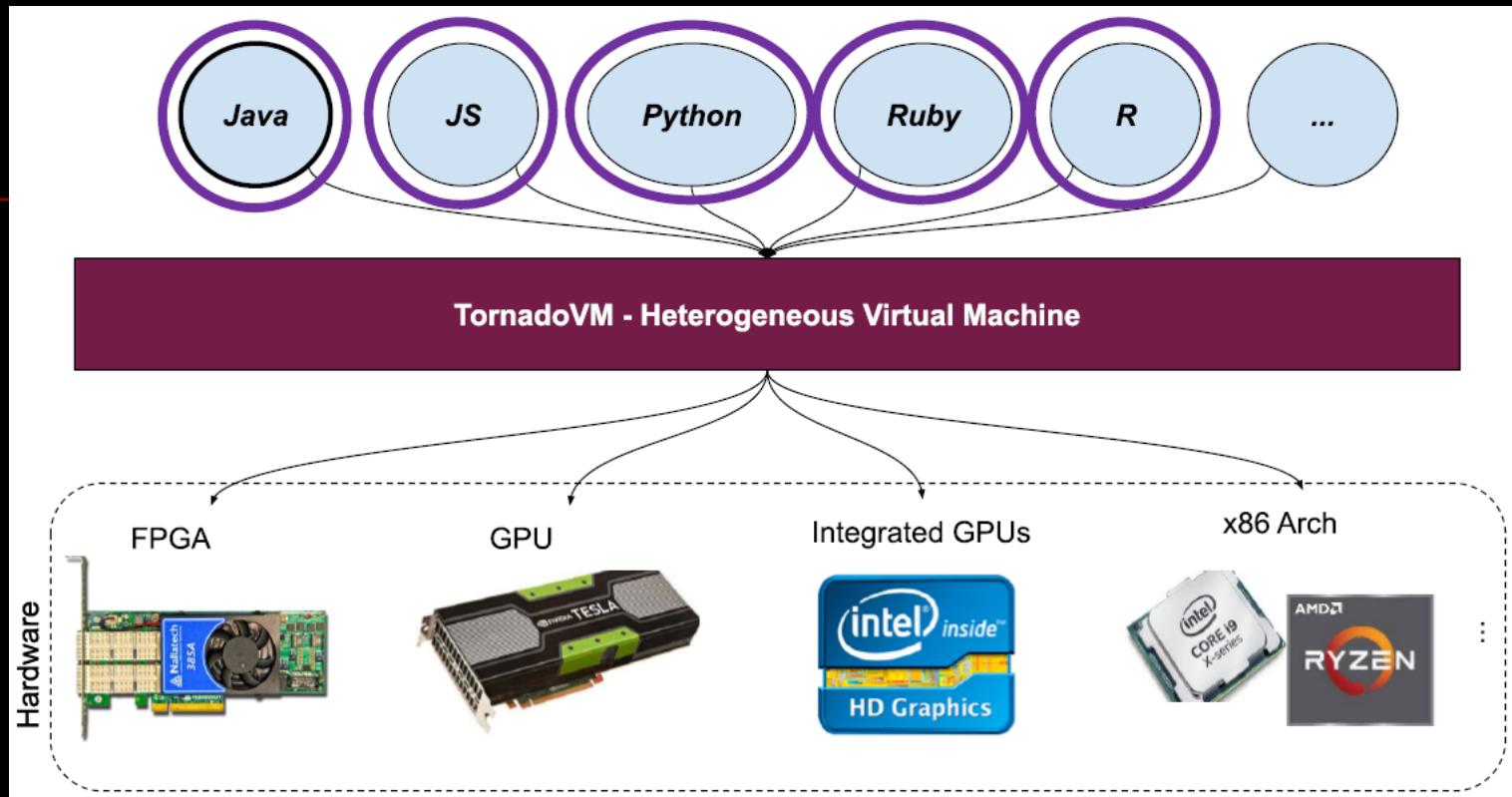
## Current Computer Systems & Prog. Lang.



Source: <https://jjfumero.github.io/talks/>



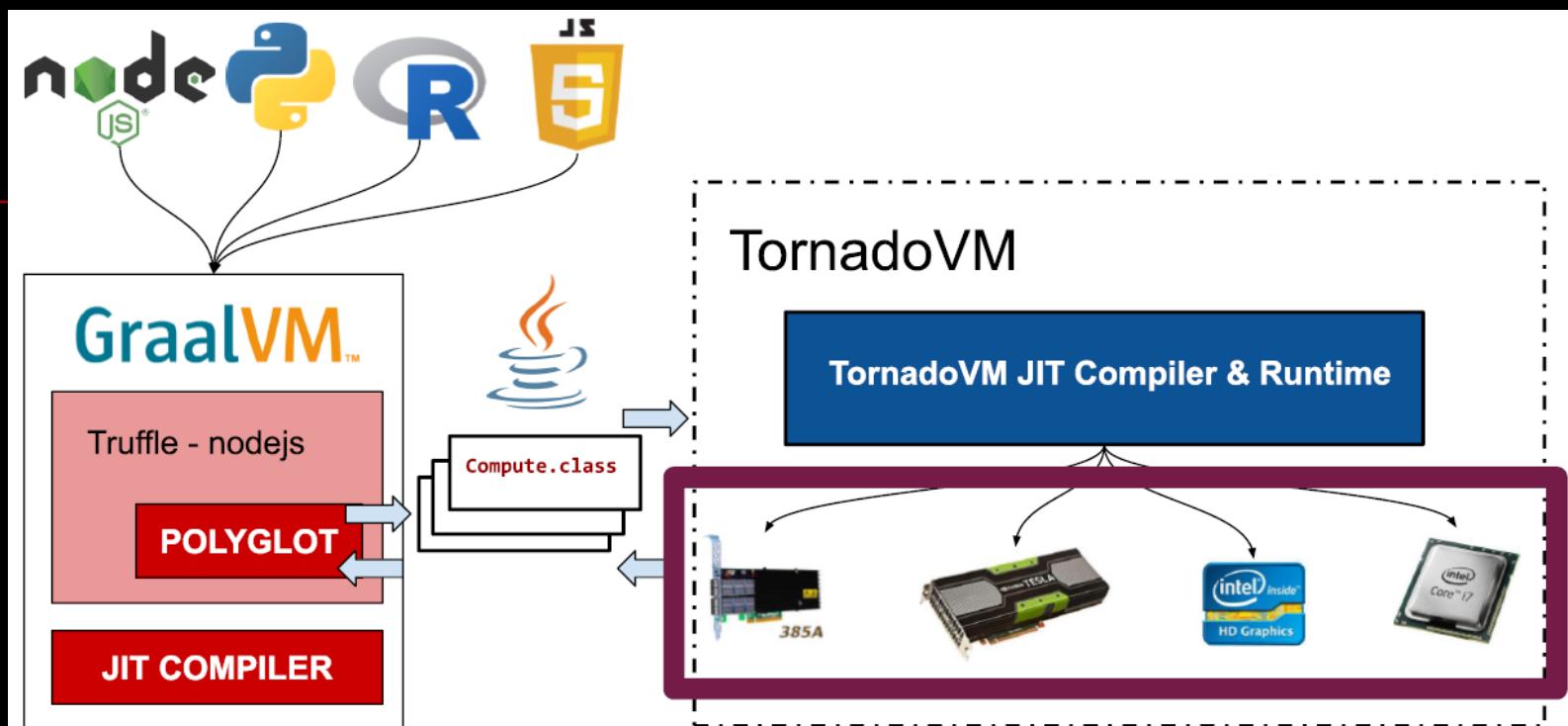
## Ideal System for Managed Languages



Source: <https://jjfumero.github.io/talks/>



## ■ TornadoVM & Dynamic/Untyped Programming Languages



Source: <https://jjfumero.github.io/talks/>



## SPIR-V backend

- [https://github.com/beehive-lab/TornadoVM/blob/master/assembly/src/docs/22\\_SPIRV\\_BACKEND\\_INSTALL.md](https://github.com/beehive-lab/TornadoVM/blob/master/assembly/src/docs/22_SPIRV_BACKEND_INSTALL.md)
- Currently only Intel oneAPI Level Zero Compute Runtime is supported.
- <https://github.com/beehive-lab/spirv-beehive-toolkit>  
Prototype for a SPIR-V assembler and disassembler. It provides a composable Java interface for generating SPIR-V code at runtime..
- ...



## TornadoVM v0.13

### ■ <https://github.com/beehive-lab/TornadoVM/releases/tag/v0.13>

- Integration with JDK 17 and Graal 21.3.0
  - JDK 11 is the default version and the support for the JDK 8 has been deprecated
- Support for extended intrinsics regarding math operations
- Native functions are enabled by default
- Support for 2D arrays for PTX and SPIR-V backends:
  - [2ef32ca](#)
- Integer Test Move operation supported:
  - [#177](#)
- Improvements in the SPIR-V Backend:
  - Experimental SPIR-V optimizer. Binary size reduction of up to 3x
    - [394ca94](#)
  - Fix malloc functions for Level-Zero
  - Support for pre-built SPIR-V binary modules using the TornadoVM runtime for OpenCL
  - Performance increase due to cached buffers on GPUs by default
  - Disassembler option for SPIR-V binary modules. Use `--printKernel`
- Improved Installation:
  - Full automatic installer script integrated
- Documentation about the installation for Windows 11
- Refactoring and several bug fixes
  - [5769418](#)
  - Vector types fixed:
    - <https://github.com/beehive-lab/TornadoVM/pull/181/files>
    - [004d61d](#)
  - Fix AtomicInteger get for OpenCL:
    - [#177](#)
- Dependencies for Math3 and Lang3 updated



## Details

- For details, you may refer to my previous talk "**GraalVM for Heterogeneous Parallel Computing**" at OSDT China 2021(Online).
-

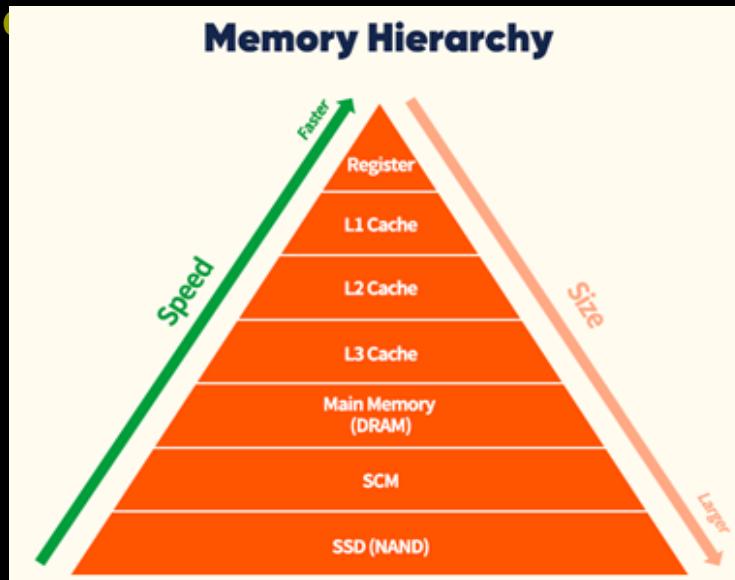


# VII. Storage

## 1) Background

### 1.1 Overview

- [https://en.wikipedia.org/wiki/Computer\\_data\\_storage](https://en.wikipedia.org/wiki/Computer_data_storage)
- 



Source: <https://www.eetimes.com/the-prospect-of-processing-in-memory-pim-in-memory-systems-for-ai-applications/>

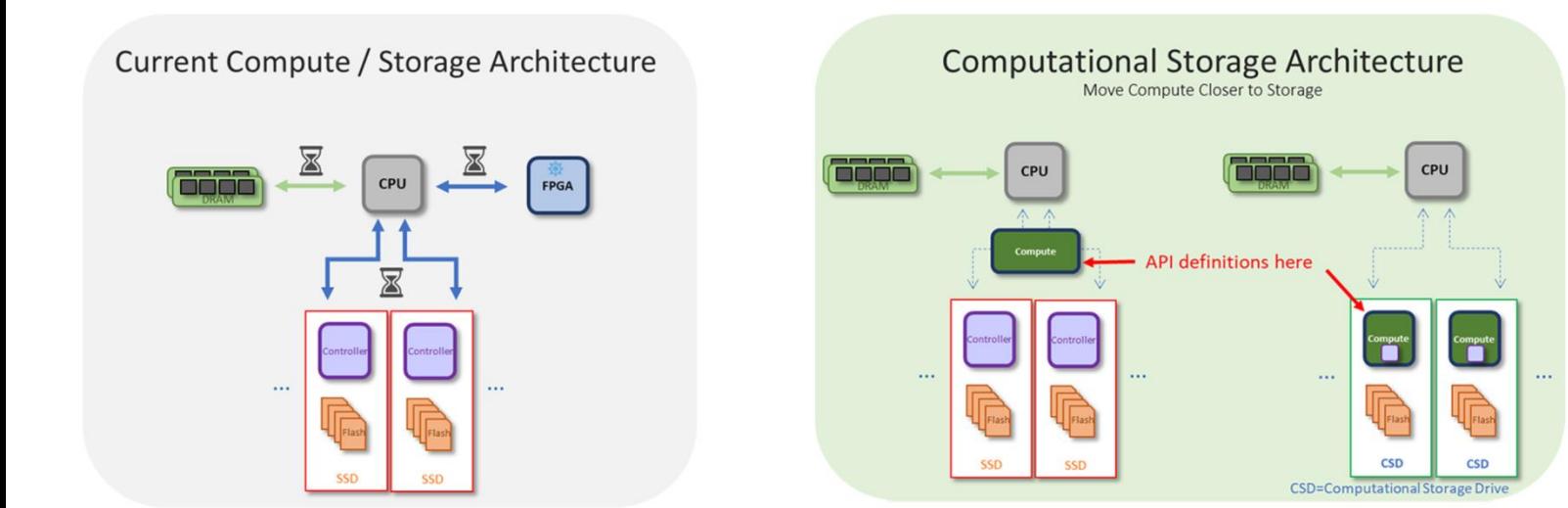
- [https://en.wikipedia.org/wiki/Solid-state\\_drive](https://en.wikipedia.org/wiki/Solid-state_drive)
- [https://en.wikipedia.org/wiki/Hard\\_disk\\_drive](https://en.wikipedia.org/wiki/Hard_disk_drive)
- [https://en.wikipedia.org/wiki/Cloud\\_storage](https://en.wikipedia.org/wiki/Cloud_storage)
- ...

## 1.2 Computational Storage

- <https://www.snia.org/education/what-is-computational-storage>

### What Is Computational Storage?

Computational Storage is defined as architectures that provide Computational Storage Functions (CSF) coupled to storage, offloading host processing or reducing data movement. These architectures enable improvements in application performance and/or infrastructure efficiency through the integration of compute resources (outside of the traditional compute & memory architecture) either directly with storage or between the host and the storage. The goal of these architectures is to enable parallel computation and/or to alleviate constraints on existing compute, memory, storage, and I/O. Learn more about [computational storage terms](#) in the SNIA Dictionary.



- <https://www.snia.org/sites/default/files/technical-work/computational/draft/SNIA-Computational-Storage-Architecture-and-Programming-Model-0.8R0-2021.06.09-DRAFT.pdf>





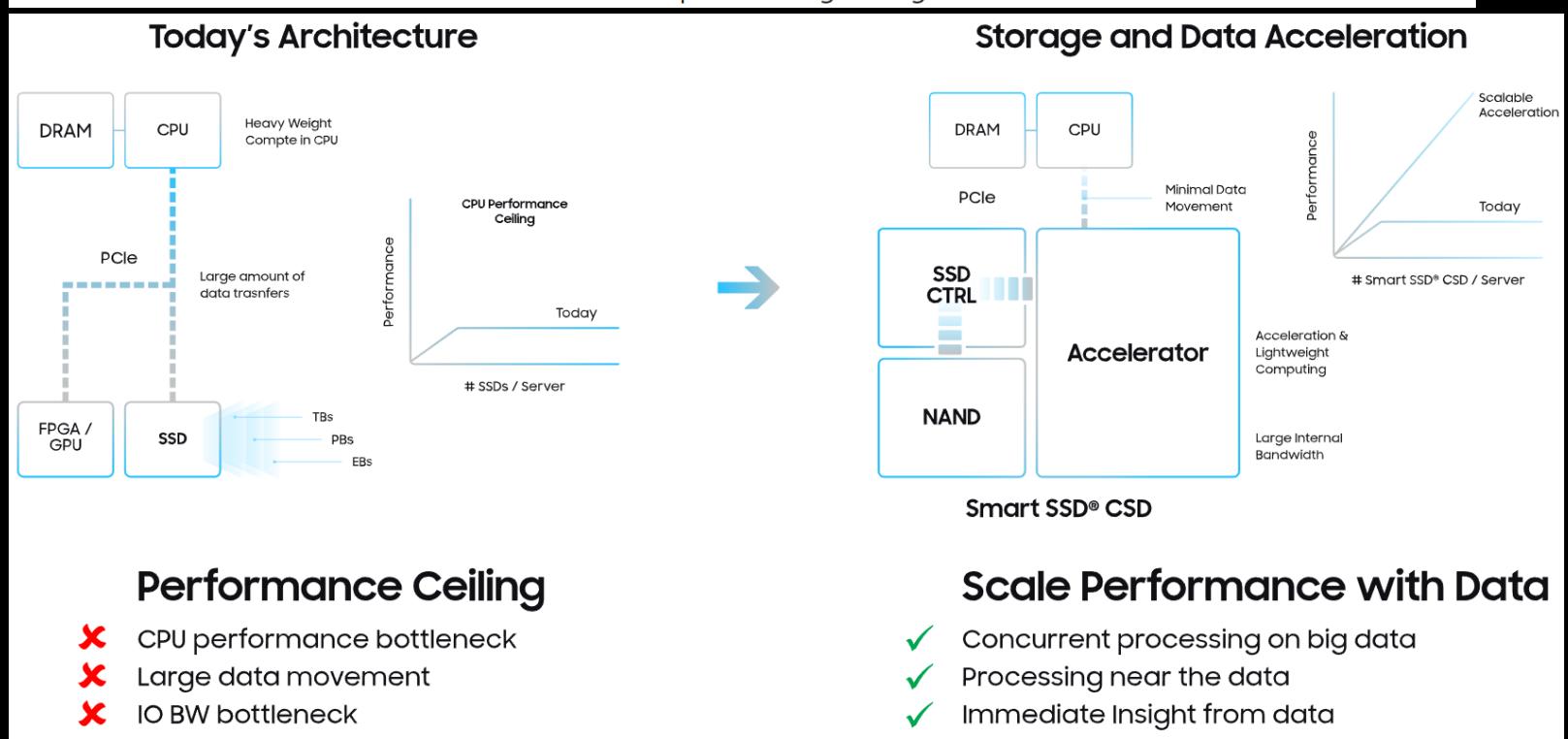


# SmartSSD

## ■ <https://samsungsemiconductor-us.com/smartssd/>

Today's architecture requires large data transfers between the CPU, GPU, SSD, and RAM. But when additional SSDs are added to a server, CPU performance causes a data bottleneck.

SmartSSD® Computational Storage Drive (CSD) adds processing power to the SSD itself, drastically reducing data movement and accelerating lightweight computing. This provides immediate insights and allows the concurrent processing of high volumes of data.





# Spec

<b>Form Factor</b>	2.5" (U.2)	
<b>Storage Capacity</b>	3.84TB (other capacities coming soon)	
<b>Host Interface</b>	Single-port PCIe Gen 3x4	
<b>Spec Compliance</b>	NVMe spec rev.1.3, PCIe base specification rev. 3.0, NVMe Management Interface (NVMe MI)1.0	
<b>Programmable Hardware Accelerator (FPGA)</b>	Xilinx Kintex Ultrascale+ KU15P FPGA	
	System Logic Cells	1.143 Million
	Available LUTs for acceleration tasks	Approx. 300k
	DSP Slices	1,968
	Internal Distributed RAM	34.6 Mbit
	Internal UltraRAM	36.0 Mbit
	Accelerator-dedicated DRAM	4 GByte DDR4 SDRAM @ 2400 Mbps
	Speed Grade	-2LE
<b>SSD Controller</b>	Enterprise class SSD controller	
<b>NAND Flash Memory</b>	Samsung V-NAND®	
	Write Endurance	1 DWPD for 5 Years
	Sequential 128k Read, QD 256	3300 MB/sec
	Sequential 128k Write, QD 256	2000 MB/sec
	Random 4k Read, QD 64	800,000 IOPS
	Random 4k Write, QD 64	110,000 IOPS
	Uncorrectable Bit Error Rate (UBER)	1 sector per $10^{17}$ bits read
	Mean Time Between Failure (MTBF)	2,000,000 hours
<b>Power Consumption</b>	Dynamic power management and throttling	
<b>Operating Temperature</b>	Commercial range	
<b>Physical Dimensions</b>	69 x 100 x 15 mm	
<b>Weight</b>	400 grams	

Source: [https://samsungsemiconductor-us.com/smartssd-archive/pdf/SmartSSD\\_ProductBrief\\_13.pdf](https://samsungsemiconductor-us.com/smartssd-archive/pdf/SmartSSD_ProductBrief_13.pdf)



# ■ Development



## Flexible Accelerator IP Development Options

- **Third party IP development:** Samsung and Xilinx partners provide IP and acceleration solutions for deployment on the SmartSSD drive. Custom IP development is also available via partners.
- **Redeployment of IP:** From Cloud to Enterprise workloads, Xilinx tools provide seamless FPGA IP mobility.
- **Simplified development:** The Xilinx Vitis environment allows development in C, C++, or OpenCL. By using a fixed "Design Support Archive" (DSA) I/O shell, developers can achieve high productivity by focusing only on the accelerator kernel under development.
- **HDL development:** The Xilinx Run Time environment allows access to the full spectrum of hardware description languages (HDLs), including Verilog and VHDL, for maximum design flexibility and optimization. This design flow also simplifies the re-use of existing accelerator IP designed in HDL for ASICs or FPGAs.

## Internal Data Path

The SSD controller provides the NAND media interface and management while the FPGA provides logic elements and CPU cores for acceleration. A private, high-speed peer-to-peer link connects the SSD controller to the FPGA and transfers data between them. This internal bandwidth scales as SmartSSDs are added to a system.

Source: [https://samsungsemiconductor-us.com/smartssd-archive/pdf/SmartSSD\\_ProductBrief\\_13.pdf](https://samsungsemiconductor-us.com/smartssd-archive/pdf/SmartSSD_ProductBrief_13.pdf)



## ■ Use cases



Life Sciences & Genomics



Immediate Insights from Data Lakes



Financial Services



Video Transcoding

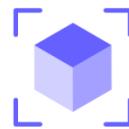


Image Recognition and Object Detection



Real-Time Log Analytics



Speedup Complex Ad-hoc Queries



Enhance Business Intelligence and Data Warehousing



Artificial Intelligence and Machine Learning

## Good Resources

- [https://en.wikipedia.org/wiki/In-situ\\_processing](https://en.wikipedia.org/wiki/In-situ_processing)
- <https://www.arm.com/solutions/storage/computational-storage>
- <https://www.xilinx.com/applications/data-center/computational-storage.html>

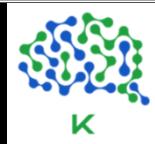
---

- <https://arxiv.org/abs/2112.09691>
- ...



## 1.3 In-memory Computing

- <https://>
  - [https://en.wikipedia.org/wiki/In-memory\\_processing](https://en.wikipedia.org/wiki/In-memory_processing)
- 





## 2) eBPF-powered Userland Filesystems

### 2.1 FUSE BPF

- <https://linuxplumbersconf.org/event/11/contributions/1048/>  
**Experimental, soon on LKML...**
- **FUSE BPF: stacking fs + passthrough + extFUSE ?**

Implement a generic stacking file system

Allow **requests** to either be handled by FUSE or the backing file system

Allow **pre** and **post filtering** of backing file system request

**Filtering** can be either **by the kernel**, or through FUSE-style requests to **userspace**

Overcome **FUSE passthrough limitations** (per-file, read/write/mmap only)

Inspiration from

- **extFUSE**, presented by Ashish Bijlani at Plumbers in 2019 (<https://linuxplumbersconf.org/event/4/contributions/415/>)
- **FUSE passthrough**
- Stacking file systems, e.g., **incremental fs**

Source: “FS stacking with FUSE: performance issues and mitigations”, Alessio Balsini & Paul Lawrence, LPC 2021.



## At a glance

■ Add to fuse\_inode:

- struct inode \*backing\_inode;
- struct bpf\_prog \*bpf;

These may be set at mount time for root, at lookup time for all other inodes

If backing\_inode exists, **all** requests will be conditionally sent **to the backing inode**, **else** we are in **classic FUSE** mode

If **no bpf**: simply forward as is (pure **passthrough** mode)

If **bpf**: format fuse\_args with in\_args and send to BPF, which may redirect request to **classic FUSE** or

1. Optionally request user-mode pre-filter with same modifiable in\_args
2. (Potentially modified) request is sent to backing file system
3. Optionally pass in\_args & out\_args to BPF post-filter
4. Optionally pass in\_args & out\_args to user-mode post-filter

Early prototypes being tested within Android team

Source: “**FS stacking with FUSE: performance issues and mitigations**”, Alessio Balsini & Paul Lawrence, **LPC 2021**.



## Some thoughts

### ■ FUSE passthrough

How can we do better for Linux?

Do we really want *FUSE\_PASSTHROUGH\_CLOSE*?

Can be done with a mapping container (e.g., *IDR*),  
but is not as simple as *fuse2* (extra spinlocks)

Source: “FS stacking with FUSE: performance issues and mitigations”, Alessio Balsini & Paul Lawrence, LPC 2021.

### FUSE BPF

BPF is a good compromise between user space  
and kernel space (good fit for FUSE)

Would the Linux community benefit from this?

Is such architecture upstreamable?



### 3) eBPF in Computational Storage

#### 3.1 eBPF CSEE

■	6 EXAMPLE COMPUTATIONAL STORAGE EXECUTION ENVIRONMENT .....	33
	6.1 OPERATING SYSTEM CSEE .....	33
	6.2 CONTAINER PLATFORM CSEE .....	33
	6.3 CONTAINER CSEE .....	33
	6.4 eBPF CSEE .....	33
	6.5 FPGA BITSTREAM CSEE .....	33

Source: <https://www.snia.org/sites/default/files/technical-work/computational/draft/SNIA-Computational-Storage-Architecture-and-Programming-Model-0.8R0-2021.06.09-DRAFT.pdf>

#### ■ 6.4 eBPF CSEE

A Berkeley Packet Filter (eBPF) CSEE provides an environment for running eBPF programs. The eBPF CSEE may contain one or more activated eBPF CSFs and supports the activation of one or more downloaded eBPF CSFs.

- ...



## 3.2 ZCSD

- <https://github.com/DantaliOn/qemu-csd>  
**eBPF Computational Storage Device (CSD) for Zoned Namespace (ZNS) SSDs in QEMU.**
- **Motivation**

The Big Data trend is putting strain on modern storage systems, which have to support high-performance I/O accesses for the large quantities of data. With the prevalent Von Neumann computing architecture, this data is constantly moved back and forth between the computing (i.e., CPU) and storage entities (DRAM, Non-Volatile Memory NVM storage). Hence, as the data volume grows, this constant data movement between the CPU and storage devices has emerged as a key performance bottleneck. To improve the situation, researchers have advocated to leverage computational storage devices (CSDs), which offer a programmable interface to run user-defined data processing operations close to the storage without excessive data movement, thus offering performance improvements. However, despite its potential, building CSD-aware applications remains a challenging task due to the lack of exploration and experimentation with the right API and abstraction. This is due to the limited accessibility to latest CSD/NVM devices, emerging device interfaces, and closed-source software internals of the devices. To remedy the situation, in this work we present an open-source CSD prototype over emerging NVMe Zoned Namespaces (ZNS) SSDs and an interface that can be used to explore application designs for CSD/NVM storage devices. In this paper we summarize the current state of the practice with CSD devices, make a case for designing a CSD prototype with the ZNS interface and eBPF (ZCSD), and present our initial findings.

Source: <https://arxiv.org/abs/2112.00142>

- **Definitions**

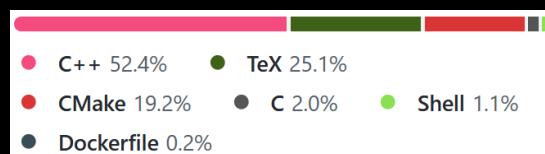
### OpenCSD

OpenCSD is an improved version of ZCSD achieving snapshot consistency log-structured filesystem (LFS) integration on Zoned Namespaces (ZNS) Computational Storage Devices (CSD). Below is a diagram of the overall architecture as presented to the end user. However, the actual implementation differs due to the use of emulation using technologies such as QEMU, uBPF and SPDK.

### ZCSD

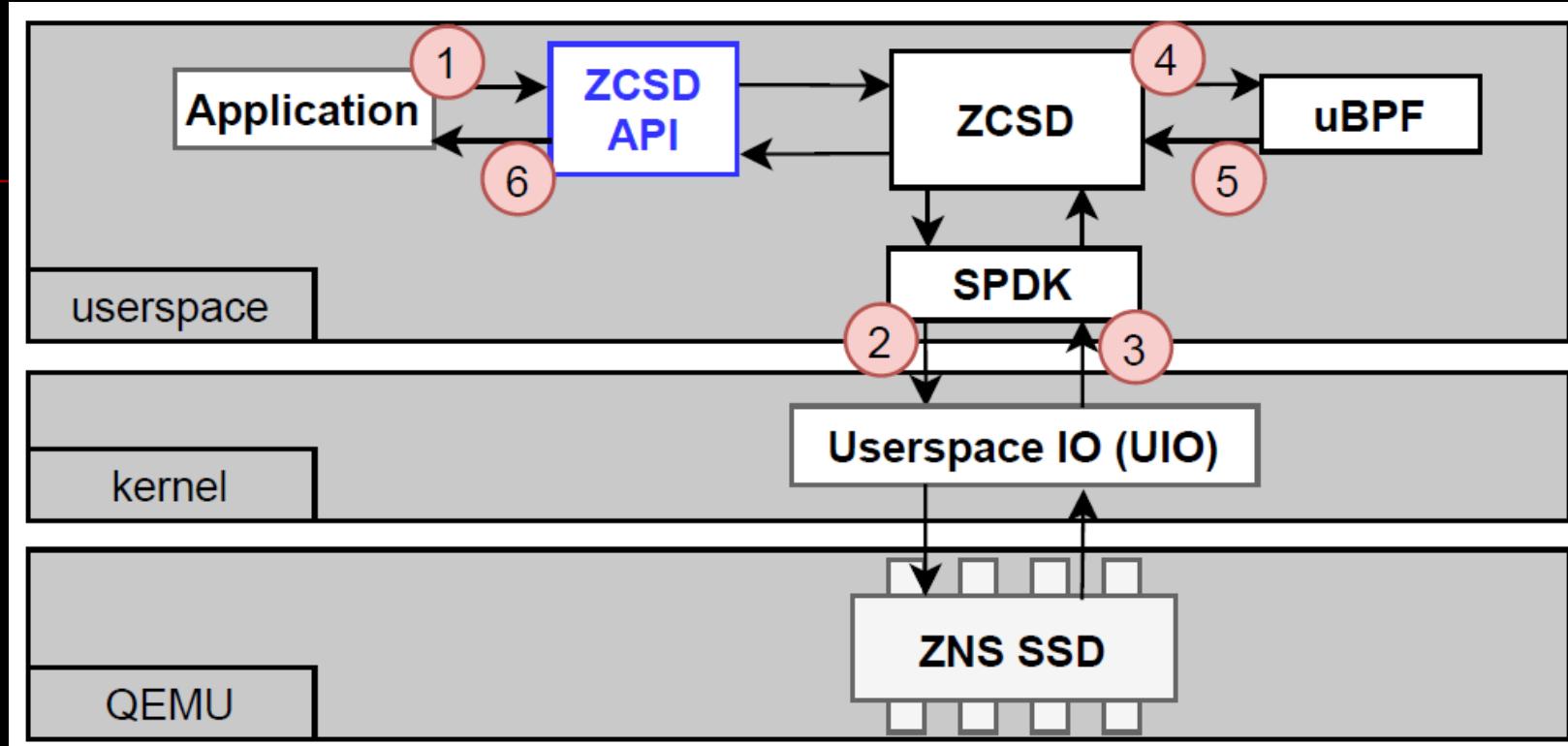
ZCSD is a full stack prototype to execute eBPF programs as if they are running on a ZNS CSD SSDs. The entire prototype can be run from userspace by utilizing existing technologies such as SPDK and uBPF. Since consumer ZNS SSDs are still unavailable, QEMU can be used to create a virtual ZNS SSD. The programming and interactive steps of individual components is shown below.

- **Languages(not includes submodules)**



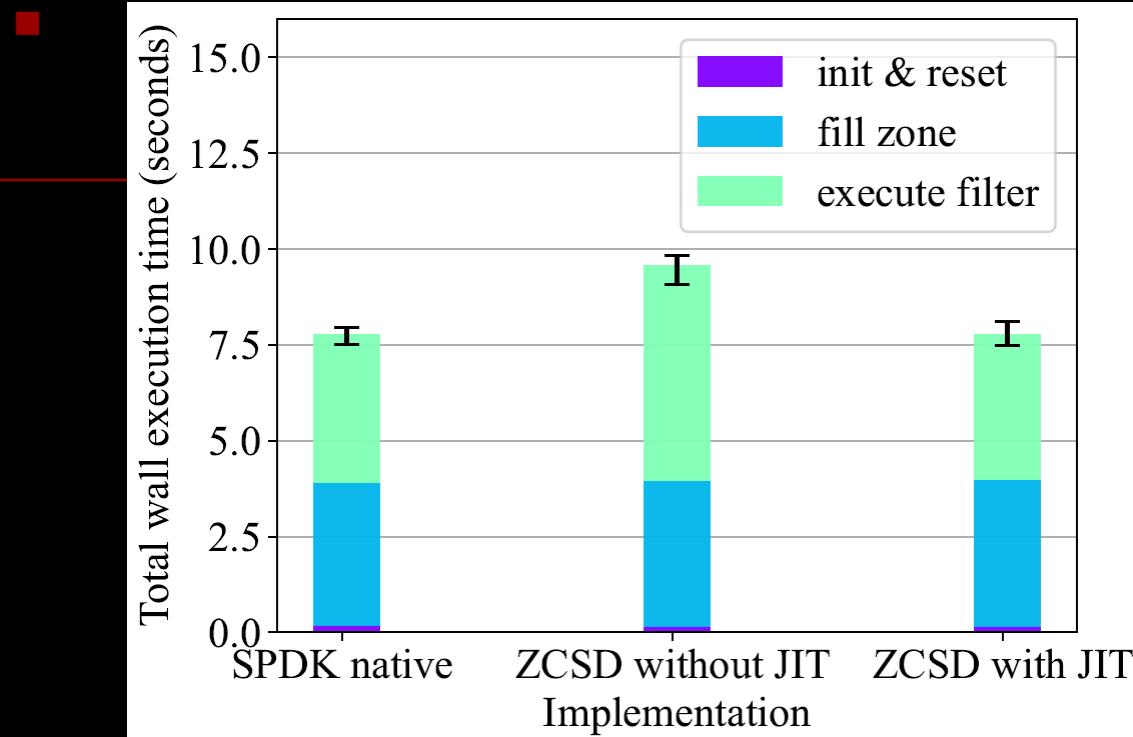


## Overview of the components in the prototype



Source: <https://arxiv.org/abs/2112.00142>

## Overview of the performance across the different scenarios

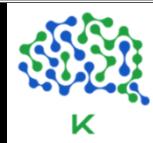


Source: <https://arxiv.org/abs/2112.00142>

## 4) eBPF-based In-kernel Storage

### Overview

- ...
- 





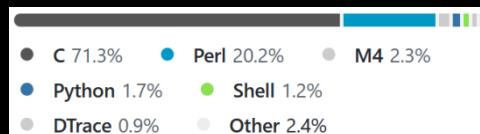
## 4.1 BMC

- <https://github.com/Orange-OpenSource/bmc-cache>

### In-kernel cache based on eBPF.

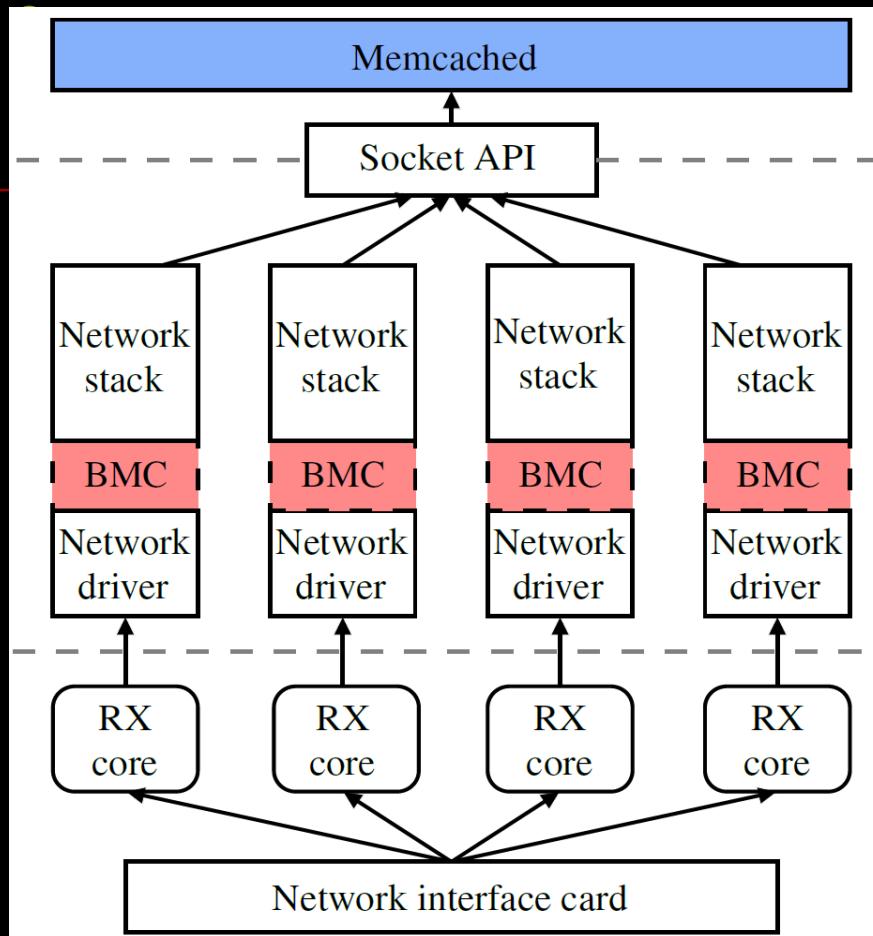
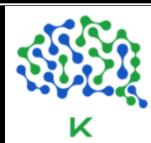
BMC (BPF Memory Cache) is an in-kernel cache for memcached. It enables runtime, crash-safe extension of the Linux kernel to process specific memcached requests before the execution of the standard network stack. BMC does not require modification of neither the Linux kernel nor the memcached application. Running memcached with BMC improves throughput by up to 18x compared to the vanilla memcached application.

- Languages



...

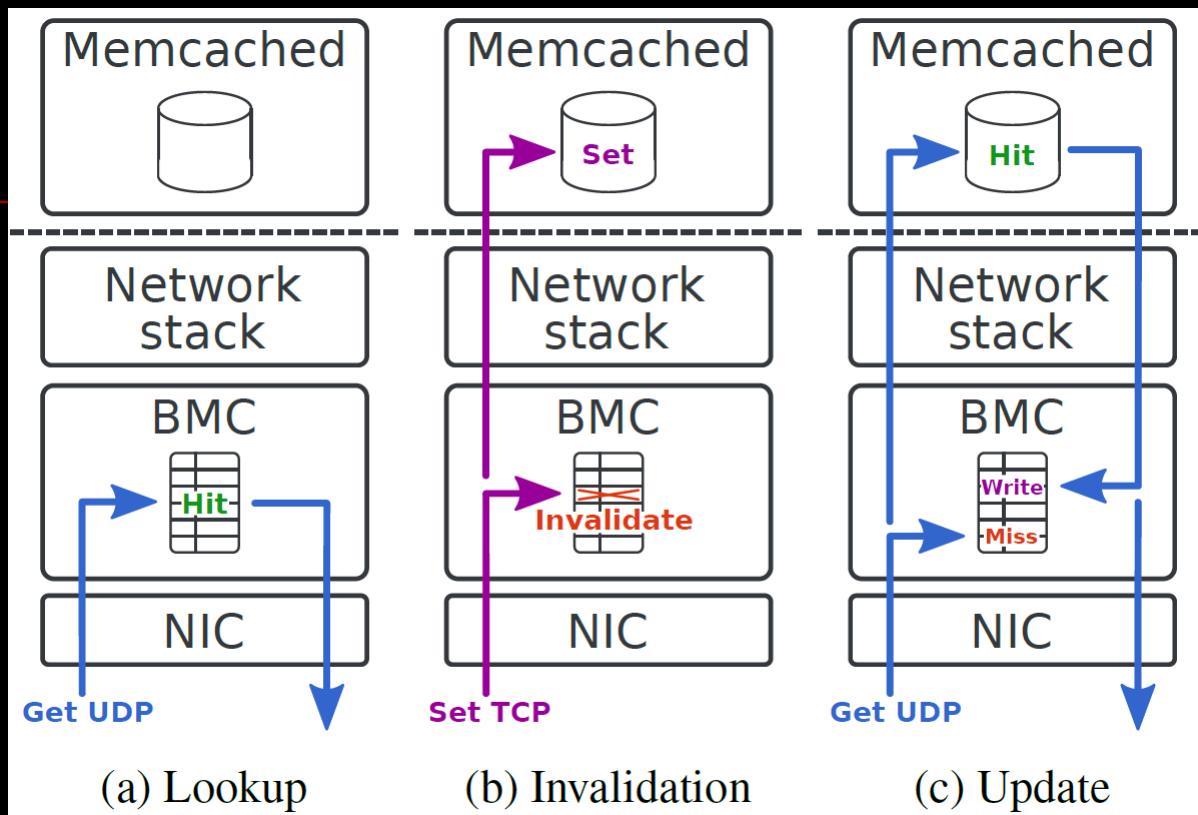
## Architecture & Design



Source: <https://www.usenix.org/system/files/nsdi21-ghigoff.pdf>



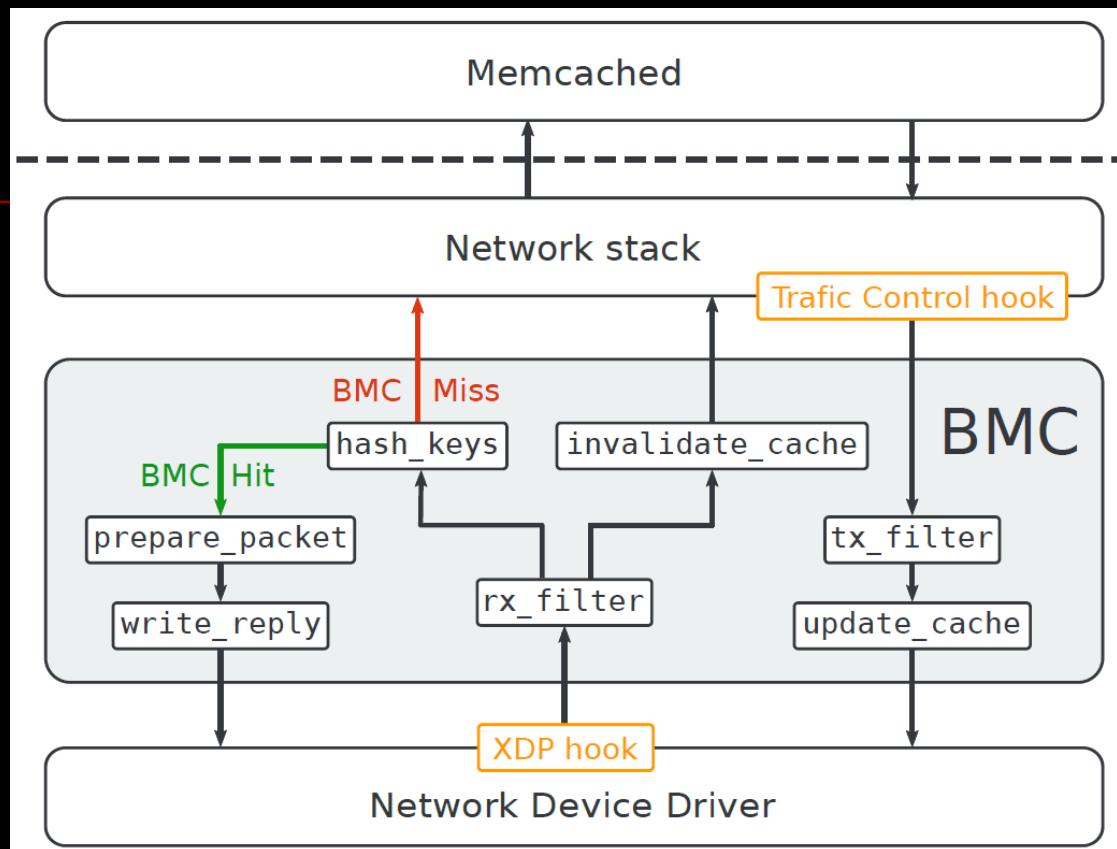
## BMC cache operations



Source: <https://www.usenix.org/system/files/nsdi21-ghigoff.pdf>



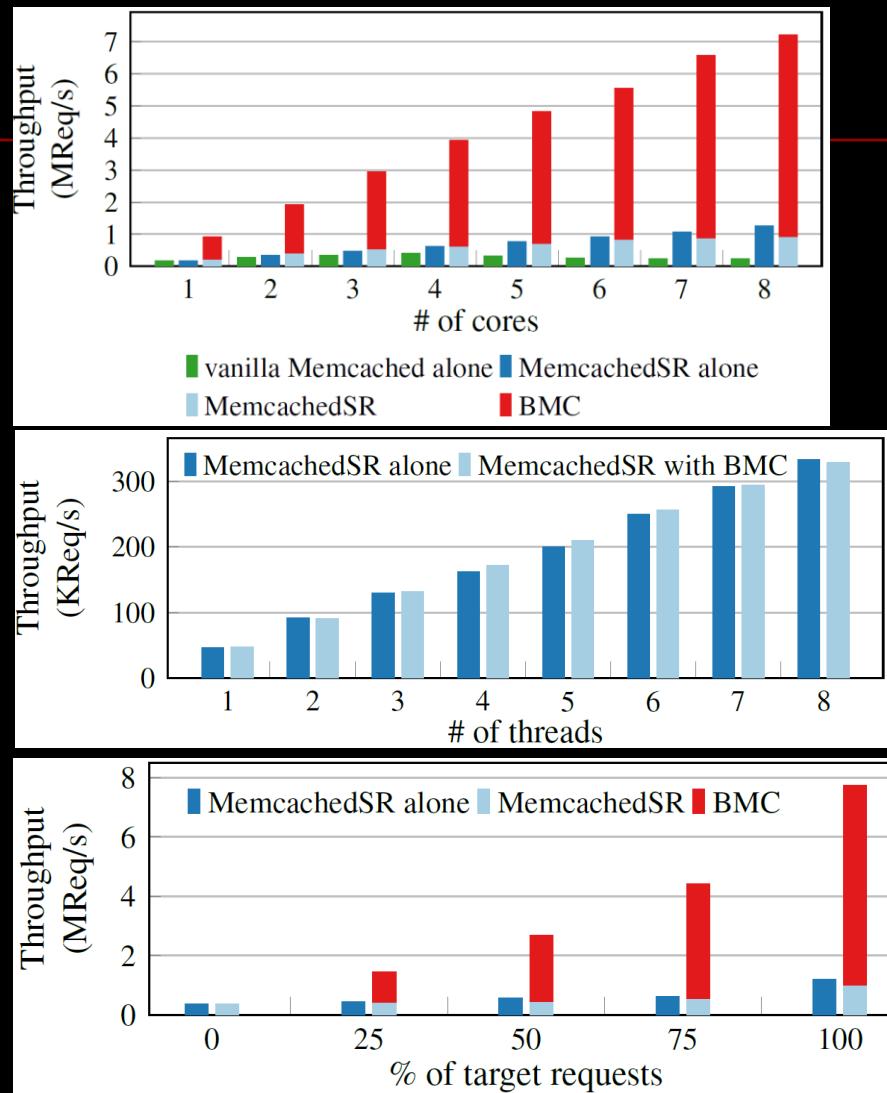
## ■ Division of BMC into seven eBPF programs



Source: <https://www.usenix.org/system/files/nsdi21-ghigoff.pdf>

## Throughput

- <https://www.usenix.org/system/files/nsdi21-ghigoff.pdf>



# VIII. eBPF in Distributed AI



## 1) Data Processing

### 1.1 Overview

---

- <https://en.wikipedia.org/wiki/Data>
- [https://en.wikipedia.org/wiki/Data\\_processing](https://en.wikipedia.org/wiki/Data_processing)
- [https://en.wikipedia.org/wiki/Data\\_type](https://en.wikipedia.org/wiki/Data_type)
- ...

## 1.2 Apache Arrow

### ■ [https://en.wikipedia.org/wiki/Apache\\_Arrow](https://en.wikipedia.org/wiki/Apache_Arrow)

Apache Arrow is a language-agnostic software framework for developing data analytics applications that process columnar data. It contains a standardized column-oriented memory format that is able to represent flat and hierarchical data for efficient analytic operations on modern CPU and GPU hardware.<sup>[3][4][5][6][7]</sup> This reduces or eliminates factors that limit the feasibility of working with large sets of data, such as the cost, volatility, or physical constraints of dynamic random-access memory.<sup>[8]</sup>

#### Interoperability [edit]

Arrow can be used with Apache Parquet, Apache Spark, NumPy, PySpark, pandas and other data processing libraries. The project includes native software libraries written in C, C++, C#, Go, Java, JavaScript, Julia, MATLAB, Python, R, Ruby, and Rust. Arrow allows for zero-copy reads and fast data access and interchange without serialization overhead between these languages and systems.<sup>[3]</sup>

#### Applications [edit]

Arrow has been used in diverse domains, including analytics,<sup>[9]</sup> genomics,<sup>[10][8]</sup> and cloud computing.<sup>[11]</sup>

#### Comparison to Apache Parquet and ORC [edit]

Apache Parquet and Apache ORC are popular examples of on-disk columnar data formats. Arrow is designed as a complement to these formats for processing data in-memory.<sup>[12]</sup> The hardware resource engineering trade-offs for in-memory processing vary from those associated with on-disk storage.<sup>[13]</sup> The Arrow and Parquet projects includes libraries that allow for reading and writing data between the two formats.<sup>[14]</sup>

### ■ <https://arrow.apache.org/>

A cross-language development platform for in-memory analytics.



A multi-language toolbox for accelerated data interchange and in-memory processing

#### LANGUAGE INDEPENDENT

Defines a language-independent columnar memory format for flat and hierarchical data

#### HIGH-PERFORMANCE TRANSPORT

Standardizes data representation to drastically improve the performance of connected systems

#### HARDWARE AGNOSTIC

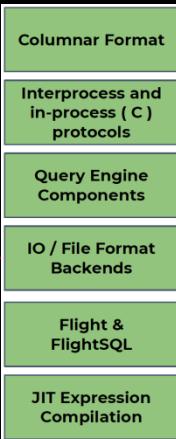
Enables efficient analytical operations on modern hardware like CPU's and GPU's

#### DEVELOPER FLEXIBILITY

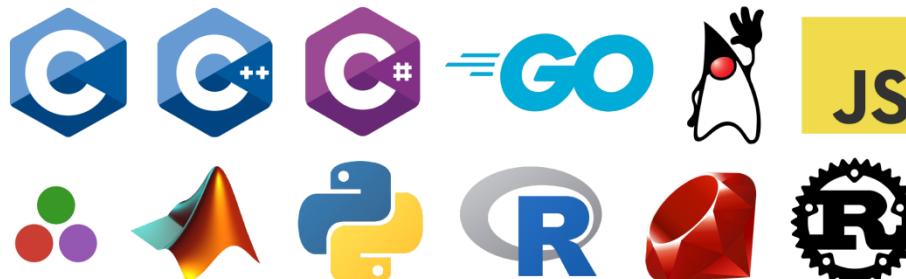
Libraries provide building blocks for various use cases, and are available in C, C++, C#, Go, Java, JavaScript, Julia, MATLAB, Python, R, Ruby, Rust

Source: <https://www.slideshare.net/wesm/apache-arrow-high-performance-columnar-data-framework>





Apache Arrow is **doing for data analytics what LLVM did for compiler infrastructure**. Modular, reusable software components for building high-performance analytics systems.



Source: <https://www.slideshare.net/wesm/apache-arrow-high-performance-columnar-data-framework>

## <https://github.com/apache/arrow>

Apache Arrow is a development platform for in-memory analytics. It contains a set of technologies that enable big data systems to process and move data fast.

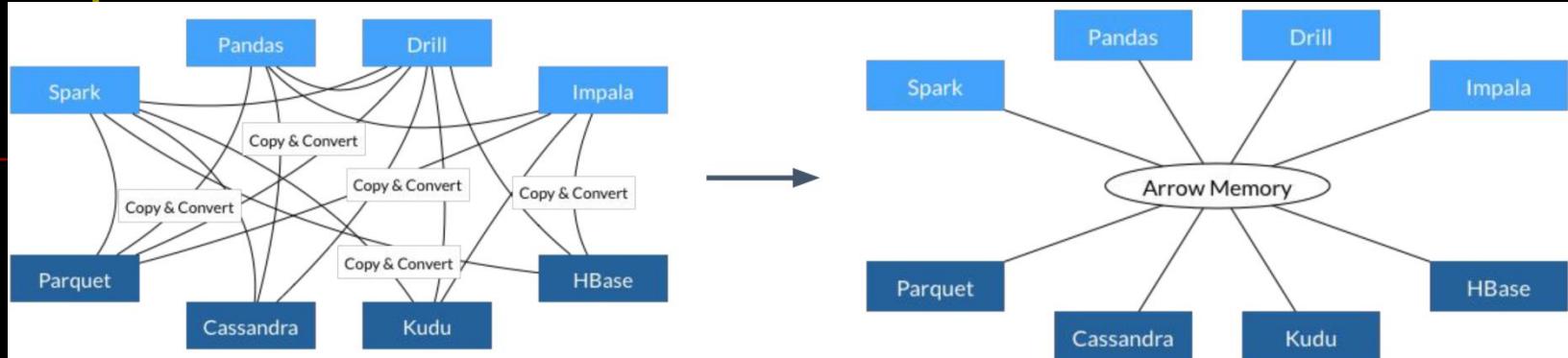
Major components of the project include:

- [The Arrow Columnar In-Memory Format](#): a standard and efficient in-memory representation of various datatypes, plain or nested
- [The Arrow IPC Format](#): an efficient serialization of the Arrow format and associated metadata, for communication between processes and heterogeneous environments
- [The Arrow Flight RPC protocol](#): based on the Arrow IPC format, a building block for remote services exchanging Arrow data with application-defined semantics (for example a storage server or a database)
- [C++ libraries](#)
- [C bindings using GLib](#)
- [C# .NET libraries](#)
- [Gandiva](#): an LLVM-based Arrow expression compiler, part of the C++ codebase
- [Go libraries](#)
- [Java libraries](#)
- [JavaScript libraries](#)
- [Plasma Object Store](#): a shared-memory blob store, part of the C++ codebase
- [Python libraries](#)
- [R libraries](#)
- [Ruby libraries](#)
- [Rust libraries](#)

## Why is it



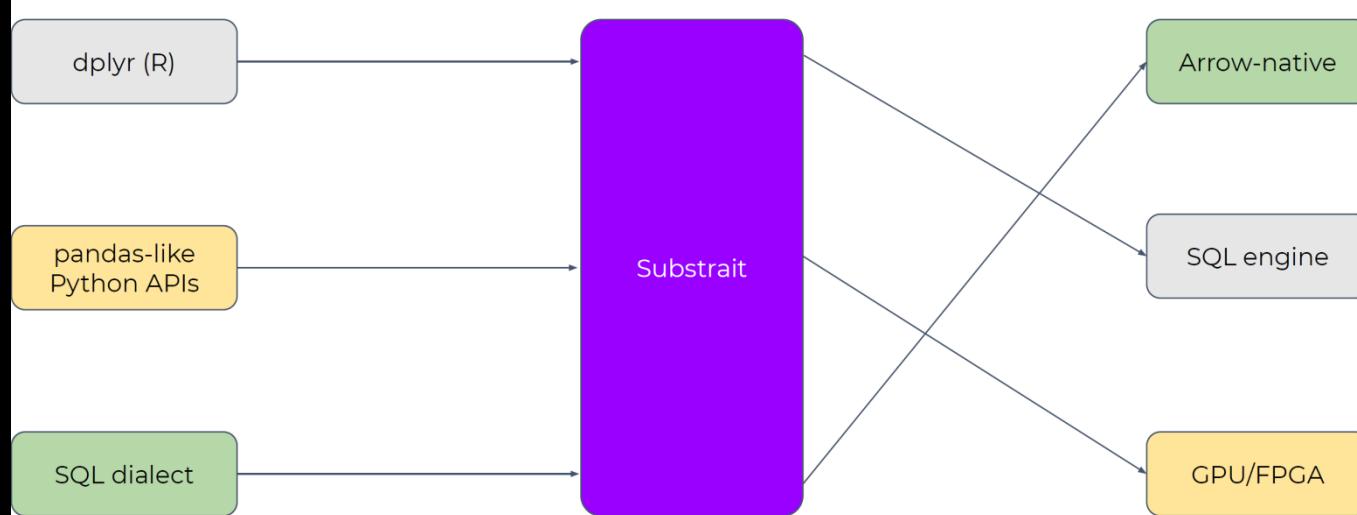
### ■ Simplified data access with the Arrow columnar format



Source: <https://www.slideshare.net/wesm/apache-arrow-high-performance-columnar-data-framework>

### ■ LEGO for data ecosystem

**Freedom** to compose **any toolchain** that speaks Arrow and Substrait



Source: <https://www.slideshare.net/wesm/apache-arrow-high-performance-columnar-data-framework>

## Arrow 6.0.0

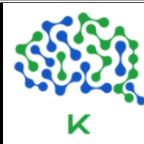
### ■ Simplified data access with the Arrow columnar format

Released on October 26, 2021

- Significant investments in the **C++** and **Rust (DataFusion)** projects providing embeddable parallel query execution
- **Increased C++ kernel coverage**, added:
  - array expressions
  - grouped aggregations
  - relational joins
- Near-complete TPC-H support **available in C++ and R**, with pyarrow support upcoming. Full TPC-H available in Rust/DataFusion.
- **Near zero-copy DuckDB integration**

Source: <https://www.slideshare.net/wesm/apache-arrow-high-performance-columnar-data-framework>

■ ...





## Arrow 7.0.0

- <https://voltrondata.com/news/apache-arrow-7-0-0-what-to-expect/>  
**Database Connectors Take Flight**  
**Aggregating Data in PyArrow**  
**Resources for New Contributors**

---
- **Arrow-native Future is (Almost) Here**
- <https://voltrondata.com/news/apache-arrow-version-7-0-0-released/>
- <https://arrow.apache.org/blog/2022/02/08/7.0.0-release>
- ...

## Good Resources

- <https://github.com/apache/arrow/tree/master/docs>
- <https://arrow.apache.org/blog/>
- <https://arrow.apache.org/faq/>
- <https://arrow.apache.org/cookbook/>
- ...





## 1.2.1 Arrow on Rust

### 1.2.1.1 arrow-rs

#### Overview

- <https://github.com/apache/arrow-rs>

#### Official Rust implementation of Apache Arrow.

This repo contains the following main components:

Crate	Description	Documentation
arrow	Core functionality (memory layout, arrays, low level computations)	(README)
parquet	Support for Parquet columnar file format	(README)
arrow-flight	Support for Arrow-Flight IPC protocol	(README)

There are two related crates in a different repository

Crate	Description	Documentation
DataFusion	In-memory query engine with SQL support	(README)
Ballista	Distributed query execution	(README)

Collectively, these crates support a vast array of functionality for analytic computations in Rust.

For example, you can write an SQL query or a `DataFrame` (using the `datafusion` crate), run it against a parquet file (using the `parquet` crate), evaluate it in-memory using Arrow's columnar format (using the `arrow` crate), and send to another process (using the `arrow-flight` crate).

Generally speaking, the `arrow` crate offers functionality for using Arrow arrays, and `datafusion` offers most operations typically found in SQL, including `joins` and window functions.

You can find more details about each crate in their respective READMEs.

...



## 1.2.1.2 Arrow2

■ <https://github.com/jorgecarleitao/arrow2/>

**Unofficial transmute-free Rust library to work with the Arrow format.**

### ■ Features

- Most feature-complete implementation of Apache Arrow after the reference implementation (C++)
  - Float 16 unsupported (not a Rust native type)
  - Decimal 256 unsupported (not a Rust native type)
- C data interface supported for all Arrow types (read and write)
- C stream interface supported for all Arrow types (read and write)
- Full interoperability with Rust's `vec`
- MutableArray API to work with bitmaps and arrays in-place
- Full support for timestamps with timezones, including arithmetics that take timezones into account
- Support to read from, and write to:
  - CSV
  - Apache Arrow IPC (all types)
  - Apache Arrow Flight (all types)
  - Apache Parquet (except deep nested types)
  - Apache Avro (all types)
  - NJSON
  - ODBC (some types)
- Extensive suite of compute operations
  - aggregations
  - arithmetics
  - cast
  - comparison
  - sort and merge-sort
  - boolean (AND, OR, etc) and boolean kleene
  - filter, take
  - hash
  - if-then-else
  - nullif
  - temporal (day, month, week day, hour, etc.)
  - window
  - ... and more ...
- Extensive set of cargo feature flags to reduce compilation time and binary size
- Fully-decoupled IO between CPU-bounded and IO-bounded tasks, allowing this crate to both be used in `async` contexts without blocking and leverage parallelism
- Fastest known implementation of Avro and Parquet (e.g. faster than the official C++ implementations)

...

## Integration

### ■ <https://github.com/apache/arrow-datafusion/issues/1532>



**Is your feature request related to a problem or challenge? Please describe what you are trying to do.**

Datafusion currently relies on the <https://github.com/apache/arrow-rs> implementation of Apache Arrow. This also means any project that is built on DataFusion is likely to end up using that implementation as well

---

There has been various talk / discussion / work on switching to arrow2 - <https://github.com/jorgecarleitao/arrow2> from [@jorgecarleitao](#)

---

**Describe the solution you'd like**

A consensus on if we want to switch datafusion to using arrow2

**Additional context**

- Arrow2 milestone; <https://github.com/apache/arrow-datafusion/milestone/3>
- PR with more discussion on this issue: #68
- Code from [@houqp](#) and [@yjshen](#) [https://github.com/houqp/arrow-datafusion/tree/arrow2\\_merge](https://github.com/houqp/arrow-datafusion/tree/arrow2_merge)
- [@Igosuki](#) 's PR: <https://github.com/Igosuki/arrow-datafusion/tree/arrow22r>
- Roadmap: <https://github.com/apache/arrow-datafusion/blob/0b8bffd6410ecdca29788f75fbc5ca15242a239/docs/source/specification/roadmap.md#runtime--infrastructure>

...



### 1.2.1.3 DataFusion

- <https://arrow.apache.org/datafusion/>  
**Apache Arrow DataFusion and Ballista query engines.**
- <https://github.com/apache/arrow-datafusion/>

DataFusion is an extensible query execution framework, written in Rust, that uses [Apache Arrow](#) as its in-memory format.

DataFusion supports both an SQL and a DataFrame API for building logical query plans as well as a query optimizer and execution engine capable of parallel execution against partitioned data sources (CSV and Parquet) using threads.

DataFusion also supports distributed query execution via the [Ballista](#) crate.

#### Use Cases

DataFusion is used to create modern, fast and efficient data pipelines, ETL processes, and database systems, which need the performance of Rust and Apache Arrow and want to provide their users the convenience of an SQL interface or a DataFrame API.

#### Why DataFusion?

- *High Performance:* Leveraging Rust and Arrow's memory model, DataFusion achieves very high performance
- *Easy to Connect:* Being part of the Apache Arrow ecosystem (Arrow, Parquet and Flight), DataFusion works well with the rest of the big data ecosystem
- *Easy to Embed:* Allowing extension at almost any point in its design, DataFusion can be tailored for your specific usecase
- *High Quality:* Extensively tested, both by itself and with the rest of the Arrow ecosystem, DataFusion can be used as the foundation for production systems.

- <https://arrow.apache.org/blog/2022/02/28/datafusion-7.0.0/>

...



## ■ Ballista: Distributed Compute with Apache Arrow and DataFusion

<https://github.com/apache/arrow-datafusion/blob/master/ballista/README.md>

Ballista is a distributed compute platform primarily implemented in Rust, and powered by Apache Arrow and DataFusion. It is built on an architecture that allows other programming languages (such as Python, C++, and Java) to be supported as first-class citizens without paying a penalty for serialization costs.

The foundational technologies in Ballista are:

- [Apache Arrow](#) memory model and compute kernels for efficient processing of data.
- [Apache Arrow Flight Protocol](#) for efficient data transfer between processes.
- [Google Protocol Buffers](#) for serializing query plans.
- [Docker](#) for packaging up executors along with user-defined code.

Ballista can be deployed as a standalone cluster and also supports [Kubernetes](#). In either case, the scheduler can be configured to use `etcd` as a backing store to (eventually) provide redundancy in the case of a scheduler failing.

### Distributed Scheduler Overview

Ballista uses the DataFusion query execution framework to create a physical plan and then transforms it into a distributed physical plan by breaking the query down into stages whenever the partitioning scheme changes.

Specifically, any `RepartitionExec` operator is replaced with an `UnresolvedShuffleExec` and the child operator of the repartition operator is wrapped in a `ShuffleWriterExec` operator and scheduled for execution.

Each executor polls the scheduler for the next task to run. Tasks are currently always `ShuffleWriterExec` operators and each task represents one *input* partition that will be executed. The resulting batches are repartitioned according to the shuffle partitioning scheme and each *output* partition is streamed to disk in Arrow IPC format.

The scheduler will replace `UnresolvedShuffleExec` operators with `ShuffleReaderExec` operators once all shuffle tasks have completed. The `ShuffleReaderExec` operator connects to other executors as required using the Flight interface, and streams the shuffle IPC files.

### How does this compare to Apache Spark?

Ballista implements a similar design to Apache Spark, but there are some key differences.

- The choice of Rust as the main execution language means that memory usage is deterministic and avoids the overhead of GC pauses.
- Ballista is designed from the ground up to use columnar data, enabling a number of efficiencies such as vectorized processing (SIMD and GPU) and efficient compression. Although Spark does have some columnar support, it is still largely row-based today.
- The combination of Rust and Arrow provides excellent memory efficiency and memory usage can be 5x - 10x lower than Apache Spark in some cases, which means that more processing can fit on a single node, reducing the overhead of distributed compute.
- The use of Apache Arrow as the memory model and network protocol means that data can be exchanged between executors in any programming language with minimal serialization overhead.



## 1.2.1.4 Polars

### ■ <https://www.pola.rs/>

**Lightning-fast DataFrame library for Rust and Python.**

#### Familiar from the start

Knowing of data wrangling habits, Polars exposes a complete Python API, including the full set of features to manipulate DataFrames using an expression language that will empower you to create readable and performant code.

#### DataFrames to the Rust ecosystem

Polars is written in Rust, uncompromising in its choices to provide a feature-complete DataFrame API to the Rust ecosystem. Use it as a DataFrame library or as query engine backend for your data models.

#### On the shoulders of a giant

Polars is built upon the [safe Arrow2 implementation](#) of the [Apache Arrow specification](#), enabling efficient resource use and processing performance. By doing so it also integrates seamlessly with other tools in the Arrow ecosystem.

### ■ <https://github.com/pola-rs/polars>

**Blazingly fast DataFrames in Rust, Python & Node.js**

Polars is a blazingly fast DataFrame library implemented in Rust using [Apache Arrow Columnar Format](#) as memory model.

- Lazy | eager execution
- Multi-threaded
- SIMD
- Query optimization
- Powerful expression API
- Rust | Python | ...

# Performance

- <https://h2oai.github.io/db-benchmark/>

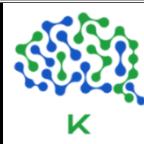
basic questions #			
Input table: 100,000,000 rows x 7 columns ( 5 GB )			
■ Polars	0.8.8	2021-06-30	43s
■ data.table	1.14.1	2021-06-30	92s
■ ClickHouse	21.3.2.5	2021-05-12	159s
■ spark	3.1.2	2021-05-31	332s
■ DataFrames.jl	1.1.1	2021-06-03	349s
■ dplyr	1.0.7	2021-06-20	370s
■ (py)datatable	1.0.0a0	2021-06-30	500s
■ pandas	1.2.5	2021-06-30	628s
■ DuckDB	0.2.7	2021-06-15	630s
■ dask	2021.04.1	2021-05-09	internal error
■ cuDF*	0.19.2	2021-05-31	internal error
■ Arrow	4.0.1	2021-05-31	not yet implemented see README
■ Modin			pending
		■ First time	
		■ Second time	

advanced questions			
Input table: 100,000,000 rows x 9 columns ( 5 GB )			
■ Polars	0.8.8	2021-06-30	57s
■ ClickHouse	21.3.2.5	2021-05-12	69s
■ DataFrames.jl	1.1.1	2021-05-15	116s
■ data.table	1.14.1	2021-06-30	120s
■ DuckDB	0.2.7	2021-06-15	157s
■ (py)datatable	1.0.0a0	2021-06-30	323s
■ pandas	1.2.5	2021-06-30	1081s
■ Arrow	4.0.1	2021-05-31	4273s
■ dplyr	1.0.7	2021-06-20	4378s
■ spark	3.1.2	2021-05-31	not yet implemented
■ dask	2021.04.1	2021-05-09	internal error
■ cuDF*	0.19.2	2021-05-31	out of memory
■ Modin		see README	pending

basic questions			
Input table: 1,000,000,000 rows x 9 columns ( 50 GB )			
■ Polars	0.8.8	2021-06-30	143s
■ data.table	1.14.1	2021-06-30	155s
■ DataFrames.jl	1.1.1	2021-05-15	200s
■ ClickHouse	21.3.2.5	2021-05-12	256s
■ cuDF*	0.19.2	2021-05-31	492s
■ spark	3.1.2	2021-05-31	568s
■ (py)datatable	1.0.0a0	2021-06-30	730s
■ dplyr	1.0.7	2021-06-20	internal error
■ pandas	1.2.5	2021-06-30	out of memory
■ dask	2021.04.1	2021-05-09	out of memory
■ Arrow	4.0.1	2021-05-31	internal error
■ DuckDB*	0.2.7	2021-06-15	out of memory
■ Modin		see README	pending



## Good Resources

- <https://pola-rs.github.io/polars-book/user-guide/index.html>
  - ...
- 





## 1.2.2 Arrow for JVM

### Challenge: Limits of JVM-Based Engines

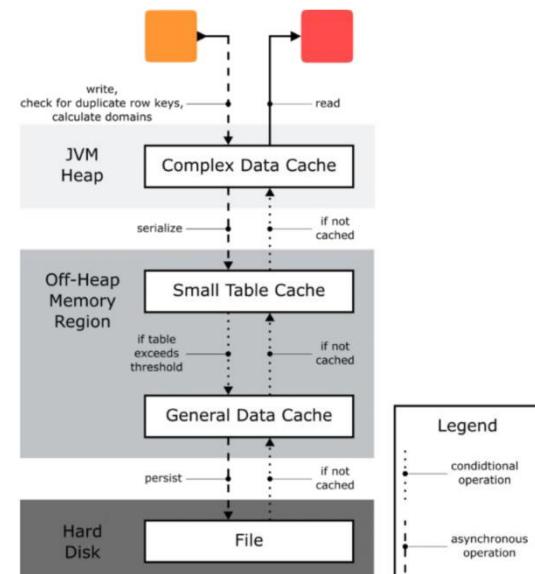
- Computing engines implemented in JVM languages including Java and Scala often suffer from **performance bottlenecks**.
- Arrow columnar data structures can **accelerate JVM-based engines** and enable use of **pluggable high-performance components** implemented in lower-level languages like C++.

Source: <https://www.slideshare.net/wesm/apache-arrow-open-source-standard-becomes-an-enterprise-necessity>

### Case Study: KNIME

#### KNIME uses Arrow in its Columnar Table Backend

- Stores data more compactly in memory, improving performance
- Eliminates the need to create Java objects to represent table elements, reducing GC pressure
- Enables use of shared memory



Source: <https://www.knime.com/blog/improved-performance-with-new-table-backend>

Source: <https://www.slideshare.net/wesm/apache-arrow-open-source-standard-becomes-an-enterprise-necessity>



## 2) Project Ray

### 2.1 Overview

- <https://www.ray.io/>

An open source project that makes it ridiculously simple to **scale** any compute-intensive **Python** workload — from deep learning to production model serving. With a rich set of libraries and integrations built on a flexible **distributed execution framework**, Ray makes distributed computing easy and accessible to every engineer.

- <https://github.com/ray-project/ray/>

An open source framework that provides a simple, universal API for building distributed applications.

- <https://www.anyscale.com/>

Easily develop and deploy distributed Python and machine learning applications, at any scale.

- <https://rise.cs.berkeley.edu/projects/ray/>

- ...



## Good Resources

- <https://docs.ray.io/en/master/index.html>
- <https://docs.ray.io/en/master/>
- <https://github.com/ray-project/tutorial>
- <https://medium.com/distributed-computing-with-ray>
- <https://www.anyscale.com/blog>
- <https://www.anyscale.com/ray-summit-2021>
- <https://github.com/ray-project/ray>

### More Information

- Documentation
- Tutorial
- Blog
- Ray 1.0 Architecture whitepaper (new)
- Ray Design Patterns (new)
- RLlib paper
- RLlib flow paper
- Tune paper

#### *Older documents:*

- Ray paper
- Ray HotOS paper

...

## Coming soon

- Please look forward to a new talk "Ray as a universal infrastructure for distributed computing"...
- 



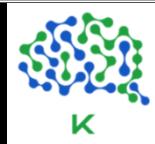
# IX. Blockchain

## 1) Overview

### 1.1

---

- ...



## 2) eBPF in Blockchain

### 2.1 Overview

#### 2.1.1

- ...
- ...





## Good Resources

- <https://stackoverflow.com/questions/65904948/why-is-having-an-userspace-version-of-ebpf-interesting>
  - ...
-



## 2.2 Solana rBPF

- <https://github.com/solana-labs/rbpf>

Rust virtual machine and JIT compiler for eBPF programs(a derivative of rBPF, but with more features).





## Solana

- [https://en.wikipedia.org/wiki/Solana\\_\(blockchain\\_platform\)](https://en.wikipedia.org/wiki/Solana_(blockchain_platform))
- <https://solana.com>  
**A decentralized blockchain built to enable scalable, user-friendly apps for the world.**
- <https://solana.com/solana-whitepaper.pdf>
- <https://github.com/solana-labs/solana>  
**Web-Scale Blockchain for fast, secure, scalable, decentralized apps and marketplaces.**



## eBPF

- <https://docs.solana.com/developing/on-chain-programs/overview>

### Berkeley Packet Filter (BPF)

Solana on-chain programs are compiled via the [LLVM compiler infrastructure](#) to an [Executable and Linkable Format \(ELF\)](#) containing a variation of the [Berkeley Packet Filter \(BPF\)](#) bytecode.

Because Solana uses the LLVM compiler infrastructure, a program may be written in any programming language that can target the LLVM's BPF backend. Solana currently supports writing programs in Rust and C/C++.

BPF provides an efficient [instruction set](#) that can be executed in an interpreted virtual machine or as efficient just-in-time compiled native instructions.

### Memory map

The virtual address memory map used by Solana BPF programs is fixed and laid out as follows

- Program code starts at 0x1000000000
- Stack data starts at 0x2000000000
- Heap data starts at 0x3000000000
- Program input parameters start at 0x4000000000

The above virtual addresses are start addresses but programs are given access to a subset of the memory map. The program will panic if it attempts to read or write to a virtual address that it was not granted access to, and an `AccessViolation` error will be returned that contains the address and size of the attempted violation.

# Stack

BPF uses stack frames instead of a variable stack pointer. Each stack frame is 4KB in size.

If a program violates that stack frame size, the compiler will report the overrun as a warning.

For example:

```
Error: Function _ZN16curve25519_dalek7edwards21EdwardsBasepointTable6create17h178b3d2411f7f082E  
Stack offset of -30728 exceeded max offset of -4096 by 26632 bytes, please minimize large stack variables
```

The message identifies which symbol is exceeding its stack frame but the name might be mangled if it is a Rust or C++ symbol. To demangle a Rust symbol use [rustfilt](#). The above warning came from a Rust program, so the demangled symbol name is:

```
$ rustfilt _ZN16curve25519_dalek7edwards21EdwardsBasepointTable6create17h178b3d2411f7f082E  
curve25519_dalek::edwards::EdwardsBasepointTable::create
```

To demangle a C++ symbol use [c++filt](#) from binutils.

The reason a warning is reported rather than an error is because some dependent crates may include functionality that violates the stack frame restrictions even if the program doesn't use that functionality. If the program violates the stack size at runtime, an [AccessViolation](#) error will be reported.

BPF stack frames occupy a virtual address range starting at 0x200000000.

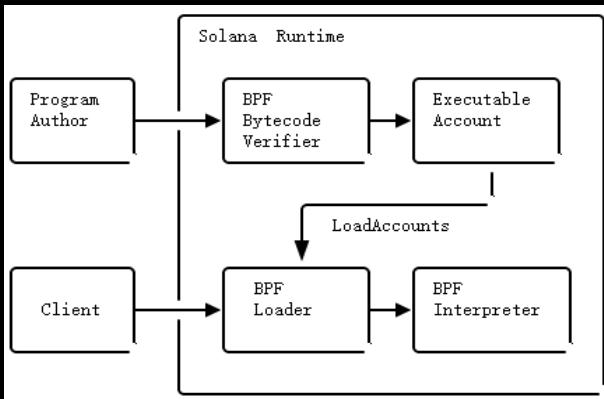
...

## Deployment

BPF program deployment is the process of uploading a BPF shared object into a program account's data and marking the account executable. A client breaks the BPF shared object into smaller pieces and sends them as the instruction data of `Write` instructions to the loader where loader writes that data into the program's account data. Once all the pieces are received the client sends a `Finalize` instruction to the loader, the loader then validates that the BPF data is valid and marks the program account as *executable*. Once the program account is marked executable, subsequent transactions may issue instructions for that program to process.

When an instruction is directed at an executable BPF program the loader configures the program's execution environment, serializes the program's input parameters, calls the program's entrypoint, and reports any errors encountered.

<https://docs.solana.com/developing/on-chain-programs/deploying>



As shown in the diagram above, a program author creates a program, compiles it to an ELF shared object containing BPF bytecode, and uploads it to the Solana cluster with a special *deploy* transaction. The cluster makes it available to clients via a *program ID*. The program ID is an *address* specified when deploying and is used to reference the program in subsequent transactions.

Upon a successful deployment the account that holds the program is marked executable. If the program is marked "final", its account data become permanently immutable. If any changes are required to the finalized program (features, patches, etc...) the new program must be deployed to a new program ID.

If a program is upgradeable, the account that holds the program is marked executable, but it is possible to redeploy a new shared object to the same program ID, provided that the program's upgrade authority signs the transaction.



## Good Resources

- <https://medium.com/coinmonks/solana-internals-part-4-the-bank-a-key-component-1d47b94cd705>
  - ...
-

# X. Security



## 1) Overview

### 1.1 Hardware-assisted Security

---

- ...



## 1.1.1 TEE

- [https://en.wikipedia.org/wiki/Trusted\\_execution\\_environment](https://en.wikipedia.org/wiki/Trusted_execution_environment)

A **trusted execution environment** (TEE) is a secure area of a [main processor](#). It guarantees code and data loaded inside to be protected with respect to [confidentiality and integrity](#)[clarification needed].<sup>[1]</sup> A TEE as an isolated execution environment provides security features such as isolated execution, integrity of applications executing with the TEE, along with confidentiality of their assets.<sup>[2]</sup> In general terms, the TEE offers an execution space that provides a higher level of security for trusted applications running on the device than a rich operating system (OS) and more functionality than a 'secure element' (SE).

### History [edit]

The [Open Mobile Terminal Platform](#) (OMTP) first defined TEE in their "Advanced Trusted Environment:OMTP TR1" standard, defining it as a "set of hardware and software components providing facilities necessary to support Applications" which had to meet the requirements of one of two defined security levels. The first security level, Profile 1, was targeted against only software attacks and while Profile 2, was targeted against both software and hardware attacks.<sup>[3]</sup>

Commercial TEE solutions based on ARM [TrustZone](#) technology, conforming to the TR1 standard, were later launched, such as Trusted Foundations developed by Trusted Logic.<sup>[4]</sup>

Work on the OMTP standards ended in mid 2010 when the group transitioned into the [Wholesale Applications Community](#) (WAC).<sup>[5]</sup>

The OMTP standards, including those defining a TEE, are hosted by [GSMA](#).<sup>[6]</sup>

### Details [edit]

The TEE typically consists of a hardware isolation mechanism, plus a secure operating system running on top of that isolation mechanism – however the term has been used more generally to mean a protected solution.<sup>[7][8][9]</sup> Whilst a GlobalPlatform TEE requires hardware isolation, others such as EMVCo use the term TEE to refer to both hardware/software and only software-based solutions.<sup>[10]</sup> FIDO uses the concept of TEE in the restricted operating environment for TEEs based on hardware isolation.<sup>[11]</sup> Only trusted applications running in a TEE have access to the full power of a device's main processor, peripherals and memory, while hardware isolation protects these from user installed apps running in a main operating system. Software and cryptographic isolation inside the TEE protect the trusted applications contained within from each other.<sup>[12]</sup>

Service providers, [mobile network operators](#) (MNO), operating system developers, [application developers](#), device manufacturers, platform providers and silicon vendors are the main stakeholders contributing to the standardization efforts around the TEE.

To prevent simulation of hardware with user-controlled software, a so-called "hardware root of trust" is used. This is a set of private keys that are embedded directly into the chip during manufacturing; one-time programmable memory such as eFuses are usually used on mobile devices. These cannot be changed, even after device resets, and whose public counterparts reside in a manufacturer database, together with a non-secret hash of a public key belonging to the trusted party (usually a chip vendor) which is used to sign trusted firmware alongside the circuits doing cryptographic operations and controlling access. The hardware is designed in a way which prevents all software not signed by the trusted party's key from accessing the privileged features. The public key of the vendor is provided at runtime and hashed; this hash is then compared to the one embedded in the chip. If the hash matches, the public key is used to verify a [digital signature](#) of trusted vendor-controlled firmware (such as a chain of bootloaders on Android devices or 'architectural enclaves' in SGX). The trusted firmware is then used to implement remote attestation.<sup>[13]</sup>

When an application is attested, its untrusted component loads its trusted component into memory; the trusted application is protected from modification by untrusted components with hardware. A [nonce](#) is requested by the untrusted party from verifier's server, and is used as a part of a cryptographic authentication protocol, proving integrity of the trusted application. The proof is passed to the verifier, which verifies it. A valid proof cannot be computed in a simulated hardware (i.e. QEMU) because in order to construct it, access to the keys baked into hardware is required; only trusted firmware has access to these keys and/or the keys derived from them or obtained using them. Because only the platform owner is meant to have access to the data recorded in the foundry, the verifying party must interact with the service set up by the vendor. If the scheme is implemented improperly, the chip vendor can track which applications are used on which chip and selectively deny service by returning a message indicating that authentication has not passed.[citation needed]

To simulate hardware in a way which enables it to illicitly pass remote authentication, an attacker would have to extract keys from the hardware, which is costly because of the equipment and technical skill required to execute it. For example, using [focused ion beams](#), [scanning electron microscopes](#), [microprobing](#), and [chip decapsulation](#)<sup>[14][15][16][17][18][19]</sup> or even impossible, if the hardware is designed in such a way that reverse-engineering destroys the keys. In most cases, the keys are unique for each piece of hardware, so that a key extracted from one chip cannot be used by others (for example [physically unclonable functions](#)<sup>[20][21]</sup>).

Though deprivation of ownership is not an inherent property of TEEs (it is possible to design the system in a way that allows only the user who has obtained ownership of the device first to control the system), in practice all such systems in consumer electronics are intentionally designed so as to allow chip manufacturers to control access to attestation and its algorithms. It allows manufacturers to grant access to TEEs only to software developers who have a (usually commercial) business agreement with the manufacturer, and to enable such use cases as [tivoization](#) and DRM.

## SGX

- [https://en.wikipedia.org/wiki/Software\\_Guard\\_Extensions](https://en.wikipedia.org/wiki/Software_Guard_Extensions)
- <https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html>
- ...



## 1.1.1.1 GlobalPlatform

- <https://globalplatform.org/technical-committees/trusted-execution-environment-tee-committee>

### TEE Committee (formerly Device Committee)

The Trusted Execution Environment (TEE) Committee is chaired by Richard Hayton from Trustonic. Full and participating GlobalPlatform members are eligible to contribute to this group.

The TEE Committee defines an open security architecture for consumer and connected devices using a TEE to secure those devices and enable the development and deployment of secure services from multiple service providers.

#### Objectives

The TEE Committee works to:

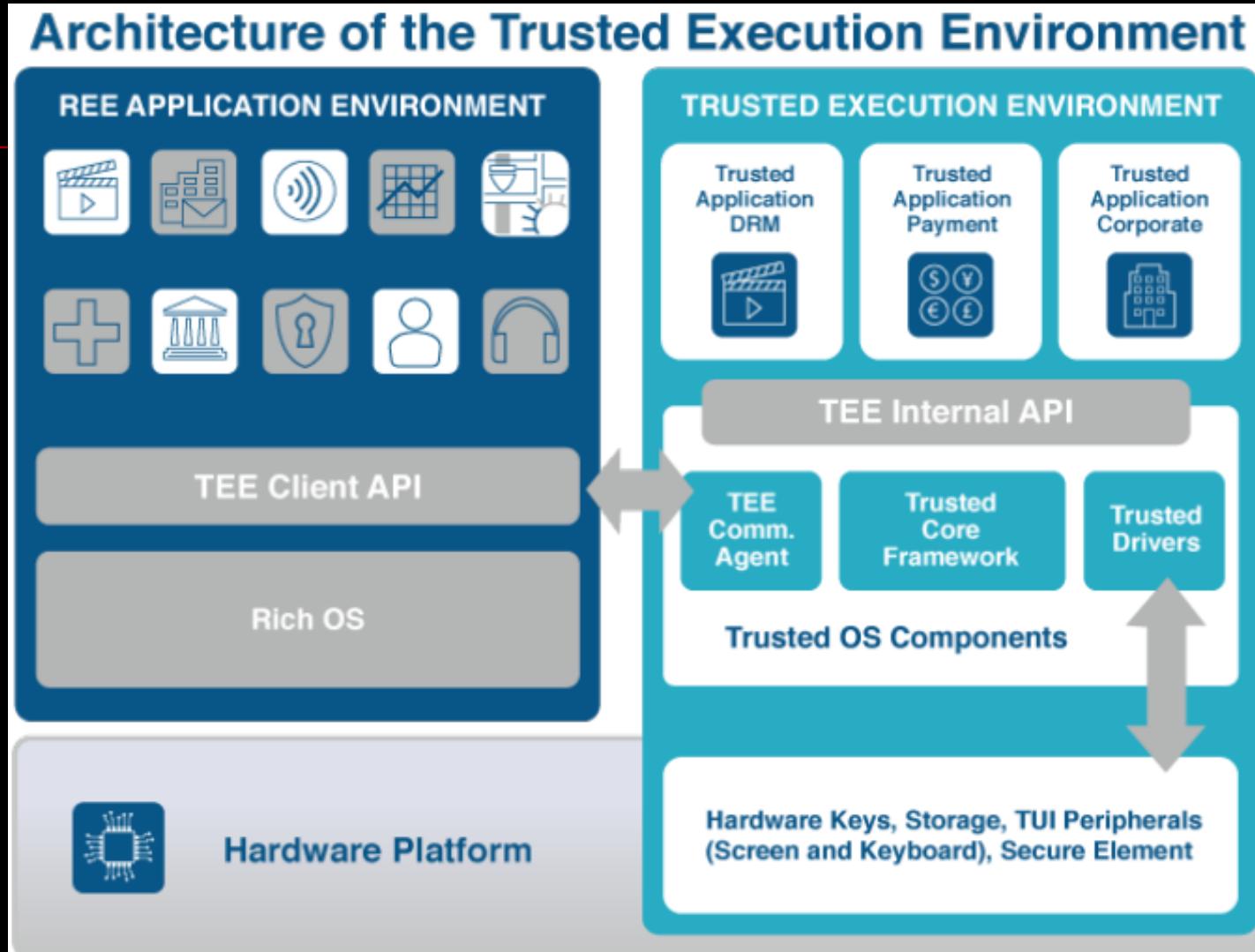
- Manage, prioritize, develop, maintain and evolve specifications for the TEE, including specifications relating to:
  - APIs to communicate to a TEE.
  - APIs to develop Trusted Applications (TAs) running within the TEE and enabling interactions with secure peripherals such as the trusted user interface, biometric peripherals and **Secure Elements (SEs)**.
  - The **TEE Management Framework (TMF)**.
  - Configurations to serve a specific class of devices.
- Advance and maintain the GlobalPlatform TEE **Functional** and **Security** Certification Programs to facilitate portability and interoperability of TA deployments on different TEE implementations, and to enable standardized security evaluations.
- Liaise, collaborate and/or coordinate activities with relevant external organizations which perform similar/complementary activities.

#### Current Priorities

- Update TEE Protection Profile to meet security requirements of different markets; produce Biometry, Secure Media Path and TUI modules.
- Development of OTrP configurations for the TEE Management Framework (TMF)
- Evolution of TEE Internal Core APIs to add new cryptographic algorithms and integrate new requirements.
- Development of TA Portability Build System.

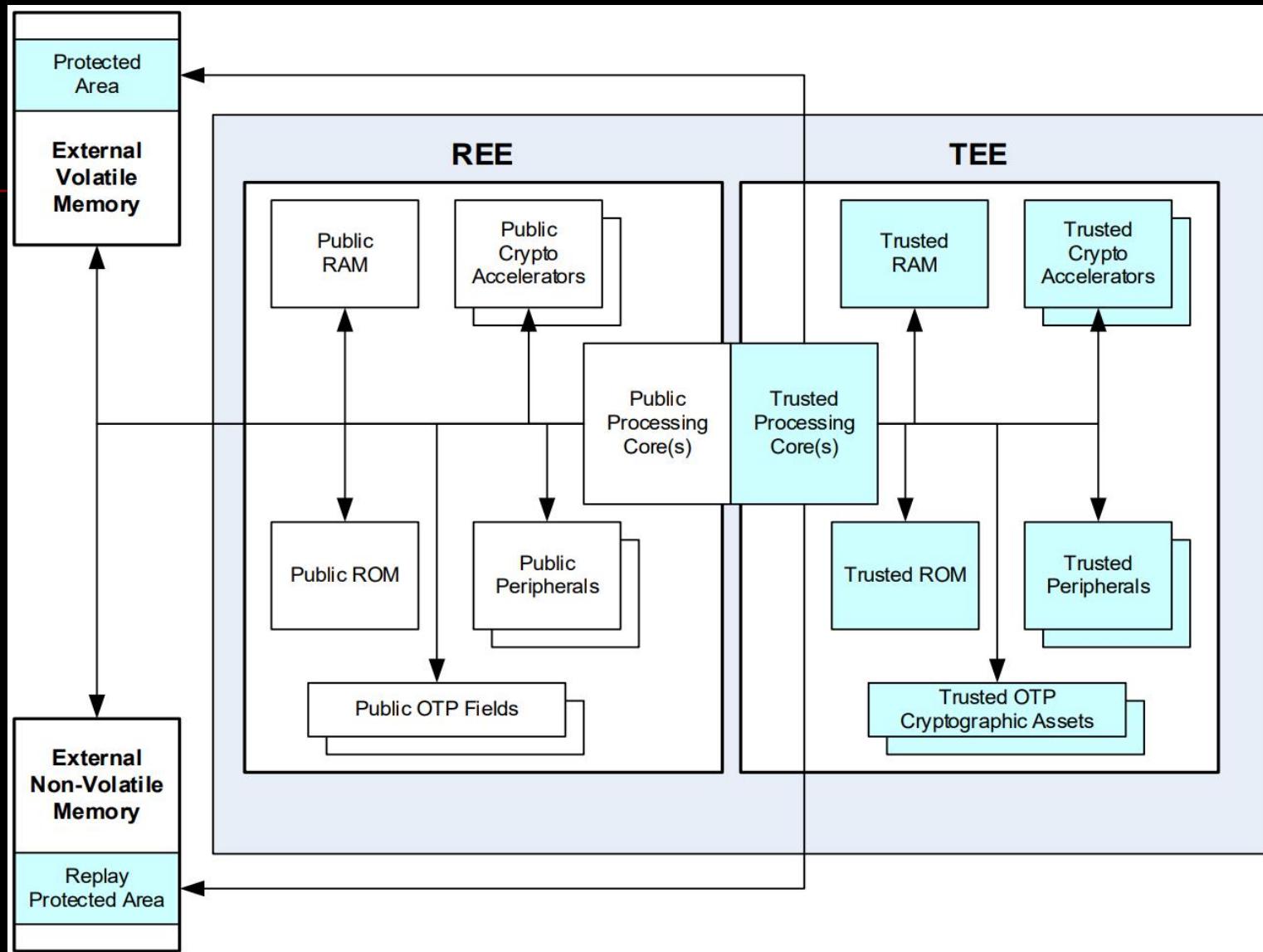
## Architecture & Design

### ■ Overview



Source: <https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf>

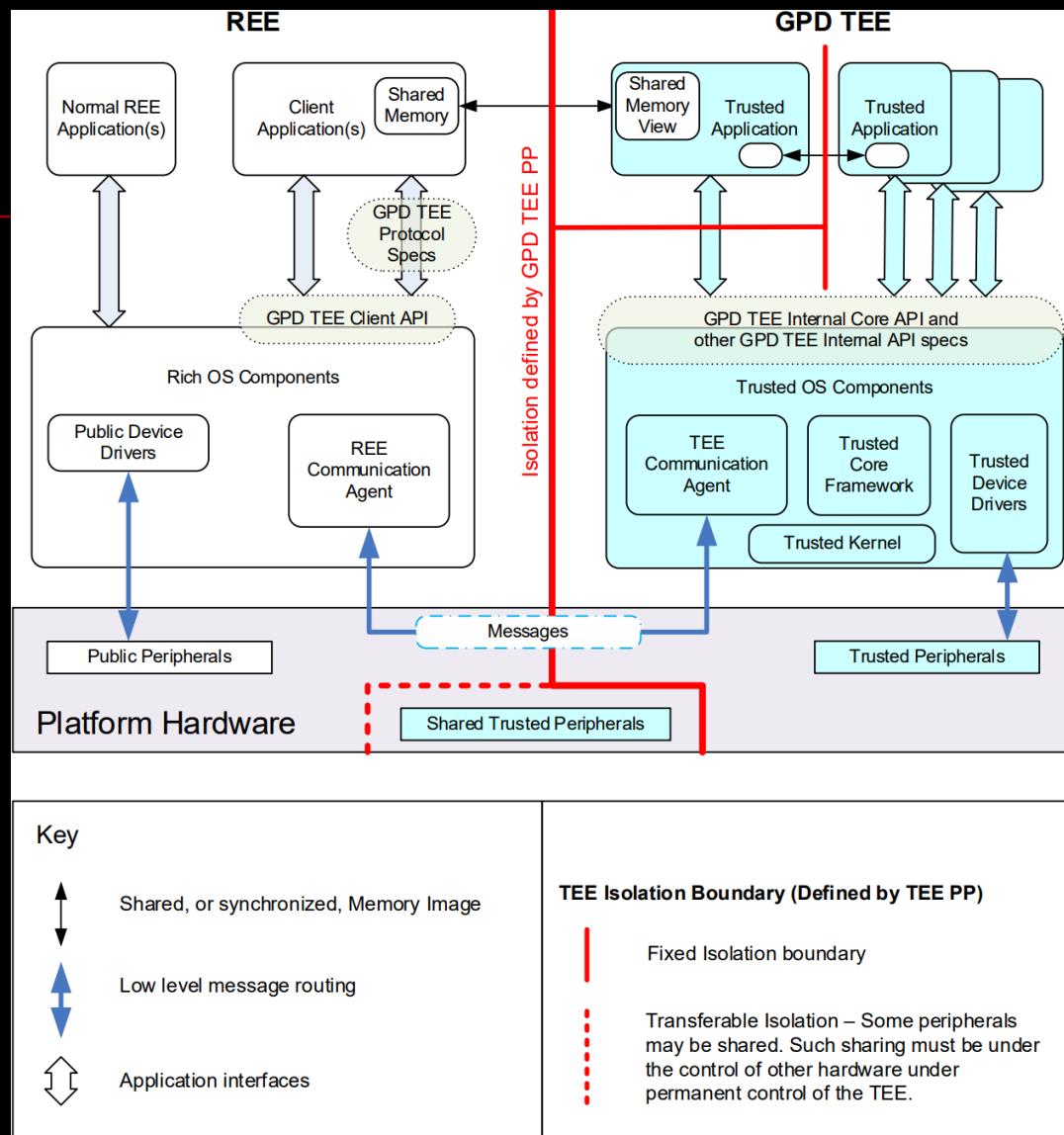
## ■ Hardware Architectural View of REE and TEE



Source: [https://globalplatform.org/wp-content/uploads/2017/01/GPD\\_TEE\\_SystemArch\\_v1.2\\_PublicRelease.pdf](https://globalplatform.org/wp-content/uploads/2017/01/GPD_TEE_SystemArch_v1.2_PublicRelease.pdf)



## ■ Software Architecture



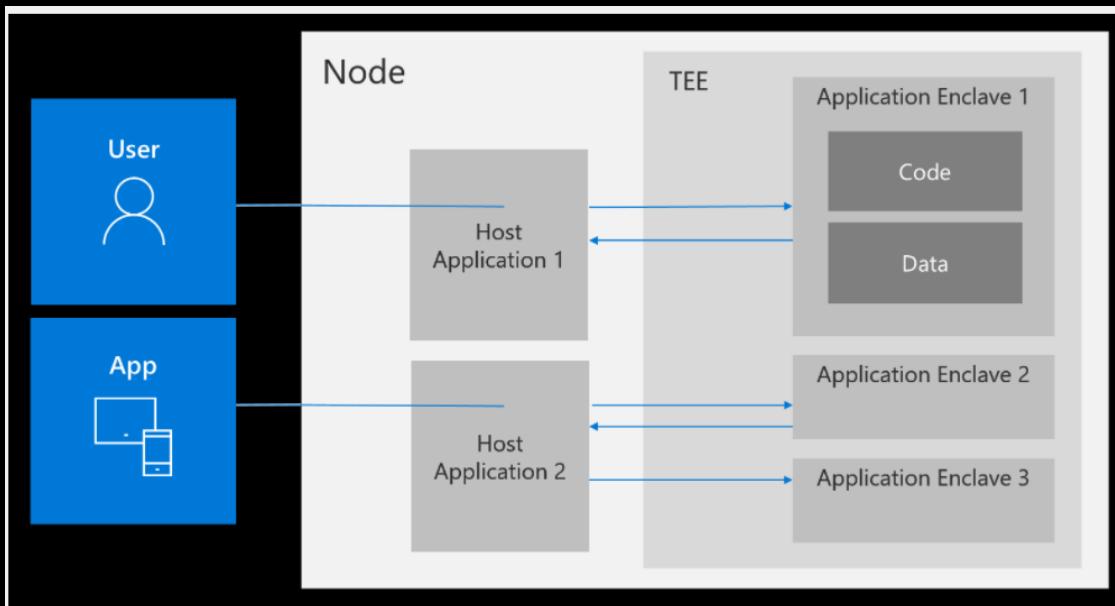
Source: [https://globalplatform.org/wp-content/uploads/2017/01/GPD\\_TEE\\_SystemArch\\_v1.2\\_PublicRelease.pdf](https://globalplatform.org/wp-content/uploads/2017/01/GPD_TEE_SystemArch_v1.2_PublicRelease.pdf)



## 1.1.1.2 Open Enclave SDK

- <https://openenclave.io/sdk/>

**Build Trusted Execution Environment based applications to help protect data in use with an open source SDK that provides consistent API surface across enclave technologies as well as all platforms from cloud to edge.**



An enclave application partitions itself into two components (1) an untrusted component (called the host) and (2) a trusted component (called the enclave). The host component runs unmodified on the untrusted operating system, while the trusted component runs within the enclave, the protected container provided by a TEE implementation. These protections allow enclaves to perform secure computations with assurances that secrets will not be compromised.



- **<https://github.com/openenclave/openenclave>**

An *enclave* is a protected memory region that provides confidentiality for data and code execution. It is an instance of a Trusted Execution Environment (TEE) which is usually secured by hardware, for example, [Intel Software Guard Extensions \(SGX\)](#).

---

This SDK aims to generalize the development of enclave applications across TEEs from different hardware vendors. The current implementation provides support for Intel SGX as well as preview support for OP-TEE OS on ARM TrustZone. As an open source project, this SDK also strives to provide a transparent solution that is agnostic to specific vendors, service providers and choice of operating systems.

- **<https://github.com/openenclave/openenclave/tree/master/docs/GettingStartedDocs>**



### 1.1.1.3 Tealize

- <https://tealize.apache.org/>

An open source universal secure computing platform, making computation on privacy-sensitive data safe and simple.

- **Features**



#### SECURE AND ATTESTABLE

Tealize adopts multiple security technologies to enable secure computing, in particular, Tealize uses Intel SGX to serve the most security-sensitive tasks with hardware-based isolation, memory encryption and attestation. Also, Tealize is written in Rust to prevent memory-safety issues.



#### FUNCTION-AS-A-SERVICE

Tealize is a function-as-a-service platform supporting tasks like privacy-preserving machine learning, private set intersection, and crypto computation. In addition, developers can also write and execute Python function. Tealize supports both general secure computing tasks and flexible multi-party secure computation.



#### EASE OF USE

Tealize builds its components in containers, therefore, it supports deployment both locally and within cloud infrastructures. Tealize also provides convenient endpoint APIs, client SDKs and command line tools.

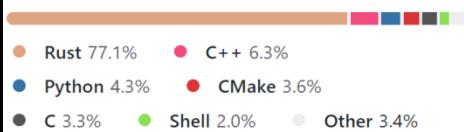


#### FLEXIBLE

Components in Tealize are designed in modular, and features like remote attestation can be easily embedded in other projects. In addition, Tealize SGX SDK and Tealize TrustZone SDK can also be used separately to write TEE apps for other purposes.

- <https://github.com/apache/incubator-tealize>

##### Languages



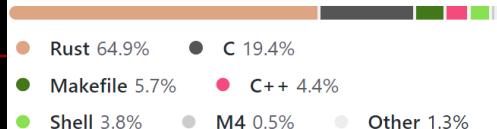
- <https://tealize.apache.org/docs/my-first-function/>



## ■ <https://github.com/apache/incubator-tealake-sgx-sdk>

Apache Tealake (incubating) SGX SDK helps developers to write Intel SGX applications in the Rust programming language, and also known as Rust SGX SDK.

### Languages



## ■ <https://github.com/apache/incubator-tealake-trustzone-sdk>

Tealake TrustZone SDK (Rust OP-TEE TrustZone SDK) provides abilities to build safe TrustZone applications in Rust. The SDK is based on the [OP-TEE](#) project which follows [GlobalPlatform](#) TEE specifications and provides ergonomic APIs. In addition, it enables capability to write TrustZone applications with Rust's standard library and many third-party libraries (i.e., crates). Tealake TrustZone SDK is a sub-project of [Apache Tealake \(incubating\)](#).

### Languages



## ■ <https://github.com/apache/incubator-tealake/blob/master/docs/executing-wasm.md>

## ■ <https://github.com/apache/incubator-tealake/blob/master/docs/inference-with-tvm.md>

## ■ <https://github.com/apache/incubator-tealake/blob/master/docs/service-internals.md>

■ ...



## 1.1.2 ARM

- [https://en.wikipedia.org/wiki/ARM\\_architecture#Security\\_extensions](https://en.wikipedia.org/wiki/ARM_architecture#Security_extensions)

### 1.1.2.1 Trustzone

- <https://developer.arm.com/ip-products/security-ip/trustzone>
- ~~<https://developer.arm.com/ip-products/security-ip/trustzone/trustzone-system-ip>~~
- <https://globalplatform.org/specs-library/?filter-committee=tee>

### Cortex-A

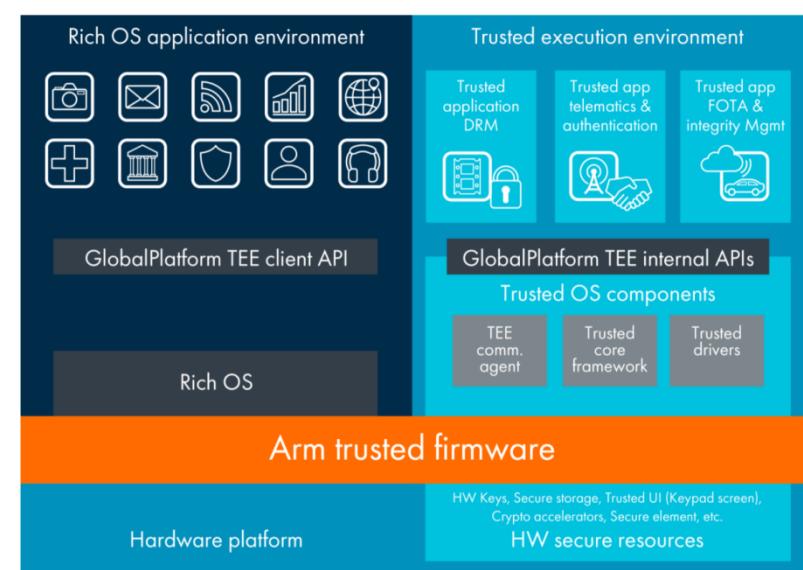
- <https://developer.arm.com/ip-products/security-ip/trustzone/trustzone-for-cortex-a>

Arm TrustZone is used on billions of applications' processors to protect high-value code and data. It is frequently used to provide a security boundary for a GlobalPlatform Trusted Execution Environment.

TrustZone is built on Secure and Non-secure worlds that are hardware separated. The partitioning of the two worlds is achieved by hardware logic present in the AMBA bus fabric, peripherals and processors.

In order to implement a Secure state in the SoC, trusted software (Trusted OS) needs to be developed to make use of the protected assets. This code typically implements trusted boot, the Secure world switch monitor, a small trusted OS and trusted apps. The combination of TrustZone based hardware isolation, trusted boot and a trusted OS make up a Trusted Execution Environment (TEE), which can be used alongside other security technology.

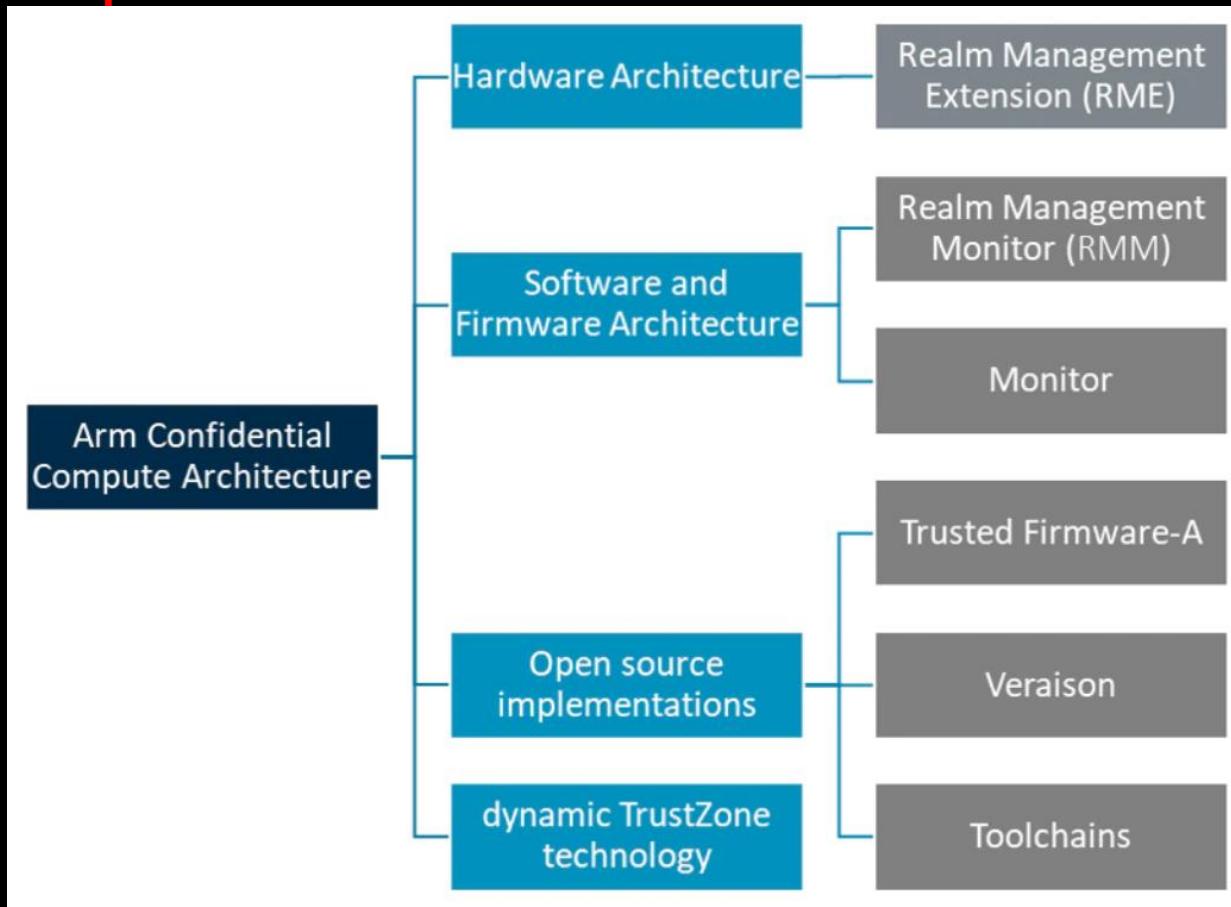
Learn more about the [GlobalPlatform TEE](#).



## 1.1.2.2 CCA(Confidential Compute Architecture)



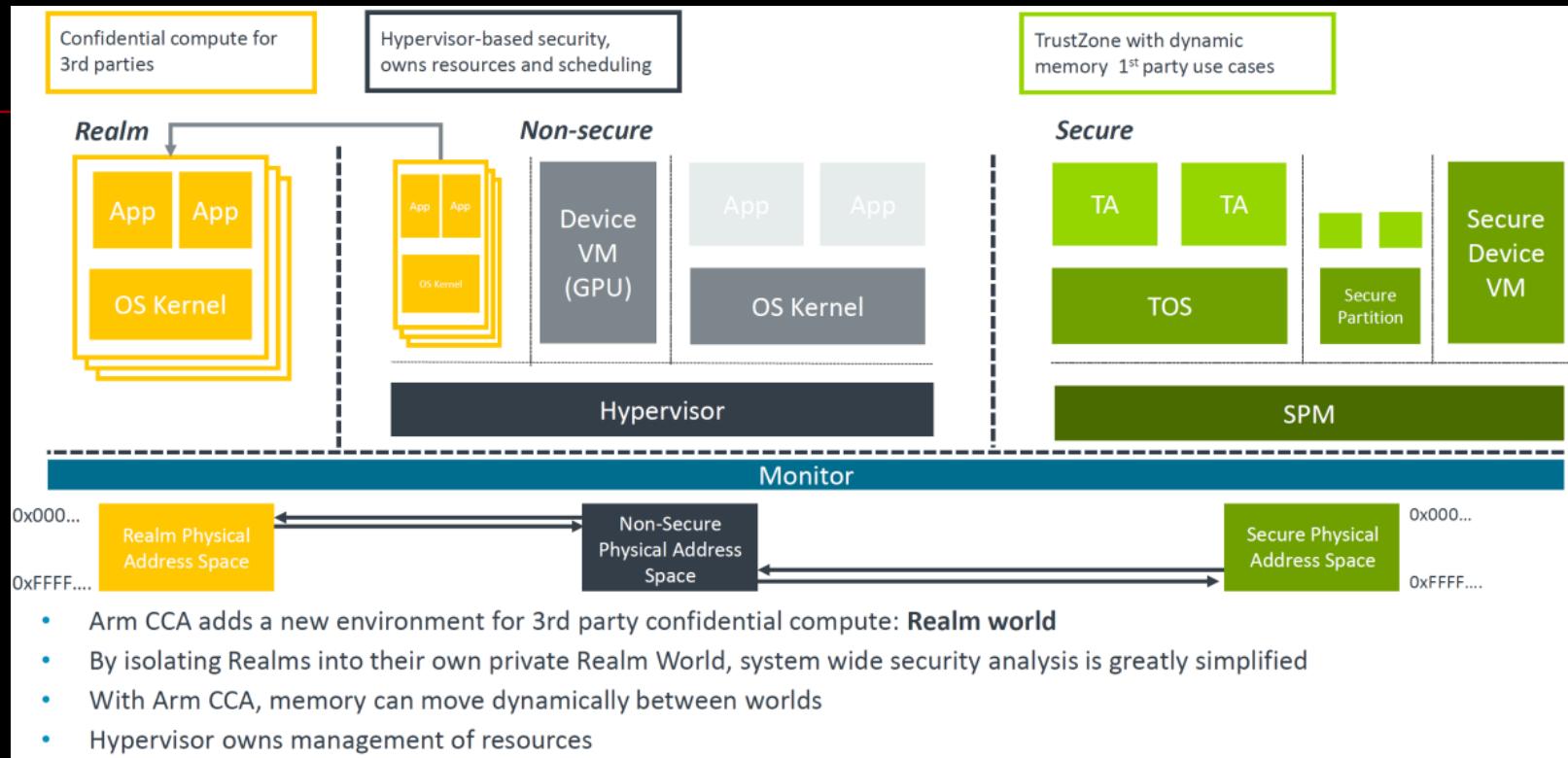
- Since ARM v9
  - <https://developer.arm.com/architectures/architecture-security-features/confidential-computing>
- ### Components





## ■ Architecture

<https://www.arm.com/ja/why-arm/architecture/security-features/arm-confidential-compute-architecture>





## ■ Benefits

Fully secures third-party data and code for its owner so that it is not accessible by platform owners.

Realms can work alongside TrustZone, ensuring minimal impact on existing trusted applications.

Applies to any market or form factor that uses microprocessors.

Democratizes secure computing for all developers, and increases scalability, not just those working closely with silicon vendors and device OEMs.

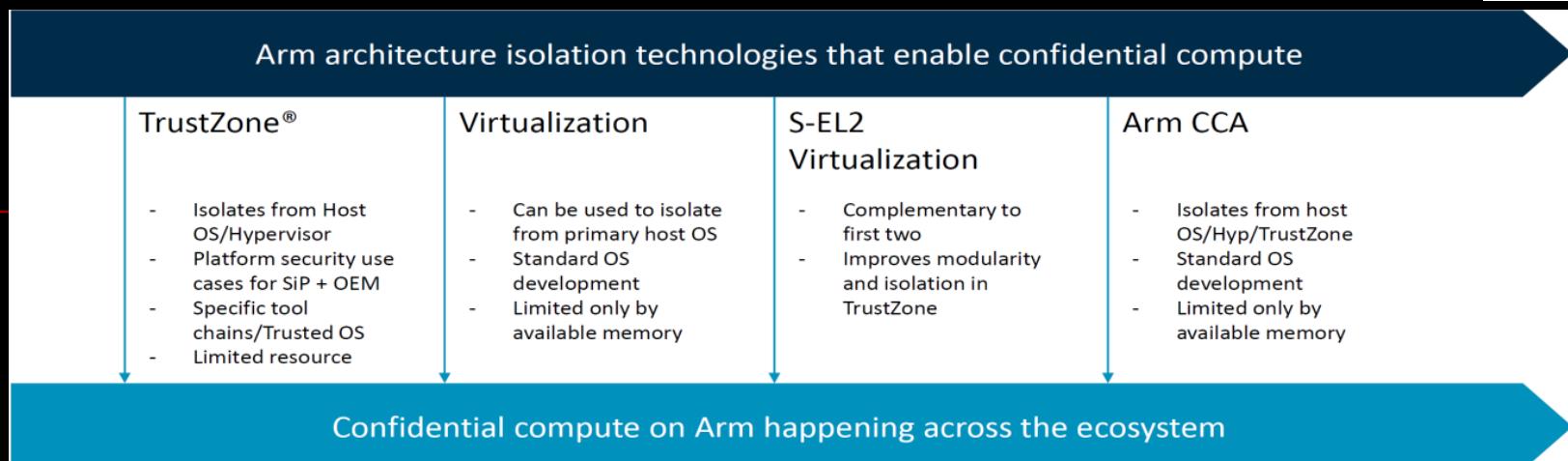
Realms can be used at the virtual machine level for seamless portability across the ecosystem of Arm devices.

Native support for attestation enables Realm owners to verify and prove the integrity of the underlying platform and Realm configuration.

Source: <https://www.arm.com/why-arm/architecture/security-features/arm-confidential-compute-architecture>



### 1.1.2.3 Evolution and Comparison



Source: <https://connect.linaro.org/resources/armcca/introduction-to-the-arm-confidential-compute-architecture/>



## 1.1.3 RISC-V

- <https://>
-



# 1.2 Provably Secure Systems

## 1.2.1 Formal verification

- [https://en.wikipedia.org/wiki/Formal\\_verification](https://en.wikipedia.org/wiki/Formal_verification)

In the context of hardware and software systems, **formal verification** is the act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain **formal specification** or property, using **formal methods** of mathematics.<sup>[1]</sup>

Formal verification can be helpful in proving the correctness of systems such as: cryptographic protocols, combinational circuits, digital circuits with internal memory, and software expressed as source code.

The verification of these systems is done by providing a **formal proof** on an abstract mathematical model of the system, the correspondence between the mathematical model and the nature of the system being otherwise known by construction. Examples of mathematical objects often used to model systems are: finite state machines, labelled transition systems, Petri nets, vector addition systems, timed automata, hybrid automata, process algebra, formal semantics of programming languages such as operational semantics, denotational semantics, axiomatic semantics and Hoare logic.<sup>[2]</sup>

### Approaches [edit]

One approach and formation is **model checking**, which consists of a systematically exhaustive exploration of the mathematical model (this is possible for **finite models**, but also for some infinite models where infinite sets of states can be effectively represented finitely by using abstraction or taking advantage of symmetry). Usually this consists of exploring all states and transitions in the model, by using smart and domain-specific abstraction techniques to consider whole groups of states in a single operation and reduce computing time. Implementation techniques include **state space enumeration**, symbolic state space enumeration, **abstract interpretation**, **symbolic simulation**, abstraction refinement.<sup>[citation needed]</sup> The properties to be verified are often described in **temporal logics**, such as **linear temporal logic (LTL)**, **Property Specification Language (PSL)**, **SystemVerilog Assertions (SVA)**,<sup>[3]</sup> or **computational tree logic (CTL)**. The great advantage of model checking is that it is often fully automatic; its primary disadvantage is that it does not in general scale to large systems; symbolic models are typically limited to a few hundred bits of state, while explicit state enumeration requires the state space being explored to be relatively small.

Another approach is deductive verification. It consists of generating from the system and its specifications (and possibly other annotations) a collection of mathematical **proof obligations**, the truth of which imply conformance of the system to its specification, and discharging these obligations using either proof assistants (interactive theorem provers) (such as **HOL**, **ACL2**, **Isabelle**, **Coq** or **PVS**), or automatic theorem provers, including in particular **satisfiability modulo theories (SMT)** solvers. This approach has the disadvantage that it may require the user to understand in detail why the system works correctly, and to convey this information to the verification system, either in the form of a sequence of theorems to be proved or in the form of specifications (invariants, preconditions, postconditions) of system components (e.g. functions or procedures) and perhaps subcomponents (such as loops or data structures).

- [https://en.wikipedia.org/wiki/Mathematical\\_proof](https://en.wikipedia.org/wiki/Mathematical_proof)

- [https://en.wikipedia.org/wiki/Formal\\_proof](https://en.wikipedia.org/wiki/Formal_proof)

- ...

## Isabelle



- <https://isabelle.in.tum.de/>
- A generic proof assistant that allows mathematical formulas to be expressed in a formal language and provides tools for proving those formulas in a logical calculus. The main application is the formalization of mathematical proofs and in particular formal verification, which includes proving the correctness of computer hardware or software and proving properties of computer languages and protocols.
- <https://github.com/seL4/isabelle>
- <https://isabelle.in.tum.de/overview.html>

```
File Edit Search Markers Folding View Utilities Macros Plugins Help
Seq.thy (ISABELLE_HOME/HOL/examples)
section ‐finite sequences‐
theory Seq imports Main
begin
datatype 'a seq = Empty | Seq "'a seq"
fun conc :: "'a seq ⇒ 'a seq ⇒ 'a seq"
where
| "conc Empty ys = ys"
| "conc (Seq x xs) ys = Seq x (conc xs ys)"
| "conc (Seq x xs) ys = Seq x (conc xs ys)"
fun reverse :: "'a seq ⇒ 'a seq"
where
| "reverse Empty = Empty"
| "reverse (Seq x xs) = conc (reverse xs) (Seq x Empty)"
lemma conc_empty: "conc xs Empty = xs"
by (induct xs) simp_all
lemma conc_assoc: "conc (conc xs ys) zs = conc xs (conc ys zs)"
by (induct xs) (simp_all add: conc_empty conc_assoc)
lemma reverse_conc: "reverse (conc xs ys) = conc (reverse ys) (reverse xs)"
by (induct xs) (simp_all add: conc_empty conc_assoc)

Proof state: Auto update Update Search - 100%
context: "a seq = 'a seq"
Found termination order: "(λ(x). size (fst x)) <=lessThan 0"

```

Isabelle may serve as a generic framework for rapid prototyping of deductive systems. These are formulated within Isabelle's logical framework *Isabelle/Pure*, which is suitable for a variety of formal calculi (e.g. axiomatic set theory). Instantiating the generic infrastructure to a particular calculus usually requires only minimal setup in the Isabelle implementation language ML. One may also write arbitrary proof procedures or even theory extension packages in ML, without breaking system soundness (Isabelle follows the well-known *LCF system approach* to achieve a secure system).

The most widespread instance of Isabelle nowadays is *Isabelle/HOL*, which provides a higher-order logic theorem proving environment that is ready to use for big applications.

*Isabelle/HOL* includes powerful specification tools, e.g. for (co)datatypes, (co)inductive definitions and recursive functions with complex pattern matching. Proofs are conducted in the structured proof language *Isar*, allowing for proof text naturally understandable for both humans and computers.

For proofs, Isabelle incorporates some tools to improve the user's productivity. In particular, Isabelle's *classical reasoner* can perform long chains of reasoning steps to prove formulas. The *simplifier* can reason with and about equations. Linear *arithmetic* facts are proved automatically, various *algebraic* decision procedures are provided. External *first-order provers* can be invoked through *sledgehammer*.

Abstract specifications are supported by a module system (known as locales), of which type classes are a special case.

Isabelle provides excellent notational support: new notations can be introduced, using normal mathematical symbols. Definitions and proofs may include LaTeX source, from which Isabelle can automatically generate typeset documents (papers, books, theses).

*Isabelle/HOL* allows to turn executable specifications directly into code in SML, OCaml, Haskell, and Scala.

Isabelle comes with a large theory library of formally verified mathematics, including elementary number theory (for example, Gauss's law of quadratic reciprocity), analysis (basic properties of limits, derivatives and integrals), algebra (up to Sylow's theorem) and set theory (the relative consistency of the Axiom of Choice). Also provided are numerous examples arising from research into formal verification. A vast collection of applications is accessible via the [Archive of Formal Proofs](#), stemming both from mathematics and software engineering.

[Isabelle/Edit](#) is the default user interface and Prover IDE for Isabelle. It is based on [jEdit](#) and Isabelle/Scala. It provides a metaphor of continuous proof checking of a versioned collection of theory sources, with instantaneous feedback in real-time and rich semantic markup for the formal text.

## HOL

- <https://www.cl.cam.ac.uk/research/hvg/HOL/>
- The HOL System is an environment for interactive theorem proving in a *higher-order logic*. Its most outstanding feature is its high degree of programmability through the *meta-language* ML. The system has a wide variety of uses from formalizing pure mathematics to verification of industrial hardware.
- <https://www.cl.cam.ac.uk/research/hvg/HOL/history.html>
- <https://hol-theorem-prover.org/>





## 1.2.1.1 seL4

- [https://en.wikipedia.org/wiki/L4\\_microkernel\\_family#High\\_assurance:\\_seL4](https://en.wikipedia.org/wiki/L4_microkernel_family#High_assurance:_seL4)

In 2006, the NICTA group commenced a from-scratch design of a third-generation microkernel, named seL4, with the aim of providing a basis for highly secure and reliable systems, suitable for satisfying security requirements such as those of Common Criteria and beyond. From the beginning, development aimed for formal verification of the kernel. To ease meeting the sometimes conflicting requirements of performance and verification, the team used a middle-out software process starting from an executable specification written in Haskell.<sup>[16]</sup> seL4 uses capability-based security access control to enable formal reasoning about object accessibility.

A formal proof of functional correctness was completed in 2009.<sup>[17]</sup> The proof provides a guarantee that the kernel's implementation is correct against its specification, and implies that it is free of implementation bugs such as deadlocks, livelocks, buffer overflows, arithmetic exceptions or use of uninitialized variables. seL4 is claimed to be the first-ever general-purpose operating-system kernel that has been verified.<sup>[17]</sup>

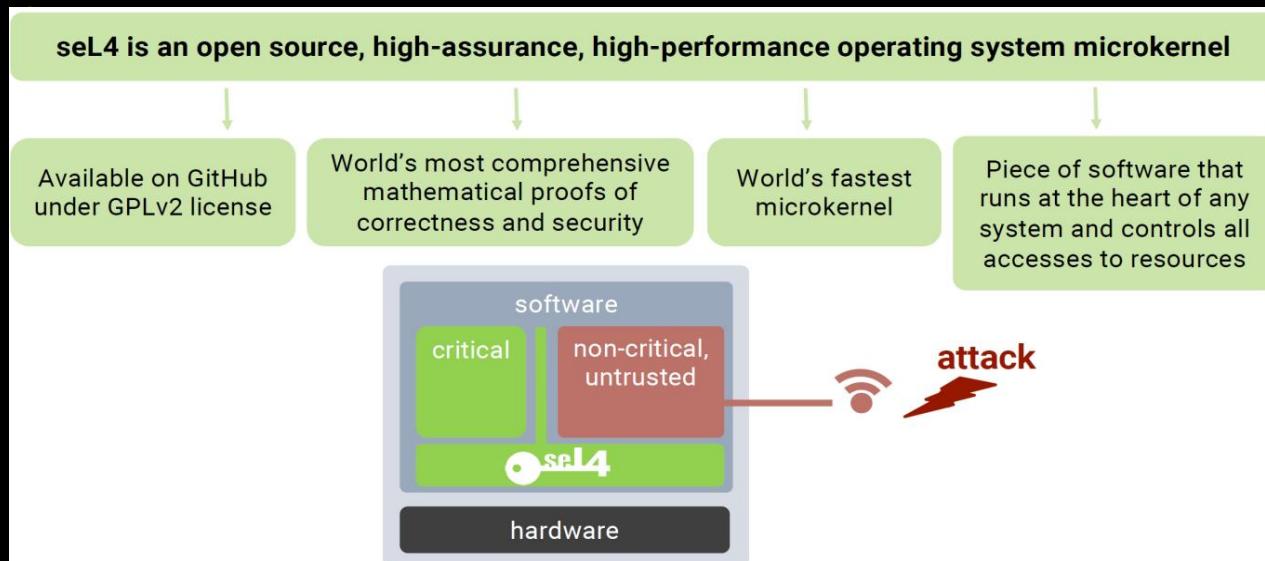
seL4 takes a novel approach to kernel resource management,<sup>[18]</sup> exporting the management of kernel resources to user level and subjects them to the same capability-based access control as user resources. This model, which was also adopted by Barreelfish, simplifies reasoning about isolation properties, and was an enabler for later proofs that seL4 enforces the core security properties of integrity and confidentiality.<sup>[19]</sup> The NICTA team also proved correctness of the translation from C to executable machine code, taking the compiler out of the trusted computing base of seL4.<sup>[20]</sup> This implies that the high-level security proofs hold for the kernel executable. seL4 is also the first published protected-mode OS kernel with a complete and sound worst-case execution-time (WCET) analysis, a prerequisite for its use in hard real-time systems.<sup>[19]</sup>

On 29 July 2014, NICTA and General Dynamics C4 Systems announced that seL4, with end to end proofs, was now released under open-source licenses.<sup>[21]</sup> The kernel source code and proofs are licensed under GNU General Public License version 2 (GPLv2), and most libraries and tools are under the BSD 2-clause. In April 2020, it was announced that the seL4 Foundation was created under the umbrella of the Linux Foundation to accelerate development and deployment of seL4.<sup>[22]</sup>

The researchers state that the cost of formal software verification is lower than the cost of engineering traditional "high-assurance" software despite providing much more reliable results.<sup>[23]</sup> Specifically, the cost of one line of code during the development of seL4 was estimated at around US\$400, compared to US\$1,000 for traditional high-assurance systems.<sup>[24]</sup>

Under the DARPA High-Assurance Cyber Military Systems (HACMS) program, NICTA together with project partners Rockwell Collins, Galois Inc, the University of Minnesota and Boeing developed a high-assurance drone based on seL4, along with other assurance tools and software, with planned technology transfer onto the optionally piloted autonomous Unmanned Little Bird helicopter under development by Boeing. Final demonstration of the HACMS technology took place in Sterling, VA in April 2017.<sup>[25]</sup> DARPA also funded several Small Business Innovative Research (SBIR) contracts related to seL4 under a program started by Dr. John Launchbury. Small businesses receiving an seL4-related SBIR included: DornerWorks, Techshot, Wearable Inc, Real Time Innovations, and Critical Technologies.<sup>[26]</sup>

- <https://sel4.systems/>



Source: "The seL4 Report", Gernot Heiser, Fosdem 2021



- <https://github.com/seL4>
- <https://sel4.systems/Use/>
- **Data61, Linux Foundation launch seL4 open source foundation**  
<https://sel4.systems/Foundation/>
- <https://microkerneldude.wordpress.com/2020/03/11/seL4-design-principles/>
- <https://docs.sel4.systems/GettingStarted.html>
- ...

## CAmkES

- <https://docs.sel4.systems/projects/camkes/>

CAmkES (component architecture for microkernel-based embedded systems) is a software development and runtime framework for quickly and reliably building microkernel-based multiserver (operating) systems. It follows a component-based software engineering approach to software design, resulting in a system that is modelled as a set of interacting software components. These software components have explicit interaction interfaces and a system design that explicitly details the connections between the components.

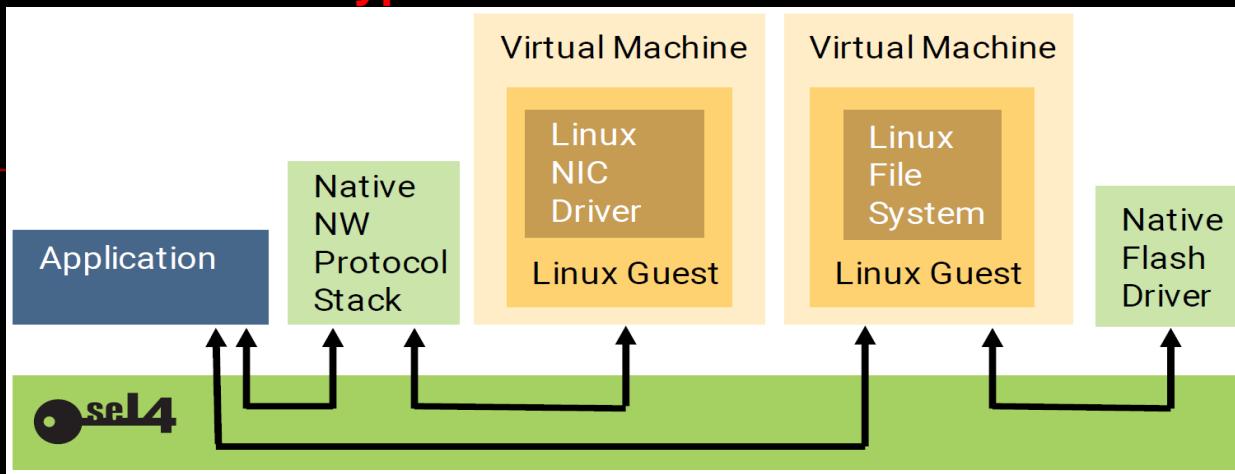
The development framework provides:

- a language to describe component interfaces, components, and whole component-based systems
- a tool that processes these descriptions to combine programmer-provided component code with generated scaffolding and glue code to build a complete, bootable, system image
- full integration in the seL4 environment and build system

- <https://docs.sel4.systems/projects/camkes/terminology>
- <https://docs.sel4.systems/Tutorials/#camkes-tutorials>
- <https://github.com/seL4/camkes>
- <https://github.com/seL4/camkes-vm>
- ...

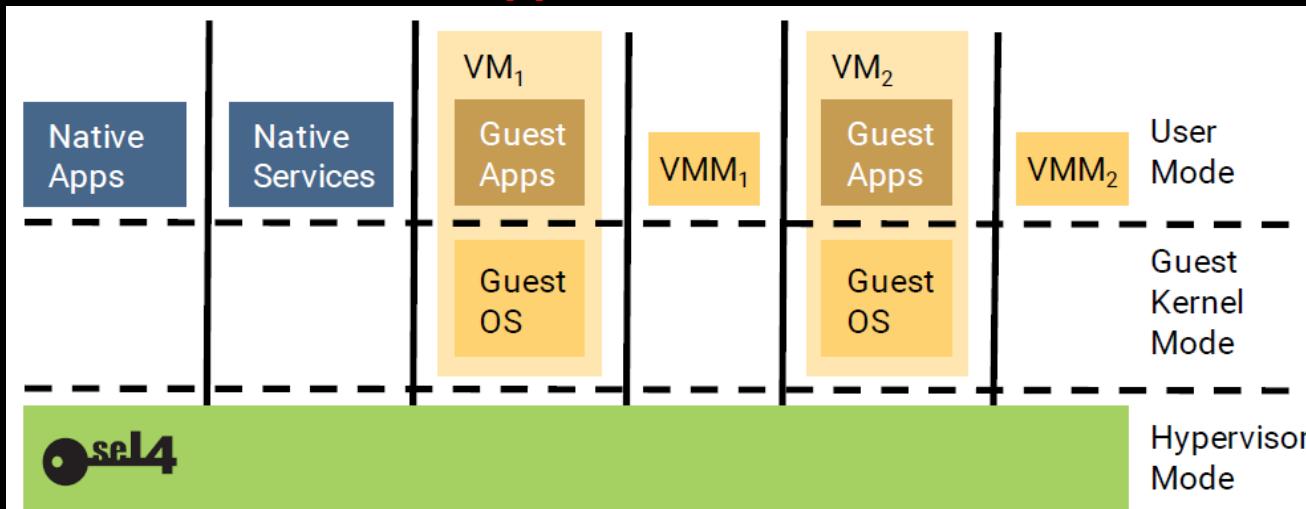
## Virtualization

- **seL4 is also a hypervisor**



Source: <https://sel4.systems/About/seL4-whitepaper.pdf>

- **seL4 virtualisation support with usermode VMMS**



Source: <https://sel4.systems/About/seL4-whitepaper.pdf>



## seL4 on RISC-V

- <https://riscv.org/announcements/2021/05/risc-v-international-and-sel4-foundation-announce-new-security-milestone/>

SAN FRANCISCO, May 5, 2021 – Today, the [seL4 Foundation](#) and [RISC-V International](#) announced that the verified seL4 microkernel on the RV64 architecture has been proved down to the executable code by CSIRO's Data61, thanks to funding provided by HENSOLDT Cyber GmbH. This guarantees that the seL4 microkernel on RV64 will operate to specification even when built with an untrusted C compiler, GCC.

Within and across open collaboration communities it is essential to work together on areas of mutual interest. RISC-V and seL4 are pleased to announce their progress and their alliances as they join forces to enable stronger overall security, combining security-oriented architecture and operating system design.

"We are excited to be one of the first architectures with secure operating system kernels with such a strong formal verification story," said Mark Himmelstein, CTO of RISC-V International. "RISC-V is continuing to increase the security features that encompass the ISA and the secure seL4 kernel is a natural complement."

"This is another milestone for seL4, which continues to define the state of the art in OS security," added Prof Gernot Heiser, Chairman of the seL4 Foundation. "Stronger aligning the two open ecosystems makes a lot of sense."

"The verified seL4 microkernel forms the core of TRENTOs, our secure operating system for our MiG-V chip, a RISC-V processor with supply chain security", said Sascha Kegreiß, CTO of HENSOLDT Cyber GmbH. "This unique combination of hardware and software security can protect critical assets from advanced persistent cyber threats."

"Translation validation ties all of our verification efforts together," said Dr Zoltan Kocsis, CSIRO Verification Engineer. "Bringing translation validation to a modern, 64-bit processor presented significant scalability challenges but, in the end, we were able to overcome them."

- <https://microkerneldude.wordpress.com/2021/05/05/sel4-on-risc-v-verified-to-binary-code/>

## 1.2.1.2 CASE(Cyber Assured Systems Engineering)

- <https://trustworthy.systems/projects/TS/CASE/>

We are working on the DARPA-sponsored [Cyber Assured Systems Engineering \(CASE\) project](#), building on our successful work in the [SMACCM](#) and [HACMS](#) projects. DARPA, the Defense Advanced Research Projects Agency (DARPA), is a US defense agency that oversees the development of emerging technologies for military use.

The goal of the CASE project is to develop re-usable design, analysis and verification tools that allow system engineers to design-in cyber resiliency when building complex, embedded computing systems. “Cyber resiliency” means that the system is tolerant to cyberattacks in the same way that safety critical systems are tolerant to random faults—they recover and continue to execute their mission function.

A novel outcome of this project will be to formally establish cyber resiliency as an explicit system property. System requirements for specific functional behaviours (e.g. to fly in a certain direction) and non-functional properties (e.g. system performance) are usually captured as “shall” statements. This approach has proven to be ineffective in engineering cyber resilient systems because cyber requirements are often statements on what the system should not do, i.e., “shall not” statements.

This project is developing ways to design and verify an embedded system when requirements are not testable (i.e., when they are expressed in “shall not” statements). We are also developing tools to automatically adapt software to new non-functional requirements and techniques to scale and provide meaningful feedback from analysis tools that reside low in the development tool chain.



### Our Partners





## 1.3 Security Analysis

### 1.3.1 Overview

- ...
-



## 1.3.2 Source Code

### 1.3.2.1 CodeQL

- <https://codeql.github.com/>

Discover vulnerabilities across a codebase with CodeQL, our industry-leading semantic code analysis engine. CodeQL lets you query code as though it were data. Write a query to find all variants of a vulnerability, eradicating it forever. Then share your query to help others do the same.

- Features

The screenshot shows the CodeQL interface. On the left, there's a sidebar with icons for file operations. The main area displays three files: `JavaConverter.java`, `UnsafeDeserialization.ql`, and `QL Query Results`.   
`JavaConverter.java` contains Java code for deserialization.   
`UnsafeDeserialization.ql` is a CodeQL query script.   
`QL Query Results` shows the results of the query, specifically alerts for unsafe deserialization of user input, listing various methods like `getContent`, `getContentAsStream`, `toBufferedInputStream`, `getInputStream`, `is`, and `ois`.

### Write and run queries in Visual Studio Code

Now that you've seen the power of the CodeQL language on LGTM.com, you're ready to write and run queries locally.

[Install CodeQL for Visual Studio Code](#)

By downloading, you agree to the [GitHub CodeQL Terms & Conditions](#).

#### Once you've installed the extension:

##### Step 1: get a CodeQL database

- Search LGTM.com for an open source project you want to research and navigate to the project page.
- Download and add the project's CodeQL database to VS Code using these instructions.

##### Step 2: query the code and find vulnerabilities

- Clone the CodeQL starter workspace and open it in VS Code.
- Run a query by right-clicking it and choosing Run Query.

See the documentation for more info.



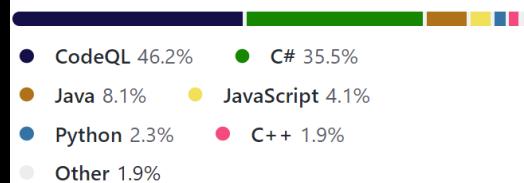
```
$ # Clone the project  
$ git clone https://github.com/m-y-mo/struts_9805  
  
$ # Create a CodeQL database  
$ codeql database create ./struts_db -s ./struts_9805 \  
-j 0 -l java --command "mvn -B -DskipTests \  
-DskipAssembly"
```

## Query open source codebases

You can create CodeQL databases yourself for any project that's under an OSI-approved open source license. To download CodeQL and get started, [visit the CodeQL CLI docs](#).

## ■ <https://github.com/github/codeql>

### Languages



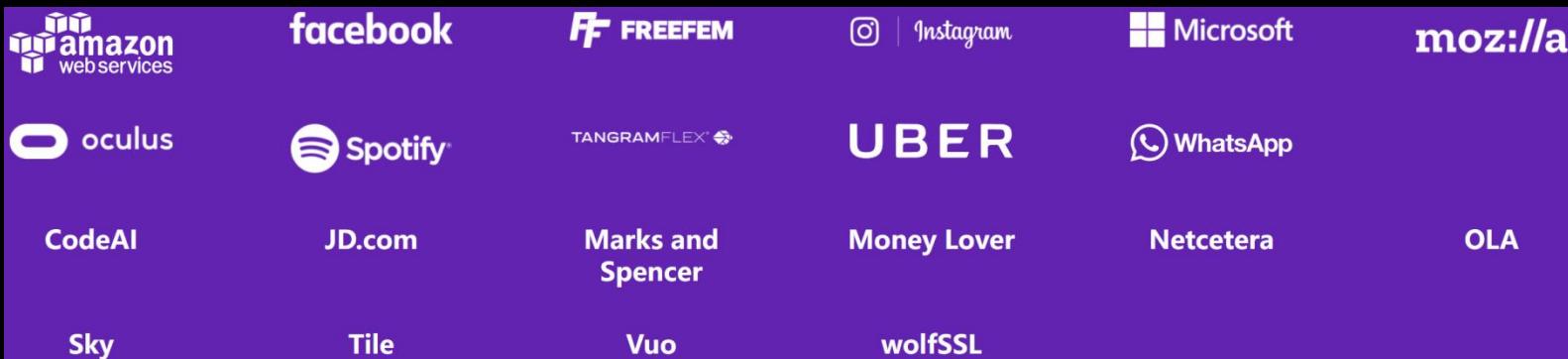


### 1.3.2.2 Infer

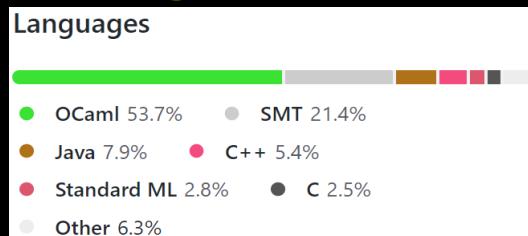
- <https://fbinfer.com/>

A static analysis tool - if you give Infer some Java or C/C++/Objective-C code it produces a list of potential bugs. Anyone can use Infer to intercept critical bugs before they have shipped to users, and help prevent crashes or poor performance.

- Who Uses Infer?



- <https://github.com/facebook/infer>



- <https://fbinfer.com/docs/getting-started/>

- ...



# 1.4 Security Models

## 1.4.1 Zero Trust

- [https://en.wikipedia.org/wiki/Zero\\_trust\\_security\\_model](https://en.wikipedia.org/wiki/Zero_trust_security_model)

The **zero trust security model** (also, **zero trust architecture**, **zero trust network architecture**, **ZTA**, **ZTNA**), sometimes known as **perimeterless security**, describes an approach to the design and implementation of **IT systems**. The main concept behind the zero trust security model is "**never trust, always verify**," which means that devices should not be trusted by default, even if they are connected to a permissioned network such as a corporate **LAN** and even if they were previously verified. Most modern corporate networks consist of many interconnected zones, **cloud services** and infrastructure, connections to remote and mobile environments, and connections to non-conventional IT, such as **IoT** devices. The reasoning for zero trust is that the traditional approach — trusting devices within a notional "corporate perimeter", or devices connected via a **VPN** — is not relevant in the complex environment of a corporate network. The zero trust approach advocates **mutual authentication**, including checking the identity and integrity of devices without respect to location, and providing access to applications and services based on the confidence of device identity and device health in combination with user authentication.<sup>[1]</sup>

### Background [edit]

The term "zero trust" was coined in April 1994 by Stephen Paul Marsh in his doctoral thesis on computer security at the [University of Stirling](#). Marsh's work studied trust as something finite that can be described mathematically, asserting that the concept of trust transcends human factors such as morality, ethics, lawfulness, justice, and judgement.<sup>[2]</sup>

The challenges of defining the perimeter to an organisation's IT systems was highlighted by the [Jericho Forum](#) in 2003, discussing the trend of what was then coined "de-perimeterisation".<sup>[citation needed]</sup> In 2009, [Google](#) implemented a zero trust architecture referred to as [BeyondCorp](#). The term zero trust model was used in 2010 by analyst John Kindervag of [Forrester Research](#) to denote stricter cybersecurity programs and access control within corporations.<sup>[3][4]</sup> However, it would take almost a decade for zero trust architectures to become prevalent, driven in part by increased adoption of mobile and cloud services.<sup>[citation needed]</sup>

In 2019 the United Kingdom [National Cyber Security Centre \(NCSC\)](#) recommended that network architects consider a zero trust approach for new IT deployments, particularly where significant use of cloud services is planned.<sup>[5]</sup>

### Principles and Definitions [edit]

In 2018, work undertaken in the [United States](#) by cybersecurity researchers at [NIST](#) and [NCCoE](#) led to the publication of [SP 800-207, Zero Trust Architecture](#).<sup>[6][7]</sup> The publication defines zero trust (ZT) as a collection of concepts and ideas designed to reduce the uncertainty in enforcing accurate, per-request access decisions in information systems and services in the face of a network viewed as compromised. A zero trust architecture (ZTA) is an enterprise's cyber security plan that utilizes zero trust concepts and encompasses component relationships, workflow planning, and access policies. Therefore, a zero trust enterprise is the network infrastructure (physical and virtual) and operational policies that are in place for an enterprise as a product of a zero trust architecture plan.

An alternative but consistent approach is taken by NCSC,<sup>[5]</sup> in identifying the key principles behind zero trust architectures:

1. Single strong source of user identity
2. User authentication
3. Machine authentication
4. Additional context, such as policy compliance and device health
5. Authorization policies to access an application
6. Access control policies within an application

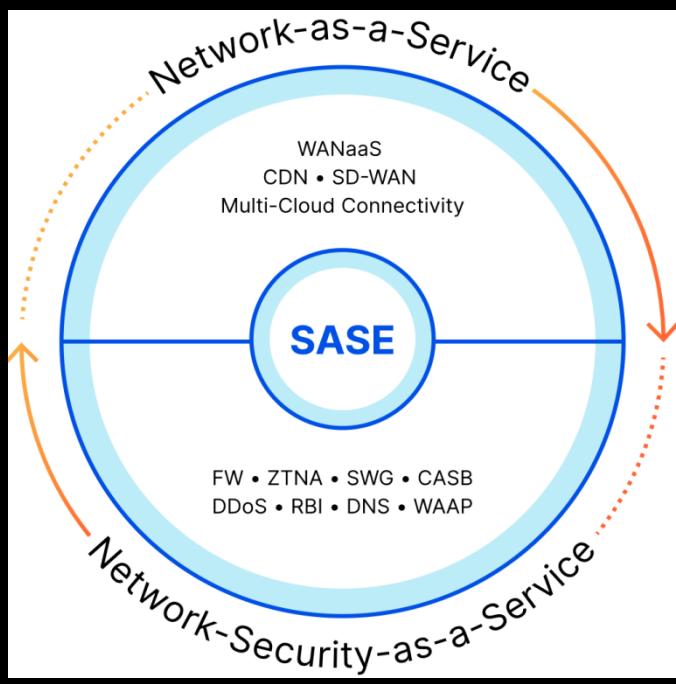


## SASE

- <https://www.cloudflare.com/learning/access-management/what-is-sase/>

Secure access service edge, or SASE, is a cloud-based IT model that bundles **software-defined networking** with network security functions and delivers them from a single service provider. [Gartner](#), a global research and advisory firm, coined the term "SASE" in 2019.

A SASE approach offers better control over and visibility into the users, traffic, and data accessing a corporate network — vital capabilities for modern, globally distributed organizations. Networks built with SASE are flexible and scalable, able to connect globally distributed employees and offices across any location and via any device.

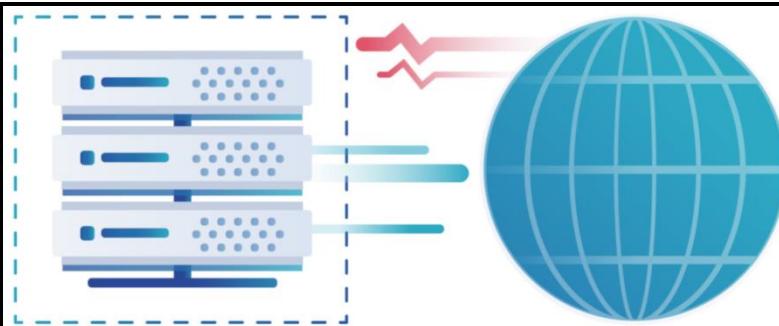




## SDP

- <https://www.cloudflare.com/learning/access-management/software-defined-perimeter/>

A software-defined perimeter (SDP) is a way to hide Internet-connected infrastructure (servers, routers, etc.) so that external parties and attackers cannot see it, whether it is hosted on-premise or in the [cloud](#). The goal of the SDP approach is to base the network perimeter on software instead of hardware. A company that uses an SDP is essentially draping a cloak of invisibility over their servers and other infrastructure so that no one can see it from the outside; however, authorized users can still access the infrastructure.



A software-defined perimeter forms a virtual boundary around company assets at the network layer, not the [application layer](#). This separates it from other [access-based controls](#) that restrict user privileges but allow wide network access. Another key difference is that an SDP authenticates devices as well as user identity. The Cloud Security Alliance [first developed](#) the SDP concept.

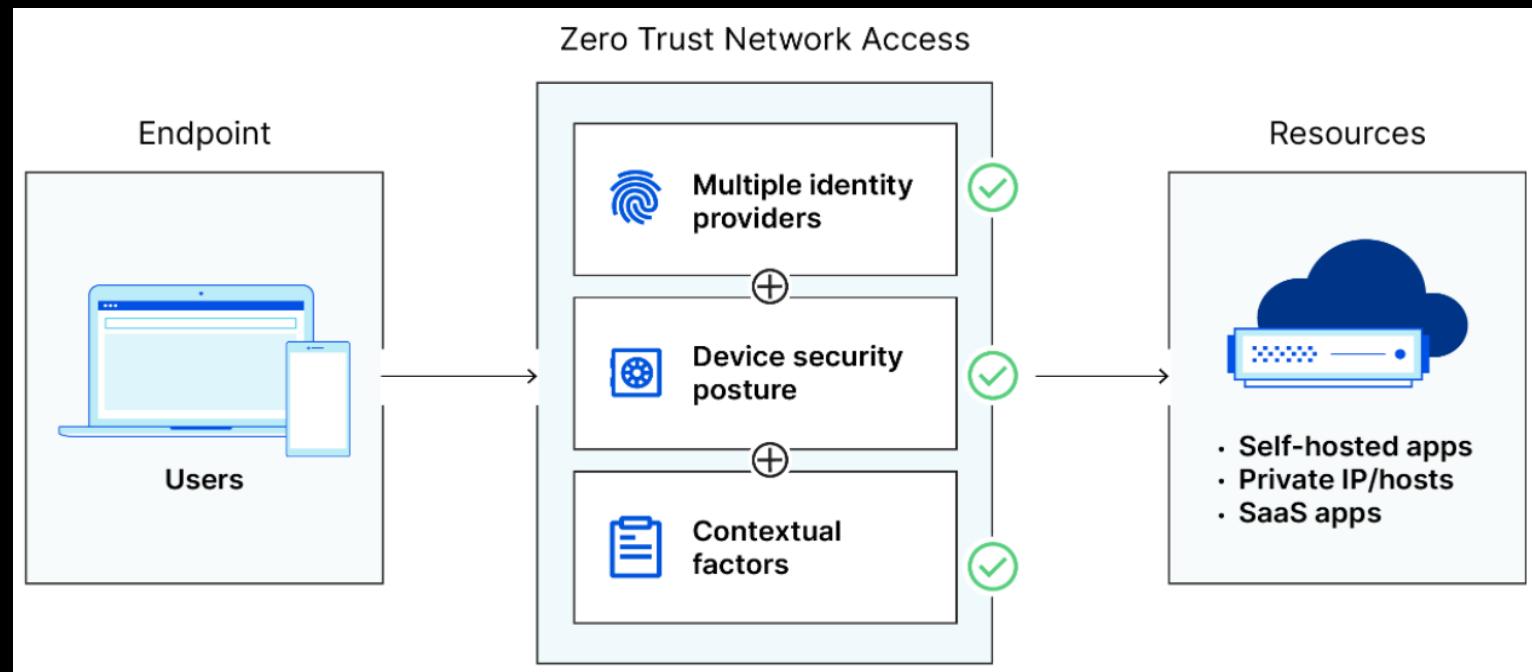
- ...



## ZNTA

- <https://www.cloudflare.com/learning/access-management/what-is-ztna/>

Zero Trust Network Access (ZTNA) is the technology that makes it possible to implement a [Zero Trust security](#) model. "Zero Trust" is an IT security model that assumes threats are present both inside and outside a network. Consequently, Zero Trust requires strict verification for every user and every device before authorizing them to access internal resources.





ZTNA is similar to the [software-defined perimeter](#) (SDP) approach to controlling access. In ZTNA, like in SDP, connected devices are not aware of any resources (applications, servers, etc.) on the network other than what they are connected to.

---

Imagine a scenario in which every resident gets a phone book with the phone numbers of every other resident of their city, and anyone can dial any number to contact any other person. Now imagine a scenario in which everyone has an unlisted phone number and one resident has to know another resident's phone number in order to call them. This second scenario offers a few advantages: no unwanted calls, no accidental calls to the wrong person, and no risk of unscrupulous persons using the city's phone book to fool or scam the residents.

ZTNA is like the second scenario. But instead of phone numbers, ZTNA uses "unlisted" [IP addresses](#), applications, and services. It sets up one-to-one connections between users and the resources they need, like when two people who need to contact each other exchange phone numbers. But unlike two people exchanging numbers, ZTNA connections need to be re-verified and recreated periodically.

...



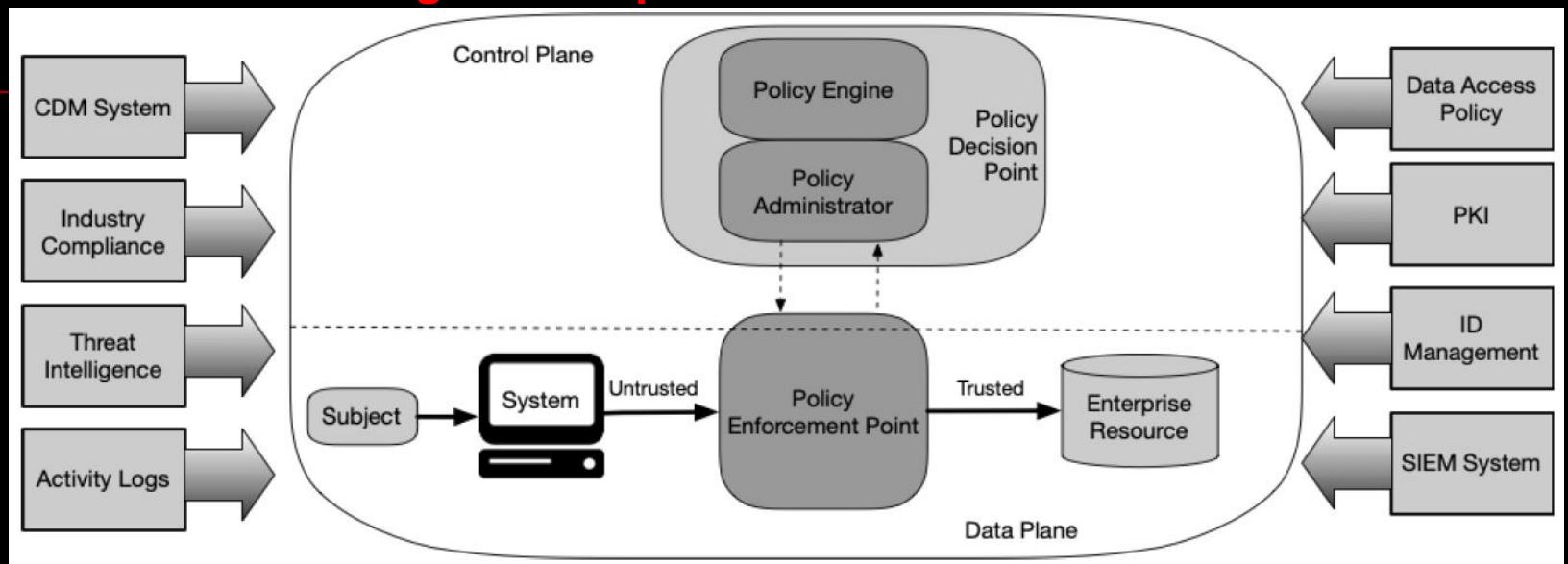
## Good Resources

- <https://www.cloudflare.com/learning/security/glossary/what-is-zero-trust/>
- <https://www.microsoft.com/security/blog/2022/02/22/the-federal-zero-trust-strategy-and-microsofts-deployment-guidance-for-all/>
- ...

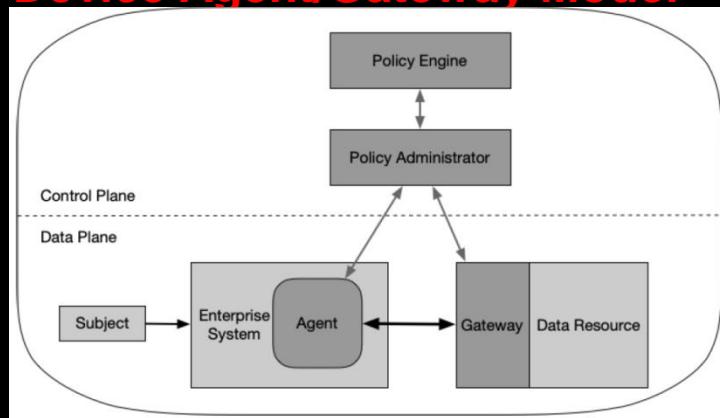


## 1.4.1.1 NIST 800-207(Zero Trust Architecture)

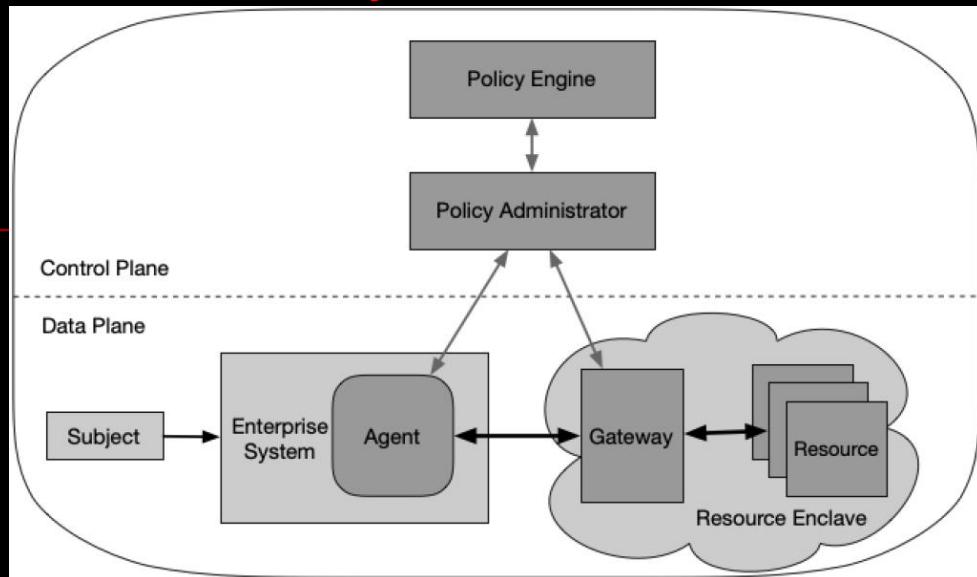
- <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf>
- Core Zero Trust Logical Components



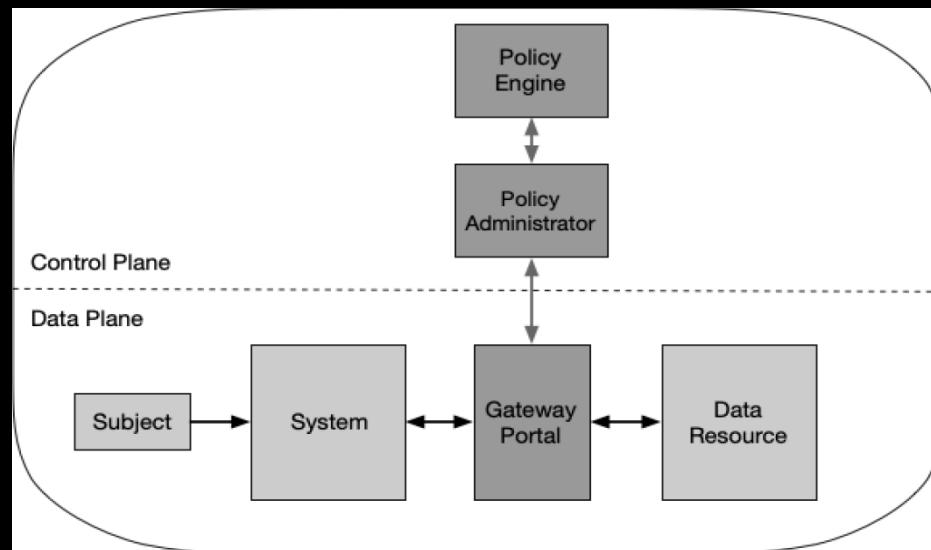
- Device Agent/Gateway Model



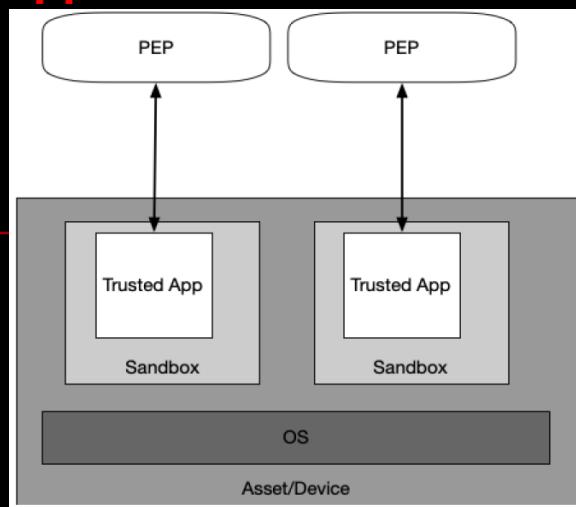
## ■ Enclave Gateway Model



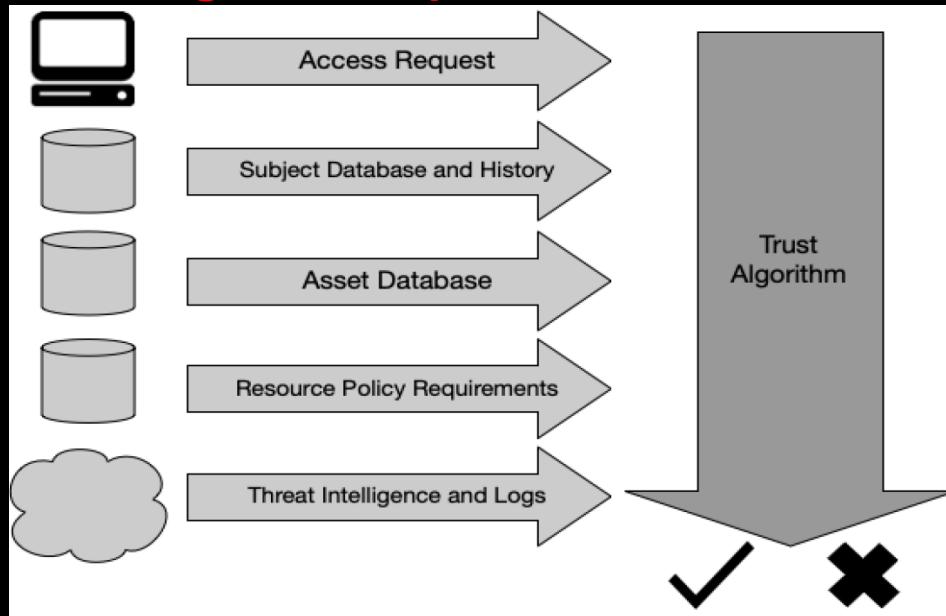
## ■ Resource Portal Model



## ■ Application Sandboxes

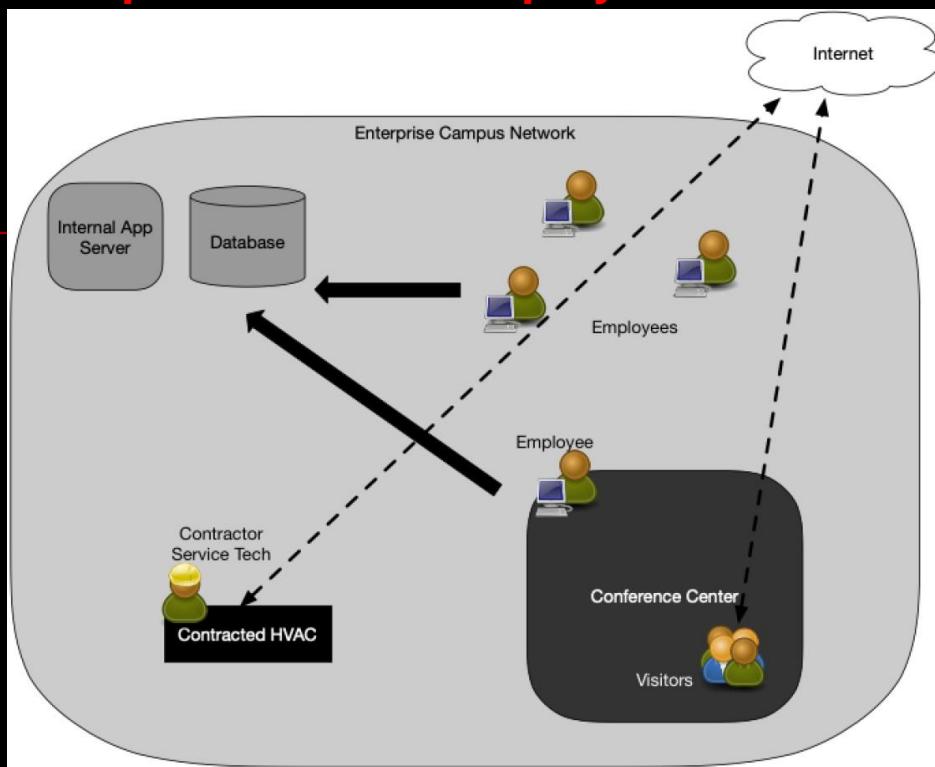


## ■ Trust Algorithm Input

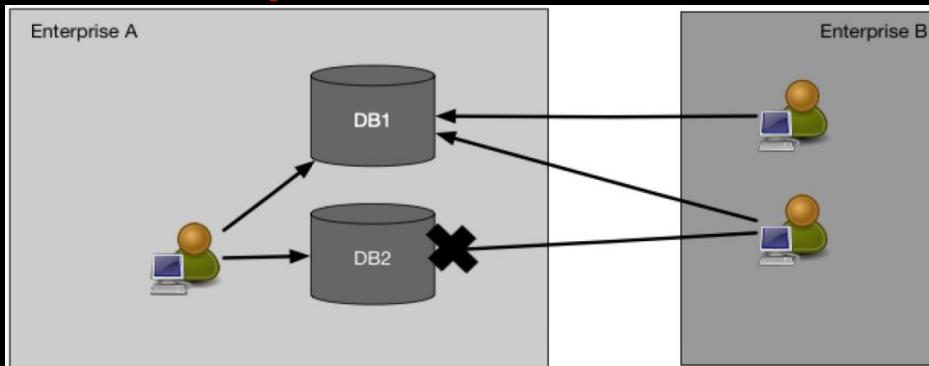




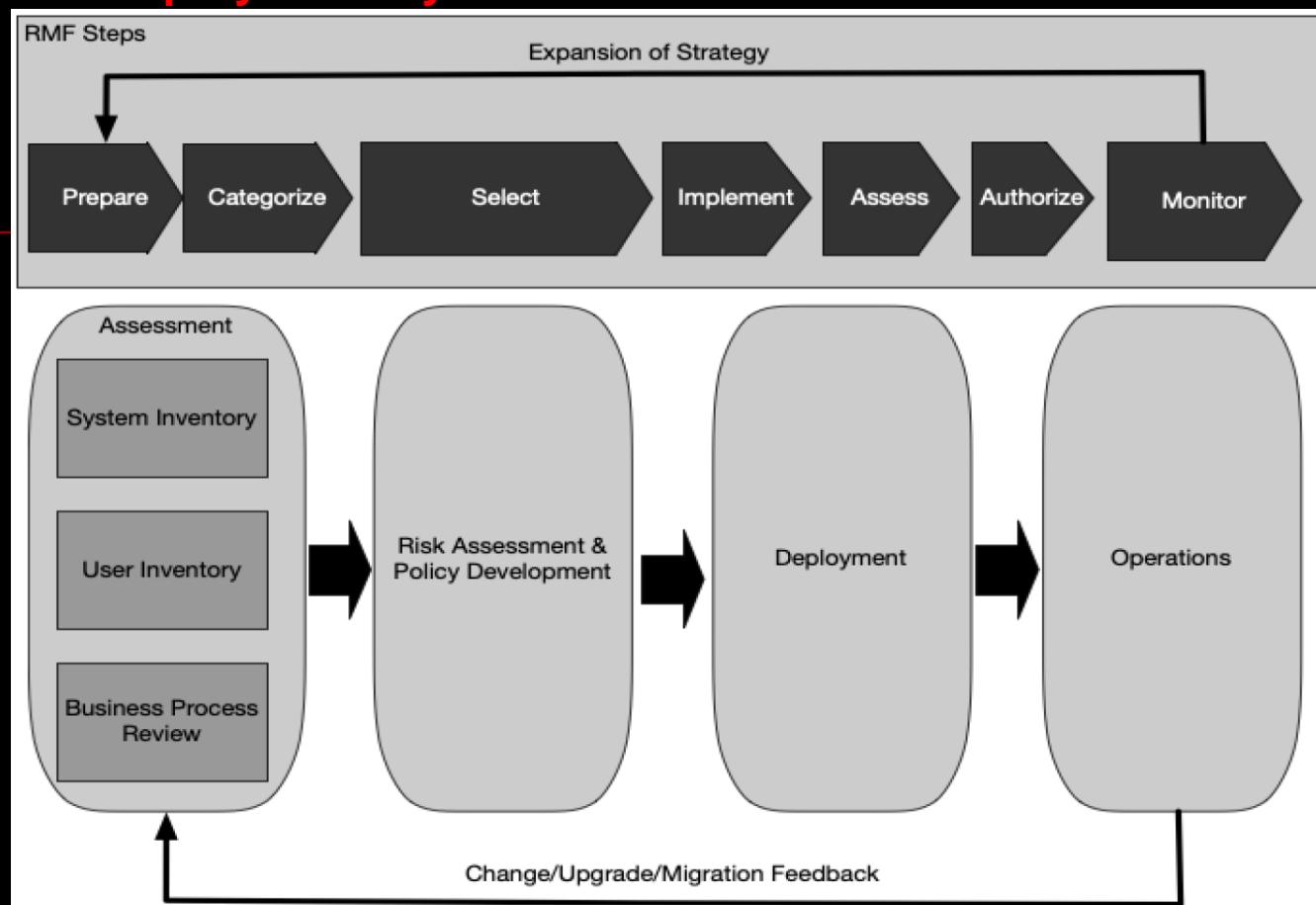
## ■ Enterprise with Nonemployee Access



## ■ Cross-Enterprise Collaboration



## ■ ZTA Deployment Cycle





## 1.5 HW-SW Co-design

### 1.5.1 Overview

- <https://en.wikipedia.org/wiki>
-



## 1.5.2 CHERI

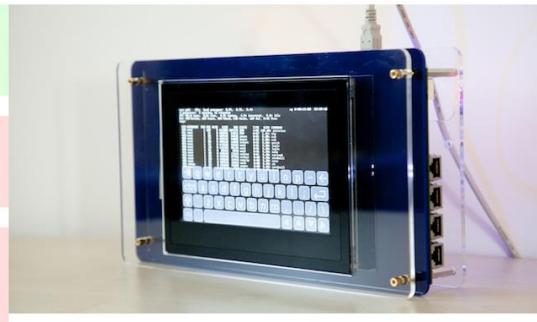
- <https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/>  
**Capability Hardware Enhanced RISC Instructions**

PIs: Robert N. M. Watson (University of Cambridge), Simon W. Moore (University of Cambridge), Peter Sewell (University of Cambridge), and Peter Neumann (SRI International)

January 2022: Arm has shipped its CHERI-enabled Morello prototype processor, SoC, and board! Read blog posts about this at [Arm](#) and [Microsoft](#), and our own thoughts at [Cambridge](#).

October 2020: We have posted **CHERI ISAv8**. This ISA version is synchronized to Arm's Morello architecture, as well as presenting a mature version of our CHERI-RISC-V ISA.

September 2019: **Learn about the CHERI architecture!** Our technical report [An Introduction to CHERI](#) is a high-level summary of our work on CHERI architecture, microarchitecture, formal modeling, and software.



### What is it

**CHERI (Capability Hardware Enhanced RISC Instructions)** is a joint research project of SRI International and the University of Cambridge to revisit fundamental design choices in hardware and software to dramatically improve system security. CHERI has been supported by the DARPA CRASH, MRC, and SSITH programs since 2010, as well as other DARPA research and transition funding. Since 2019, development of Arm's experimental CHERI-enabled Morello processor, SoC, and board has been supported by UKRI. We gratefully acknowledge DARPA, UKRI, and our other supporters including EPSRC, ERC, Google, and Arm.

CHERI extends conventional hardware Instruction-Set Architectures (ISAs) with new architectural features to enable fine-grained memory protection and highly scalable software compartmentalization. The CHERI memory-protection features allow historically memory-unsafe programming languages such as C and C++ to be adapted to provide strong, compatible, and efficient protection against many currently widely exploited vulnerabilities. The CHERI scalable compartmentalization features enable the fine-grained decomposition of operating-system (OS) and application code, to limit the effects of security vulnerabilities in ways that are not supported by current architectures.

CHERI is a **hybrid capability architecture** in that it is able to blend **architectural capabilities** with conventional MMU-based architectures and microarchitectures, and with conventional software stacks based on virtual memory and C/C++. This approach allows incremental deployment within existing software ecosystems, which we have demonstrated through extensive hardware and software prototyping.

CHERI is a **hybrid capability architecture** in that it is able to blend **architectural capabilities** with conventional MMU-based architectures and microarchitectures, and with conventional software stacks based on virtual memory and C/C++. This approach allows incremental deployment within existing software ecosystems, which we have demonstrated through extensive hardware and software prototyping.

CHERI interacts with the design of the full hardware-software stack. We have developed:

- An abstract CHERI protection model that introduces architectural capabilities, hardware-supported descriptions of permissions that can be used, in place of integer virtual addresses, to refer to data, code, and objects in protected ways;
- A set of ISA extensions to 64-bit MIPS, 32-bit RISC-V, 64-bit RISC-V, and (in collaboration with Arm) 64-bit Armv8-A, showing that the model is applicable to a range of contemporary ISA designs.
- New microarchitecture demonstrating that capabilities can be implemented efficiently in hardware, including capability compression and tagged memory to protect capabilities in memory; and
- Formal models of these ISA extensions enabling mechanised statements and proofs of their security properties, automatic test generation, and automatic construction of executable ISA-level simulators.

These features enable new software constructs that are incrementally deployable within existing software ecosystems. Through extensive prototyping and co-design, we have demonstrated and evaluated:

- New software construction models that use capabilities to provide fine-grained memory protection and scalable software compartmentalization;
- Language and compiler extensions to use capabilities in implementing memory-safe C and C++, higher-level managed languages, and Foreign Function Interfaces (FFIs);
- OS extensions to use fine-grained memory protection, and to support applications that use CHERI, including through spatial, referential, and temporal memory safety;
- OS extensions that provide new CHERI-based abstractions including in-kernel and intra-process compartmentalization, and new efficient Inter-Process Communication (IPC);
- Application-level adaptations to operate correctly with CHERI memory protection; and
- Application-level adaptations to introduce new and more affordable software compartmentalization.



## CHERI for various ISA

In addition to our CHERI-MIPS FPGA prototypes and software, as well as formal models, we are working actively to bring CHERI to the ARMv8-A and RISC-V ecosystems:

### ARMv8-A

Since 2014, supported by DARPA, we have been collaborating with Arm to develop an experimental integration of CHERI with 64-bit ARMv8-A. InnovateUK will be jointly funding the creation of **an experimental superscalar CHERI-ARM processor (based on the Neoverse N1), SoC, and evaluation board ("Morello")** to be available for academic and industrial research from late 2021.

We will bring our full CHERI software stack to Morello, as well as performing formal modeling and verification as part of the effort, and are already able to use the full CheriBSD software stack on an early ISA-level model. EPSRC and ESRC have announced funding calls to support CHERI-related research on the board. We look forward to supporting this programme of research through software prototypes and formal models. More information on this effort can be found in our [Digital Security by Design](#) page.

Arm released a software simulator and open-source toolchain (based on CHERI Clang/LLVM) in late 2020, and we have open sourced CheriBSD for Morello as well as CHERI GDB.

### RISC-V

Since 2017, supported by DARPA, we have been creating an experimental adaptation of CHERI to the 32-bit and 64-bit RISC-V ISAs. This includes multiple FPGA prototypes based on Bluespec and MIT BSV-language cores: 3-stage, 32-bit MMU-free RISC-V; 5-stage 64-bit RISC-V; and superscalar 64-bit RISC-V. CHERI ISAv7 includes a fully elaborated version of the CHERI-RISC-V ISA. CHERI Clang/LLVM are already up and running, and we are able to boot a pure-capability CheriFreeRTOS on the 32-bit core. We are in the process of bringing up the remainder of our software stack including CheriBSD on the first 64-bit core. This work is available as open source in our GitHub repositories.

## Prototype CHERI-MIPS processor on FPGA

We have developed a prototype of the CHERI ISA using the [Bluespec Extensible RISC Implementation \(BERI\)](#), a 64-bit MIPS FPGA soft core implemented in the Bluespec HDL. The FreeBSD operating system has also been ported to CHERI in order to allow us to compare, side-by-side, traditional software compartmentalisation approaches (based on a translation look-aside buffer (TLB)), with those supported by a capability coprocessor. We run lightly modified commodity software stacks (see below) on this prototype, allowing us to validate our hybrid design, evaluating compatibility, performance, and security implications of our changes to hardware and software.



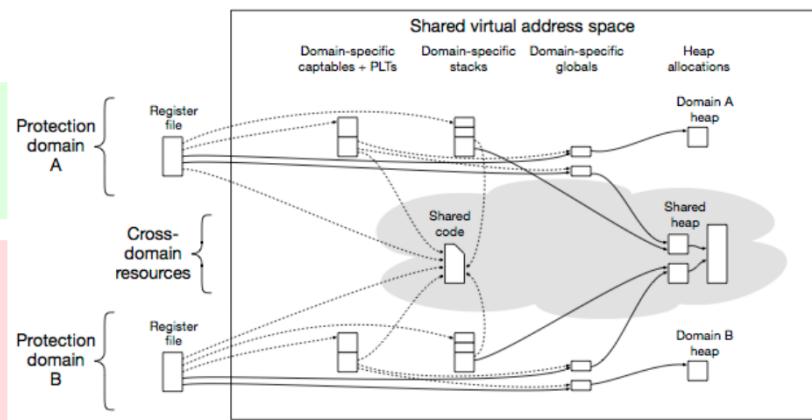
## Software Stack

- <https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/cheri-software.html>

### CHERI Software Stack

February 2021: We have open sourced a **CHERI adaptation of the WebKit browser framework and JavaScript interpreter**, which has been developed in close collaboration with Arm. This is the first open-source JIT available for CHERI, and runs on Arm's Morello architecture.

April 2019: We are pleased to announce that our **ASPLOS 2019 paper on CHERI and OS design, CheriABI: Enforcing Valid Pointer Provenance and Minimizing Pointer Privilege in the POSIX C Runtime Environment**, on the general-purpose OS design implications of CHERI when used for ubiquitous memory safety, has won an ASPLOS Best Paper award.



### CHERI Clang/LLVM compiler suite, LLD linker, and GDB debugger

**CHERI Clang/LLVM and LLD** are a complete compiler suite and compile-time linker for use with the CHERI architecture as instantiated for 64-bit MIPS and 32/64-bit RISC-V. We have also adapted the GDB debugger to use CHERI. The suite targets two C interpretations and ABIs:

#### **Hybrid-capability C and binary code**

*Hybrid-capability C/C++* starts with the conventional integer implementation of C/C++ pointers, but allows types to be optionally qualified in order to trigger implementation using CHERI capabilities. This model could also transparently introduce capability use where it does not interfere with existing ABIs. This model is typically used in low-level OS components and compatibility layers, such as in allowing the pure-capability CheriABI process environment to be implemented over a hybrid-capability kernel.

#### **Pure-capability C and binary code**

*Pure-capability C/C++* implements all C/C++ pointers using CHERI capabilities. This includes explicit pointers (such as explicitly declared pointers to functions, global variables, local variables, and heap allocations), and also implicit pointers (such as return addresses, and GOT pointers). This model is suitable for a broad variety of uses, including in providing strong memory safety for operating-system kernels and user applications. CHERI C is documented in our **CHERI C/C++ Programming Guide**.



## CheriBSD operating system

**CheriBSD** is an adaptation of the open-source FreeBSD operating system to utilize the CHERI Architecture's capability-system model. CheriBSD provides strong CHERI-based memory protection for the kernel and userspace, as well as support for scalable single-address-space software compartmentalization. By default, the kernel is compiled as a hybrid-capability binary, but we have experimental extensions that allow it to run as a pure-capability binary. Both kernels support running existing off-the-shelf userspace MIPS and RISC-V binaries, as well as pure-capability binaries running in the CheriABI process environment. CheriBSD runs on 64-bit CHERI-MIPS and 64-bit CHERI-RISC-V.

## QEMU-CHERI

We have also developed a **QEMU CHERI-MIPS** and **CHERI-RISC-V** implementation, which provides an ISA-level emulation of our CHERI extensions to the 64-bit MIPS and RISC-V ISAs. While not micro-architecturally realistic, this emulation can be useful for software development, especially in the absence of an FPGA or access to Bluespec. It is faster than the Sail-generated C emulator, but less directly based on the Sail CHERI-MIPS and CHERI-RISC-V ISA specifications.

## Other ported software

In addition to software packages such as OpenSSL and OpenSSH that are included in **CheriBSD**, we have ported a number of other applications to CHERI. These include:

### **nginx**

Webserver

### **newlib**

Embedded/baremetal C standard library.

### **PostgreSQL**

Enterprise relational database.

### **Qt**

Cross-platform GUI framework.

### **SQLite**

Embedded relational database.

### **WebKit**

The WebKit browser platform and JavaScript Core (JSC) language runtime. This includes a CHERI-adapted JIT for Morello

...



## Good Resources

- <https://www.arm.com/why-arm/architecture/cpu/morello>
- <https://www.ukri.org/what-we-offer/our-main-funds/industrial-strategy-challenge-fund/artificial-intelligence-and-data-economy/digital-security-by-design-challenge/>
- <https://git.morello-project.org/morello/docs>
- ...

## 1.5.2.1 Implementation

### Morello

#### ■ <https://www.morello-project.org/>

Morello is a research program led by Arm in association with partners and funded by the UKRI as part of the UK government **Digital Security by Design (DSbD) programme**. It defines a new prototype security architecture based on CHERI (Capability Hardware Enhanced RISC Instructions).

A DSbD technology platform prototype (the Morello board) provides a SoC implementation of the architecture. This was created to enable software developers and researchers to explore real-world use cases and inform future development.

### ***Development Platforms***

#### **Morello Platform Model**

The Morello Platform Model is an open access FVP (Fixed Virtual Platform) implementation aligned with the development board. It is available to download from Arm's [Ecosystem FVP Developer page](#).

FVPs use Arm binary translation technology to create a register level functional model of system hardware (including processor, memory and peripherals) that can be run as an executable in a development environment. They implement a programmer's view model suitable for software development, enabling execution of full software stacks on a widely available platform.

#### **Morello Hardware Development Platform**

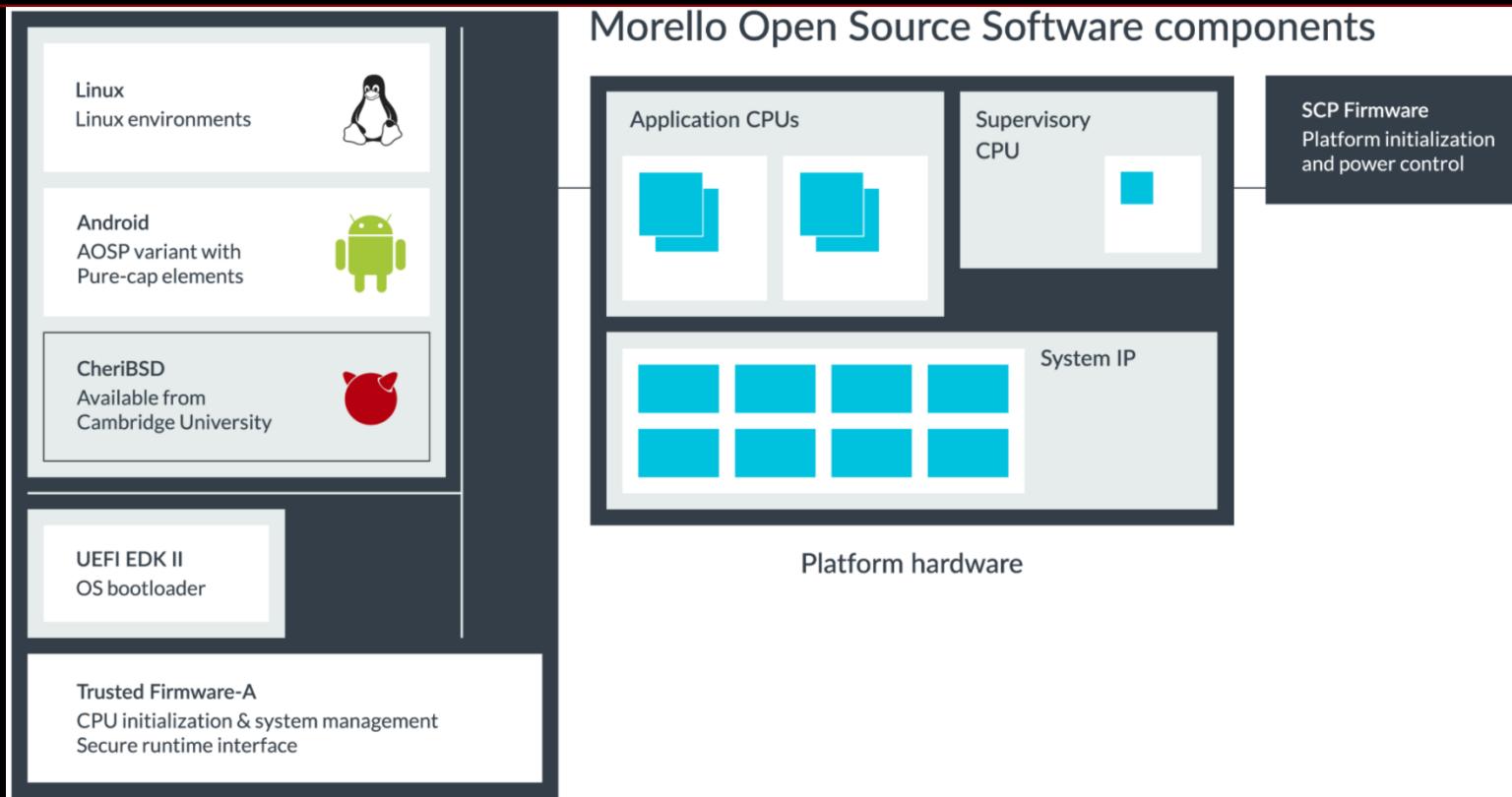
The Morello hardware development platform (available Q1 2022 onwards) is supported by the same range of software stacks as the existing Platform model. Availability of hardware will be limited - platforms will be restricted to partners involved in defined research activities.





## Software Stack

The diagram shows a high-level view of the software stacks targeting the Morello hardware and FVP platforms. Above the firmware, which environment is best suited for development is dictated by a range of factors relating to the specific aims of individual research projects. The CheriBSD port for Morello provides a mature environment for general research and userspace experimentation. However, the majority of commercial devices are based on Android and Linux, which is why Arm are also focused on exploring the application of the Morello prototype architecture to these environments.

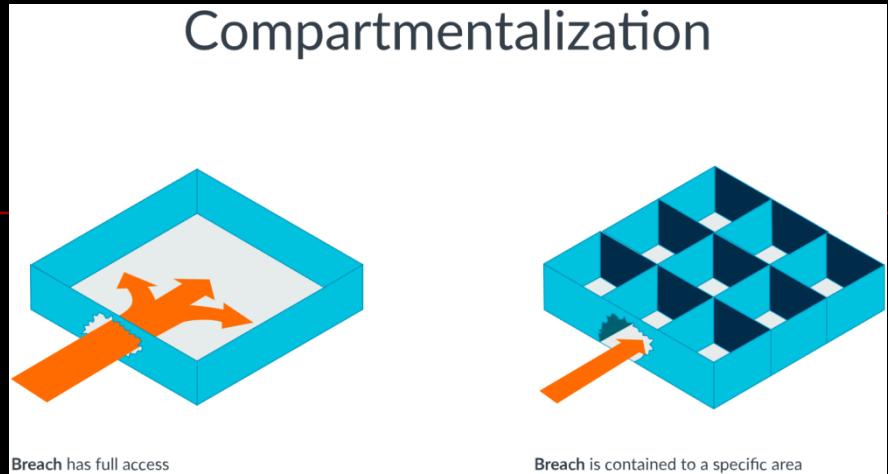


These stacks and the supporting tooling are intended to provide a foundation for ecosystem research, enabling collaboration on existing work packages and new work on alternate RTOS/OS environments, tools and workloads. Functionality will evolve in stages throughout the lifetime of the Morello Program.



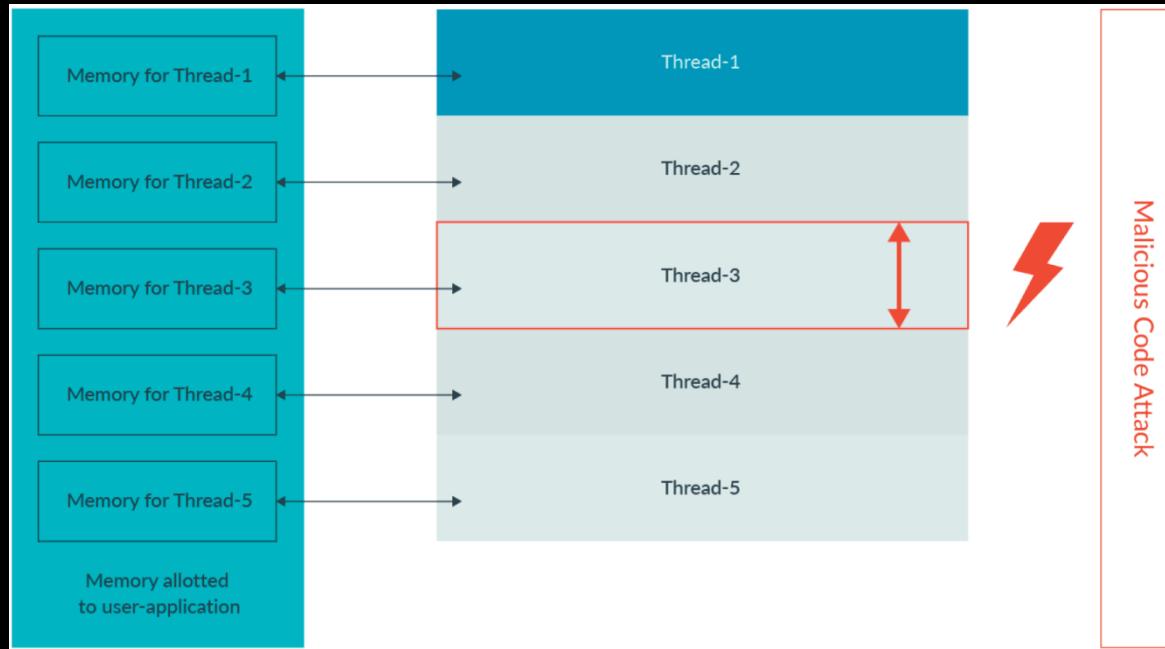
## Arch & Design

### Compartmentalization

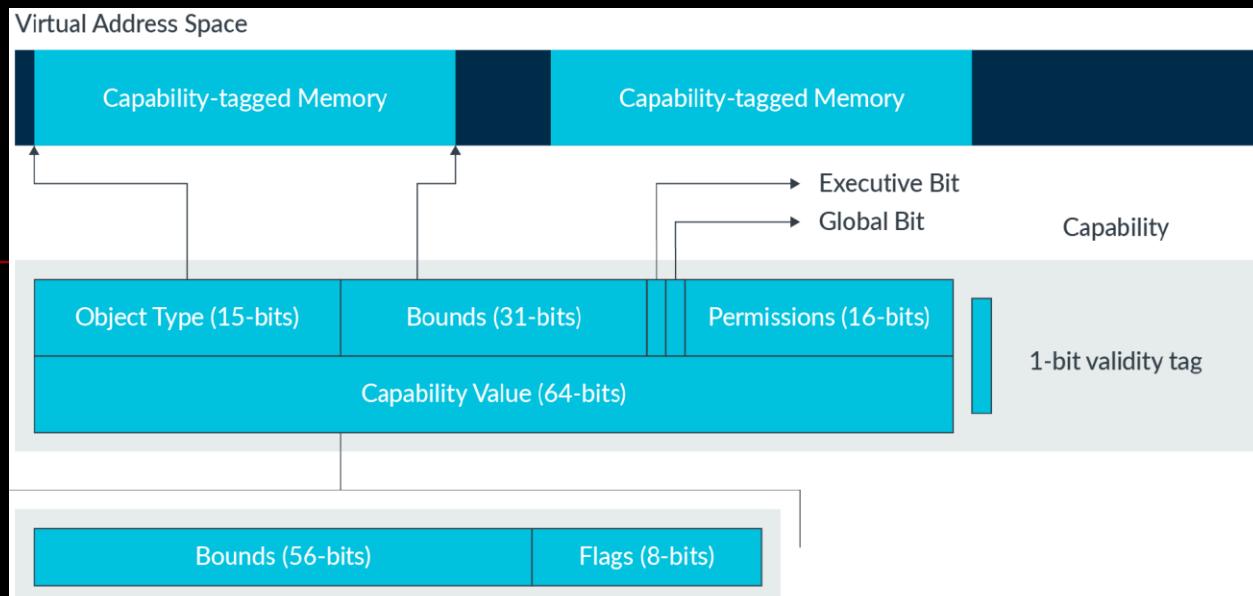


Breach has full access

Breach is contained to a specific area

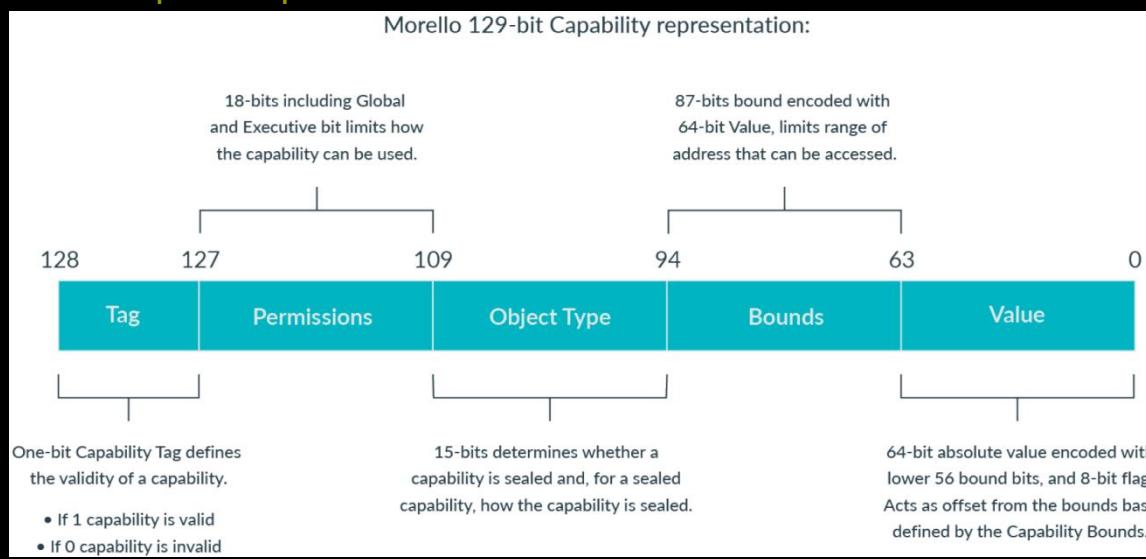


Source: <https://developer.arm.com/documentation/den0133/latest>



Source: <https://developer.arm.com/documentation/den0132/latest/>

Morello 129-bit Capability representation:

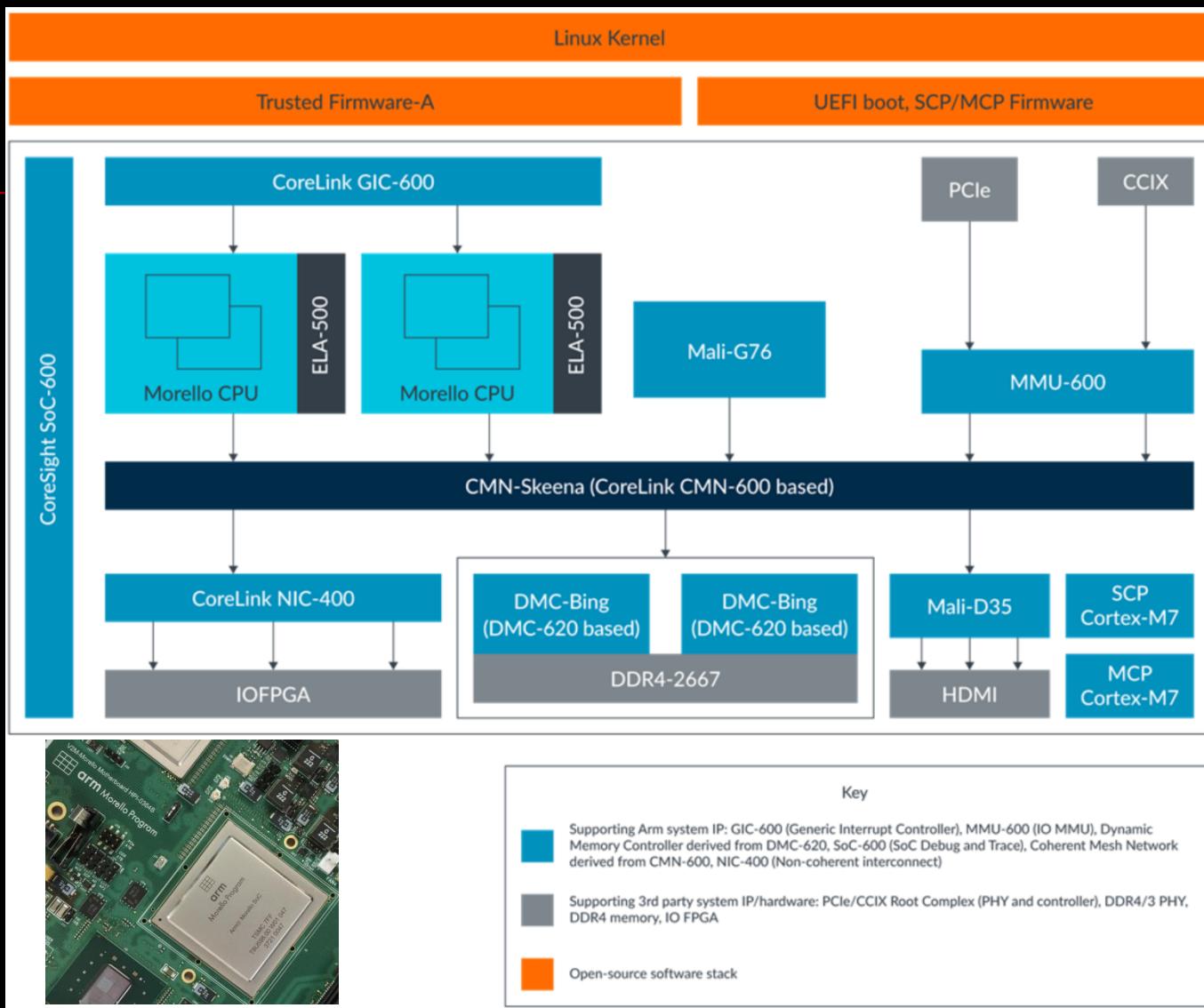


Source: <https://developer.arm.com/documentation/den0133/latest>

\*\*\*



## Prototype SoC design



Source: [https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/creating-the-morello-technology-demonstrator?\\_ga=2.91341786.1569610424.1648308280-758543983.1648308280](https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/creating-the-morello-technology-demonstrator?_ga=2.91341786.1569610424.1648308280-758543983.1648308280)



Src

<https://git.morello-project.org/morello>

Morello		Group ID: 6			
Subgroups and projects		Shared projects	Archived projects	Search by name	Last updated
A	Android	4	0	1	1 year ago
K	Kernel	0	1	2	2 months ago
U	Utilities	0	1	1	1 week ago
B	Binutils GDB	★ 0			1 year ago
B	board-firmware	★ 0			2 months ago
B	Build scripts	★ 0			1 week ago
D	Documentation	★ 1			2 months ago
E	EDK II	★ 0			1 week ago
E	EDK II platforms	★ 0			2 days ago
L	Ilvm-project	★ 3			5 days ago
L	Ilvm-project-releases	★ 0			5 days ago
M	Manifest	★ 0			1 week ago
M	meta-arm	★ 0			6 months ago
M	Model scripts	★ 0			1 week ago
C	Morello CI Containers	★ 0			4 months ago
M	Morello CI Documentation	★ 0			8 months ago
C	Morello CI Pipelines	★ 0			1 week ago
M	Morello CI Trigger	★ 0			4 months ago
M	morello-aarch64	★ 0			1 week ago
M	Musl	★ 2			3 weeks ago
N	Newlib	★ 0			8 months ago
S	SCP Firmware	★ 0			1 week ago
T	Trusted Firmware-A	★ 0			1 week ago
Y	yocto-platform-config	★ 0			6 months ago



## 1.6 Summary

### Addons

- For Virtualization and Security(especially on ARM), you may refer to my previous talks:

"**seL4 for secure Edge Cloud infrastructure**" at CID 2021(Shanghai)

"**Cloud-Hypervisor on ARM**" at the Rust China Meetup 2021(Hangzhou) and the upcoming follow-ups.



## 2) eBPF-based Linux System Security

### 2.1 Overview

#### 2.1.1 LSM (Linux Security Module)

- [https://en.m.wikipedia.org/wiki/Linux\\_Security\\_Modules](https://en.m.wikipedia.org/wiki/Linux_Security_Modules)
- <https://www.kernel.org/doc/html/latest/admin-guide/LSM/index.html>
- <https://docs.kernel.org/security/lsm.html>
- [https://www.kernel.org/doc/html/latest/bpf/bpf\\_lsm.html](https://www.kernel.org/doc/html/latest/bpf/bpf_lsm.html)
- ...



## 2.1.1.1 KRSI (Kernel Runtime Security Instrumentation)

### ■ eBPF + LSM

<https://lwn.net/Articles/798157/>

<https://lwn.net/Articles/808048/>

<https://lwn.net/Articles/809841/>

<https://lwn.net/Articles/813261/>

//Kernel runtime security instrumentation

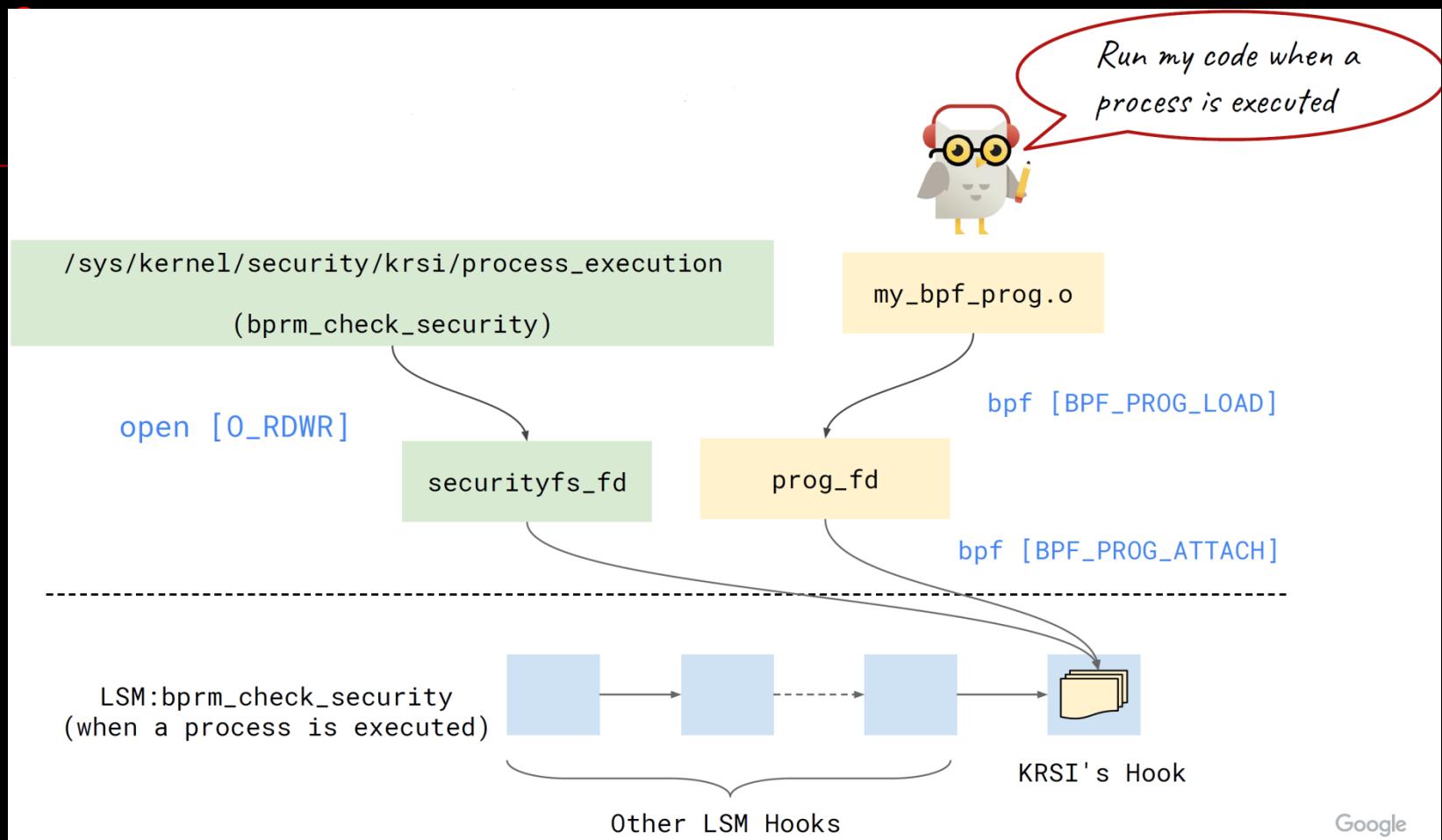
//KRSI — the other BPF security module

//KRSI and proprietary BPF programs

//Impedance matching for BPF and LSM



## How it works

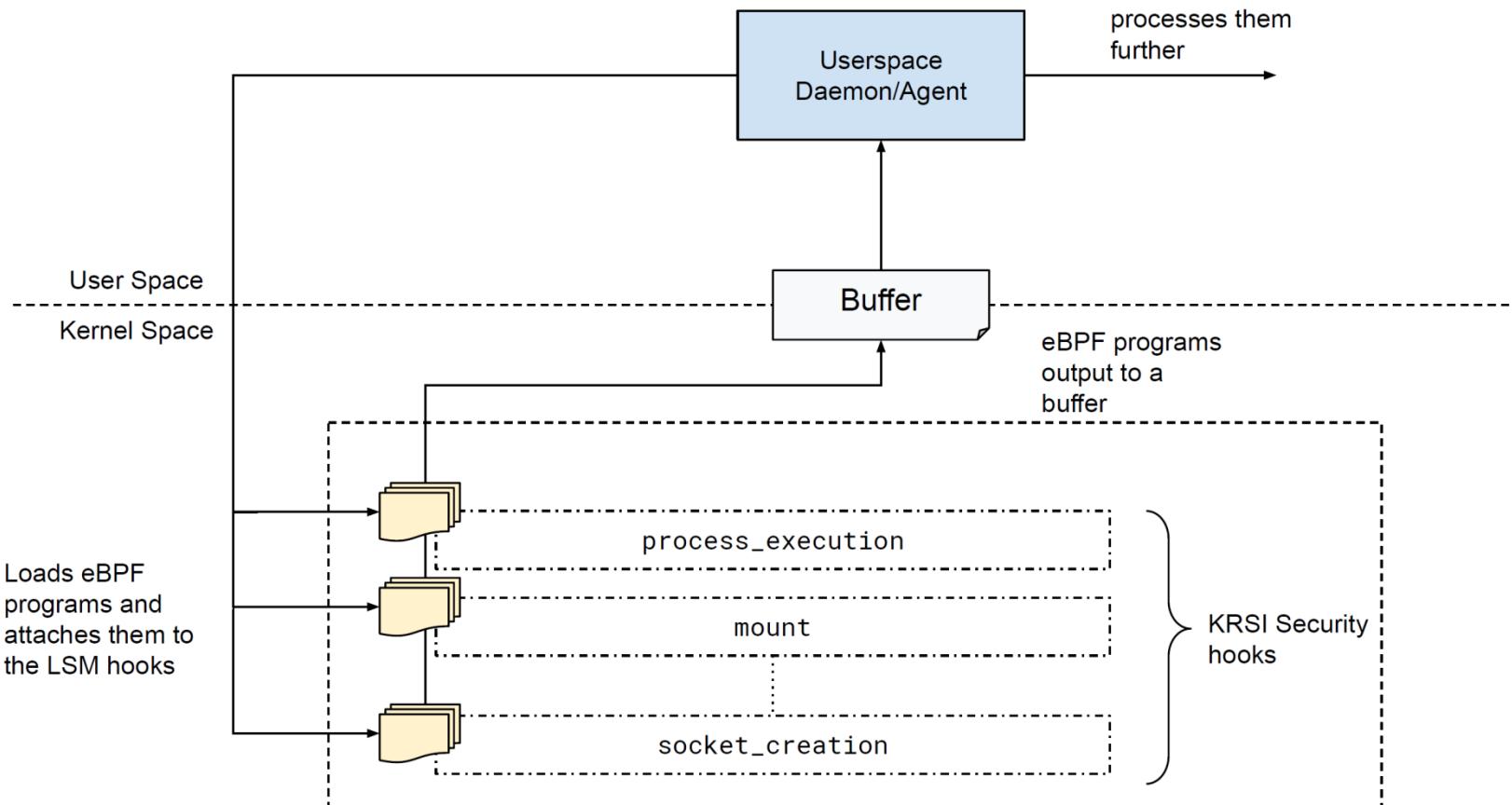


Source: “Kernel Runtime Security Instrumentation”, KP Singh, LPC 2021.

Google



# Tying it all Together



Source: “Kernel Runtime Security Instrumentation”, KP Singh, LPC 2021.



## 2.1.2 Tracee Overview

- Linux Runtime Security and Forensics using eBPF
- <https://www.aquasec.com/products/tracee/>

```
vagrant@ubuntu-hirsute:~/dev/demo$ docker run --rm --privileged --pid=host -v /tmp/tracee:/tmp/tracee -it aquasec/tracee
Loaded signature(s): [TRC-1 TRC-2 TRC-3 TRC-4 TRC-5 TRC-6 TRC-7]

*** Detection ***
Time: 2021-08-01T14:58:46Z
Signature ID: TRC-5
Signature: Fileless Execution
Data: map[]
Command: elfexec
Hostname: ubuntu-hirsute

*** Detection ***
Time: 2021-08-01T14:58:51Z
Signature ID: TRC-2
Signature: Anti-Debugging
Data: map[]
Command: strace
Hostname: ubuntu-hirsute
[]

vagrant@ubuntu-hirsute:~/dev/demo$ ./remote_memfd
Hello, world!
vagrant@ubuntu-hirsute:~/dev/demo$ ./dontlook
Hello World!
vagrant@ubuntu-hirsute:~/dev/demo$
```

### Detect suspicious behavior patterns with Tracee Rules

Curated behavioral indicators identify defense evasion techniques, as defined in the MITRE Att&CK framework, based on the events collected by the Tracee's eBPF engine. Indicators include activity such as fileless execution, anti-debugging and kernel module loading.

[View Tracee on Github >](#)

The most powerful, accurate and reliable eBPF-based detection engine

Tracee has used eBPF since inception and collects 330 syscalls (and other non syscall events) right out of the box. Unlike solutions built on kernel modules, eBPF is safe and fast. And Tracee uses cutting edge eBPF features to prevent evasion by attackers.

#### Blog

Using LSM Hooks with Tracee to Overcome Gaps with Syscall Tracing

[Read the Blog >](#)

```
Time: 2021-10-07T15:07:03Z
Signature ID: TRC-5
Signature: Fileless Execution
Data: map[]
Command: python
Hostname: c6180b7274e6
```

```
*** Detection ***
Time: 2021-10-07T15:07:03Z
Signature ID: TRC-4
Signature: Dynamic Code Loading
Data: map[]
Command: 3
Hostname: c6180b7274e6
```



```
[root@sapling tracee (~ |tadmin@traceecluster:default)]# kubectl create -f deploy/kubernetes/
daemonset.apps/tracee created
configmap/tracee-webhook-config created
deployment.apps/tracee-webhook created
service/tracee-webhook created
[root@sapling tracee (~ |tadmin@traceecluster:default)]# |
```

## Easy deployment, maximum portability and easy integrations

Deploying Tracee with Kubernetes and Docker is a simple “kubectl create” or “docker run” command. Run Tracee with and without BTF support and enjoy maximum portability for different Linux versions using CO:RE mode. Send Tracee data to external notification tools such as Slack or GitHub Actions via projects such as [Postee](#).

[Blog](#)

Tracee: Tracing Containers with eBPF

[Read More >](#)

**Customize and filter intel by relevance and priority**

Use filters to customize where to look for events within specific clusters, containers and hosts. Capture artifacts such as network packets and executables for further analysis only from the most meaningful locations. View insights easily from output templates ranging from JSON files to a GO template for customization.

[View a Tracee Demo >](#)

```
~/dev/tracee/tracee-ebpf/dist
./tracee -t '!container'
TIME(s) UTS_NAME PID TID RET EVENT ARGs
237175.694941 d192e24211d0 0 runc:[2:INIT] 1 /4290 1 /4290 0 execve security_bpmp_check pathname: /bin/busyb
237175.695023 d192e24211d0 0 runc:[2:INIT] 1 /4290 1 /4290 0 execve cap_capable cap: CAP_SYS_ADMIN
237175.695058 d192e24211d0 0 runc:[2:INIT] 1 /4290 1 /4290 0 execve cap_capable cap: CAP_SYS_ADMIN
237175.695060 d192e24211d0 0 runc:[2:INIT] 1 /4290 1 /4290 0 sched_process_exit
237175.696060 d192e24211d0 0 sh: 1 /4290 1 /4290 0
^C
End of events stream
Stats: {eventCounter:5 errorCounter:0 lostEvCounter:0 lostWrCounter:0}
~/dev/tracee/tracee-ebpf/dist
> sudo ./tracee -t '!container'
```

## Relied on for industry-leading threat detection

Tracee is the eBPF engine behind industry-first commercial capabilities of the Aqua Platform such as Dynamic Threat Analysis (DTA), the container sandbox, and Cloud Native Detection and Response (CNDR). DTA, CNDR and Tracee are the only solutions in the industry to combine behavioral indicators from a dedicated cloud native security research team, Nautilus, with eBPF events for real-time threat detection in runtime.

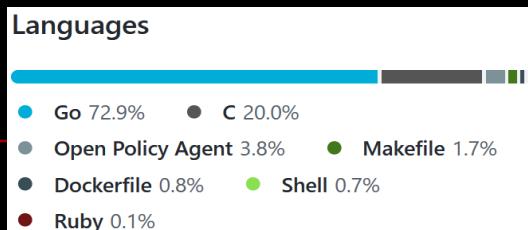
[Blog](#)

Uncover Malware Payload Executions Automatically with Tracee

[Read the Blog >](#)



- <https://aquasecurity.github.io/tracee/latest>
- <https://github.com/aquasecurity/tracee>



- ## Components

**Tracee-eBPF - Linux Tracing and Forensics using eBPF**  
<https://aquasecurity.github.io/tracee/dev/tracee-ebpf/>

...

**Tracee-Rules** - a rule engine to process Tracee-eBPF's events and detect suspicious behavior based on built-in and user-defined signatures

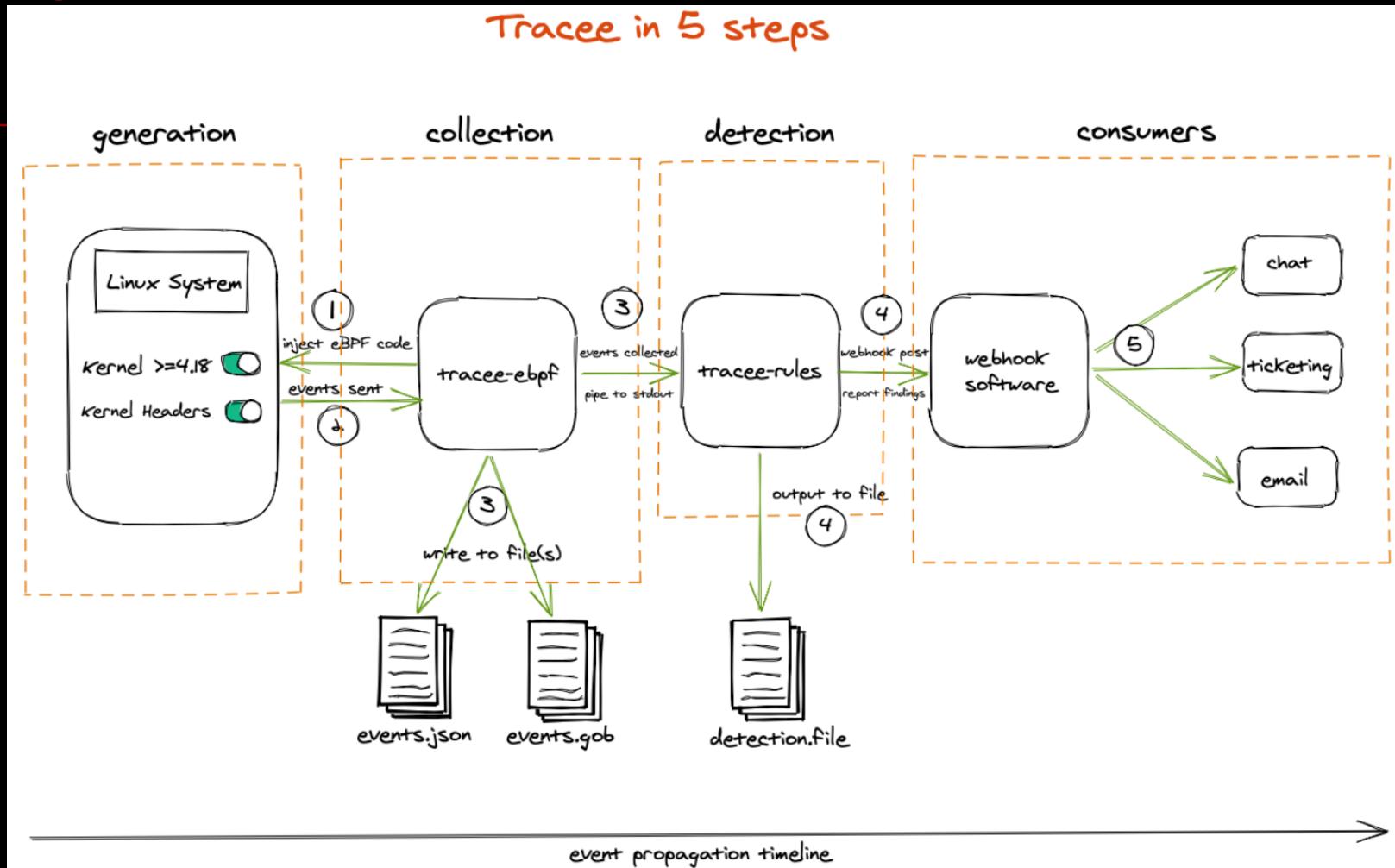
<https://aquasecurity.github.io/tracee/dev/rules-authoring/>  
<https://aquasecurity.github.io/tracee/dev/rules-index/>

...



# Architecture

## ■ High level overview



Source: <https://aquasecurity.github.io/tracee/v0.6.5/architecture/>



## Good Resources

- <https://blog.aquasec.com/open-source-container-runtime-security>
- <https://golangexample.com/tracee-linux-runtime-security-and-forensics-using-ebpf/>
- <https://blog.aquasec.com/ebpf-container-tracing-malware-detection>
- ...



### 3) eBPF in Firewall

#### 3.1 bpfilter

##### Overview

###### ■ BPF based firewall

- 
- <https://lwn.net/Articles/747551/> //BPF comes to firewalls
  - <https://lwn.net/Articles/749108/> //Designing ELF modules
  - <https://lwn.net/Articles/755919/> //Bpfilter (and user-mode blobs) for 4.18
  - <https://lwn.net/Articles/822744/> //Rethinking bpfilter and user-mode helpers

...

###### ■ Patches

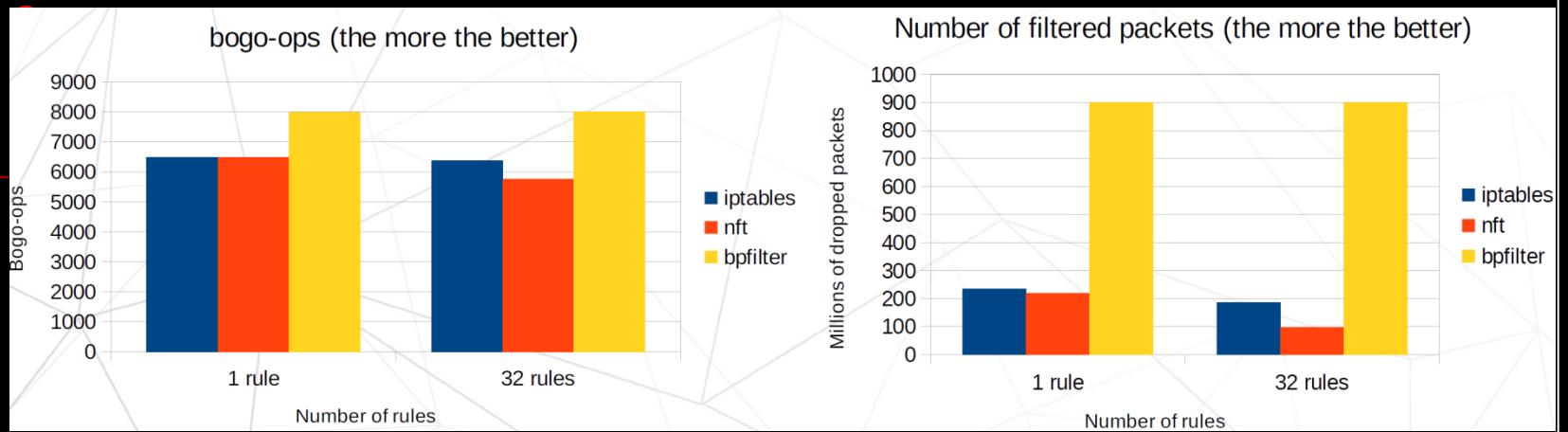
...

<https://lkml.kernel.org/netdev/20210829183608.2297877-12-me@ubique.spb.ru/T/>

###### ■ Reference

<https://lwn.net/Articles/867185/> //Nftables reaches 1.0

## Performance



Source: “**bpfilter – BPF based firewall**”, Dmitrii Banshchikov, **LPC 2021**.



## 3.2 eBPFSnitch

### Overview

- <https://github.com/harporoeder/ebpfsnitch>

### **Linux Application Level Firewall based on eBPF and NFQUEUE.**

eBPFSnitch is a Linux Application Level Firewall based on eBPF and NFQUEUE. It is inspired by [OpenSnitch](#) and [Douane](#) but utilizing modern kernel abstractions - without a kernel module.

The eBPFSnitch daemon is implemented in C++ 20. The control interface is implemented in Python 3 utilizing Qt5.

### **Disclaimer**

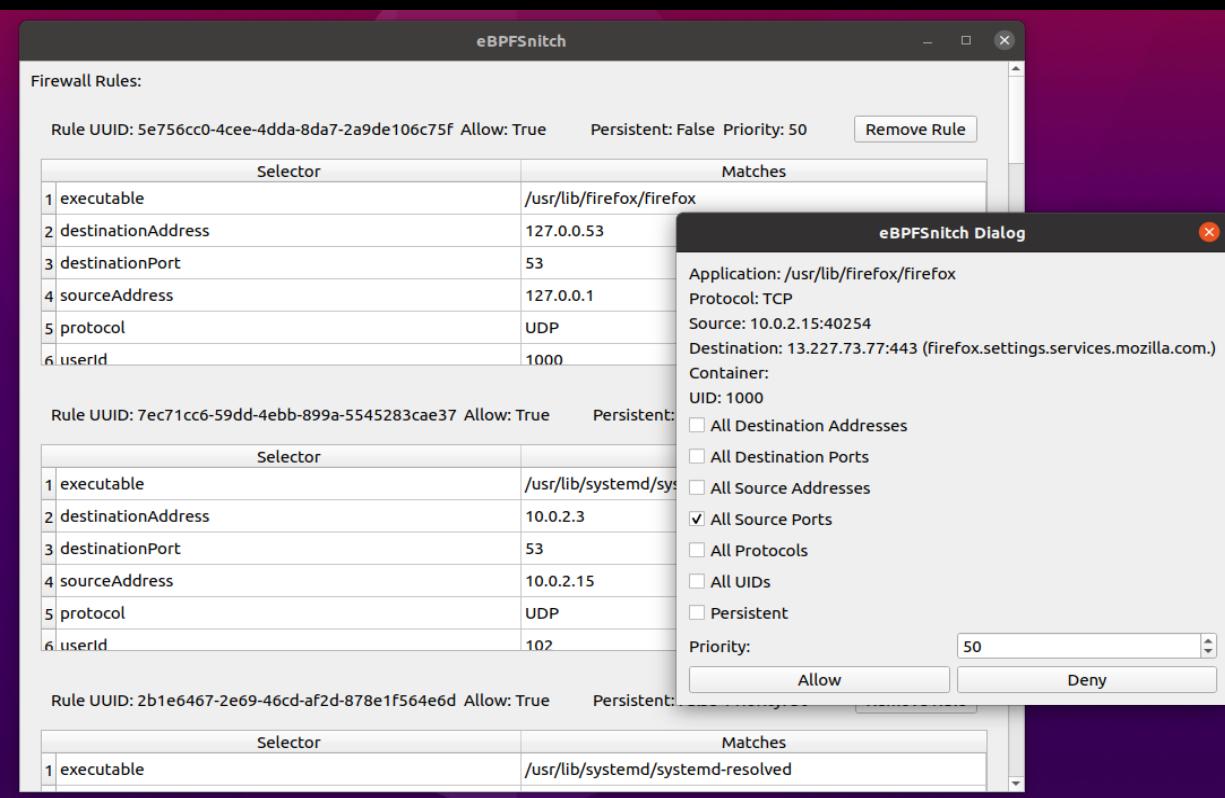
This is an experimental project. The security of this application has not been audited by a 3rd party, or even myself. There are likely mechanisms by which it could be bypassed. Currently the daemon control socket is unauthenticated and an attacker could impersonate the user interface to self authorize.

### **Features**

eBPFSnitch supports filtering all outgoing IPv4 / IPv6 based protocols (TCP / UDP / ICMP / etc). Filtering incoming connections should be supported in the near future.

A core goal of this project is to integrate well with containerized applications. If an application is running in a container that container can be controlled independently of the base system or other containers.

Additionally targeting can occur against specific system users. Blanket permissions for every instance of Firefox for every user are not required.



Source: <https://github.com/harpoeder/ebpfnsnitch/blob/main/screenshot.png>

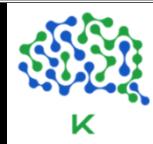
## Languages



## 4) eBPF for Cloud-native Security

### Overview

- ...
- 





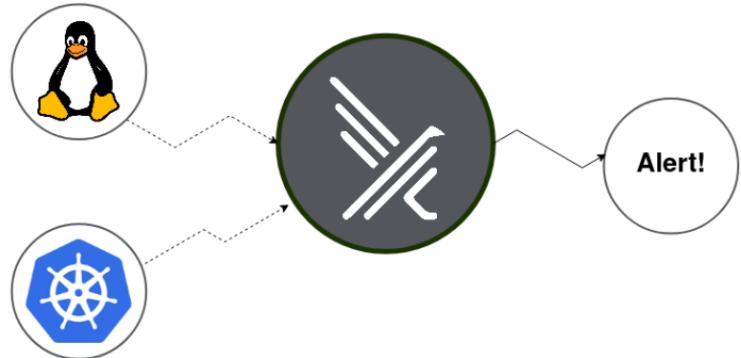
## 4.1 Falco

- <https://falco.org/>

**Falco**, the cloud-native runtime security project, is the de facto **Kubernetes threat detection engine**

Falco is the first runtime security project to join CNCF as an incubation-level project. Falco acts as a security camera detecting unexpected behavior, intrusions, and data theft in real time.

Created by sysdig



## What is it

- <https://falco.org/docs/>

### What does Falco do?

Falco uses system calls to secure and monitor a system, by:

- Parsing the Linux system calls from the kernel at runtime
- Asserting the stream against a powerful rules engine
- Alerting when a rule is violated



## What does Falco check for?

Falco ships with a default set of rules that check the kernel for unusual behavior such as:

- Privilege escalation using privileged containers
- Namespace changes using tools like `setns`
- Read/Writes to well-known directories such as `/etc`, `/usr/bin`, `/usr/sbin`, etc
- Creating symlinks
- Ownership and Mode changes
- Unexpected network connections or socket mutations
- Spawning processes using `execve`
- Executing shell binaries such as `sh`, `bash`, `csh`, `zsh`, etc
- Executing SSH binaries such as `ssh`, `scp`, `sftp`, etc
- Mutating Linux `coreutils` executables
- Mutating login binaries
- Mutating `shadowutil` or `passwd` executables such as `shadowconfig`, `pwck`, `chpasswd`, `getpasswd`, `change`, `useradd`, etc, and others.

## What are Falco rules?

Rules are the items that Falco asserts against. They are defined in the Falco configuration file, and represent the events you can check on the system. For more information about writing, managing, and deploying rules, see Falco [Rules](#).



## What are Falco alerts?

Alerts are configurable downstream actions that can be as simple as logging to `STDOUT` or as complex as delivering a gRPC call to a client. For more information about configuring, understanding, and developing alerts, see [Falco Alerts](#). Falco can send alerts to :

- Standard Output
- A file
- Syslog
- A spawned program
- A HTTP[s] end point
- A client through the gRPC API

## What are the Components of Falco?

Falco is composed of three main components:

- Userspace program - is the CLI tool `falco` that you can use to interact with Falco. The userspace program handles signals, parses information from a Falco driver, and sends alerts.
- Configuration - defines how Falco is run, what rules to assert, and how to perform alerts. For more information, see [Configuration](#).
- Driver - is a software that adheres to the Falco driver specification and sends a stream of system call information. You cannot run Falco without installing a driver. Currently, Falco supports the following drivers:
  - (Default) Kernel module built on `libscap` and `libsinsp` C++ libraries
  - BPF probe built from the same modules
  - Userspace instrumentation

For more information, see [Falco Drivers](#).

## Falco Drivers

### ■ <https://falco.org/docs/event-sources/drivers/>

Falco depends on a driver that taps into the stream of system calls on a machine and passes those system calls to user space.

The kernel module called `falco` is the default driver. Alternatively, an eBPF probe can be used.

The Falco project has three different kind of drivers.

- Kernel module
- eBPF probe
- Userspace instrumentation

...

### eBPF probe

The eBPF probe is an alternative to the one described above. Note that eBPF is a feature supported only by recent kernels.

To install the eBPF probe, please refer to the [Installation](#) page.

To enable the eBPF support in Falco set the `FALCO_BPF_PROBE` environment variable to an empty value (ie. `FALCO_BPF_PROBE=""`) or otherwise explicitly set it to the path where the eBPF probe resides.

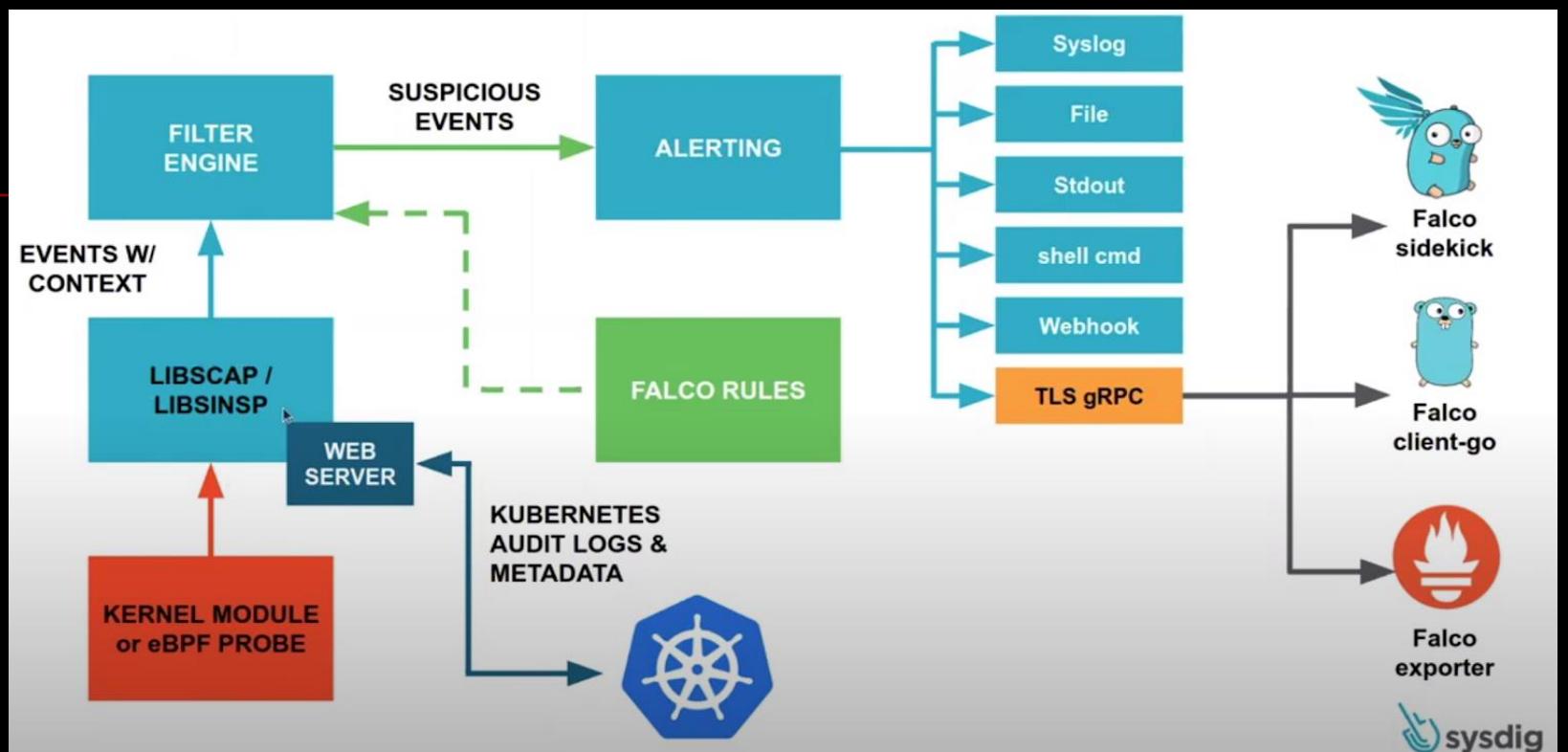
...

### <https://falco.org/docs/getting-started/installation/#install-driver>





## How Falco uses eBPF



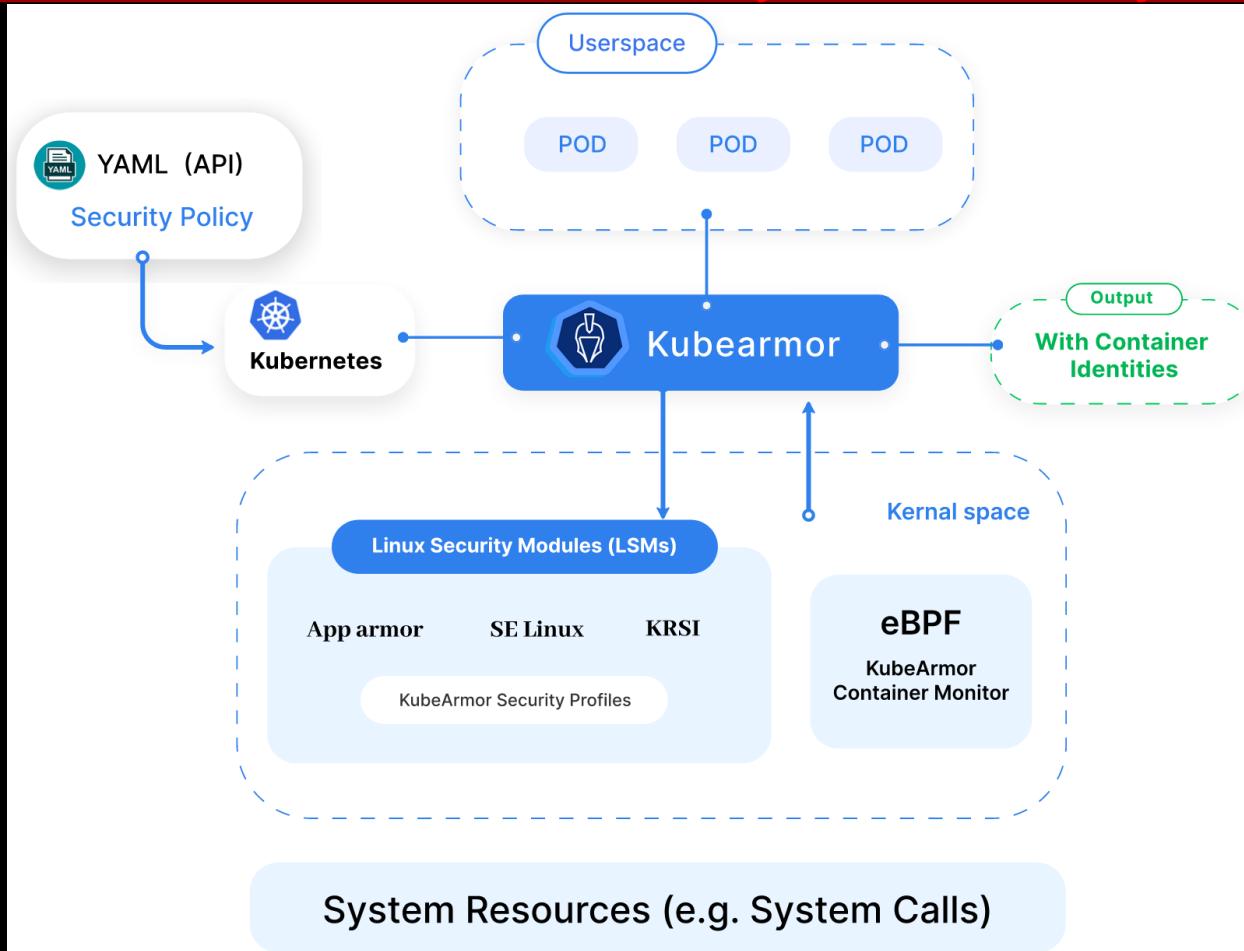
Source: <https://thenewstack.io/ebpf-tools-an-overview-of-falco-inspektor-gadget-hubble-and-cilium/>



## 4.2 KubeArmor

### 4.2.1 Overview

- <https://www.accuknox.com/kubearmor/>  
**Container-aware Runtime Security Enforcement System**





## ■ Key Features



**Restrict the behavior of containers at the system level**  
KubeArmor provides the ability to filter process executions, file accesses, networking operations, and resource utilization inside containers at the system level.



**Provide effortless semantics for policy definitions**  
KubeArmor manages internal complexities associated with LSMs and provides easy semantics for policy definitions.



**Support network security enforcement among containers**  
KubeArmor allows applying policy settings at the level of network system calls, controlling interactions among containers.



**Enforce security policies to containers in runtime**  
KubeArmor directly enforces security policies into Linux Security Modules (LSMs) for each container based on the identities (e.g., labels) of given containers and security policies.



**Produce container-aware alert logs against policy violations.**  
KubeArmor produces alert logs for policy violations that happen in containers by monitoring the operations of containers' processes using its eBPF-based system monitor.



**Provide Kubernetes-native security enforcement engine**  
KubeArmor allows operators to define security policies based on Kubernetes metadata and simply apply them into Kubernetes.



## ■ Sample Policies

```
apiVersion: security.accuknox.com/v1
kind: KubeArmorPolicy
metadata:
  name: ksp-wordpress-config-block
  namespace: wordpress-mysql"
spec:
  severity: 10
  selector:
    matchLabels:
      app: wordpress
  file:
    matchPaths:
      - path: /var/www/html/wp-config.php
  group: alice
  fromSource:
    - path: /usr/sbin/apache2
```

```
apiVersion: security.accuknox.com/v1
kind: KubeArmorPolicy
metadata:
  name: ksp-wordpress-config-block
  namespace: wordpress-mysql"
spec:
  severity: 3
  selector:
    matchLabels:
      app: wordpress
  process:
    matchPaths:
      - path: /usr/bin/apt
      - path: /usr/bin/apt-get
  action: Block
```

```
apiVersion: security.accuknox.com/v1
kind: KubeArmorPolicy
metadata:
  name: ksp-mysql-dir-audit
  namespace: wordpress-mysql"
spec:
  severity: 5
  selector:
    matchLabels:
      app: mysql
  file:
    matchDirectories:
      - dir: /var/lib/mysql/p
        recursive: true
  action: Audit
```

```
apiVersion: security.accuknox.com/v1
kind: KubeArmorPolicy
metadata:
  name: ksp-wordpress-sa-block
  namespace: wordpress-mysql"
spec:
  severity: 8
  tags: ["MITRE"]
  message: "block the k8s credential access"
  selector:
    matchLabels:
      app: wordpress
  file:
    matchDirectories:
      - dir : /run/secrets/kubernetes.io/serviceaccount/
        recursive: true
  action: Block
```



## ■ Github

**<https://github.com/kubearmor/KubeArmor>**

KubeArmor is a cloud-native runtime security enforcement system that restricts the behavior (such as process execution, file access, and networking operation) of containers and nodes at the system level.

KubeArmor operates with [Linux security modules \(LSMs\)](#), meaning that it can work on top of any Linux platforms (such as Alpine, Ubuntu, and Container-optimized OS from Google) if Linux security modules (e.g., [AppArmor](#), [SELinux](#), or [BPF-LSM](#)) are enabled in the Linux Kernel. KubeArmor will use the appropriate LSMs to enforce the required policies.

KubeArmor allows operators to define security policies and apply them to Kubernetes. Then, KubeArmor will automatically detect the changes in security policies from Kubernetes and enforce them to the corresponding containers and nodes.

If there are any violations against security policies, KubeArmor immediately generates alerts with container identities. If operators have any logging systems, it automatically sends the alerts to their systems as well.

### Languages



...



## ***Good Resources***

- <https://docs.kubearmor.com/kubearmor/>
  - ...
-

# XI. System Debugging, Tuning, and Monitoring

1) ...

---

- ...





# 1) eBPF-based Unified Debugger

## 1.1 Drgn

- <https://github.com/osandov/drgn>

### Programmable debugger.

drgn (pronounced "dragon") is a debugger with an emphasis on programmability. drgn exposes the types and variables in a program for easy, expressive scripting in Python. For example, you can debug the Linux kernel:

```
>>> from drgn.helpers.linux import list_for_each_entry
>>> for mod in list_for_each_entry('struct module',
...                                 prog['modules'].address_of_(),
...                                 'list'):
...     if mod.refcnt.counter > 10:
...         print(mod.name)
...
(char [56])"snd"
(char [56])"evdev"
(char [56])"i915"
```

Although other debuggers like [GDB](#) have scripting support, drgn aims to make scripting as natural as possible so that debugging feels like coding. This makes it well-suited for introspecting the complex, inter-connected state in large programs. It is also designed as a library that can be used to build debugging and introspection tools; see the [official tools](#).

- **A debugger for both kernel space and userland**

drgn was developed at [Meta](#) for debugging the Linux kernel (as an alternative to the [crash](#) utility), but it can also debug userspace programs written in C. C++ support is in progress.

- ...



## Getting started

- <https://drgn.readthedocs.io/en/latest/installation.html>
  - [https://drgn.readthedocs.io/en/latest/user\\_guide.html](https://drgn.readthedocs.io/en/latest/user_guide.html)
  - ...
-



## Good Resources

- <https://lwn.net/Articles/878309/> //drgn: How the Linux Kernel Team at Meta Debugs the Kernel at Scale (Meta)
- <https://lwn.net/Articles/789641/> //A kernel debugger in Python
- ...



## 2) eBPF in System and Application Profiling

### 2.1 Pyroscope

- <https://pyroscope.io/>
- <https://github.com/pyroscope-io/pyroscope>

**Continuous Profiling Platform! Debug performance issues down to a single line of code.**

- Find performance issues and bottlenecks in your code
- Resolve issues with high CPU utilization
- Understand the call tree of your application
- Track changes over time

- **Features**

- Can store years of profiling data from multiple applications
- You can look at years of data at a time or zoom in on specific events
- Low CPU overhead
- Efficient compression, low disk space requirements
- Snappy UI

- <https://pyroscope.io/blog/>
- <https://pyroscope.io/docs/ebpf/>
- ...



## Supported Integrations

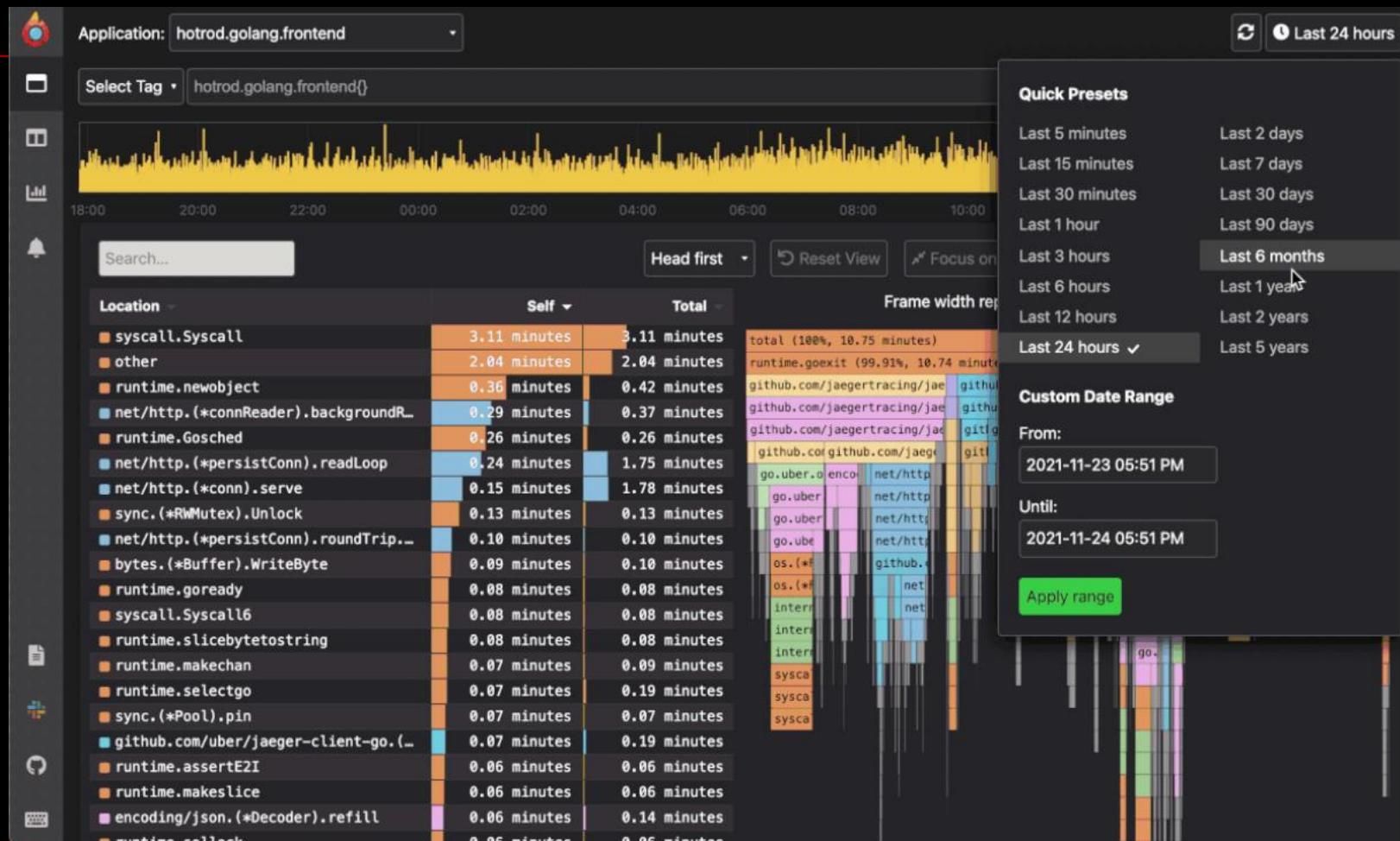
- - Ruby (via `rbspy`)
  - Python (via `py-spy`)
  - Go (via `pprof`)
  - Linux eBPF (via `profile.py` from `bcc-tools`)
  - PHP (via `phpspy`)
  - .NET (via `dotnet trace`)
  - Java (via `async-profiler`)
  - Rust (in progress via `pprof-rs`)
  - Node (seeking contributors)

■ <https://pyroscope.io/docs/supported-integrations>

## Live demo

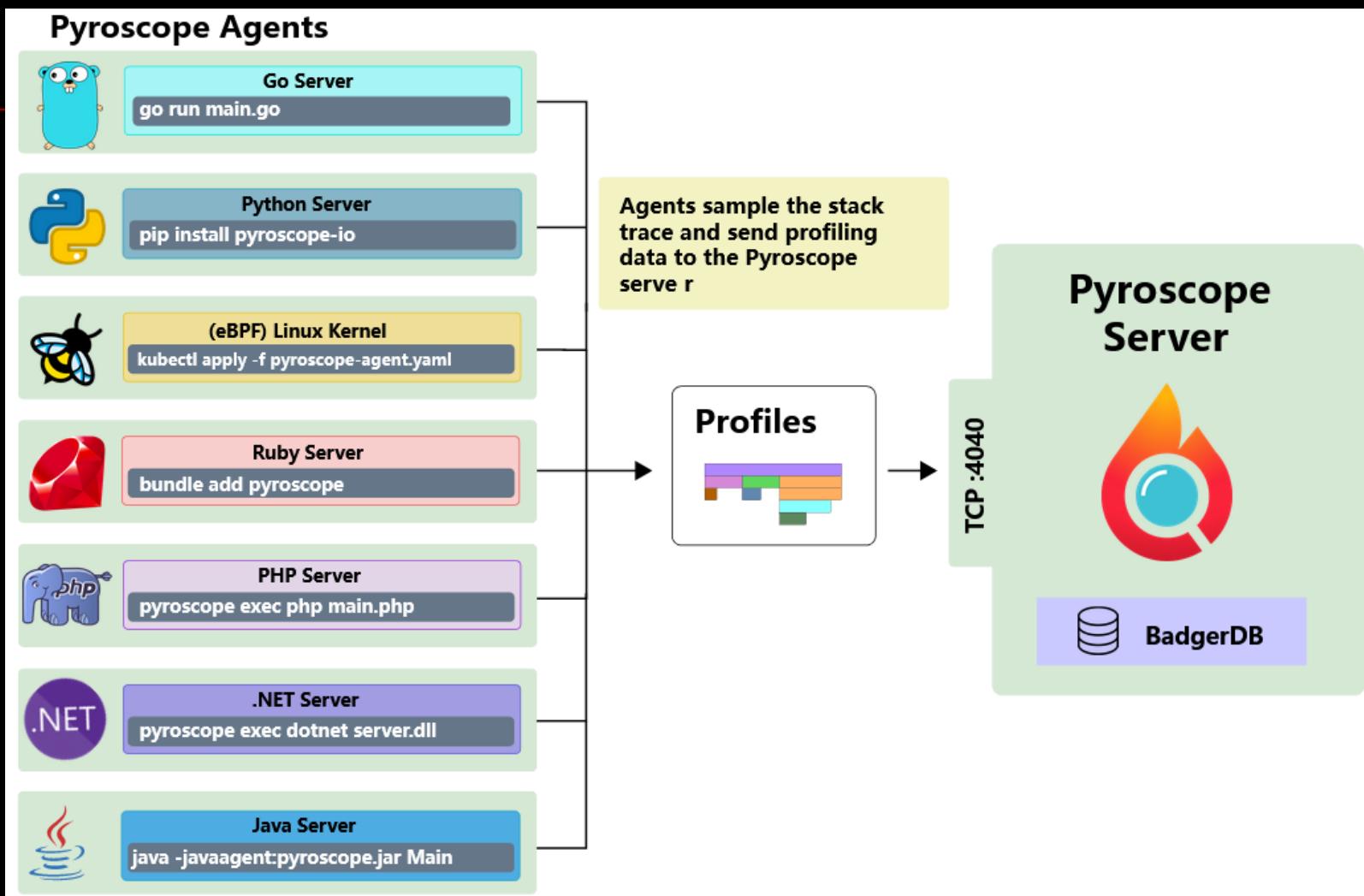
■ <https://demo.pyroscope.io/>

E.g.:



## How it works

- <https://pyroscope.io/docs/>





## 2.2 gProfiler Overview

- <https://gprofiler.io/>

**An open-source, continuous profiler for production – across any environment, at any scale.**

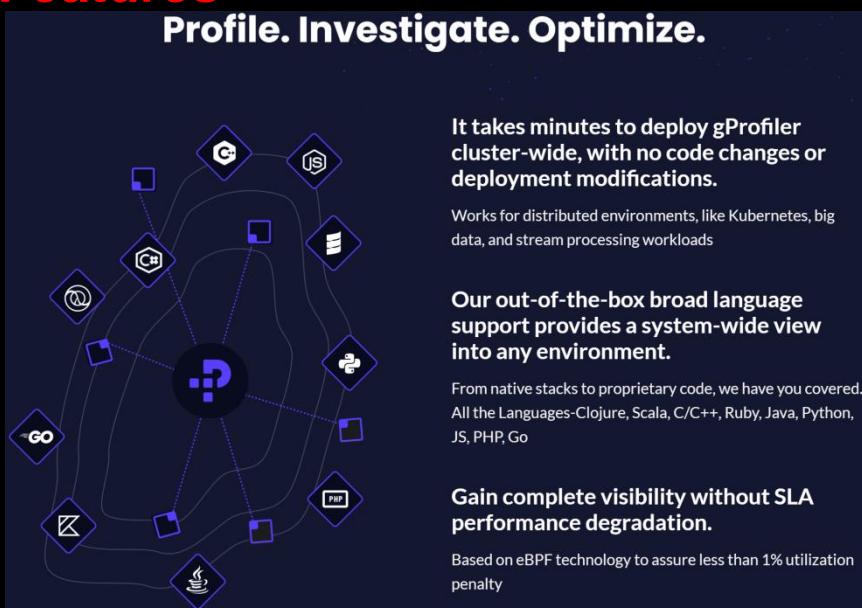
**The must-have tool for performance and cost optimization**

gProfiler enables any team to leverage cluster-wide profiling to investigate performance with minimal overhead.

By continuously analyzing code performance across your entire environment, you can optimize the most resource-consuming parts of your code, improve application performance, and reduce costs.

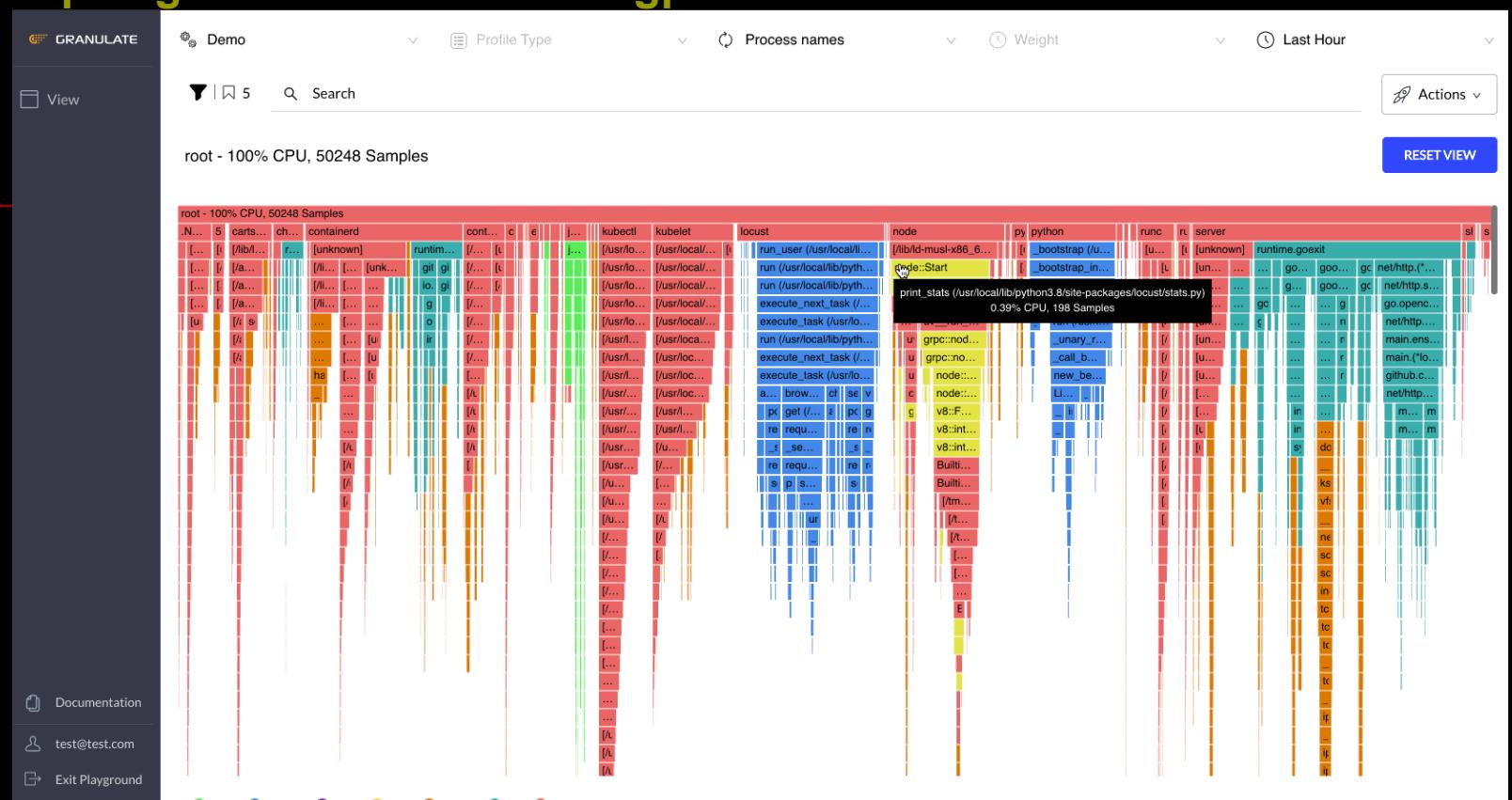
- **Features**

**Profile. Investigate. Optimize.**

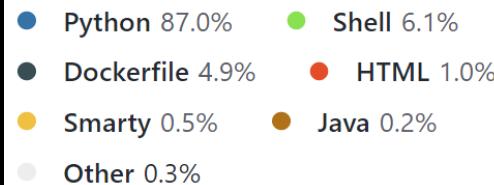




# https://github.com/Granulate/gprofiler



## Languages



## Python profiling options

- `--no-python` : Alias of `--python-mode disabled`.
- `--python-mode` : Controls which profiler is used for Python.
  - `auto` - (default) try with PyPerf (eBPF), fall back to py-spy.
  - `pyperf` - Use PyPerf with no py-spy fallback.
  - `pypy` / `py-spy` - Use pypy.
  - `disabled` - Disable profilers for Python.

Profiling using eBPF incurs lower overhead & provides kernel & native stacks.

### 3) eBPF for System Observability and Monitoring

#### 3.1 Netdata

- <https://www.netdata.cloud/>  
**Monitor everything in real time. For free.**





#### See everything

View the health of your entire IT infrastructure from any browser.



#### Miss nothing

Get per-second performance metrics for every system and app in real time.



#### Stay alert

Get notified of anomalies or outages with preconfigured alarm notifications.



#### Work together

Bring data and teams together for faster, easier troubleshooting.



#### React faster

Speed your time to resolution with tools designed to help identify anomalies faster.



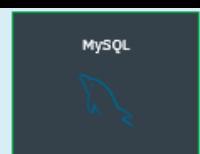
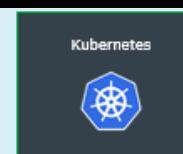
#### Scale smarter

Get operational visibility and insight to help drive growth, not costs.

#### Hundreds of integrations. Thousands of metrics. Zero configuration.

View real-time metrics from your favorite operating systems, hardware, applications, and other monitoring solutions with an always-expanding selection of collectors. Then, send alarms about anomalous behavior or performance degradation to your favorite notification apps.

[View all integrations](#)





■ <https://github.com/netdata/netdata>

## **High-fidelity infrastructure monitoring and troubleshooting. Open-source, free, preconfigured, opinionated, and always real-time.**

Netdata's distributed, real-time monitoring Agent collects thousands of metrics from systems, hardware, containers, and applications with zero configuration. It runs permanently on all your physical/virtual servers, containers, cloud deployments, and edge/IoT devices, and is perfectly safe to install on your systems mid-incident without any preparation.

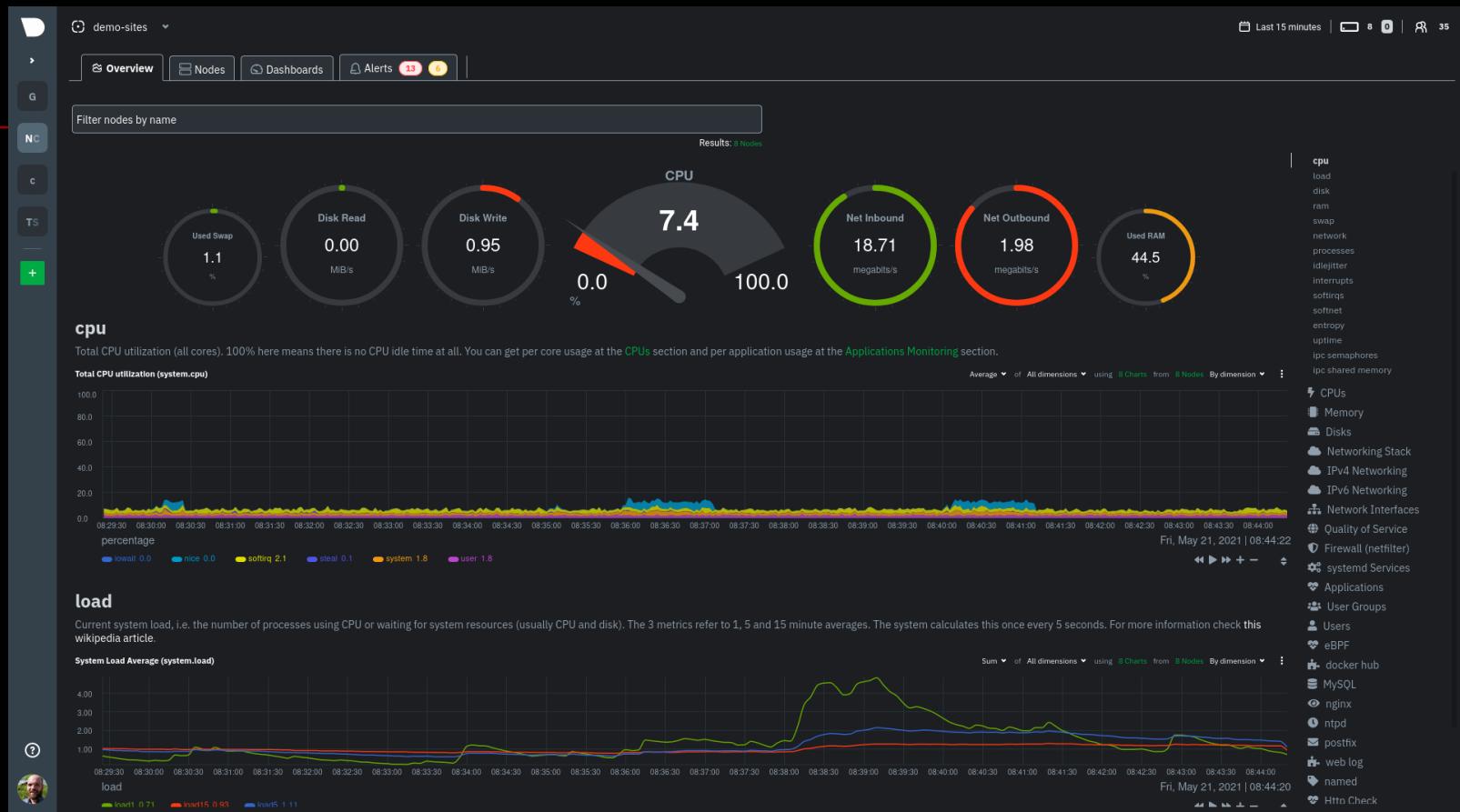
You can install Netdata on most Linux distributions (Ubuntu, Debian, CentOS, and more), container platforms (Kubernetes clusters, Docker), and many other operating systems (FreeBSD, macOS). No `sudo` required.

Netdata is designed by system administrators, DevOps engineers, and developers to collect everything, help you visualize metrics, troubleshoot complex performance problems, and make data interoperable with the rest of your monitoring stack.



# ■ Features

<https://learn.netdata.cloud/docs/overview/why-netdata>





## Here's what you can expect from Netdata:

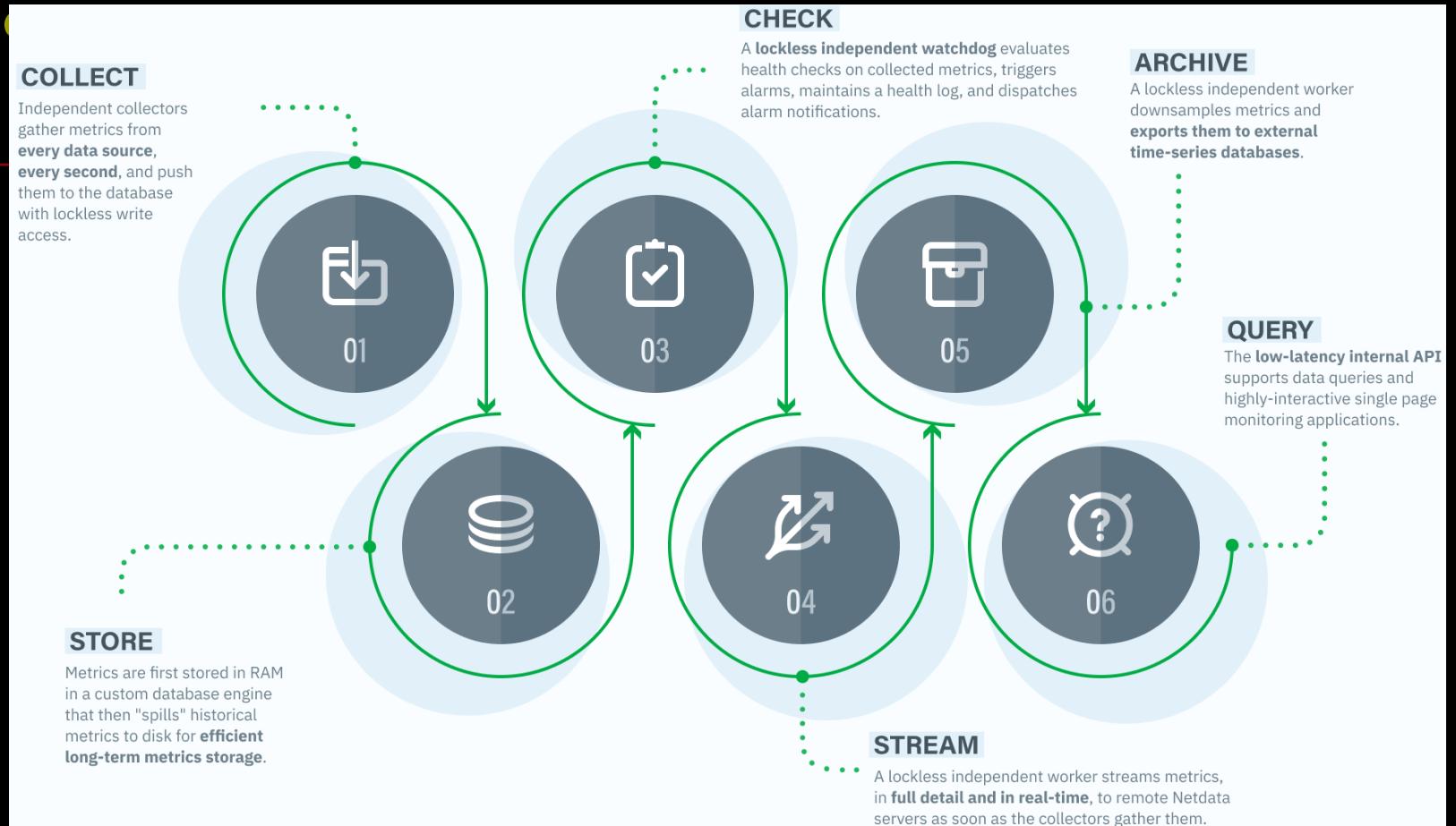
- **1s granularity:** The highest possible resolution for all metrics.
- **Unlimited metrics:** Netdata collects all the available metrics—the more, the better.
- **1% CPU utilization of a single core:** It's unbelievably optimized.
- **A few MB of RAM:** The highly-efficient database engine stores per-second metrics in RAM and then "spills" historical metrics to disk long-term storage.
- **Minimal disk I/O:** While running, Netdata only writes historical metrics and reads `error` and `access` logs.
- **Zero configuration:** Netdata auto-detects everything, and can collect up to 10,000 metrics per server out of the box.
- **Zero maintenance:** You just run it. Netdata does the rest.
- **Stunningly fast, interactive visualizations:** The dashboard responds to queries in less than 1ms per metric to synchronize charts as you pan through time, zoom in on anomalies, and more.
- **Visual anomaly detection:** Our UI/UX emphasizes the relationships between charts to help you detect the root cause of anomalies.
- **Scales to infinity:** You can install it on all your servers, containers, VMs, and IoT devices. Metrics are not centralized by default, so there is no limit.
- **Several operating modes:** Autonomous host monitoring (the default), headless data collector, forwarding proxy, store and forward proxy, central multi-host monitoring, in all possible configurations. Use different metrics retention policies per node and run with or without health monitoring.

## Netdata works with tons of applications, notifications platforms, and other time-series databases:

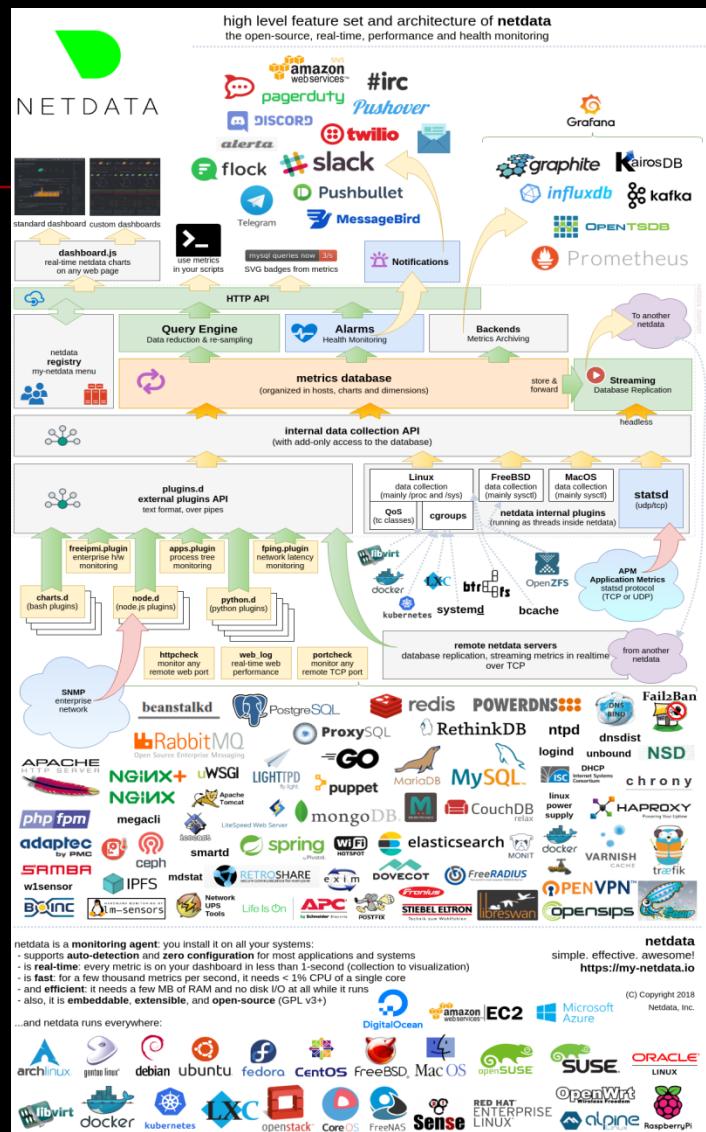
- **300+ system, container, and application endpoints:** Collectors autodetect metrics from default endpoints and immediately visualize them into meaningful charts designed for troubleshooting. See [everything we support](#).
- **20+ notification platforms:** Netdata's health watchdog sends warning and critical alarms to your favorite [platform](#) to inform you of anomalies just seconds after they affect your node.
- **30+ external time-series databases:** Export resampled metrics as they're collected to other [local- and Cloud-based databases](#) for best-in-class interoperability.



# How it works



# Infographic



## eBPF

- <https://learn.netdata.cloud/docs/agent/collectors/ebpf.plugin>
- <https://github.com/netdata/netdata/tree/master/collectors/ebpf.plugin>
- <https://learn.netdata.cloud/guides/troubleshoot/monitor-debug-applications-ebpf>

---

- <https://github.com/netdata/kernel-collector>
- ...



## What's new(v1.33.0)

- <https://github.com/netdata/netdata/releases>

### Highlights

- Netdata is now distributed as pre-built packages on many Linux distributions
- Stream compression (tech preview)
- eBPF CO-RE support

In v1.32 we added some major improvements to our eBPF support. For this release, we're taking the next step by gradually introducing BPF CO-RE support!

Today, the distribution of eBPF programs is very challenging, because trying to compile an eBPF program with so many different Linux kernels is so complex. We want to make eBPF widely available to everyone without worrying about compatibility. And here is where eBPF CO-RE (Compile Once, Run Everywhere), part of libbpf, comes to the rescue.

CO-RE is a modern approach to writing portable BPF applications that can run on multiple kernel versions and configurations without modifications and runtime source code compilation on the target machine. We now have the opportunity to focus on what matters, add more features, and improve performance of our eBPF offering!

Furthermore, in this release we also introduce two new eBPF charts:

- Threads info: Displays the total number of active eBPF threads and the number of all eBPF threads.
- Load info: Measures the number of eBPF threads running on legacy code or CO-RE.





## What's new(v1.32.0)

- <https://github.com/netdata/netdata/releases>

### Highlights

#### New Cloud backend and Agent communication protocol

This Agent release supports our new Cloud backend. From here, we will be offering much faster and simpler communication, reliable alerts and exchange of metadata, and first-time support for the parent-child relationship of Netdata agents. This is the first Agent release that allows Netdata Cloud to use the Netdata Agent as a distributed time-series database that supports replication and query routing, for every metric!

#### eBPF latency monitoring, container monitoring, and more

We use eBPF to monitor all running processes, without the cooperation of the processes and without sniffing data traffic. This new release includes 13 new eBPF monitoring features, including I/O latency, BTRFS, EXT4, NFS, XFS and ZFS latencies, IRQs latencies, extended swap monitoring, and more.

#### Machine learning (ML) powered anomaly detection

This release links Netdata Agent with [dlib](#), the popular C++ machine learning algorithms library, which we use to automatically detect anomalies out-of-the-box, at the edge! Once enabled, Netdata trains an ML model for every metric, which is then used to detect outliers in real-time. The resulting "anomaly bit" (where 0=normal, 1=anomalous) associated with each database entry is stored alongside the raw metric value with zero additional storage overhead! This feature is still in development, so it is disabled by default. If you would like to test it and provide feedback, you can enable the feature using the instructions provided in the [Detailed release highlights](#) section.

# XII. Cloud-native



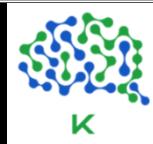
## Overview

- <https://github.com/cncf/toc/blob/main/DEFINITION.md>
  - ...
-

# 1) eBPF for Containers

## 1.1 Overview

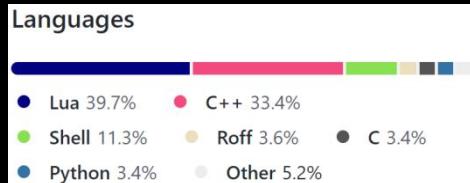
- ...
- 





## 1.2 Sysdig Overview

- <https://sysdig.com/>  
**Security for containers, Kubernetes, and cloud.**
- <https://github.com/draios/sysdig>



- <https://github.com/draios/sysdig/wiki/Sysdig-Overview>

Sysdig is a simple tool for deep system visibility, with native support for containers.

We built sysdig to give you *easy* access to the actual behavior of your Linux systems and containers. Honestly, the best way to understand sysdig is to [try it](#) - its super easy! Or here's a quick video introduction to csysdig, the curses-based UI for sysdig:  
<https://www.youtube.com/watch?v=UJ4wVrbP-Q8>

Far too often, system-level monitoring and troubleshooting still involves logging into a machine with SSH and using a plethora of dated tools with very inconsistent interfaces. And many of these classic Linux tools breakdown completely in containerized environments. Sysdig unites your Linux toolkit into a single, consistent, easy-to-use interface. And sysdig's unique architecture allows deep inspection into containers, right out of the box, without having to instrument the containers themselves in any way.

Sysdig provides additional value by focusing on a few key principles:

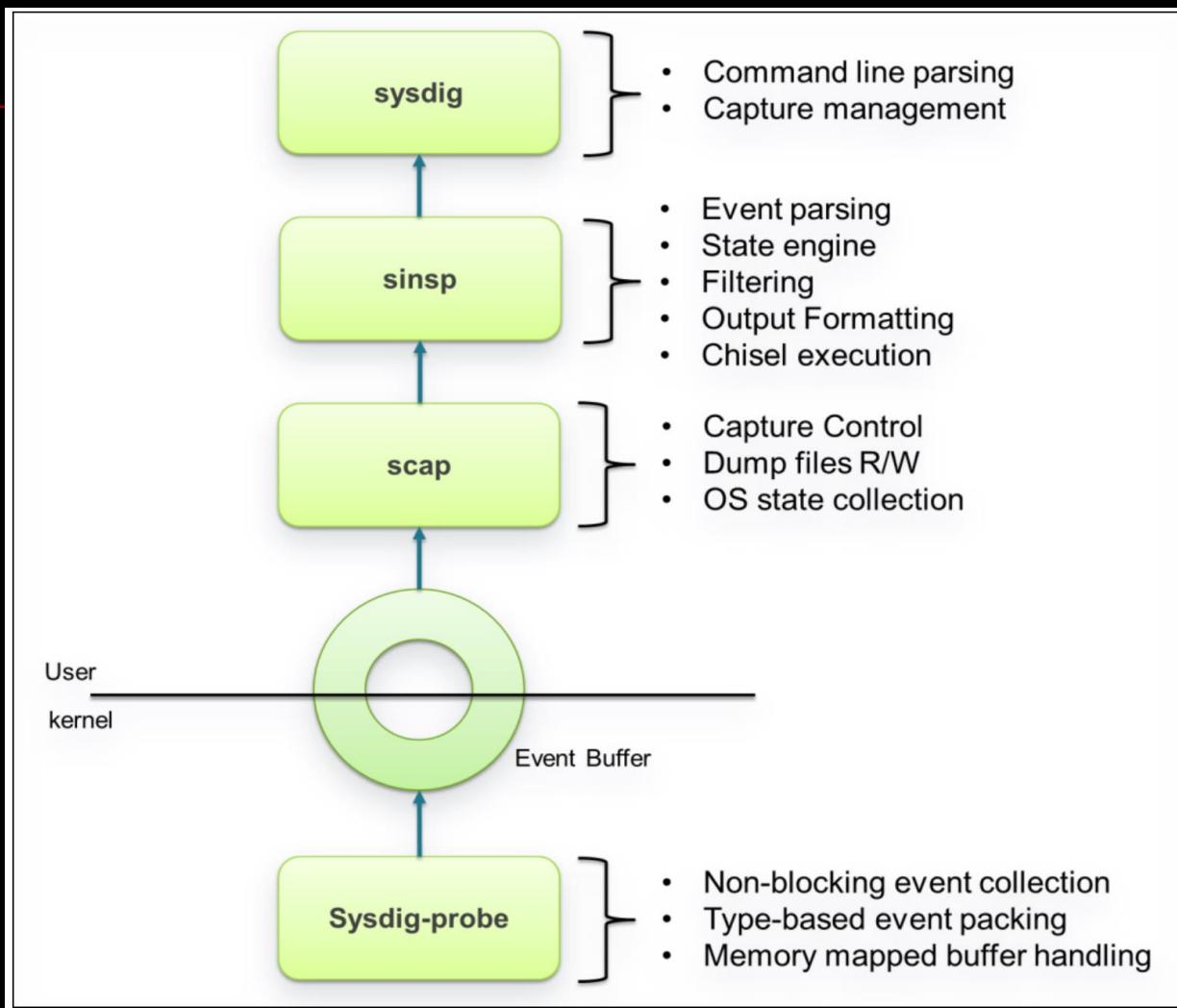
- offering native support for all Linux container technologies, including Docker, LXC, etc
- offering unified, coherent, and granular visibility into the storage, processing, network, and memory subsystems
- making it possible to create trace files for system activity, similarly to what you can do for networks with tools like tcpdump and Wireshark, so that the problem can be analyzed at a later time, without losing important information
- including rich system state in the trace files, so that the captured activity can be put into full context
- offering a filtering language to dig into the information in a natural and interactive way
- including a rich library of Lua scripts to solve common problems, which we call [chisels](#) (to carve up the data you unearthed... get it?).
- offering an simple, intuitive, and fully customizable curses-based UI called [csysdig](#)

Think about sysdig as strace + tcpdump + htop + iftop + lsof + ...awesome sauce.



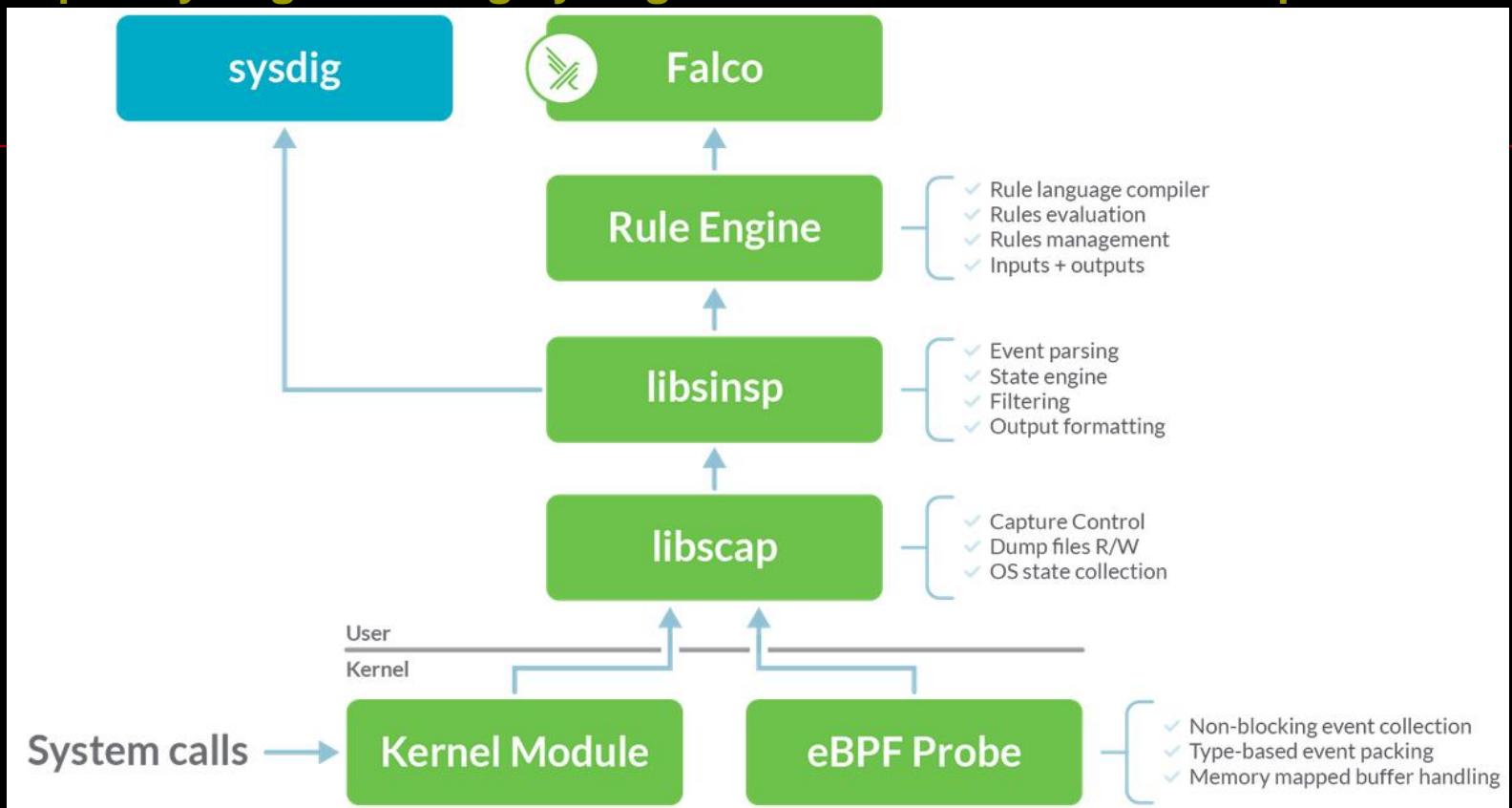
## How it works

- <https://sysdig.com/blog/sysdig-vs-dtrace-vs-strace-a-technical-discussion/>



## Gontributes Falco's kernel module, eBPF probe, and libs to the CNCF

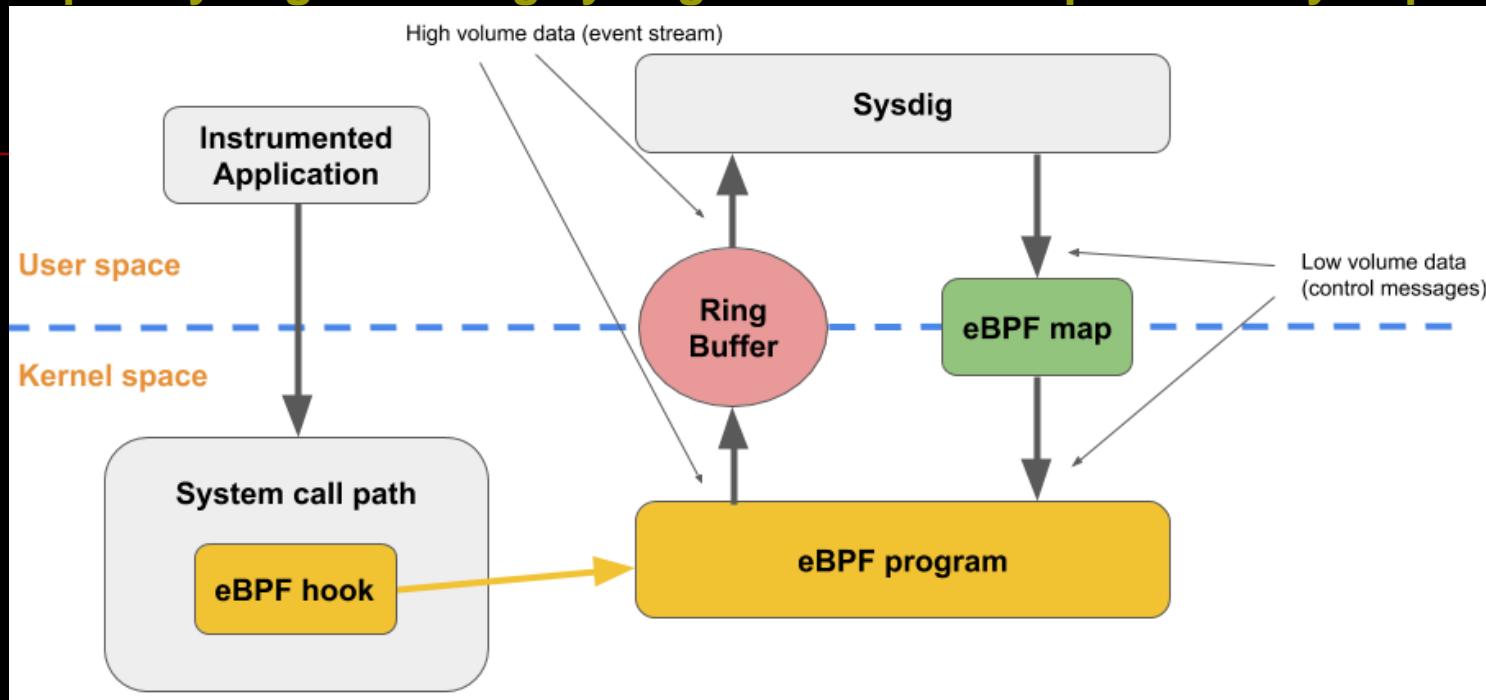
- <https://sysdig.com/blog/sysdig-contributes-falco-kernel-ebpf-cncf/>





## sysdig + eBPF

- <https://sysdig.com/blog/sysdig-and-falco-now-powered-by-ebpf/>



- <https://sysdig.com/blog/the-art-of-writing-ebpf-programs-a-primer/>
- <https://github.com/draios/sysdig/wiki/eBPF>
- ...



## Good Resources

- <https://sysdig.com/blog/>
- <https://github.com/draios/sysdig/wiki>
- <https://opensourcesecurity.io/2021/03/14/episode-262-a-discussion-with-loris-and-pop-from-sysdig/>
- ...

## 2) eBPF in Kubernetes Ecosystem

### 2.1 Overview

- ...
- 



## 2.2 Pixie Overview

- <https://px.dev/>

Open source Kubernetes observability for developers.



The screenshot displays the Pixie UI interface. At the top, there's a navigation bar with tabs for 'Script', 'Live View', and 'Pod List'. The 'Live View' tab is active, showing a service graph for the 'online-boutique' namespace in the 'gke\_skylab1' cluster. The graph shows nodes for various services like 'frontend', 'recommendationservice', and 'cartservice', connected by bidirectional arrows representing traffic flow. Below the graph, there's a 'Service List' table with columns for service, latency\_ms (p99), requests\_per\_s, error\_rate\_pct, inbound\_bytes\_per\_s, and outbound\_bytes\_per\_s. The table includes entries for 'online-boutique/frontend-external', 'online-boutique/checkoutservice', 'online-boutique/productcatalogservice', and 'online-boutique/shippingservice'. On the left side, there's a sidebar with sections for 'Outgoing Traffic' and 'Status Code'. The bottom of the screen shows a footer with links for 'Underlying Data', 'Execution Stats', and the 'PIXIE' logo.

service	latency_ms (p99)	requests_per_s	error_rate_pct	inbound_bytes_per_s	outbound_bytes_per_s
online-boutique/frontend-external, online-boutique/frontend	1226.45	2.01	0	2.01	801.47
online-boutique/checkoutservice	292.65	0.23	0	19.55	27.08
online-boutique/productcatalogservice	0.62	9.79	0	130.07	2899
online-boutique/shippingservice	0.63	0.69	0	54.76	25.94

- Key Features  
Auto-instrumented. Scriptable. Kubernetes native.



PIXIE CLI    PIXIE LIVE

## No Instrumentation

Access metrics, events, traces and logs in seconds without changing code via dynamic eBPF probes and ingestors. Add logging only for custom data.

[Read More](#)

px deploy

Running Cluster checks

- ✓ Kernel version > 4.14
- ✓ K8s version > 1.12.0
- ✓ Kubectl v1.18.0 is present
- ✓ Docker v19.03.12 is present

Deploying to Pixie

Deploying Pixie to t

Is the cluster correct? Found 5 nodes

Installing version

- ✓ Creating namespace
- ✓ Deleting stale Pixie
- ✓ Pulling image
- ✓ Loading secrets
- ✓ Downloading Vizier
- ✓ Deploying NATS
- ✓ Deploying Cloud C
- ✓ Waiting for Cloud C
- ✓ Initiating deploy
- ✓ Waiting for state
- ✓ Wait for PEMs/Kel
- ✓ Wait for healthcheck

Pixie is Live!

Visit: <https://work>.

Run some scripts using:

- px script list : t
- px run px/service, coming soon!)

### Namespace Overview

cluster: gke\_skylab-us-west1-a\_skylab1%20start\_time=t

Services\_List

service	latency	reqs	rate	inbytes	outbytes
online-boutiquefr...	2.13	0	2.13	955.29	
online-boutiqueus...	0.10	0	0.10	25.60	
online-boutiqueuk...	0.77	0	79.81	79.81	
online-boutiquecn...	0.10	0	0.10	25.60	
online-boutiquebr...	11.97	0	160.07	3538.08	
online-boutiqueid...	0.90	0	0.90	69.43	
online-boutiqueau...	0.10	0	94.04	54.47	

Browsing 7 of 7 records

### Pod List

pod	create_time	status
online-boutiquefr-pod-0000000000-0qjkn	7/31/2020, 3:36:05 PM	green
online-boutiqueus-pod-0000000000-8nq2z	7/31/2020, 3:38:05 PM	green
online-boutiqueuk-pod-0000000000-1bf72	7/31/2020, 3:38:04 PM	green
online-boutiquecn-pod-0000000000-64t4d-09-74-1ba-4v	7/31/2020, 3:38:05 PM	green

Underlying Data: Pod List Namespace Service Graph Services List Execution Stats **PIXIE**

**</> PXL SCRIPTS**

https://work.withpixie.ai/live/clusters/gke\_pixie-skylab\_us-west1-a\_skylab1%20start\_time=t

### Basic service SLAs

script: skylab/service\_stats svc: px-sock-shop start:=10m

PXL Script

```
1 def service_qps(start_time: str, service: str):
2     px.Service():
3
4         requests_dataframe =
5             query_http_table(start_time)
6
7         qps_dataframe =
8             calc_http_qps(requests_dataframe,
9                         [service, 'timestamp'])
10
11    return qps_dataframe
```

VIS Spec

### Request per Second

Request per Second

3:05:20 pm    3:13:30 pm

px-sock-shop/foreground: 94.70  
px-sock-shop/user: 80.80

Run community, team or custom scripts to debug as code. Publish and share your sessions as code with your team and global Pixienaut community.

[Read More](#)

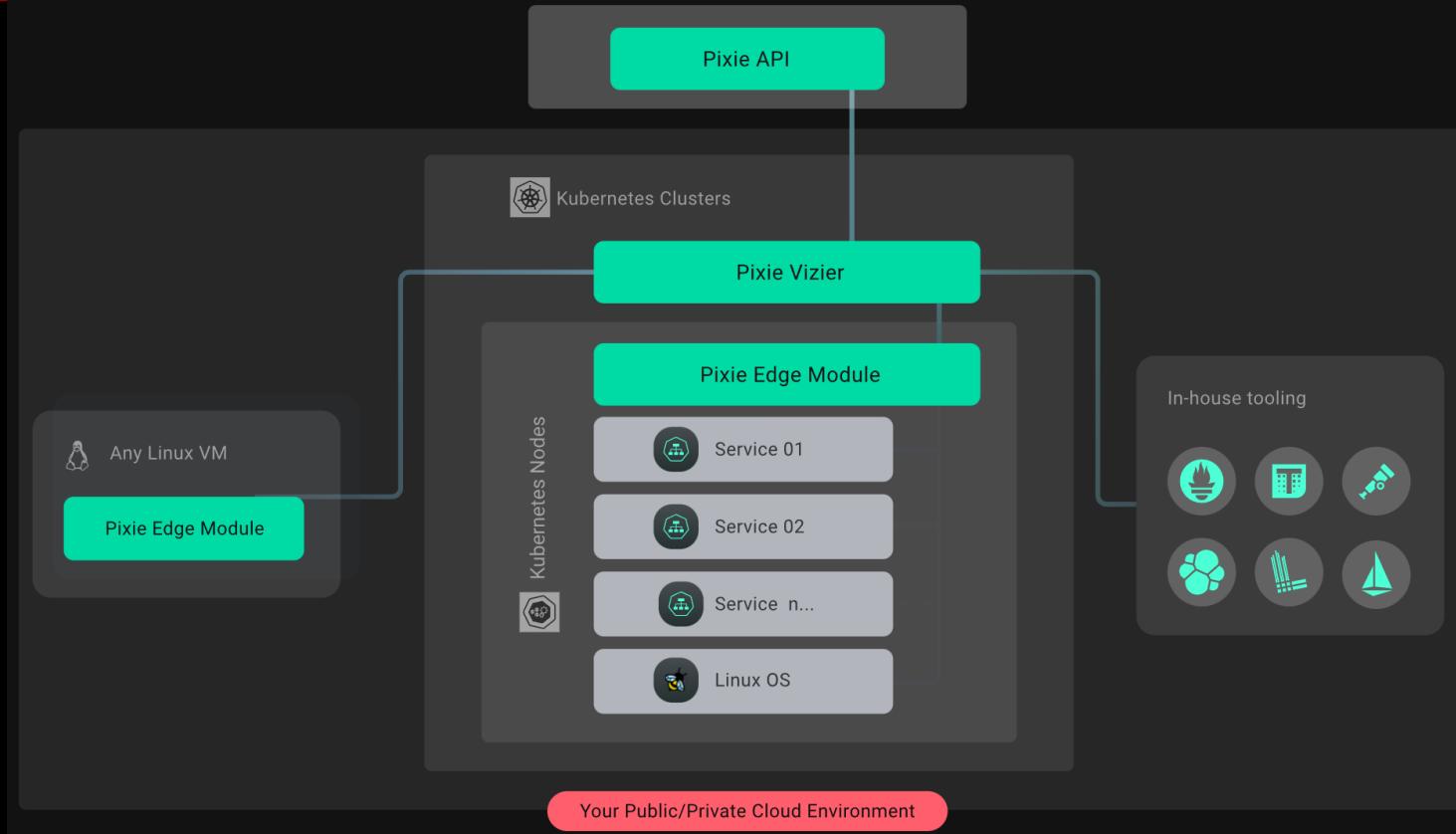
## Debug with Scripts



## Kubernetes Native

Pixie runs entirely inside your Kubernetes clusters without storing any customer data outside. Avoid trading-off depth of visibility due to the hassle and cost of trucking petabytes of telemetry off-cluster.

[Read More](#)





## ■ Supported Protocols

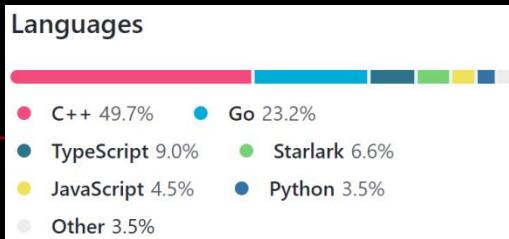
<https://docs.px.dev/about-pixie/data-sources/>

Pixie automatically traces the following protocols:

Protocol	Support	Notes
HTTP	Supported	
HTTP2/gRPC	Partially Supported	Currently only for Golang apps with <a href="#">debug information</a> .
DNS	Supported	
NATS	Supported	Requires a NATS build with <a href="#">debug information</a> .
MySQL	Supported	
PostgreSQL	Supported	
Cassandra	Supported	
Redis	Supported	
Kafka	Supported	

# Src

- <https://github.com/pixie-io/pixie>



- <https://github.com/pixie-io/jattach>

## JVM Dynamic Attach utility

The utility to send commands to remote JVM via Dynamic Attach mechanism.

All-in-one jmap + jstack + jcmt + jinfo functionality in a single tiny program.  
No installed JDK required, works with just JRE. Supports Linux containers.

This is the lightweight native version of HotSpot Attach API

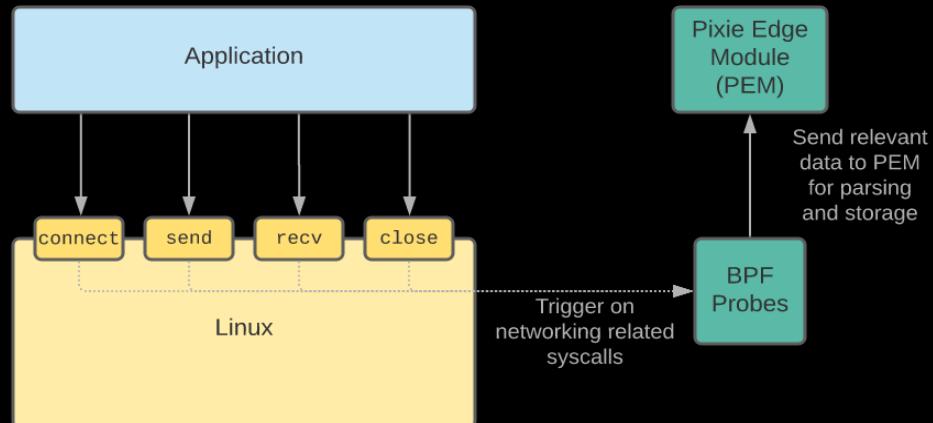
<https://docs.oracle.com/javase/8/docs/jdk/api/attach/spec/>

- <https://github.com/pixie-io/grafana-plugin>
- <https://github.com/pixie-io/pxapi.go>
- <https://github.com/pixie-io/pixie-demos>
- ...

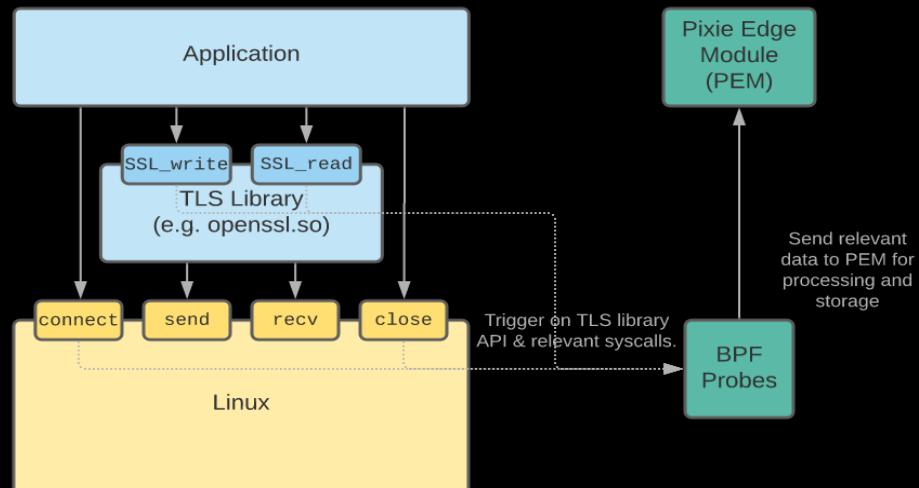


## eBPF in Pixie

- <https://docs.px.dev/about-pixie/pixie-ebpf/>  
**Protocol Tracing**

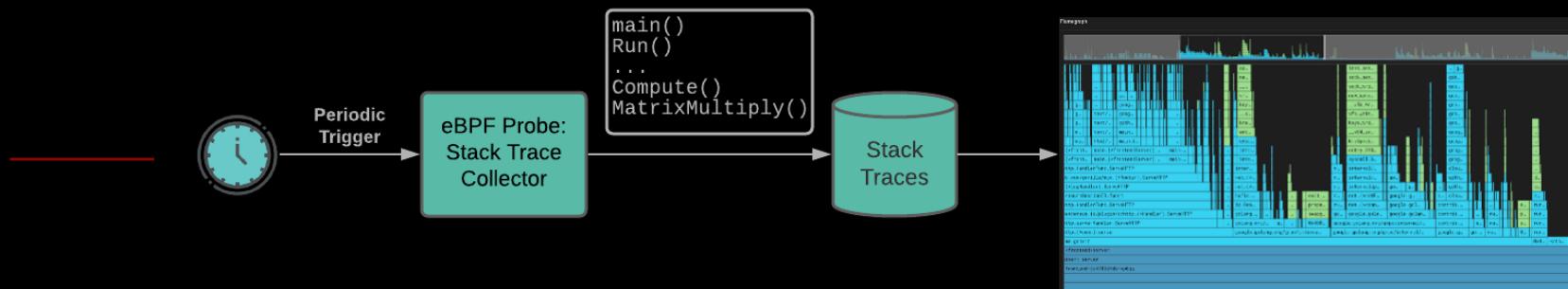


## Tracing TLS/SSL Connections





# Application CPU Profiling



## Distributed bpftrace Deployment

Distributed bpftrace Deployment

**DEMO**

Distributed bpftrace deployment using Pixie

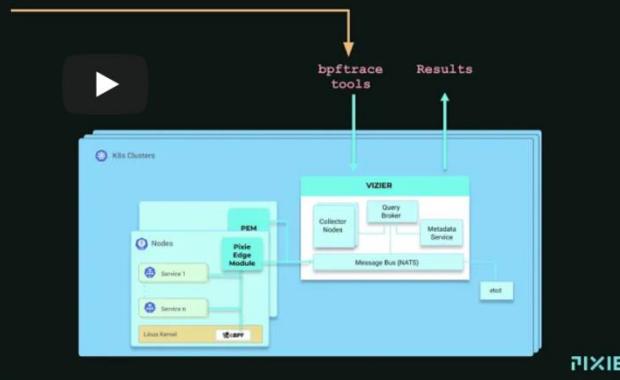
稍后观看 分享

Omid Azizi Yaxiong Zhao

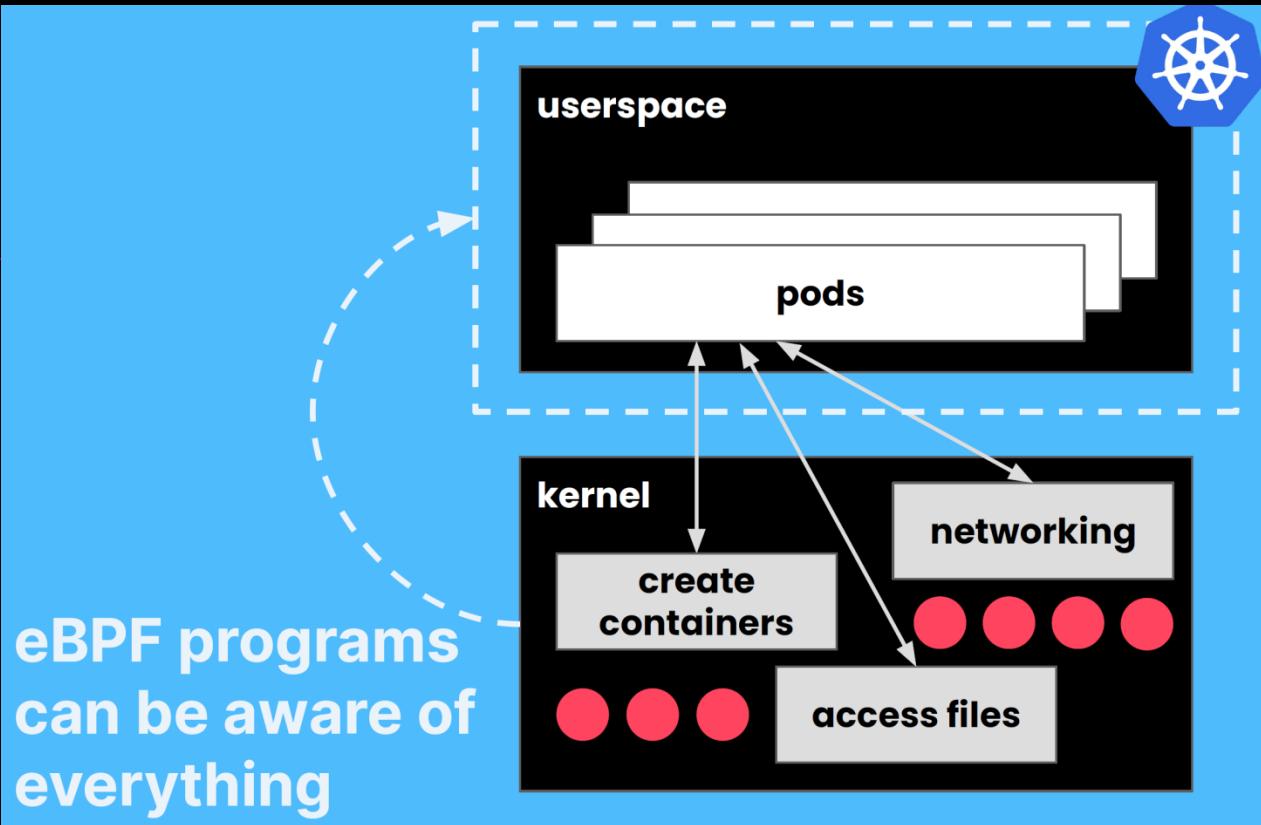
<https://github.com/iovisor/bpftrace>

- tools/bashheadline.bt: Print entered bash commands system wide. Examples.
- tools/bislatency.bt: Block I/O latency as a histogram. Examples.
- tools/biosnoop.bt: Block I/O tracing tool, showing per I/O latency. Examples.
- tools/biosicks.bt: Show disk I/O latency with initialization stacks. Examples.
- tools/bitsizez.bt: Show disk I/O size as a histogram. Examples.
- tools/capable.bt: Trace security capability checks. Examples.
- tools/cpupiuk.bt: Sample which CPUs are executing processes. Examples.
- tools/dtcsnoop.bt: Trace directory entry cache (dcache) lookups. Examples.
- tools/execnoop.bt: Trace new processes via execve syscalls. Examples.
- tools/getrstatusname.bt: Show latency for getraddrinfo/gethostbyname[2] calls. Examples.
- tools/killnsnoop.bt: Trace signals issued by the kill() syscall. Examples.
- tools/loads.bt: Print load averages. Examples.
- tools/mrfush(bt: Trace md flush events. Examples.
- tools/naptme.bt: Show voluntary sleep calls. Examples.
- tools/openinno(bt: Trace open() syscalls showing filenames. Examples.
- tools/oomkill.bt: Trace OOM killer. Examples.
- tools/pidpersec.bt: Count new processes (via fork). Examples.
- tools/rundqbt: CPU scheduler run queue latency as a histogram. Examples.
- tools/runlenbt: CPU scheduler run queue length as a histogram. Examples.
- tools/suidbs: Trace the setuid syscalls: privilege escalation. Examples.
- tools/istatnsop.bt: Trace stat() syscalls for general debugging. Examples.
- tools/memmon.bt: Show memory by processes. Examples.
- tools/memmon(bt: Trace memory pages of syscalls. Examples.

通过以下平台观看:



<https://docs.px.dev/tutorials/custom-data/distributed-bpftrace-deployment/>



Source: “Cloud Native Superpowers with eBPF”, Liz Rice, KubeCon & CloudNativeCon NA 2021.



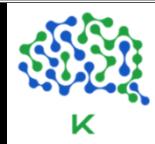
## Good Resources

- <https://blog.px.dev/>
- <https://pixielabs.ai/>
- <https://docs.px.dev/reference/pxl/>
- <https://docs.px.dev/using-pixie/>
- <https://blog.px.dev/cpu-profiling/>
- <https://docs.px.dev/tutorials/custom-data/dynamic-go-logging/>
- <https://aws.amazon.com/blogsopensource/gathering-insights-on-kubernetes-applications-services-and-network-traffic-with-pixie/>
- ...

## 3) eBPF in Microservices

### 3.1 Overview

- ...
- 





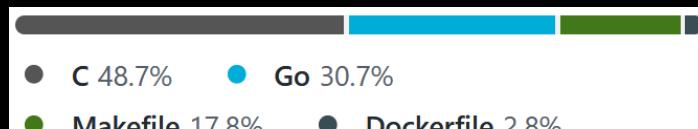
## 3.2 eBPF for Service Mesh

### 3.2.1 Merbridge

- <https://github.com/merbridge/merbridge>

Use eBPF to speed up your Service Mesh like crossing an Einstein-Rosen Bridge.

- Language

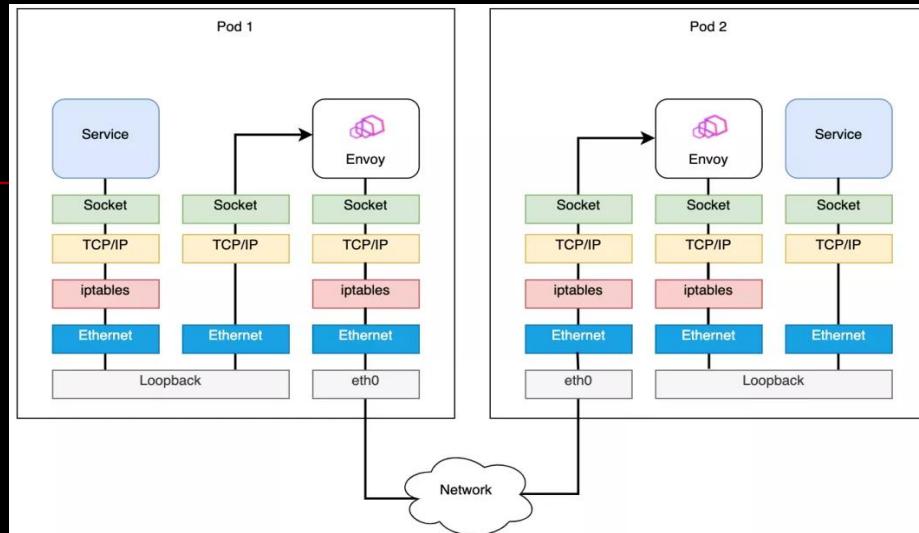


- ...

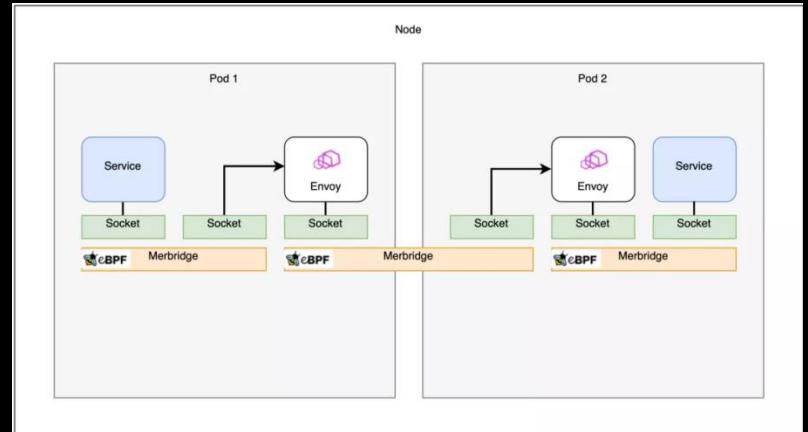
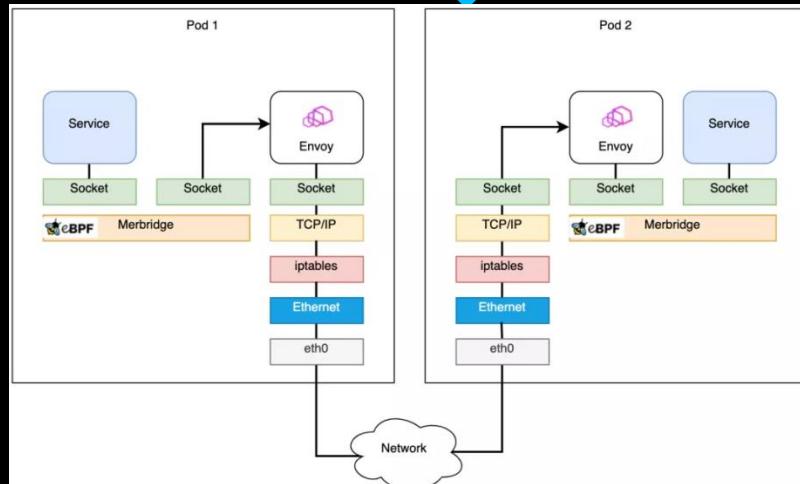


## Architecture & Design

■



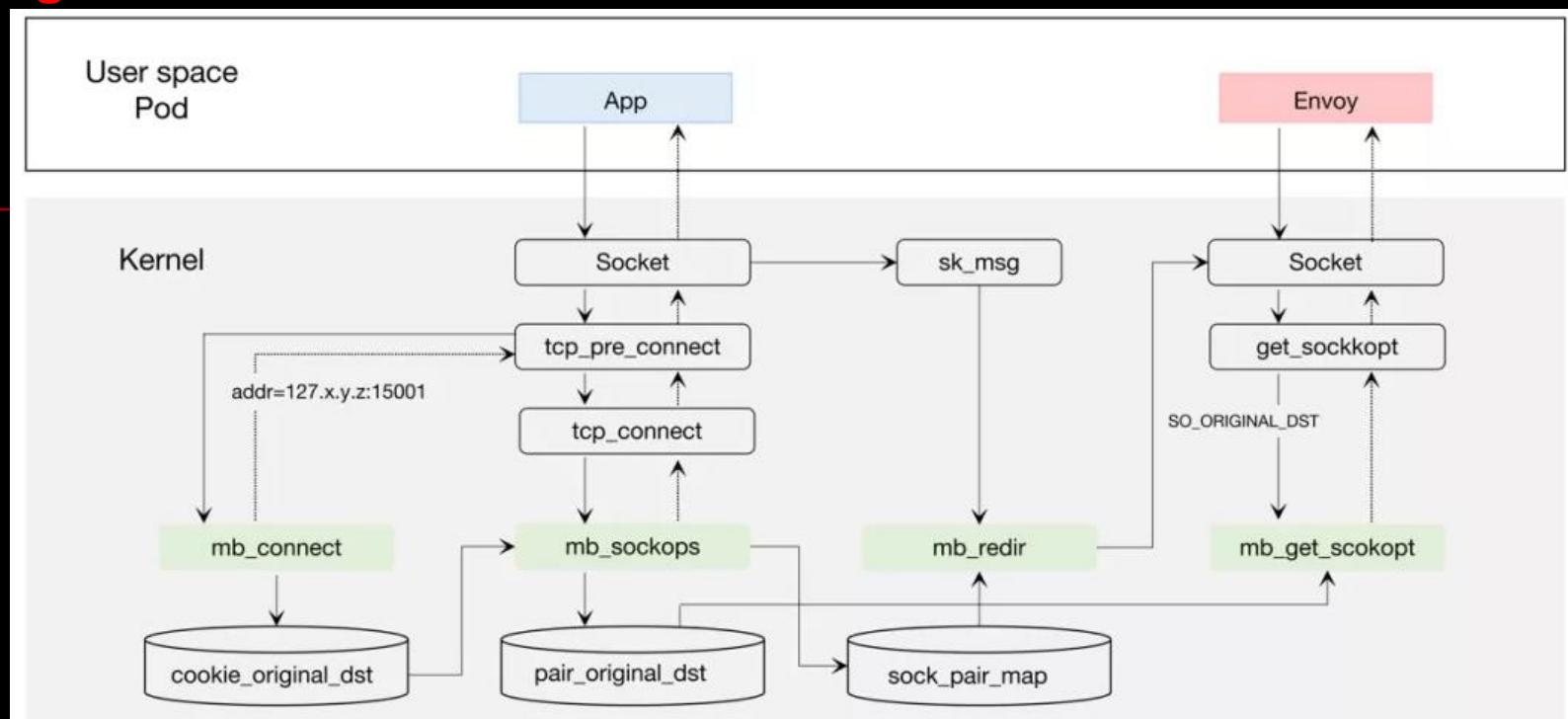
Or if the two pods on the same machine



Source: <https://www.daocloud.io/posts/7949>



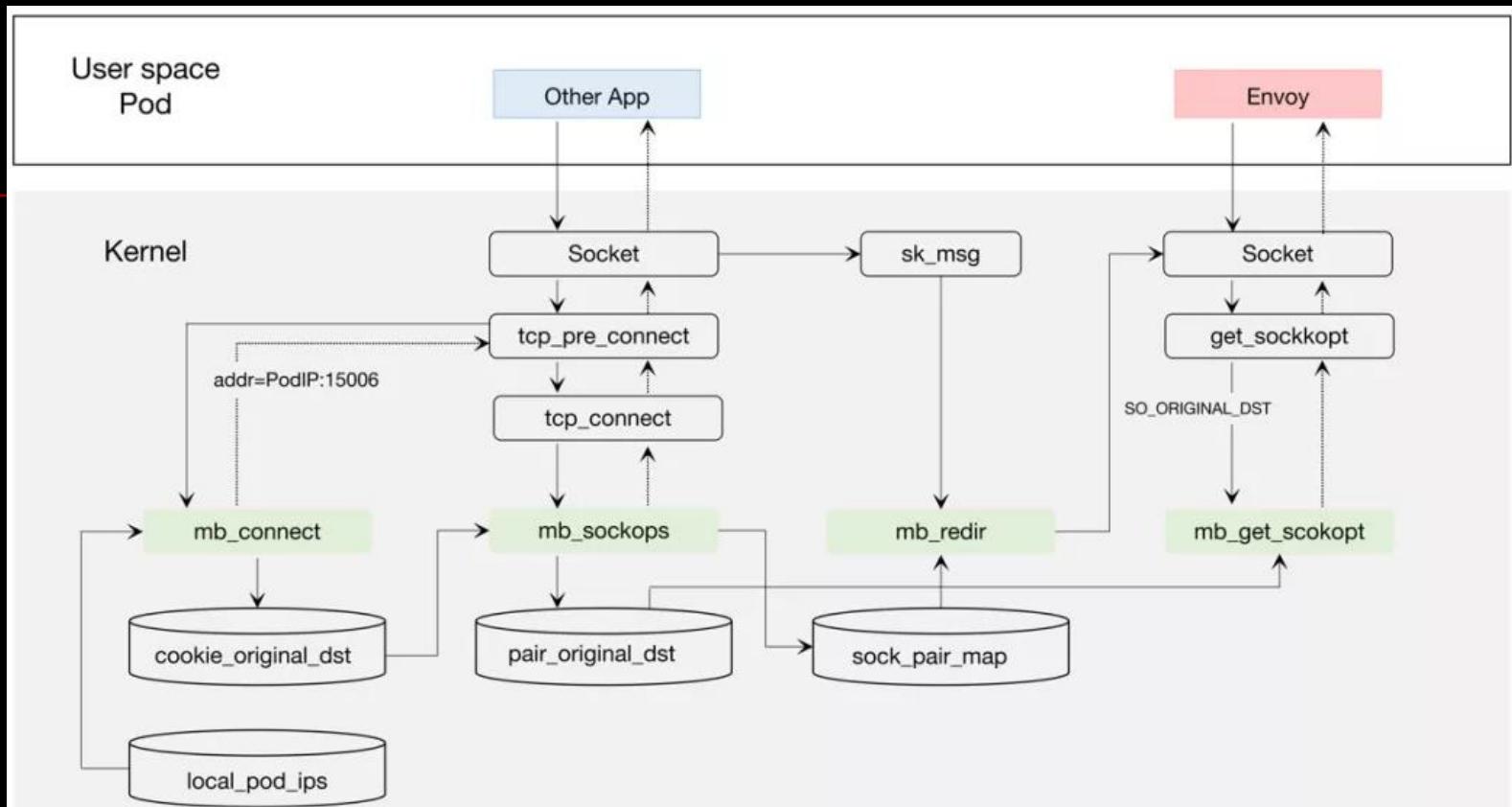
## Egress



Source: <https://www.daocloud.io/posts/7949>



## Ingress



Source: <https://www.daocloud.io/posts/7949>



## 4) Project Cilium Overview

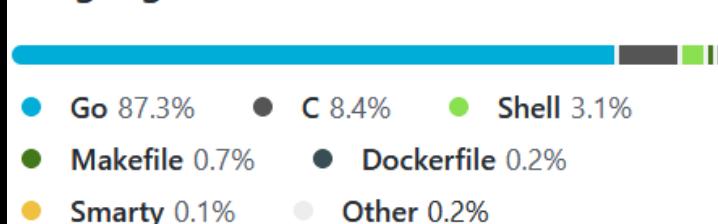
- <https://cilium.io/>
- <https://github.com/cilium/cilium>

### eBPF-based Networking, Security, and Observability

Cilium is open source software for providing and transparently securing network connectivity and loadbalancing between application workloads such as application containers or processes. Cilium operates at Layer 3/4 to provide traditional networking and security services as well as Layer 7 to protect and secure use of modern application protocols such as HTTP, gRPC and Kafka. Cilium is integrated into common orchestration frameworks such as Kubernetes.

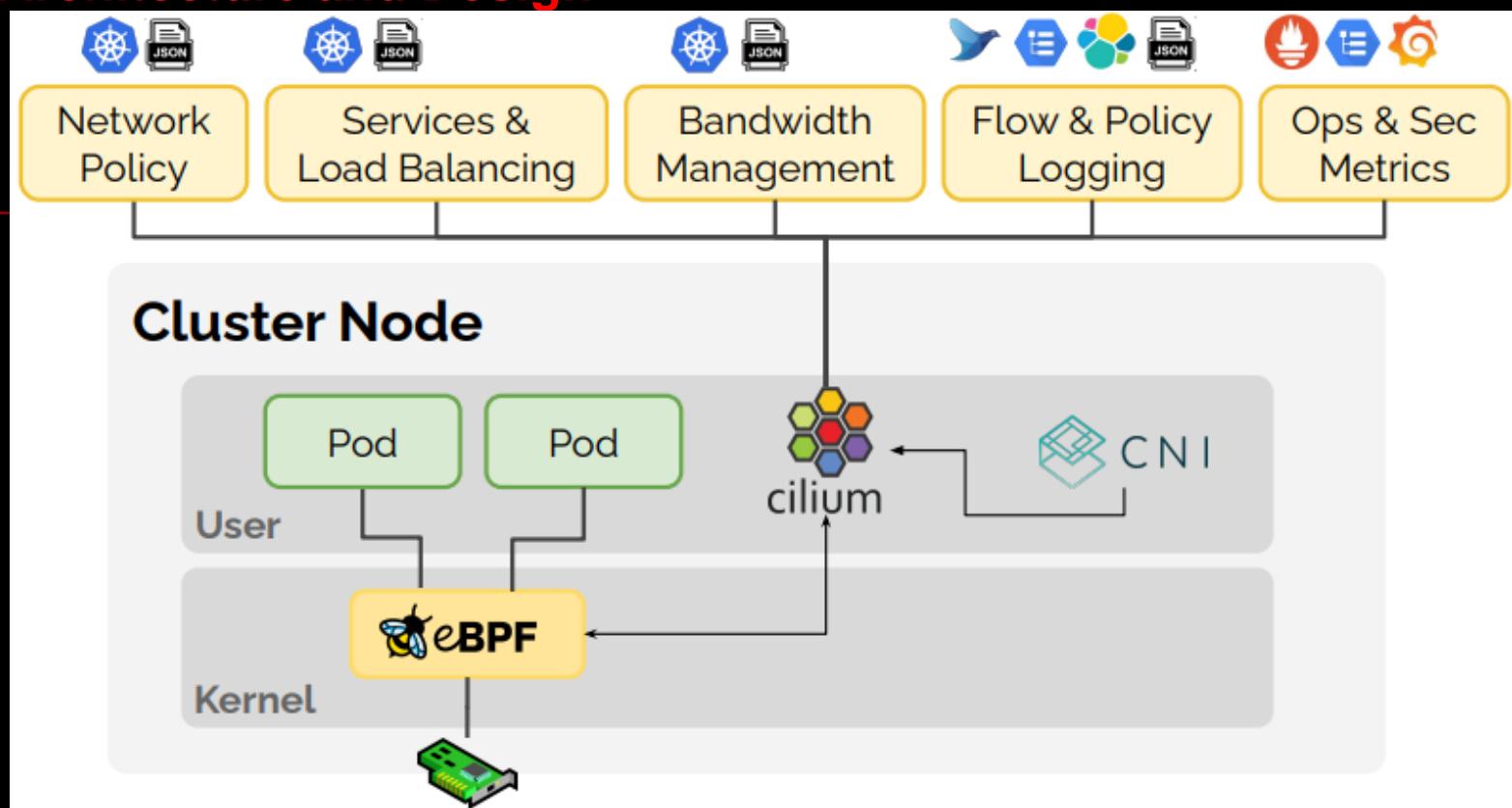
A new Linux kernel technology called eBPF is at the foundation of Cilium. It supports dynamic insertion of eBPF bytecode into the Linux kernel at various integration points such as: network IO, application sockets, and tracepoints to implement security, networking and visibility logic. eBPF is highly efficient and flexible. To learn more about eBPF, visit [eBPF.io](http://eBPF.io).

#### Languages





## Architecture and Design





## ■ Features

### Networking



Native support for service type Load Balancer and Egress



Scalable Kubernetes CNI



Multi-cluster Connectivity

### Observability



Identity-aware Visibility



Advanced Self Service Observability



Network Metrics + Policy Troubleshooting

### Security



Transparent Encryption



Security Forensics + Audit



Advanced Network Policy



## 4.1 eBPF

### \$SRC\_CILIUM/bpf

```
[mydev@fedora cilium-master]$ exa -T -L 1 bpf  
bpf
```

```
└── bpf_alignchecker.c  
└── bpf_features.h  
└── bpf_host.c  
└── bpf_lxc.c  
└── bpf_network.c  
└── bpf_overlay.c  
└── bpf_sock.c  
└── bpf_xdp.c  
└── cilium-probe-kernel-hz.c  
└── complexity-tests  
└── COPYING  
└── custom  
└── ep_config.h  
└── filter_config.h  
└── include  
└── init.sh  
└── lib  
└── Makefile  
└── Makefile.bpf  
└── mock  
└── netdev_config.h  
└── node_config.h  
└── sockops  
└── tests
```

```
[mydev@fedora cilium-master]$
```

```
...
```

```
[mydev@fedora cilium-master]$ tokei bpf  
=====  
Language      Files    Lines   Code  Comments  Blanks  
=====  
BASH          1        12     10      1       1  
C             12      5724   4176    759     789  
C Header      114     25109  15664   6720    2725  
Dockerfile    1        10     10      0       0  
Go            1        396    334     11      51  
Makefile      4        452    341     35      76  
ReStructuredText 1        131    97      0       34  
Shell          3        592    459     65      68  
Plain Text    20       36     0       36      0  
YAML          1        15     8       4       3  
-----  
Markdown      1        76     0       45      31  
|- BASH       1        6      6       0       0  
(Total)      82      32559  21105   7676    3778  
=====  
[mydev@fedora cilium-master]$
```

## eBPF Library for Go

### ■ <https://github.com/cilium/ebpf>

eBPF is a pure Go library that provides utilities for loading, compiling, and debugging eBPF programs. It has minimal external dependencies and is intended to be used in long running processes.

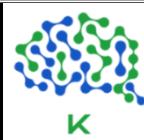
The library is maintained by [Cloudflare](#) and [Cilium](#).

### ■ Packages

This library includes the following packages:

- [asm](#) contains a basic assembler, allowing you to write eBPF assembly instructions directly within your Go code.  
(You don't need to use this if you prefer to write your eBPF program in C.)
- [cmd/bpf2go](#) allows compiling and embedding eBPF programs written in C within Go code. As well as compiling the C code, it auto-generates Go code for loading and manipulating the eBPF program and map objects.
- [link](#) allows attaching eBPF to various hooks
- [perf](#) allows reading from a `PERF_EVENT_ARRAY`
- [ringbuf](#) allows reading from a `BPF_MAP_TYPE_RINGBUF` map
- [features](#) implements the equivalent of `bptool feature probe` for discovering BPF-related kernel features using native Go.
- [rlimit](#) provides a convenient API to lift the `RLIMIT_MEMLOCK` constraint on kernels before 5.11.

■ ...





## 4.2 Hubble Overview

### ■ <https://github.com/cilium/hubble>

### Network, Service & Security Observability for Kubernetes using eBPF.

Hubble is a fully distributed networking and security observability platform for cloud native workloads. It is built on top of [Cilium](#) and [eBPF](#) to enable deep visibility into the communication and behavior of services as well as the networking infrastructure in a completely transparent manner.

Hubble can answer questions such as:

#### Service dependencies & communication map:

- What services are communicating with each other? How frequently? What does the service dependency graph look like?
- What HTTP calls are being made? What Kafka topics does a service consume from or produce to?

#### Operational monitoring & alerting:

- Is any network communication failing? Why is communication failing? Is it DNS? Is it an application or network problem? Is the communication broken on layer 4 (TCP) or layer 7 (HTTP)?
- Which services have experienced a DNS resolution problems in the last 5 minutes? Which services have experienced an interrupted TCP connection recently or have seen connections timing out? What is the rate of unanswered TCP SYN requests?

#### Application monitoring:

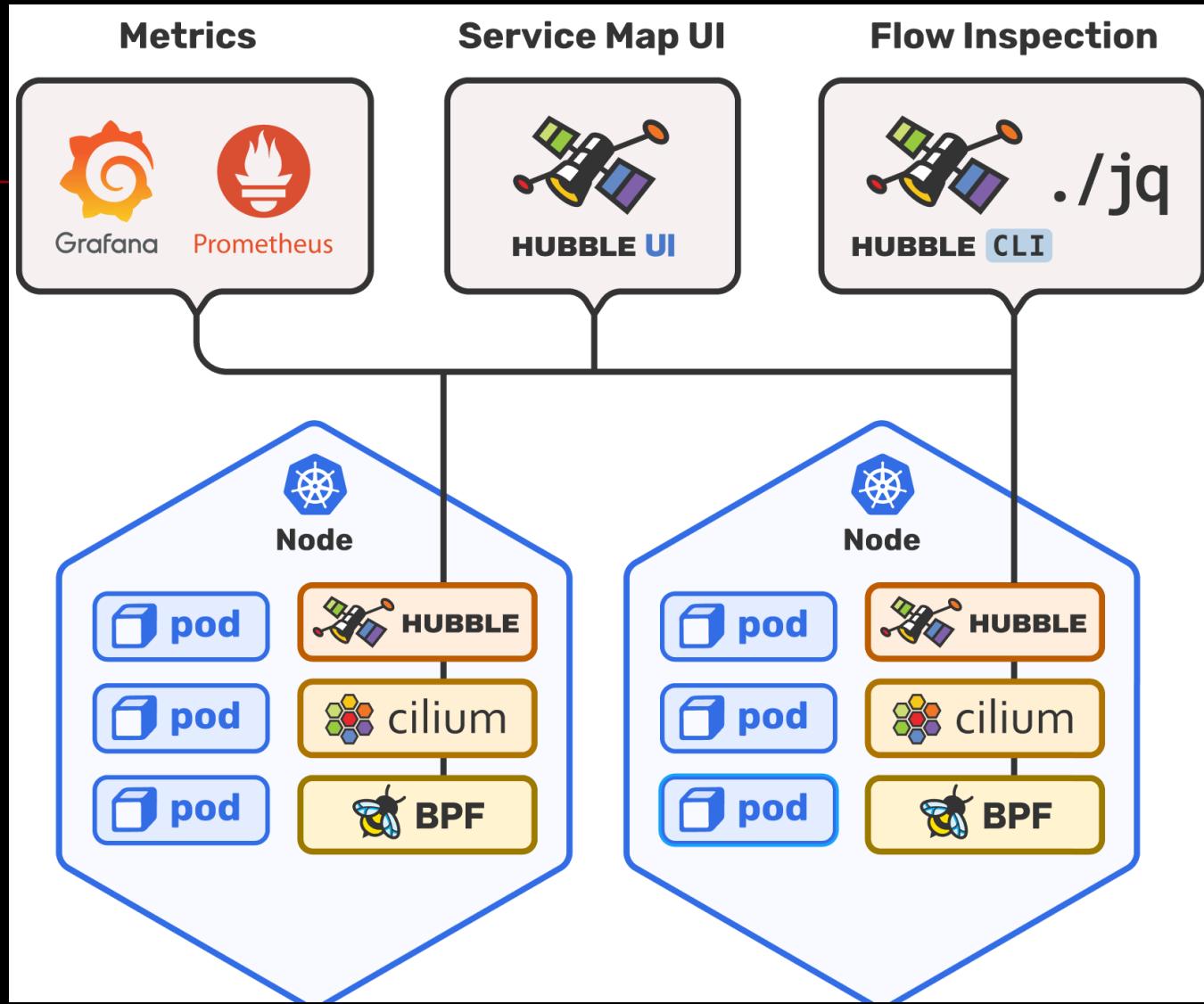
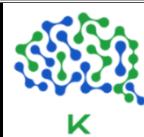
- What is the rate of 5xx or 4xx HTTP response codes for a particular service or across all clusters?
- What is the 95th and 99th percentile latency between HTTP requests and responses in my cluster? Which services are performing the worst? What is the latency between two services?

#### Security observability:

- Which services had connections blocked due to network policy? What services have been accessed from outside the cluster? Which services have resolved a particular DNS name?

...

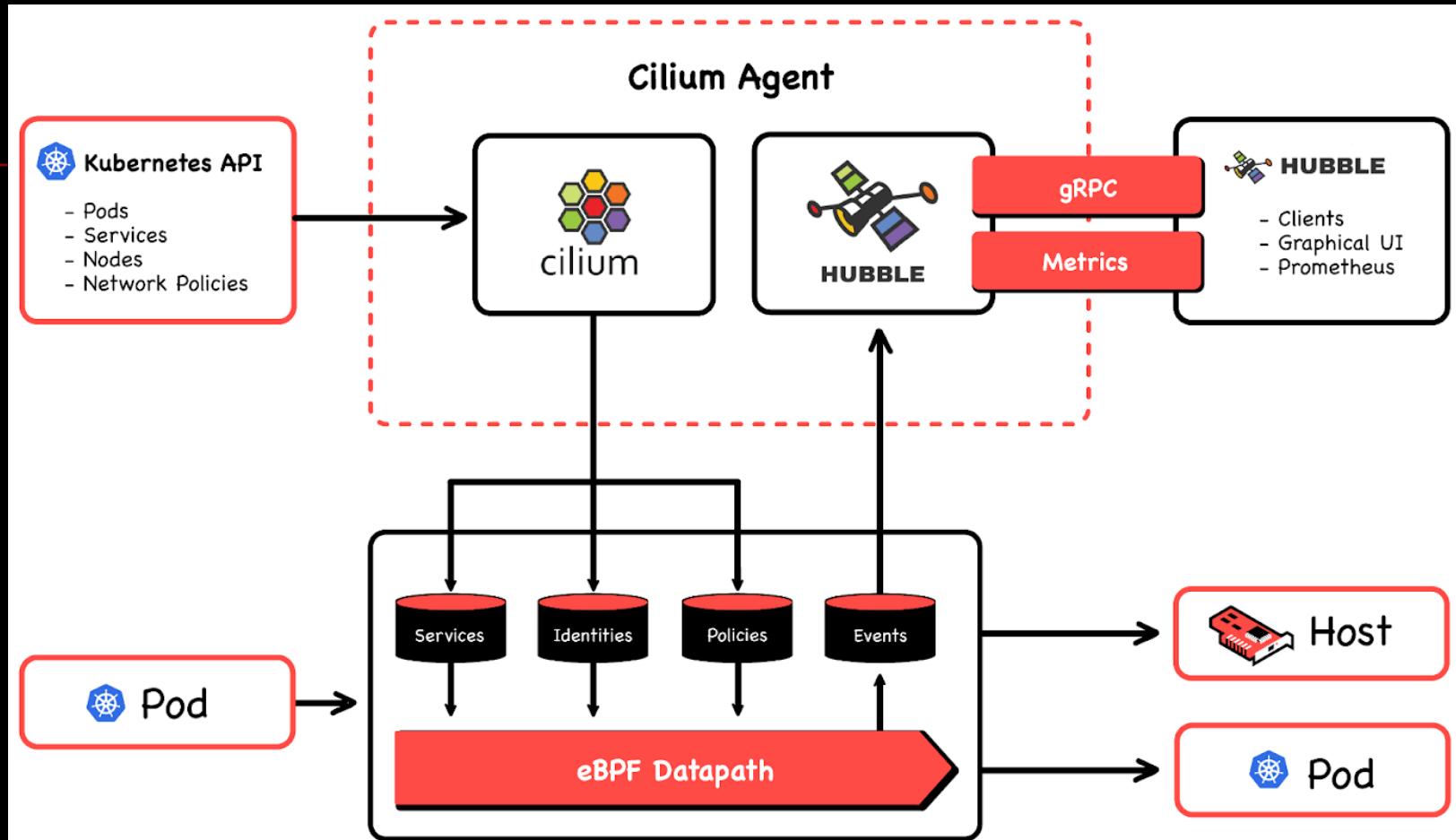
## ■ Architecture and Design



Source: [https://raw.githubusercontent.com/cilium/hubble/master/Documentation/images/hubble\\_arch.png](https://raw.githubusercontent.com/cilium/hubble/master/Documentation/images/hubble_arch.png)



## How it works

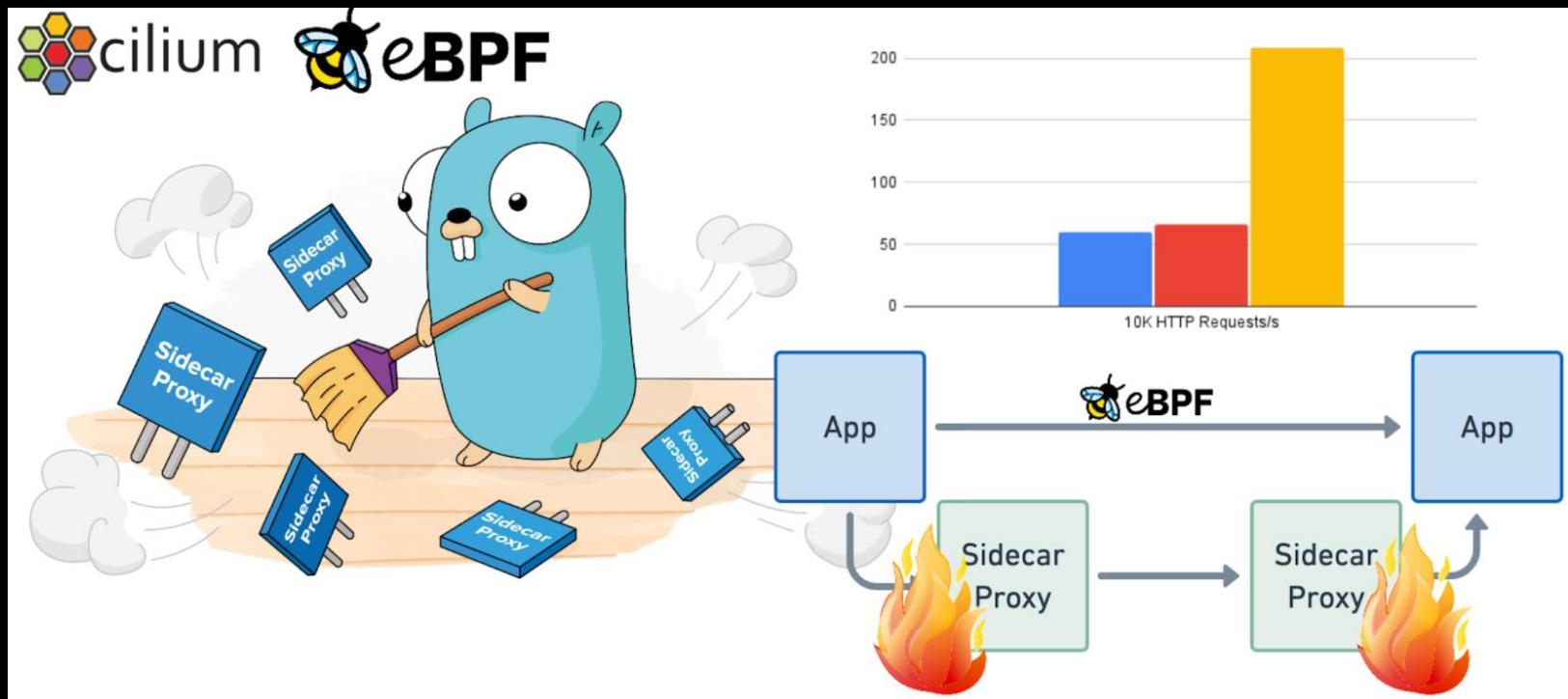


Source: <https://blog.container-solutions.com/ebsf-cloud-native-tools-an-overview-of-falco-inspekto-gadget-hubble-and-cilium>

## 4.3 eBPF-powered Cilium Service Mesh

### Overview

- <https://github.com/cilium/cilium-service-mesh-beta>  
**Instructions and issue tracking for Service Mesh capabilities of Cilium.**



Source: <https://isovalent.com/blog/post/2021-12-08-ebpf-servicemesh>



## ■ Feature Status

*Please note: although we'll only change things for good reasons, there are no backward-compatibility guarantees with the configuration of Cilium Service Mesh during beta, and the code has only been through limited testing. Please be mindful of this when you choose where to deploy Cilium Service Mesh - we do not recommend using it in production or staging environments yet.*

---

**Alpha** Very early release of features that have not been through extensive testing yet. No guarantee that future releases will be back compatible with CRDs or other configuration settings. Please don't be surprised if you encounter bugs.

**Beta** Early release of features that we expect to be stable enough for testing by end users. No guarantee that future releases will be back compatible with CRDs or other configuration settings.

**v1.11** Features already merged into the Cilium v1.11 release (as well as in the Service Mesh beta-specific builds)

Feature	Status
Kubernetes Ingress support	Beta
Open Telemetry support	Alpha, v1.11
L7-aware Traffic Management	Alpha

Other features will be added as the beta progresses.

# What does Cilium bring to Service Mesh?

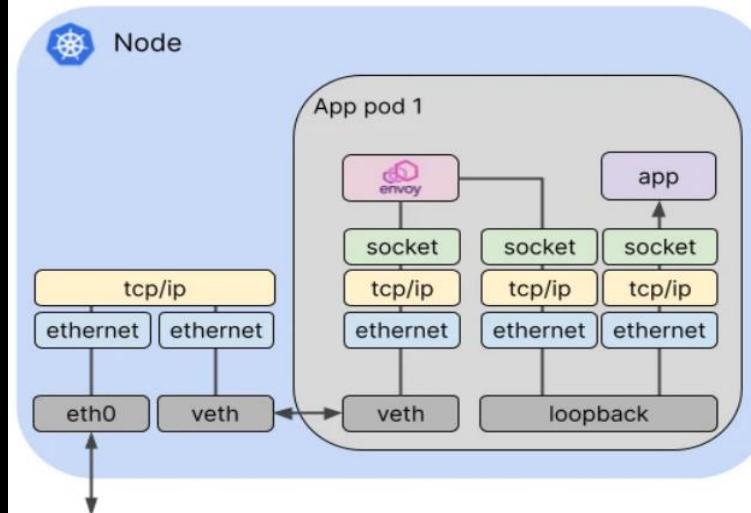


- <https://thenewstack.io/how-ebpf-streamlines-the-service-mesh/>

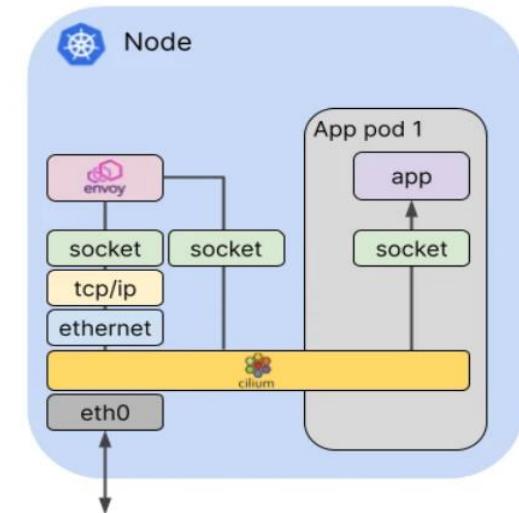
The term “Service Mesh” encompasses a wide range of features, including service discovery, encryption, service authentication, load balancing, observability, canary roll-outs and more. Some of these features overlap with established Cilium capabilities - for example, Cilium has offered load balancing, Kubernetes service awareness, multi-cluster connectivity, and visibility of network traffic at layer 3-7, for ages. Cilium already uses Envoy for L7 policy and observability for some protocols, and this same component is used as the sidecar proxy in many popular Service Mesh implementations. So it's a natural step to extend Cilium to offer more of the features commonly associated with Service Mesh.

In a typical Service Mesh, all network packets need to pass through a sidecar proxy container on their path to or from the application container in a Pod. In Cilium Service Mesh, we're moving that proxy container onto the host and kernel so that sidecars for each application pod are no longer required. Because eBPF allows us to intercept packets at the socket as well as at the network interface, Cilium can dramatically shorten the overall path for each packet. (Read more about [sidecarless, eBPF-based Service Mesh](#).)

Service mesh with traditional networking



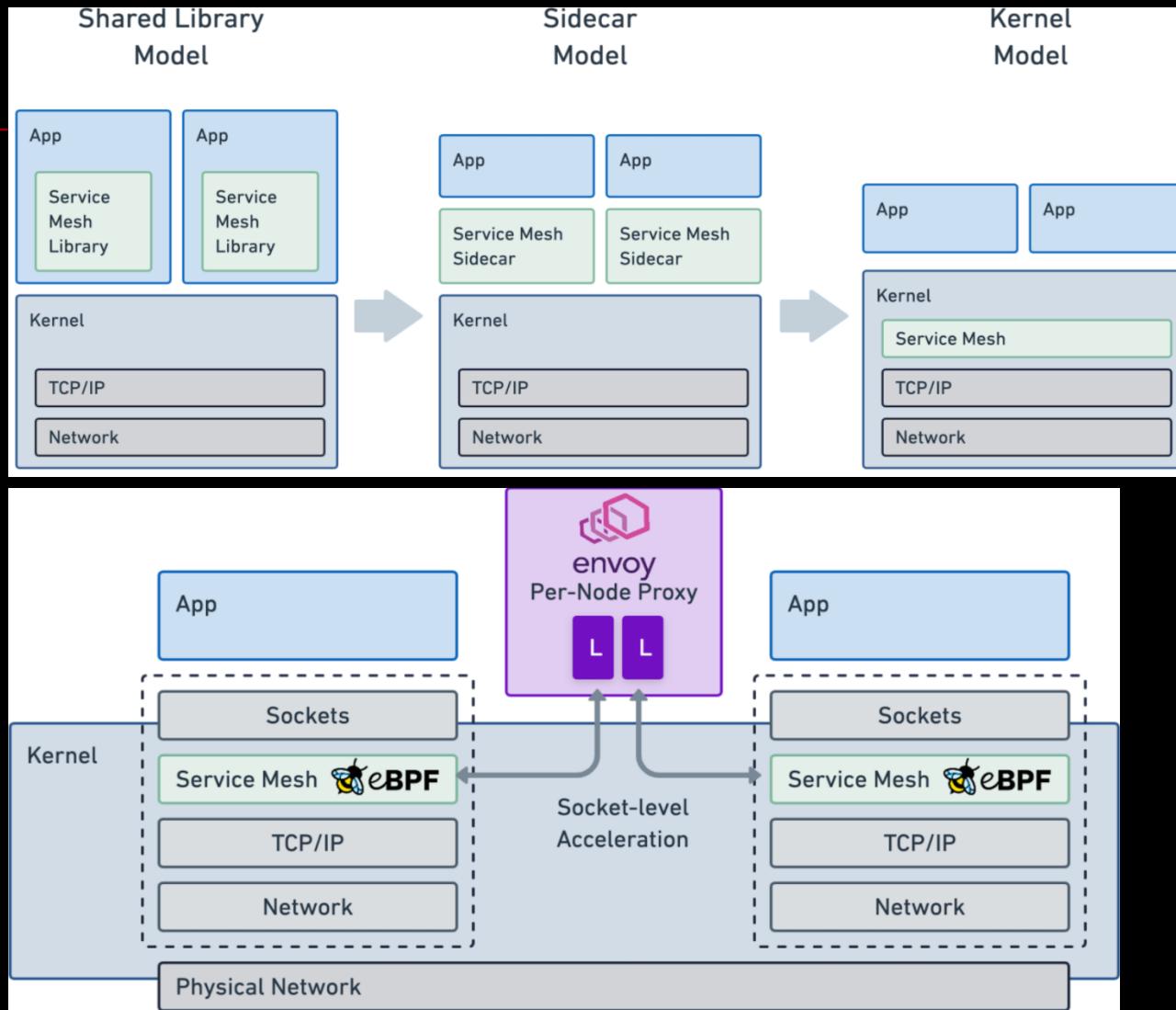
Sidecarless model, eBPF acceleration





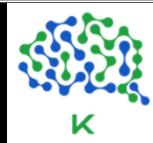
## Architecture & Design

- <https://isovalent.com/blog/post/2021-12-08-ebpf-servicemesh>



## Good Resources

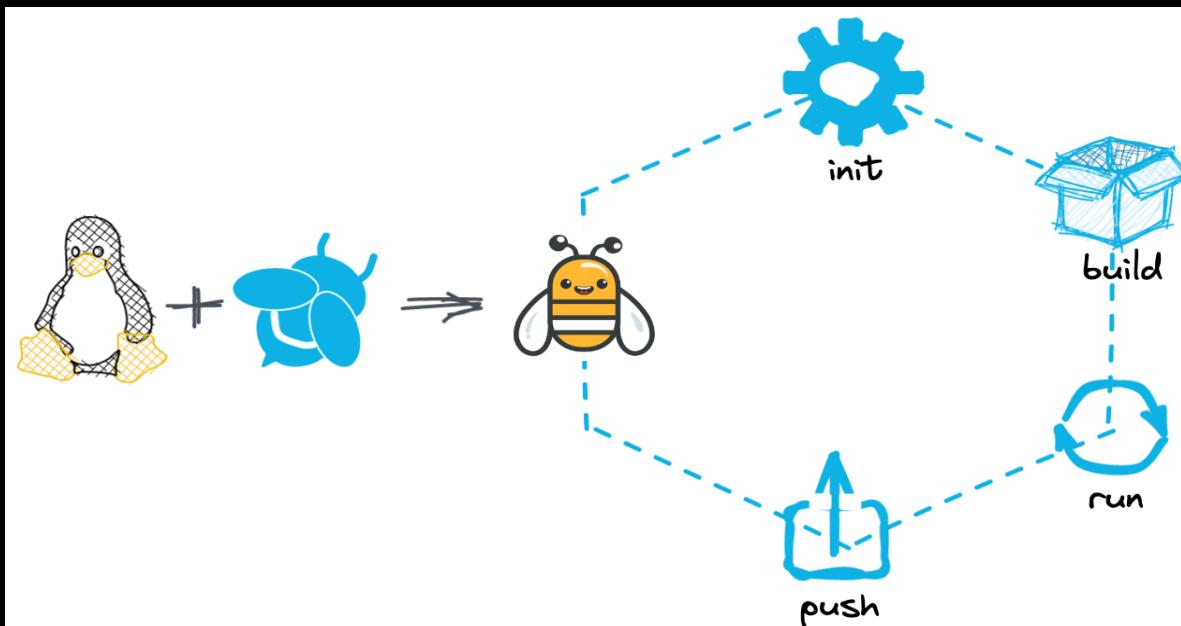
- <https://cilium.io/blog/2021/12/01/cilium-service-mesh-beta>
  - <https://thenewstack.io/how-ebpf-streamlines-the-service-mesh/>
  - ...
- 





## 5) Project BumbleBee Overview

- <https://bumblebee.io/>  
**simplifies building eBPF tools and allows you to package, distribute, and run them anywhere. Just focus on the eBPF portion of your code and BumbleBee automates away the boilerplate, including the userspace code.**
- **Features**



Source: <https://www.solo.io/blog/get-started-with-ebpf-using-bumblebee/>



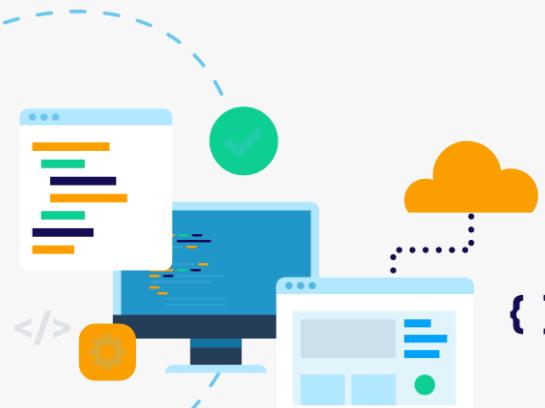
## Publish

Leveraging BTF and OCI packaging, the eBPF code you write with BumbleBee is portable and can plug into existing publishing workflows. Push and pull your eBPF code to any OCI compliant registry to publish to other users.



## Build

Getting the eBPF tool chain “just right” is hard. BumbleBee automates the build process and lets you focus on the code. BumbleBee packages your eBPF code as an OCI image so you can distribute it across your infrastructure.



## Run

With BumbleBee, you focus on your eBPF code and run it anywhere. BumbleBee also builds the userspace code and can expose the eBPF maps as logs, metrics, and histograms. BumbleBee leverages BTF introspection to know what types to display.



## Getting started

- [https://github.com/solo-io/bumblebee/blob/main/docs/getting\\_started.md](https://github.com/solo-io/bumblebee/blob/main/docs/getting_started.md)

...

The first option you will be confronted with is the language with which you will develop your probe. Currently only `c` is supported, but support for `Rust` is planned as well.

? What language `do` you wish to use `for` the filter:  
▶ `C`



...

...

## Good Resources

- <https://github.com/solo-io/bumblebee/blob/main/docs>
  - <https://www.solo.io/blog/get-started-with-ebpf-using-bumblebee/>
  - ...
- 



# XIII. 5G-6G

## 1) 5G & 6G

### 1.1 5G

---

- ...





## 1.1.2 Architecture

### 1.1.2.1

- ...
-

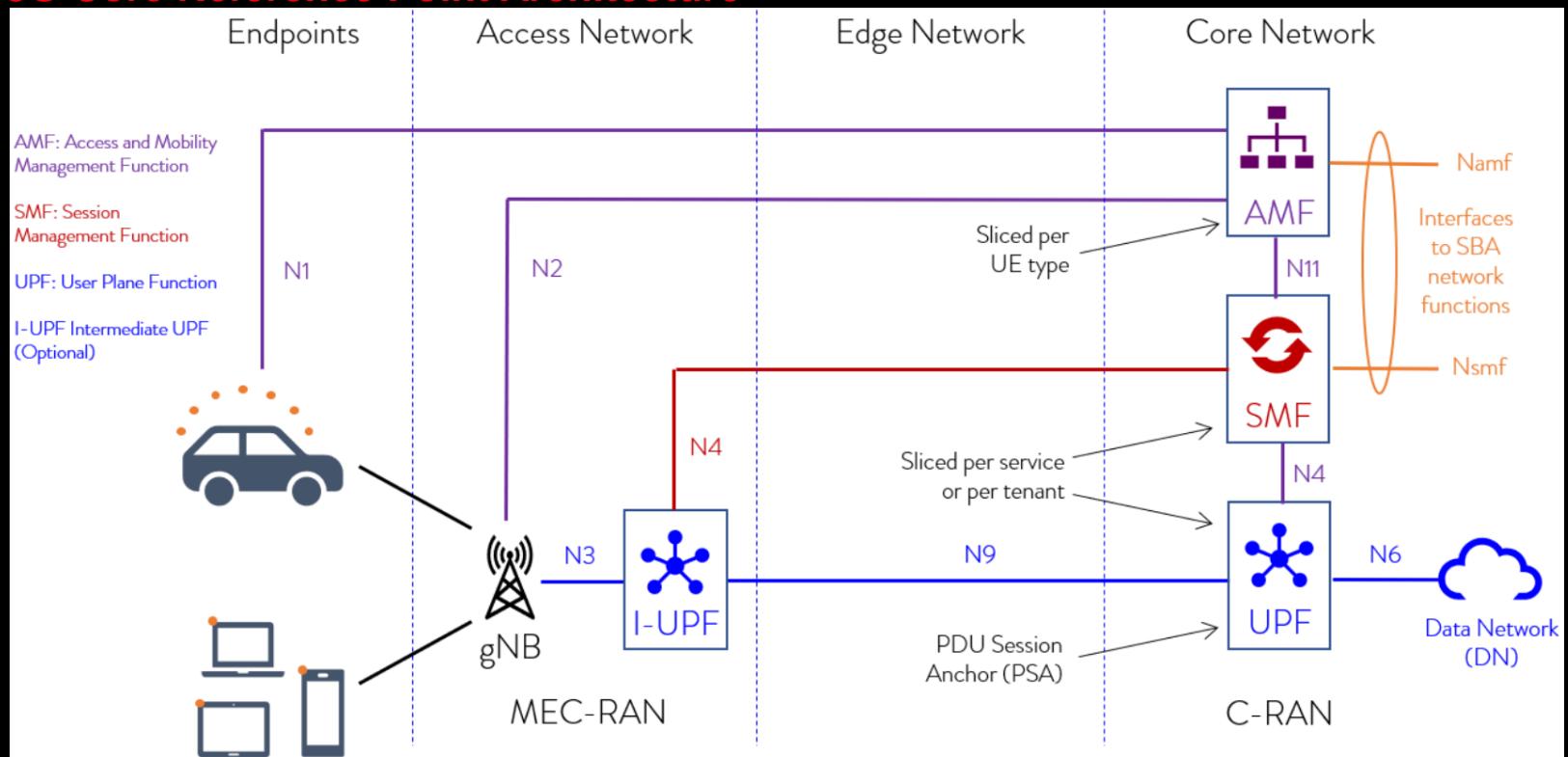


## 1.1.2.2 UPF

### Overview

- <https://www.metaswitch.com/knowledge-center/reference/what-is-the-5g-user-plane-function-upf>

### 5G Core Reference Point Architecture

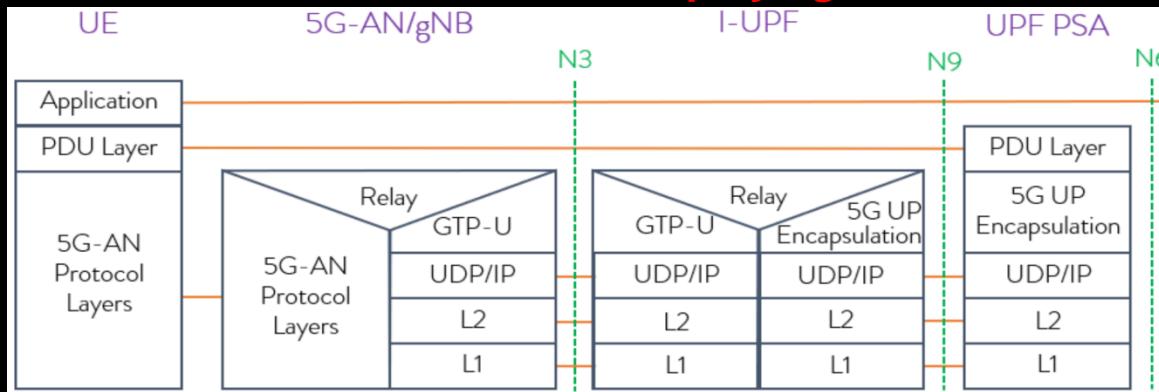




The User Plane Function has four distinct reference points:

- N3: Interface between the RAN (gNB) and the (initial) UPF.
- N9: Interface between two UPF's (i.e the Intermediate I-UPF and the UPF Session Anchor)
- N6: Interface between the Data Network (DN) and the UPF
- N4: Interface between the Session Management Function (SMF) and the UPF

## 5G User Plane Protocol Stack, employing STP with 5G header extensions



## Summary

To meet 5G's fundamental mandate for granular capacity or application-driven instantiation and up/down-scaling, the UPF must be implemented as a pure cloud native network function using modern microservices methodologies and deployable within a serverless framework. In order to achieve high packet throughputs with complex pipelines and low latencies, such components have historically been realized using dedicated hardware and custom silicon. Delivering a cloud native UPF on shared resources demands solving complex real-time data processing problems and providing a high degree of flexibility in the pipeline to support emerging traffic tunneling, mobility and infrastructure overlays. UPF instantiation must also be highly automated and orchestrated. This means developing or enhancing user-space data plane acceleration, runtime programmability techniques and tightly integrating with numerous cloud orchestration systems, such as Kubernetes.

The User Plane Function is a critical new component for supporting the new generation of service-based architectures. The velocity of innovation in this area, however, should also allow the early majority of perspective CUPS adopters to adopt the UPF over interim PGW/SGW-U solutions, smoothing the migration from 4G to 5G.



## 2) XDP for UPF

### 2.1 upf-bpf

- <https://github.com/navarrothiago/upf-bpf>  
**An In-Kernel Solution Based on eBPF/XDP for 5G UPF.**

An open source C++ library powered by eBPF/XDP for user plane in the mobile core network (5G/LTE).

The key pillars of this project are:

- In-kernel fast packet processing
- Flexible and programmable dataplane
- Portable to different systems

These points are achieved mainly by eBPF/XDP and CO-RE (Compile Once - Run Everywhere) technologies.

This project is based on the following 3GPP Technical Specification:

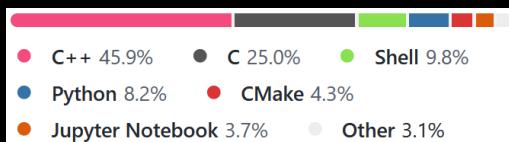
- LTE; 5G; Interface between the Control Plane and the User Plane nodes (3GPP TS 29.244 version 16.5.0 Release 16)
- 5G; System architecture for the 5G System (5GS) (3GPP TS 23.501 version 16.5.0 Release 16)

The main goal is to enable in-kernel fast packet processing in third-party UPF/5G or SPGWu/LTE components in order to:

1. Boost them for those which does not have any fast packet processing enabled, or
2. Co-locate them with other fast packet processing solutions (e.g. DPDK)

Possible scenarios that take advantage of this type of technology: MEC, 5G NPN (Non Public Networks), on-premise, 5G enterprise, and much more.

- **Languages (without submodules)**





## ■ Features

As described in 3GPP TS 29.244, the Information Elements (IEs) are part of the PFCP context. The PFCP context is created by sending a PFCP Session Establishment Request message. The main features supported in this project are:

### Management Layer - CRUD

- PFCP Session
- PDR (Packet Detection Rule)
- FAR (Forwarding Action Rule)

### Fast Datapath Layer

- UDP and GTP parse
- Traffic classification based on PDR
- Traffic forwarding based on FAR

The logical data model between PFCP Session and IEs is shown in the image below. For more detail, see 3GPP TS 29.244 version 16.5.0 Release 16.

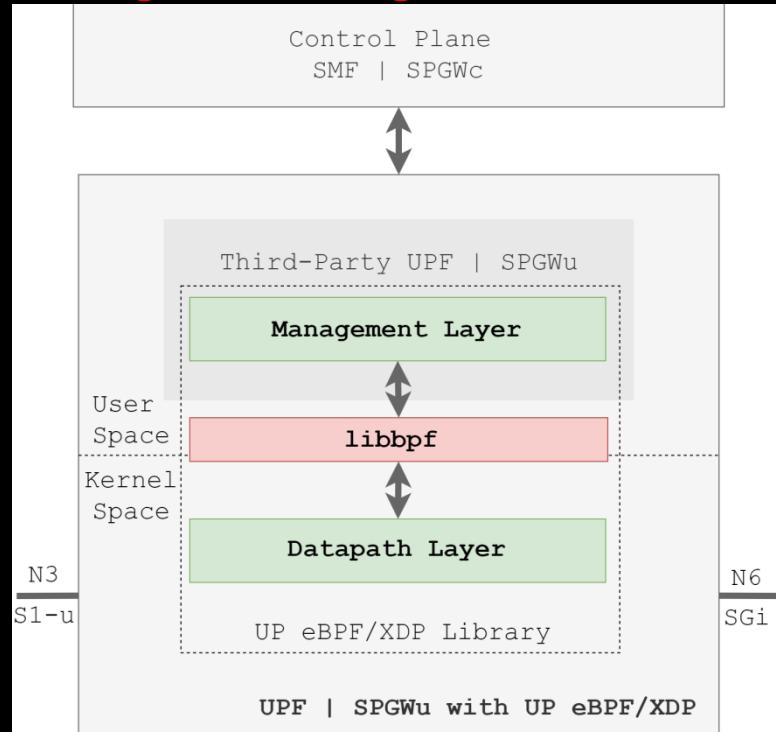


# Architecture and Design

- The library is divided in layers:

- Management Layer:** An user space layer responsible to receive requests from the third-party UPF/SPGWu components to manage PFCP sessions and eBPF programs lifecycle
- Datapath Layer:** A kernel space layer representing by eBPF/XDP programs responsible to handle the user traffic (datapath) for fast packet processing

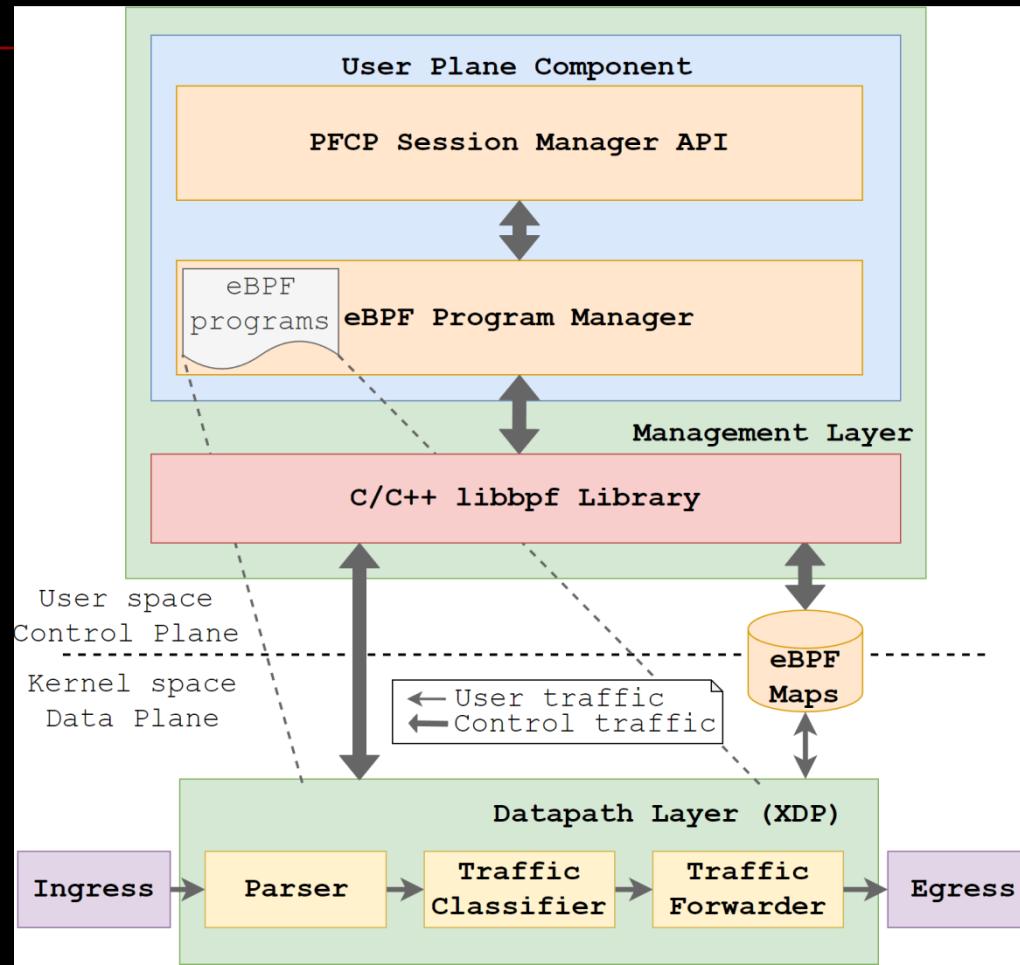
## The high level design



Source: <https://github.com/navarrothiago/up-bpf/blob/master/img/up-ebpf-xdp-high-level.svg>



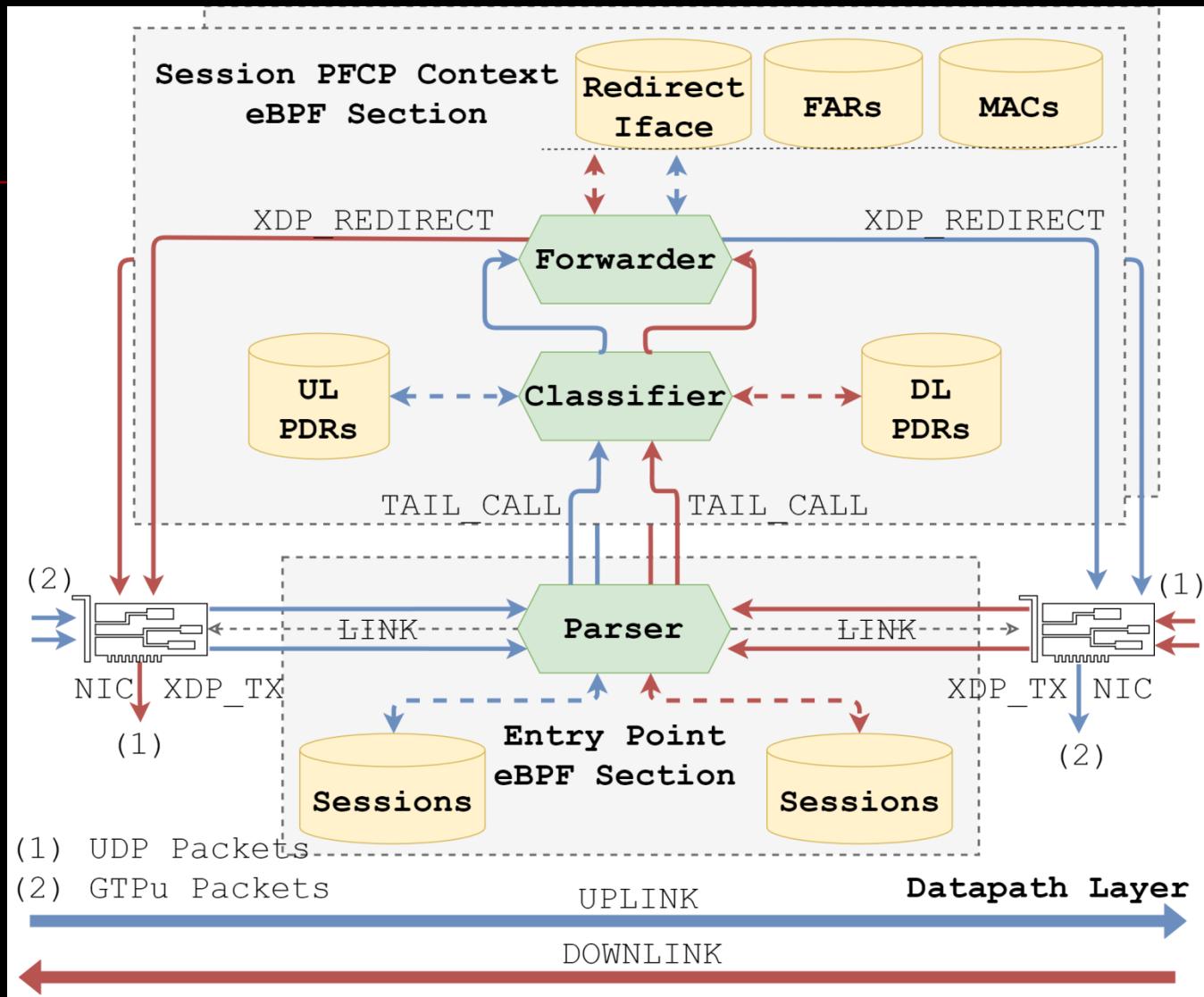
The library has a component, called `PFCP Session Manager`, which is a C++ API responsible for managing PFCP (Packet Forwarding Control Protocol) sessions. For each session, there is an eBPF program that represents the PFCP context in the fast path. These programs are managed by the `eBPF Program Manager` component. The fast path is composed of three main functions: parser, traffic classifier and traffic forwarder. The image below shows this in more detail.



Source: <https://github.com/navarrothiago/upf-bpf/blob/master/img/up-ebpf-xdp-high-level2.svg>



## A low-level design (Datapath Layer)



Source: <https://github.com/navarrothiago/upf-bpf/blob/master/img/up-ebpf-xdp-low-level.svg>

# XIV. xBPF



## 1) Overview

What does xBPF mean?

- xBPF stands for eBPF/uBPF and all of their derivatives, together with the technologies base on them.



## 2) Femto-Containers

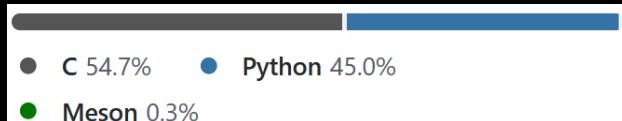
### Overview

- <https://github.com/future-proof-iot/Femto-Container>

### **Minimal eBPF based virtual machine for small embedded devices.**

Femto-Containers is a minimal lightweight virtual machine environment for embedded devices. The virtual machine ISA is adapted from Linux eBPF. Femto-Containers is pure C and makes use of some GCC extensions for efficiency. There is some auxiliary Python tooling to convert compiled Femto-Container applications into efficient representations.

- **Languages**



- **Features**

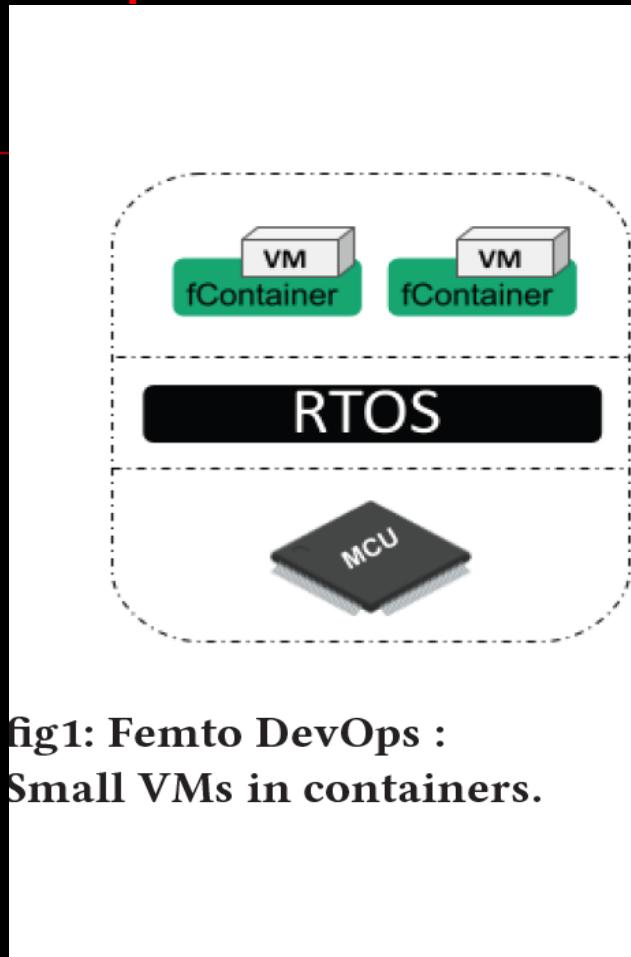
- Simple virtual machine
- Hardware independent
- Short-lived, Event driven
- Integration with RIOT
- Based on Linux eBPF
- Minimal footprint

Source: <https://summit.riot-os.org/2021/wp-content/uploads/sites/16/2021/09/s04-02.pdf>



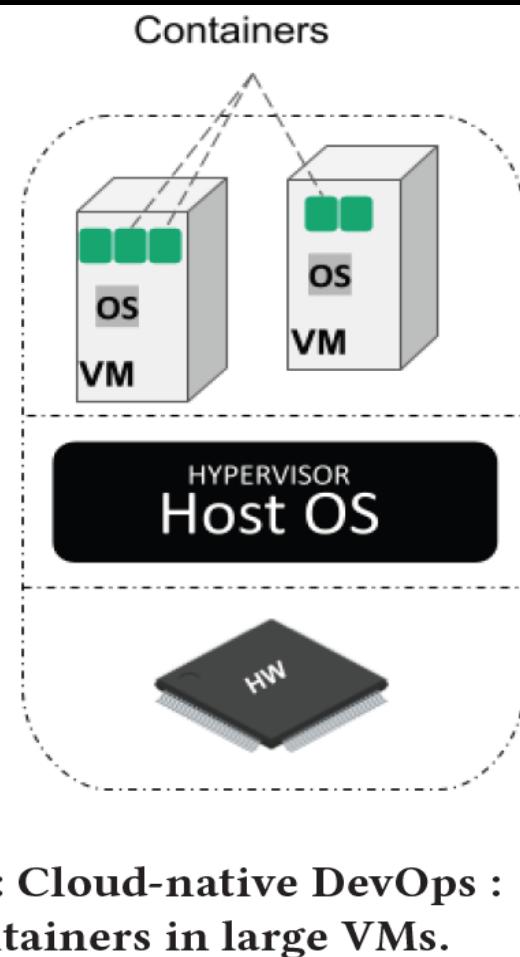
## Architecture and Design

- DevOps architecture: Femto vs Cloud.



**fig1: Femto DevOps :**  
**Small VMs in containers.**

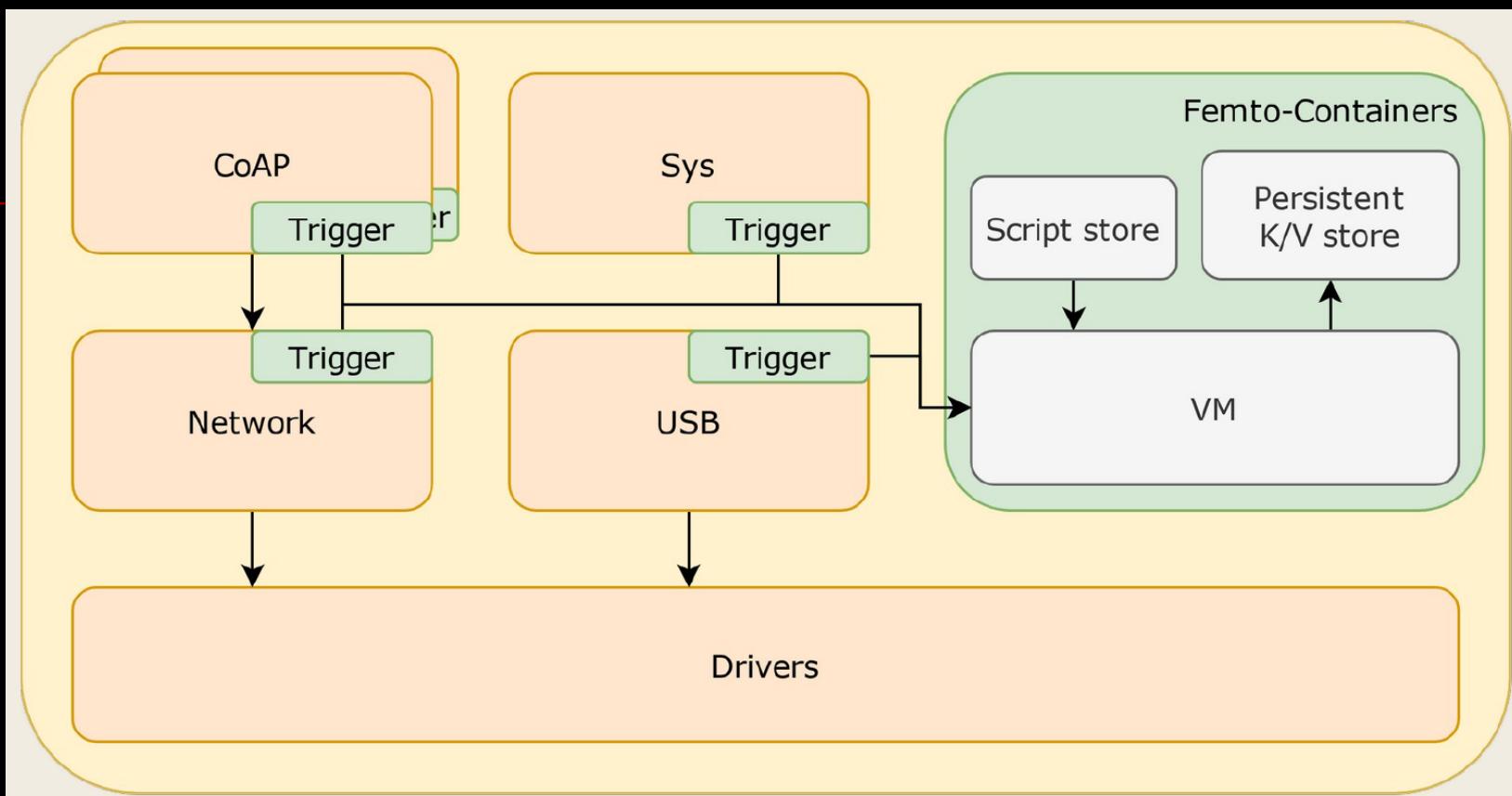
Source: <https://arxiv.org/pdf/2106.12553.pdf>



**fig2: Cloud-native DevOps :**  
**Containers in large VMs.**



## ■ How it works



Source: <https://summit.riot-os.org/2021/wp-content/uploads/sites/16/2021/09/s04-02.pdf>

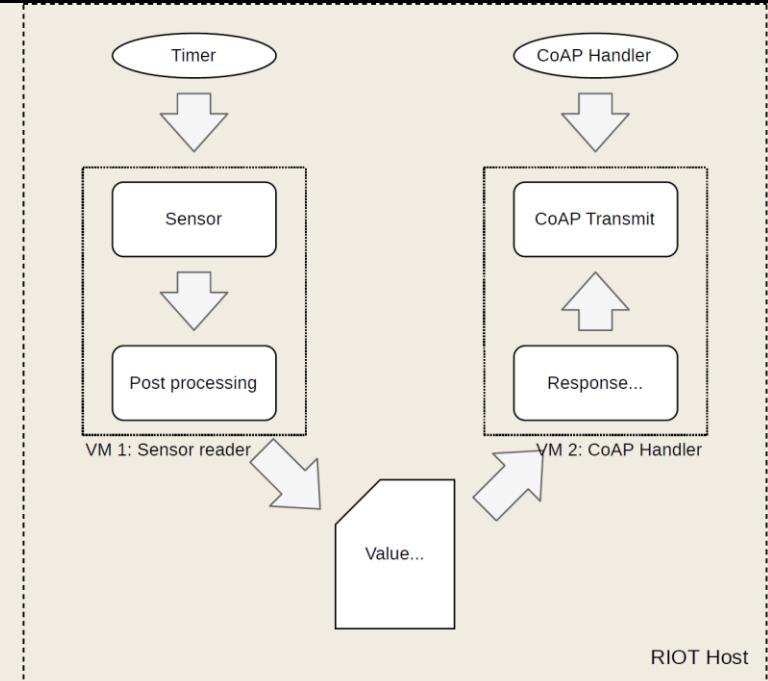


## Events

Event triggered:

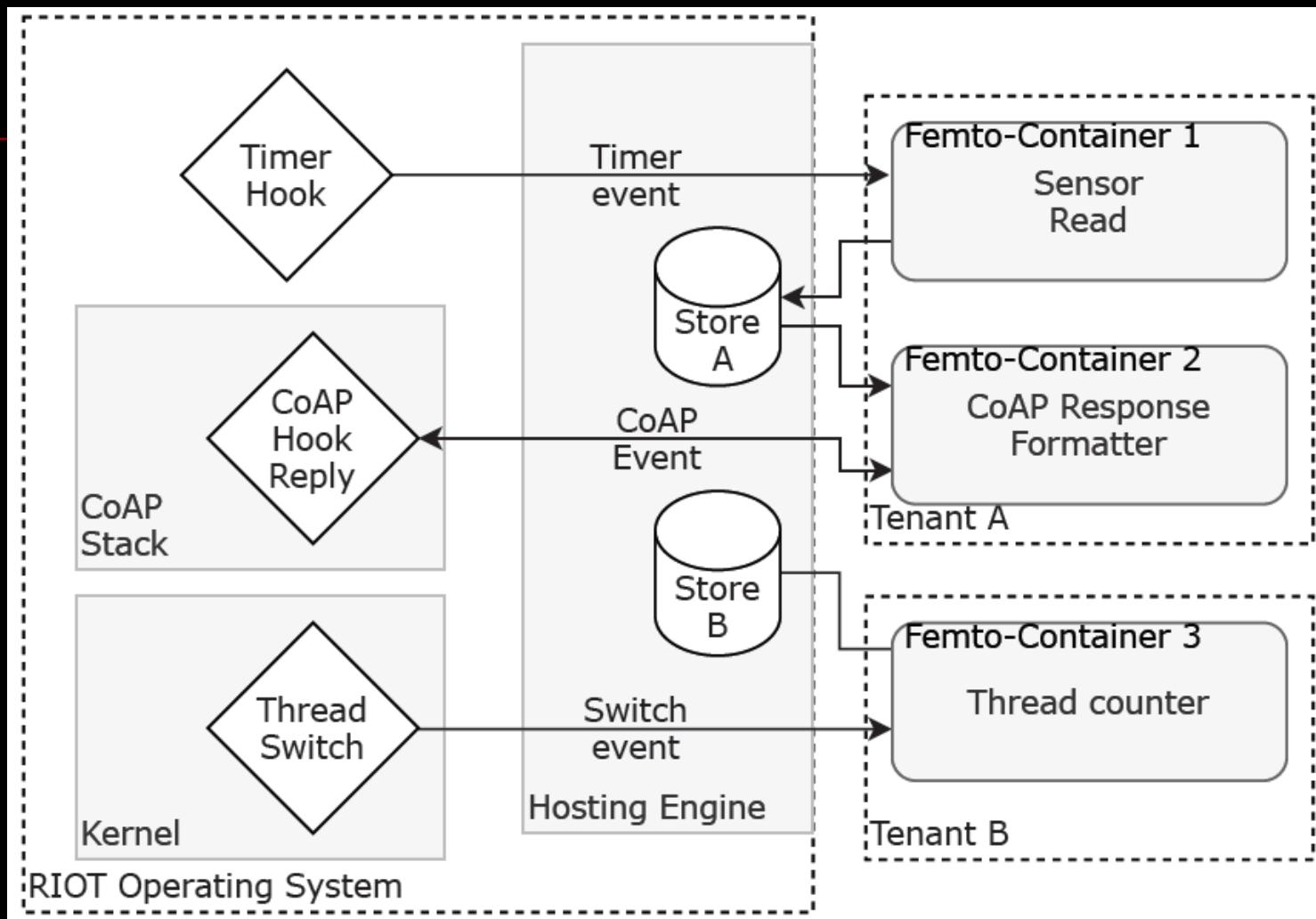
- Network
- USB
- System events
- Timers

Adding hooks is cheap



Source: <https://summit.riot-os.org/2021/wp-content/uploads/sites/16/2021/09/s04-02.pdf>

## event and value flow when hosting multiple containers for different tenants

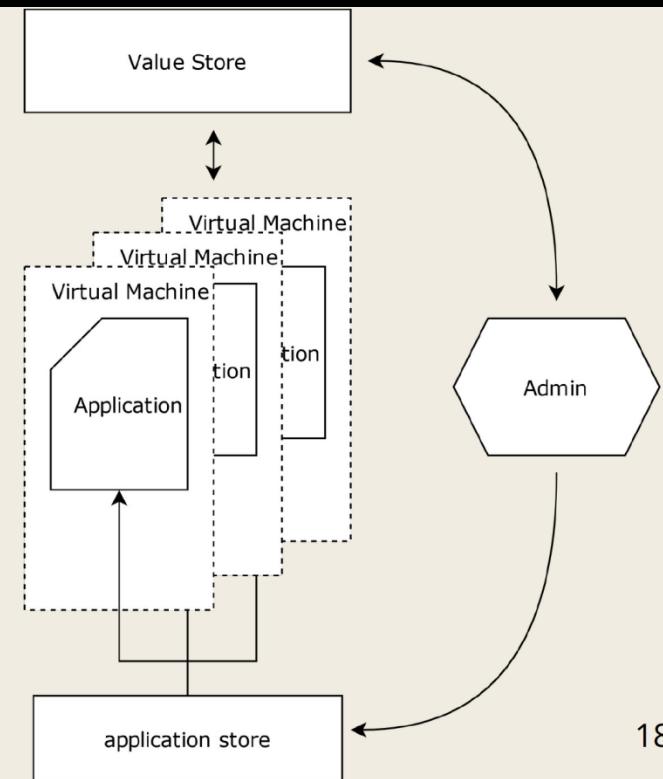


Source: <https://arxiv.org/pdf/2106.12553.pdf>



## OS Interaction

- Context and return value
  - Packet and Allow/Reject
- Bindings
  - Calls to OS, e.g. saul\_read
- Value store
  - Store simple values

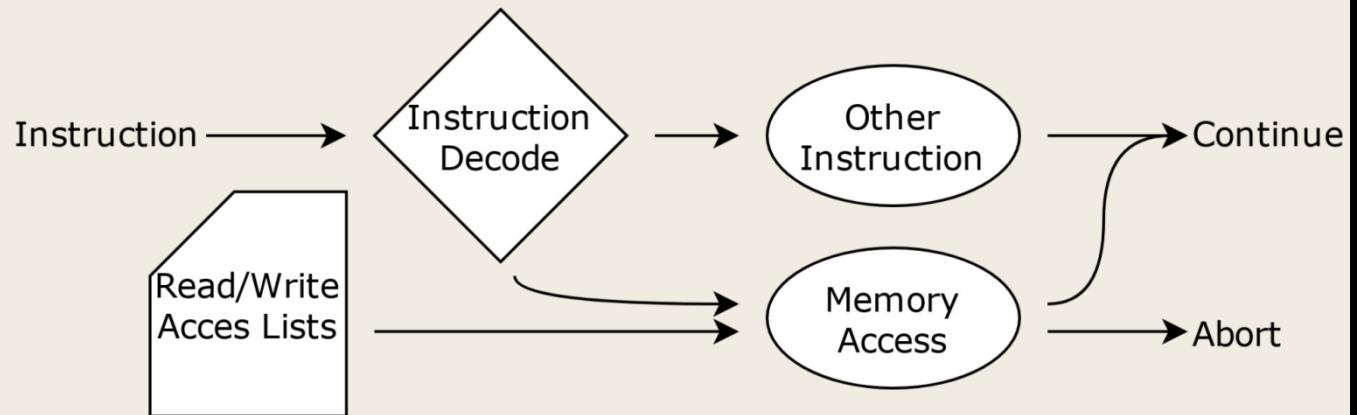


Source: <https://summit.riot-os.org/2021/wp-content/uploads/sites/16/2021/09/s04-02.pdf>



## ■ Isolation

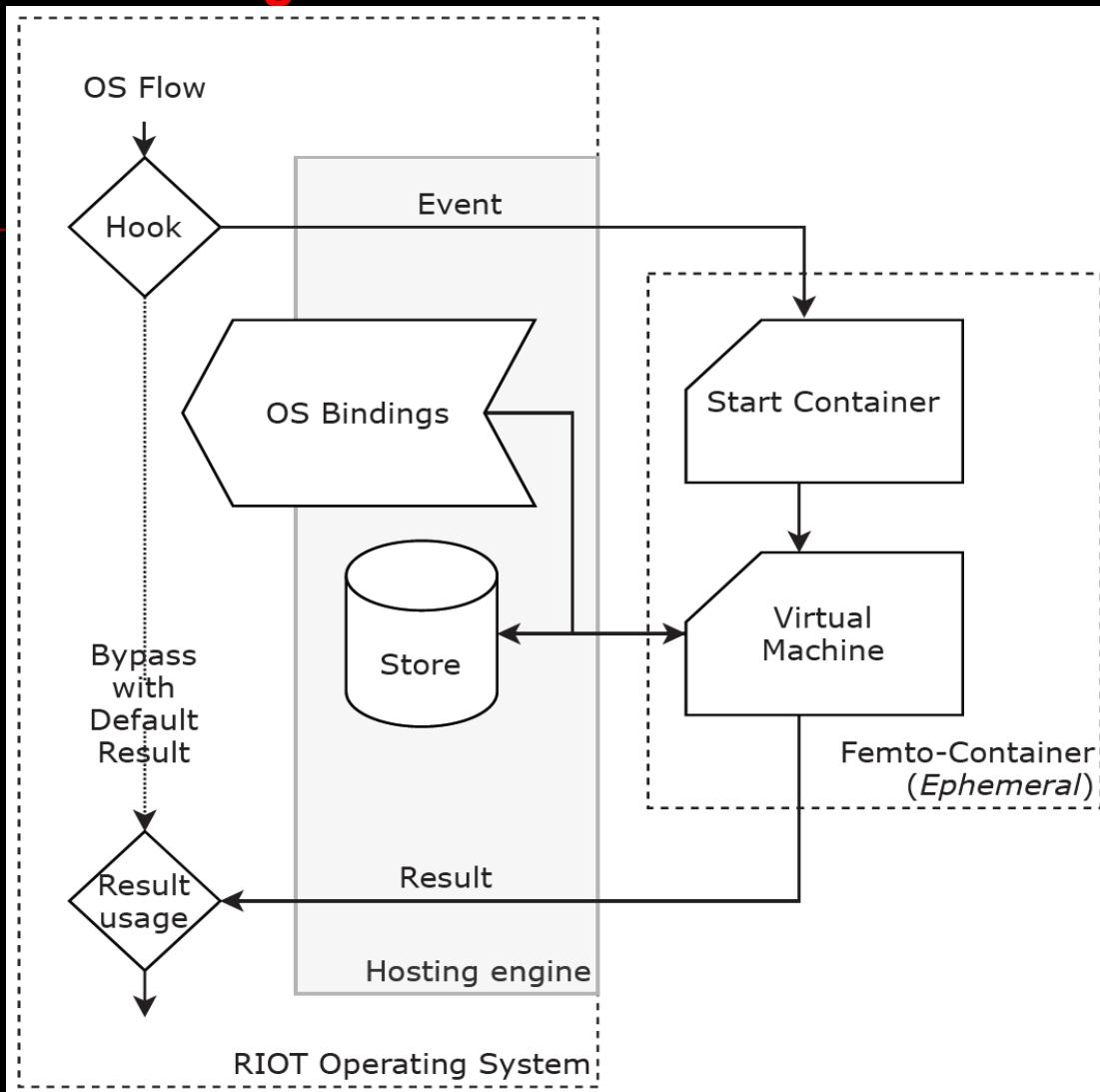
- Sandboxed from the host
  - Pre-flight checks
  - Memory access guards



Source: <https://summit.riot-os.org/2021/wp-content/uploads/sites/16/2021/09/s04-02.pdf>



## RTOS Integration



Source: <https://arxiv.org/pdf/2106.12553.pdf>

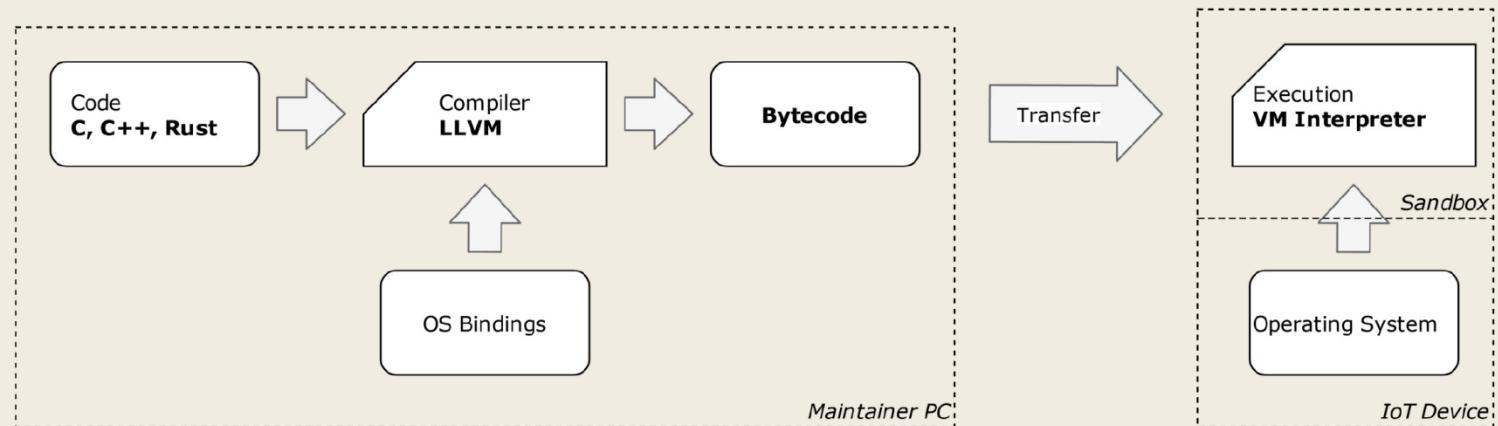


## Workflow

### ■ Sample

[https://github.com/bergzand/RIOT/tree/wip/bpf/examples/rbpf\\_sched/](https://github.com/bergzand/RIOT/tree/wip/bpf/examples/rbpf_sched/)

- 1) Write code
- 2) Compile
- 3) Transfer
- 4) Run



Source: <https://summit.riot-os.org/2021/wp-content/uploads/sites/16/2021/09/s04-02.pdf>



# Start VM from RIOT

- Our code is compiled-in for simplicity
- RIOT executes the VM when switching threads
- Run the code
- Query the value store counters

```
static void sched_rbpf_cb(kernel_pid_t active_thread,
                           kernel_pid_t next_thread)
{
    sched_ctx_t ctx = {
        .previous = active_thread,
        .next = next_thread,
    };

    int64_t res;

    bpf_hook_execute(BPF_HOOK_SCHED, &ctx, sizeof(ctx), &res);
}
```

```
main(): This is RIOT! (Version: 2021.
bpf scheduler example app
All up, running the shell now
> bpf_keyval
bpf_keyval
+-----+-----+
| key   | value |
+-----+-----+
|      2 |      5 |
|      3 |      5 |
|      6 |      3 |
+-----+-----+
```

Source: <https://summit.riot-os.org/2021/wp-content/uploads/sites/16/2021/09/s04-02.pdf>



## Results & Preliminary Analysis

■

	ROM size	RAM size
WASM3 Interpreter	64 KiB	85 KiB
rBPF Interpreter	4.4 KiB	0.6 KiB
RIOTjs	121 KiB	18 KiB
MicroPython	101 KiB	8.2 KiB
Host OS (without VM)	52.5 KiB	16.3 KiB

**Table 1: Memory requirements for VM interpreters.**

	code size	startup time	run time
Native C	74 B	–	27 µs
WASM3	322 B	17 096 µs	980 µs
rBPF	456 B	1 µs	2133 µs
RIOTjs	593 B	5589 µs	14 726 µs
MicroPython	497 B	21 907 µs	16 325 µs

**Table 2: Size and performance of fletcher32 logic hosted in different VMs.**

Source: <https://arxiv.org/pdf/2106.12553.pdf>



## ***Good Resources***

- [https://github.com/future-proof-iot/Femto-Container\\_tutorials](https://github.com/future-proof-iot/Femto-Container_tutorials)
- <https://github.com/bergzand/RIOT/tree/wip/bpf>
- <https://github.com/future-proof-iot/rBPF>
- ...

# XV. Tittle-tattle

## 1) eBPF for more Linux subsystems

### Overview

- ...





# 1.1 Scheduling

## 1.1.1 CPU Scheduler

- [https://lwn.net/Articles/873244/ //Controlling the CPU scheduler with BPF](https://lwn.net/Articles/873244/)  
<https://lwn.net/ml/linux-kernel/20210916162451.709260-1-guro@fb.com/>

Gushchin's patch set creates a new BPF program type (`BPF_PROG_TYPE_SCHED`) for programs that influence CPU-scheduler decisions. There are three attachment points for these programs:

- `cfs_check_preempt_tick` is called during the handling of the scheduler's periodic timer tick; a BPF program attached here can then look at which process is running. If that process should be allowed to continue to run, the hook can return a negative number to prevent preemption. A positive return value, instead, informs the scheduler that it should switch to a different process, thus forcing preemption to happen. Returning zero leaves the decision up to the scheduler as if the hook hadn't been run.
- `cfs_check_preempt_wakeup` is called when a process is woken by the kernel; a negative return value will prevent this process from preempting the currently running process, a positive value will force preemption, and zero leaves it up to the scheduler.
- `cfs_wakeup_preempt_entity` is similar to `cfs_check_preempt_wakeup`, but it is called whenever a new process is being selected for execution and can influence the decision. A negative return indicates no preemption, positive forces it, and zero leaves the decision to other parts of the scheduler.

Gushchin notes that, at Facebook, the first experiments using these hooks "look very promising". By posting the patch set, he hoped to start a conversation on how BPF could be used within the scheduler.



...



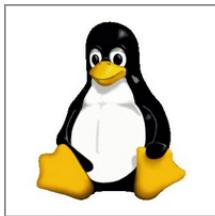
## 1.2 HID

### 1.2.1 Input Devices

- [https://www.phoronix.com/scan.php?page=news\\_item&px=Linux-eBPF-For-HID](https://www.phoronix.com/scan.php?page=news_item&px=Linux-eBPF-For-HID)

### Red Hat Eyeing Innovative eBPF Uses For Linux's HID Subsystem

Written by Michael Larabel in [Linux Kernel](#) on 24 February 2022 at 03:14 PM EST. [7 Comments](#)



eBPF for sandboxed programs running in the kernel have shown to be very useful beyond the original BPF origins in the networking subsystem to also be very practical for other security, tracing, and other general use-cases for an in-kernel JIT virtual machine. Red Hat has sent out initial patches extending eBPF for making use of it within the HID subsystem for input devices.

Benjamin Tissoires of Red Hat's elite input Linux team has sent out a set of patches introducing eBPF support for HID devices. One of the very useful areas this eBPF support can be used within the HID area is for buggy/quirky devices. Currently there is lots of simple drivers and quirks for just correcting a key or byte in the report descriptor for input events. Unfortunately with the current approach with the input drivers being in the mainline kernel and the time it takes for upstreaming and getting down to vendor kernels is painful for users. The idea is that these "fixups" could be externalized in some external repository and ship these fixes as various eBPF programs that would be loaded at boot time to avoid needing a new kernel for quirky/buggy hardware.



## ■ initial patches

<https://lore.kernel.org/lkml/20220224110828.2168231-1-benjamin.tissoires@redhat.com/>

Benjamin Tissoires (6):

HID: initial BPF implementation  
HID: bpf: allow to change the report descriptor from an eBPF program  
HID: bpf: add hid\_{get|set}\_data helpers  
HID: bpf: add new BPF type to trigger commands from userspace  
HID: bpf: tests: rely on uhid event to know if a test device is ready  
HID: bpf: add bpf\_hid\_raw\_request helper function

drivers/hid/Makefile	1 +
drivers/hid/hid-bpf.c	327 +++++++
drivers/hid/hid-core.c	31 +-
include/linux/bpf-hid.h	98 +++
include/linux/bpf_types.h	4 +
include/linux/hid.h	25 +
include/uapi/linux/bpf.h	33 +
include/uapi/linux/bpf_hid.h	56 ++
kernel/bpf/Makefile	3 +
kernel/bpf/hid.c	653 ++++++++++++++
kernel/bpf/syscall.c	12 +
samples/bpf/.gitignore	1 +
samples/bpf/Makefile	4 +
samples/bpf/hid_mouse_kern.c	91 ???
samples/bpf/hid_mouse_user.c	129 +++
tools/include/uapi/linux/bpf.h	33 +
tools/lib/bpf/libbpf.c	9 +
tools/lib/bpf/libbpf.h	2 +
tools/lib/bpf/libbpf.map	1 +
tools/testing/selftests/bpf/prog_tests/hid.c	685 ++++++++++++++
tools/testing/selftests/bpf/progs/hid.c	149 +++
21 files changed, 2339 insertions(+), 8 deletions(-)	
create mode 100644 drivers/hid/hid-bpf.c	
create mode 100644 include/linux/bpf-hid.h	
create mode 100644 include/uapi/linux/bpf_hid.h	
create mode 100644 kernel/bpf/hid.c	
create mode 100644 samples/bpf/hid_mouse_kern.c	
create mode 100644 samples/bpf/hid_mouse_user.c	
create mode 100644 tools/testing/selftests/bpf/prog_tests/hid.c	
create mode 100644 tools/testing/selftests/bpf/progs/hid.c	

## 2) eBPF in Edge Computing

### Overview

- ...
- 



# 2.1 fledge.io

## Overview

- <https://www.fledge.io/>  
**Evolving multi-cloud and edge applications.**

- **Use Cases**

fledge.io Cloud offers a turnkey solution for several use cases that require applications to be distributed geographically across different clouds, datacenters and edges.



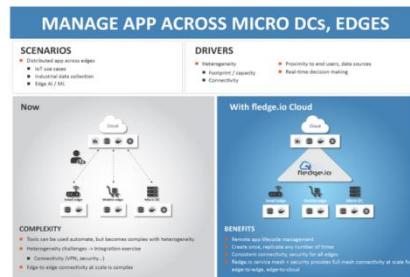
### Entering New Markets?

Are you expanding your services in a new geography?



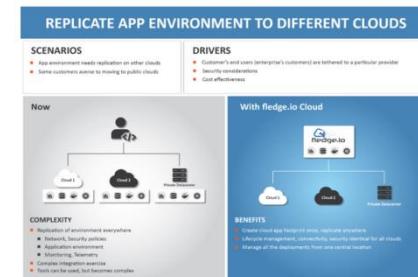
### Need App Bursting?

Do you need app bursting to manage demand spikes?



### Managing Edge Apps?

Are you looking to manage apps spanning multiple edge locations?



### Need App Replication?

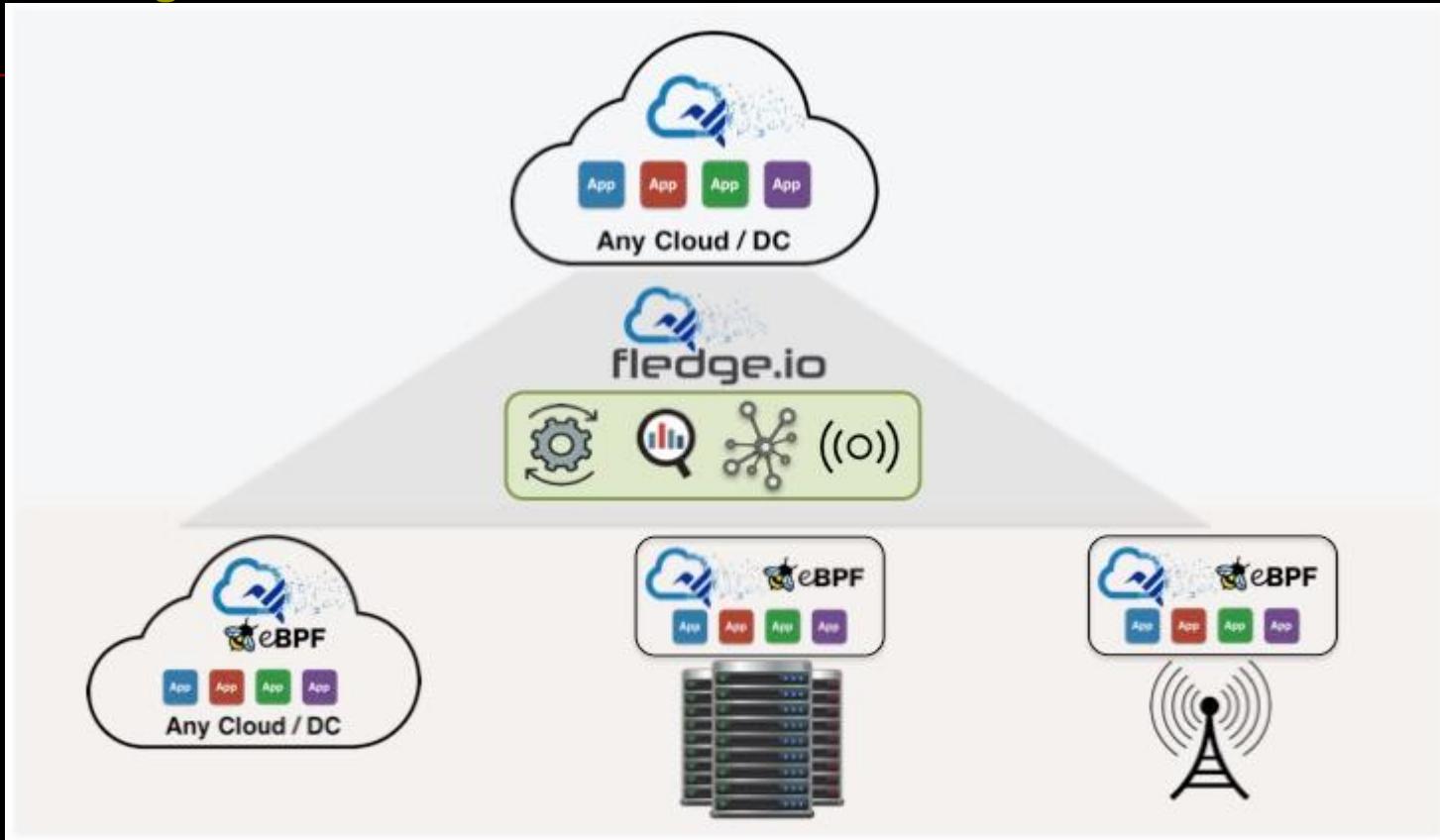
Do you need your app to be replicated on different providers?





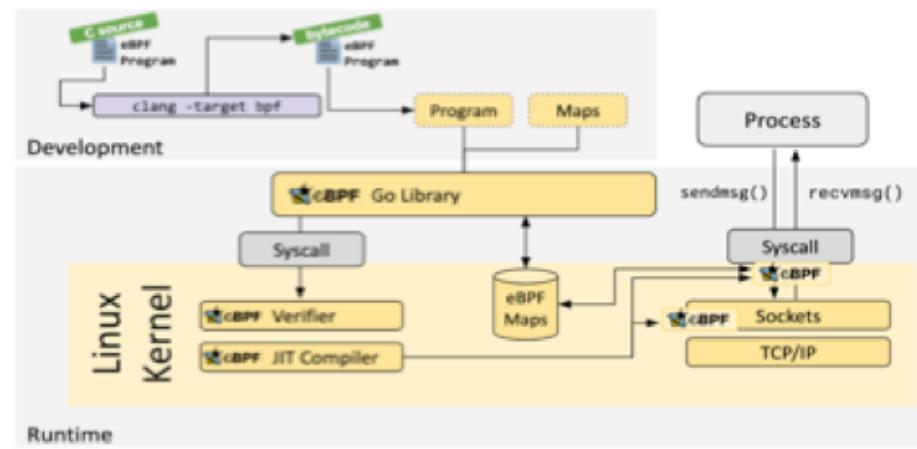
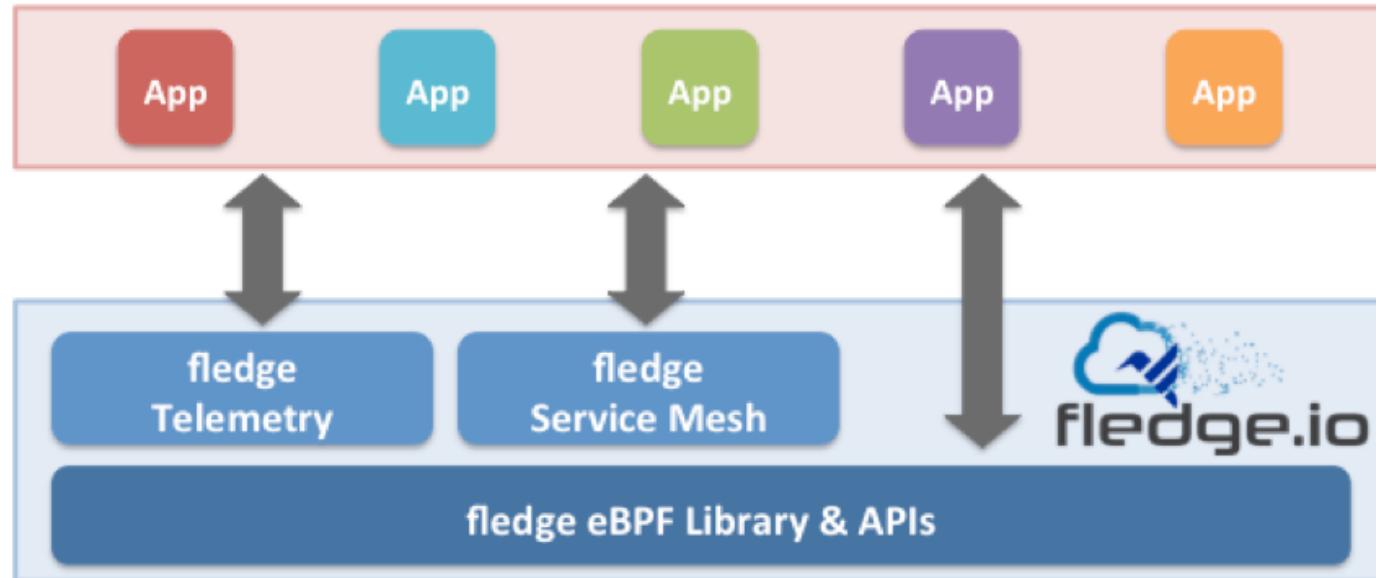
## 2.1.1 brings eBPF to multi-cloud and edge

- <https://www.fledge.io/2021/02/24/fledge-io-brings-ebpf-to-multi-cloud-and-edge/>





## Compute Node (Any Cloud or Edge)





### 3) Accelerating Python

#### 3.1 Why is Python so important

##### 3.1.1 Ranking

###### TIOBE

- <https://www.tiobe.com/tiobe-index/>

Oct 2021	Oct 2020	Change	Programming Language	Ratings	Change
1	3	▲	Python	11.27%	-0.00%
2	1	▼	C	11.16%	-5.79%
3	2	▼	Java	10.46%	-2.11%
4	4		C++	7.50%	+0.57%
5	5		C#	5.26%	+1.10%
6	6		Visual Basic	5.24%	+1.27%
7	7		JavaScript	2.19%	+0.05%
8	10	▲	SQL	2.17%	+0.61%
9	8	▼	PHP	2.10%	+0.01%
10	17	▲	Assembly language	2.06%	+0.99%

###### PYPL

- <http://pypl.github.io/PYPL.html>

Worldwide, Oct 2021 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	29.66 %	-2.1 %
2		Java	17.18 %	+0.8 %
3		JavaScript	8.81 %	+0.4 %
4		C#	7.3 %	+1.1 %
5	▲	C/C++	6.48 %	+0.7 %
6	▼	PHP	5.92 %	+0.1 %
7		R	4.09 %	+0.2 %
8		Objective-C	2.24 %	-1.2 %
9	▲	TypeScript	1.91 %	+0.1 %
10	▲▲	Kotlin	1.9 %	+0.3 %

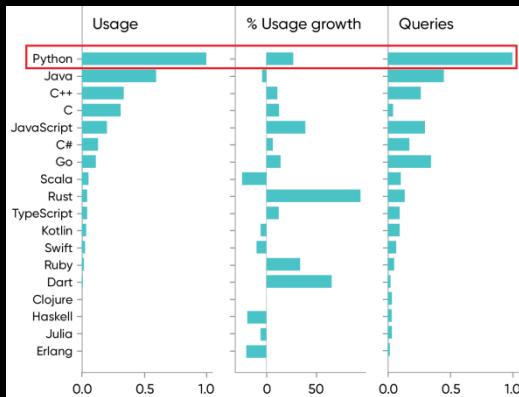


- <https://spectrum.ieee.org/top-programming-languages/>

1	Python~	🌐	💻	⚙️	100.0
2	Java~	🌐	💻	⚙️	95.4
3	C~	💻	⚙️	⚙️	94.7
4	C++~	💻	🌐	⚙️	92.4
5	JavaScript~	🌐			88.1
6	C#~	🌐	💻	⚙️	82.4
7	R~		💻		81.7
8	Go~	🌐	💻		77.7
9	HTML~	🌐			75.4
10	Swift~	💻	🌐		70.4

## O'Reilly

- <https://www.oreilly.com/radar/where-programming-ops-ai-and-the-cloud-are-headed-in-2021/>



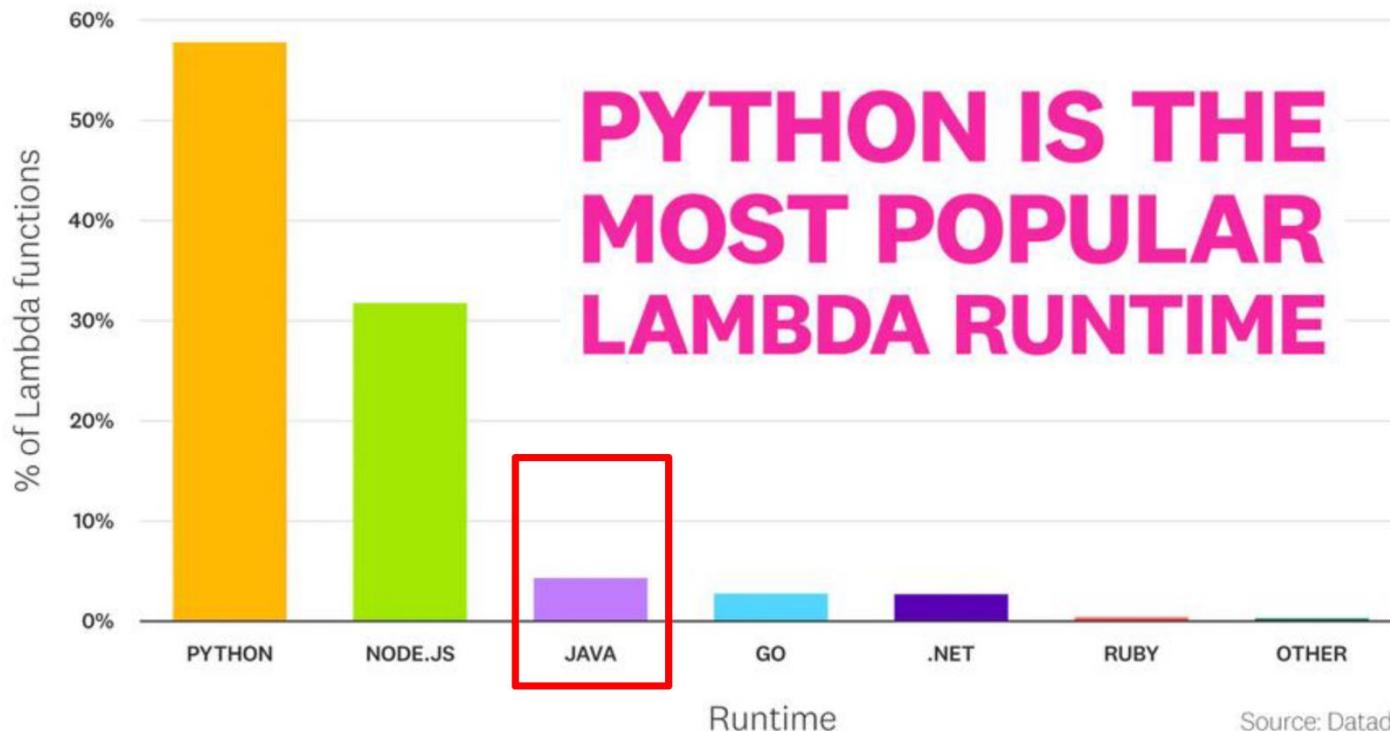


## 3.1.2 Python in Cloud

### AWS Lambda

- 

Most Popular Runtimes by Distinct Functions



**PYTHON IS THE  
MOST POPULAR  
LAMBDA RUNTIME**



## 3.2 Making Python faster

- For details, you may refer to my previous talk "**A survey of current Python implementations**" at PyCon China 2021(Online) and the upcoming follow-ups.
-



## 4) Lua

### 4.1 Overview

- [https://en.wikipedia.org/wiki/Lua\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Lua_(programming_language))
  - <https://www.lua.org/>
-

# Lua 5.4



## ■

### ❖ Changes since Lua 5.3

Here are the main changes introduced in Lua 5.4. The [reference manual](#) lists the [incompatibilities](#) that had to be introduced.

#### Main changes

- new generational mode for garbage collection
- to-be-closed variables
- const variables
- userdata can have multiple user values
- new implementation for math.random
- warning system
- debug information about function arguments and returns
- new semantics for the integer 'for' loop
- optional 'init' argument to 'string.gmatch'
- new functions 'lua\_resetthread' and 'coroutine.close'
- string-to-number coercions moved to the string library
- allocation function allowed to fail when shrinking a memory block
- new format '%p' in 'string.format'
- utf8 library accepts codepoints up to 2^31

## ■

<https://www.lua.org/manual/5.4/contents.html>

## ■

<https://www.lua.org/manual/5.4/manual.html>

## ■

<https://lwn.net/Articles/826134/>

//What's new in Lua 5.4

## ■

...



## 4.2 Redis

### 4.2.1 Redis 6.x

- <https://redis.com/blog/diving-into-redis-6/>
- <https://www.infoworld.com/article/3541356/redis-6-arrives-with-multithreading-for-faster-io.html>
- ...



## 4.2.2 Redis 7.x

- <https://github.com/redis/redis/releases/tag/7.0-rc1>

Redis 7.0 includes several new user-facing features, significant performance optimizations, and many other improvements. It also includes changes that potentially break backwards compatibility with older versions. We urge users to review the release notes carefully before upgrading.

In particular, users should be aware of the following changes:

1. Redis 7 stores AOF as multiple files in a folder; see Multi-Part AOF below.
2. Redis 7 uses a new version 10 format for RDB files, which is incompatible with older versions.
3. Redis 7 converts ziplist encoded keys to listpacks on the fly when loading an older RDB format. Conversion applies to loading a file from disk or replicating from a Redis master and will slightly increase loading time.
4. See sections about breaking changes mentioned below.



...



## 4.2.3 Rust for Redis

- [\*\*https://github.com/RedisLabsModules/redismodule-rs\*\*](https://github.com/RedisLabsModules/redismodule-rs)

This crate provides an idiomatic Rust API for the [Redis Modules API](#). It allows writing Redis modules in Rust, without needing to use raw pointers or unsafe code.

- [\*\*https://github.com/mitsuhiko/redis-rs\*\*](https://github.com/mitsuhiko/redis-rs)
- 

- [\*\*https://github.com/RediSearch/redisearch-api-rs\*\*](https://github.com/RediSearch/redisearch-api-rs)
- [\*\*https://github.com/dpbriggs/redis-oxide\*\*](https://github.com/dpbriggs/redis-oxide)
- ...



## 4.3 RedisJSON Overview

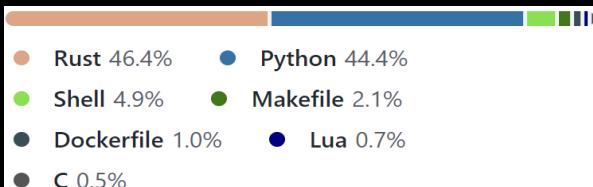
### ■ <https://oss.redis.com/redisjson/>

RedisJSON is a [Redis](#) module that implements [ECMA-404 The JSON Data Interchange Standard](#) as a native data type. It allows storing, updating and fetching JSON values from Redis keys (documents).

Primary features:

- Full support of the JSON standard
- [JSONPath](#)-like syntax for selecting elements inside documents
- Documents are stored as binary data in a tree structure, allowing fast access to sub-elements
- Typed atomic operations for all JSON values types

### ■ <https://github.com/RedisJSON/RedisJSON> Languages (without submodules)



■ ...

## ***Performance***

- <https://redis.com/blog/redisjson-public-preview-performance-benchmarking/>
- <https://oss.redis.com/redisjson/performance/>
- ~~<https://oss.redis.com/redisjson/ram/>~~
- ...





## 4.4 Lua & Rust

### 4.4.1 mlua

#### ■ <https://github.com/khvzak/mlua>

`mlua` is bindings to [Lua](#) programming language for Rust with a goal to provide *safe* (as far as it's possible), high level, easy to use, practical and flexible API.

Started as `rlua` fork, `mlua` supports Lua 5.4, 5.3, 5.2 and 5.1 including LuaJIT (2.0.5 and 2.1 beta) and allows to write native Lua modules in Rust as well as use Lua in a standalone mode.

`mlua` tested on Windows/macOS/Linux including module mode in [GitHub Actions](#) on `x86_64` platform and cross-compilation to `aarch64` (other targets are also supported).

#### ■ Feature flags

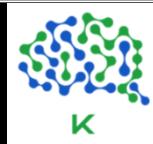
`mlua` uses feature flags to reduce the amount of dependencies, compiled code and allow to choose only required set of features. Below is a list of the available feature flags. By default `mlua` does not enable any features.

- `lua54` : activate Lua [5.4](#) support
- `lua53` : activate Lua [5.3](#) support
- `lua52` : activate Lua [5.2](#) support
- `lua51` : activate Lua [5.1](#) support
- `luajit` : activate [LuaJIT](#) support
- `luajit52` : activate [LuaJIT](#) support with partial compatibility with Lua 5.2
- `vendored` : build static Lua(JIT) library from sources during `mlua` compilation using `lua-src` or `luajit-src` crates
- `module` : enable module mode (building loadable `cdylib` library for Lua)
- `async` : enable async/await support (any executor can be used, eg. `tokio` or `async-std`)
- `send` : make `mlua::Lua` transferable across thread boundaries (adds `Send` requirement to `mlua::Function` and `mlua::UserData`)
- `serialize` : add serialization and deserialization support to `mlua` types using `serde` framework
- `macros` : enable procedural macros (such as `chunk!` )

## 4.5 OpenResty

### Overview

- <https://openresty.org/>
  - o
- 





## 4.6 Lua in Kernel

### Overview

- <https://lwn.net/Articles/830154/> //Lua in the kernel?
  - You may also refer to my previous talk "**In-Kernel VM & Service**" at OSDT 2019(Beijing).
-

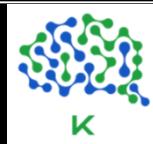


## 4.7 Good Resources

- <https://redis.com/blog/diving-into-redis-6/>
- <https://github.com/eatonphil/lust>
- <https://github.com/danii/hepatita>
- <https://github.com/amethyst/rlua>
- <https://cloud.tencent.com/developer/article/1922607>
- ...

## 5) Integration Overview

- ...
  - ...
- 





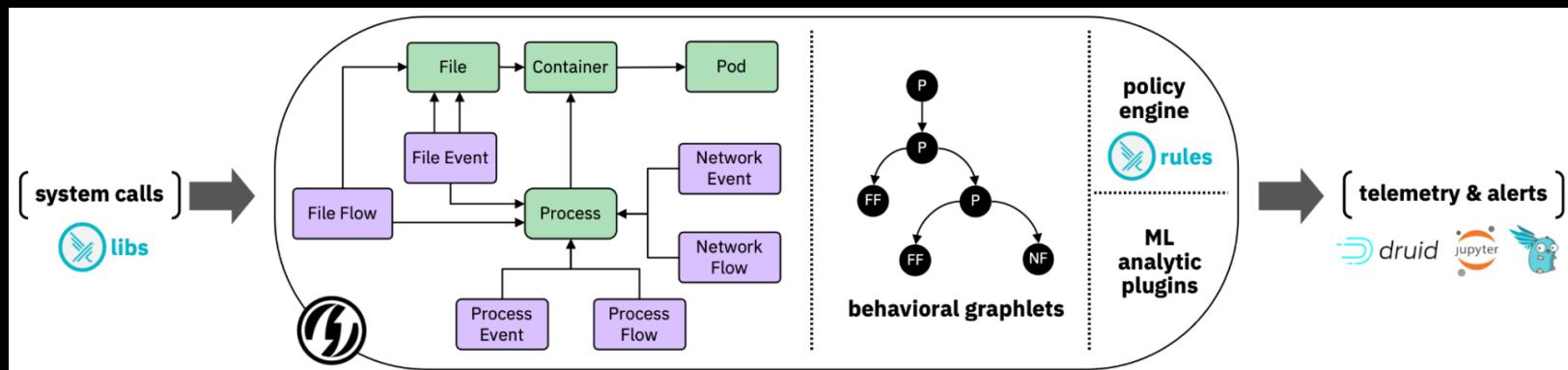
## 5.1 SysFlow and Falco

- <https://falco.org/blog/sysflow-falco-sidekick/>

### SysFlow

- <https://sysflow.io/>

A cloud-native system telemetry framework that enables the creation of security analytics on a scalable, pluggable open-source platform.



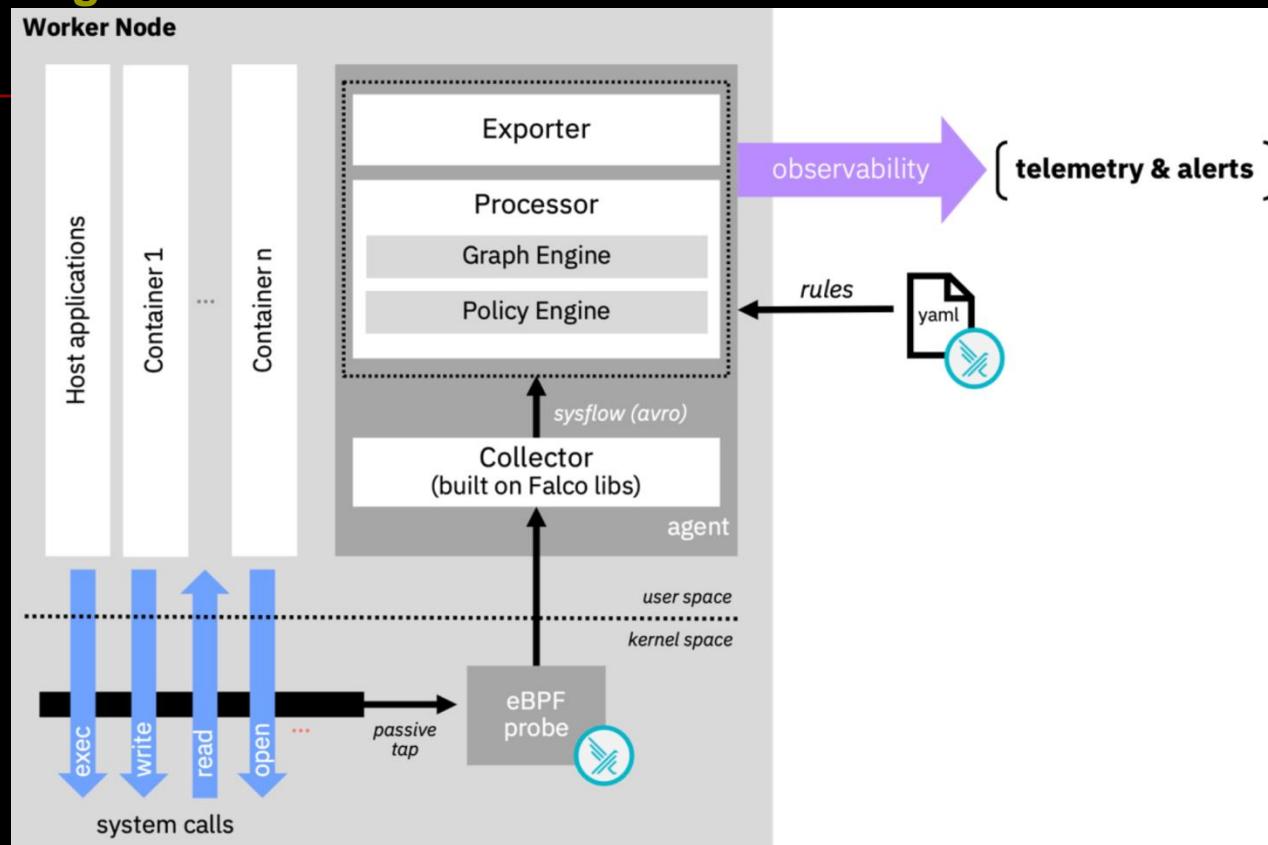
### What challenges does it solve?

The following are the key reasons that set SysFlow apart:

- Reduces data footprints drastically when compared to raw system call collection.
- Reduces event fatigue (a.k.a. "too many alerts") thanks to its underlying entity-relational model and flow abstractions.
- Minimizes the requirements for hard filters.
- Provides useful context by linking together system event data at the data format level. This expedites security analysis, enabling analytics beyond simple manual policies, including machine learning, stateful analytics, and automated policy generation.
- Treats system security monitoring as a data science problem through rapid edge analytics building, easy-to-use APIs, and support for multiple data serialization and export formats.



- The framework builds on **Falco libs** and the **Falco rules language** to create the plumbing required for system telemetry as shown in the diagram below:



- <https://github.com/falcosecurity/libs>
- The SysFlow framework is designed as a **pluggable edge processing architecture**, which includes a policy engine, and what's more noteworthy, an experimental graph engine.



- <https://github.com/falcosecurity/plugins>
- Beyond the built-in plugins and exporters provided in the SysFlow stack, users can write and plug their own real-time analytics, and consume new telemetry sources using our Golang APIs. The framework also includes Python packages and a pre-built Jupyter container to facilitate interactive SysFlow data exploration.
- <https://github.com/sysflow-telemetry/sf-processor>

## Sub-projects

The SysFlow framework consists of the following sub-projects:

- [sf-apis](#) provides the SysFlow schema and programmatic APIs in go, python, and C++.
- [sf-collector](#) monitors and collects system call and event information from hosts and exports them in the SysFlow format using Apache Avro object serialization.
- [sf-processor](#) provides a performance optimized policy engine for processing, enriching, filtering SysFlow events, generating alerts, and exporting the processed data to various targets.
- [sf-exporter](#) exports SysFlow traces to S3-compliant storage systems for archival purposes.
- [sf-deployments](#) contains deployment packages for SysFlow, including Docker, Helm, and OpenShift.
- [sysflow](#) is the documentation repository and issue tracker for the SysFlow framework.

...

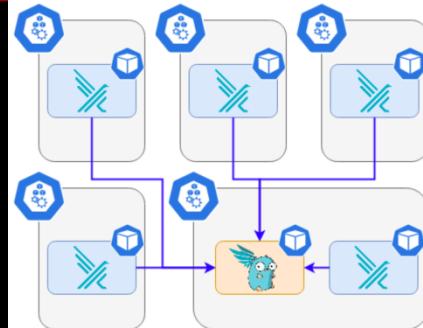
# Falco Sidekick

- <https://github.com/falcosecurity/falcosidekick>  
**Connect Falco to your ecosystem.**



A simple daemon for connecting [Falco](#) to your ecosystem. It takes a [Falco](#)'s events and forward them to different outputs in a fan-out way.

It works as a single endpoint for as many as you want [Falco](#) instances :



The screenshot shows the Falcosidekick UI interface. On the left, there's a sidebar with a dashboard, events, query data, and Jupyter Notebook options. The main area displays a log entry from 2021-10-05 02:40:23:

**Suspicious behavior observed in application spawning another process**

**critical** Suspicious process spawned

**Process Tree:**

- P 22300425748293072603362824470214162495  
/usr/local/bin/node [app.js]  
2025, 0  
root 0 root 0 False
- FF (1531776715030353952,=) EXEC (1531776715041743005,=)
- 1615670594600517595414854311475367655490  
1, 0, 6537, 1 /tmp/exfil.py
- FF 98283704851029983858613304370313093499  
/bin/sh [-c /tmp/exfil.py -a]  
17753, 1531776715040686177  
root 0 root 0 False  
mите: T1106, mите: T1574
- FF (1531776715048770544,=) CLONE (1531776715048770544,=)
- 18693389293292062040211194925249940032  
2, 832, 1, 0, 0 /lib/x86\_64-linux-gnu/libc.so.6
- FF 296923321479947111565410335072170460001  
/bin/sh [-c /tmp/exfil.py -a]  
17754, 1531776715048770544  
root 0 root 0 False

**Container entropy "node" starts shell sub-process**

**Event Headers:**

- of\_type: PE of\_node\_id: 10.187.39.243 of\_uefi\_exec: 0 of\_proc\_id: 17753 of\_uefi\_log: mите: T1106, mите: T1574 of\_container\_proileged: 0 of\_container\_id: 6021266160112490253 of\_uefi\_uefi\_id: 664594064440fbc of\_uefi\_uefi\_name: /bin/sh -c /tmp/exfil.py + of\_uefi\_uefi\_version: 0
- of\_uefi\_uefi\_image: nodejs@sha256:c822730f88bf58d31169228e029bd36277e5caf074eaad20b4960b1a33e953/



## 6) Emerging Technologies

### 6.1 UMCG

#### ■ User Managed Concurrency Groups

The UMCG kernel code is summed up as "*User Managed Concurrency Groups is a fast context switching and in-process userspace scheduling framework. Two main use cases are security sandboxes and userspace scheduling. Security sandboxes: fast X-process context switching will open up a bunch of light-weight security tools, e.g. gVisor, or Tor Project's Shadow simulator, to more use cases. In-process userspace scheduling is used extensively at Google to provide latency control and isolation guarantees for diverse workloads while maintaining high CPU utilization.*"

The UMCG build switch adds, "*Enable UMCG core wait/wake/swap operations as well as UMCG group/server/worker API. The core API is useful for fast IPC and context switching, while the group/server/worker API, together with the core API, form the basis for an in-process M:N userspace scheduling framework implemented in lib/umcg.*"

- <https://lore.kernel.org/lkml/20211012232522.714898-1-posk@google.com/>

#### Fibers User-Space Scheduling Framework

- ...



## Good Resources

- [https://www.phoronix.com/scan.php?page=news\\_item&px=Google-UMCG-Linux-v0.7](https://www.phoronix.com/scan.php?page=news_item&px=Google-UMCG-Linux-v0.7)
- [https://www.phoronix.com/scan.php?page=news\\_item&px=Google-UMCG-0.2-Fibers](https://www.phoronix.com/scan.php?page=news_item&px=Google-UMCG-0.2-Fibers)
- [https://www.phoronix.com/scan.php?page=news\\_item&px=Google-Fibers-Toward-Open](https://www.phoronix.com/scan.php?page=news_item&px=Google-Fibers-Toward-Open)
- ...



## 6.2 ghOSt

- **Fast & Flexible User-Space Delegation of Linux Scheduling**
- **<https://github.com/google/ghost-userspace>**

ghOSt is a general-purpose delegation of scheduling policy implemented on top of the Linux kernel. The ghOSt framework provides a rich API that receives scheduling decisions for processes from userspace and actuates them as transactions. Programmers can use any language or tools to develop policies, which can be upgraded without a machine reboot. ghOSt supports policies for a range of scheduling objectives, from  $\mu$ s-scale latency, to throughput, to energy efficiency, and beyond, and incurs low overheads for scheduling actions. Many policies are just a few hundred lines of code. Overall, ghOSt provides a performant framework for delegation of thread scheduling policy to userspace processes that enables policy optimization, non-disruptive upgrades, and fault isolation.

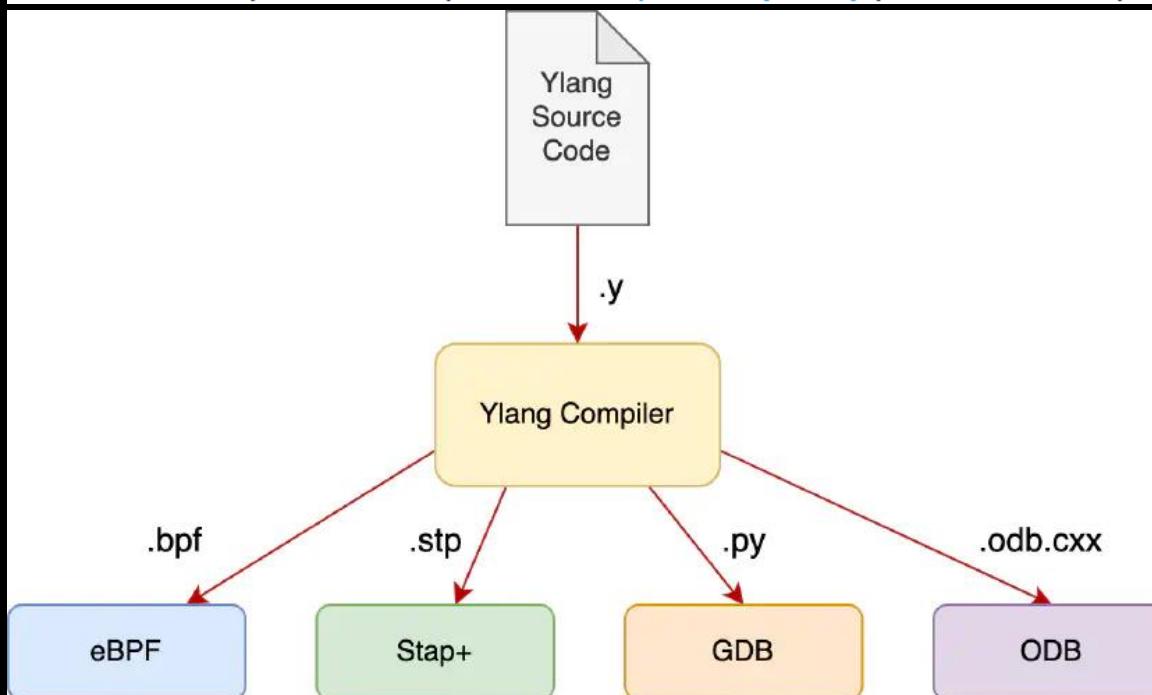
- **<https://github.com/google/ghost-kernel>**
- ...



## 6.3 Ylang

- <https://doc.openresty.com.cn/en/ylang/>  
**Universal Language for eBPF, Stap+, GDB, and More.**
- <https://blog.openresty.com/en/ylang-intro-part1/>

Y or Ylang is a universal **dynamic tracing** language that targets various **dynamic tracing** frameworks and toolchains. We provide it as part of the [OpenResty XRay](#) platform developed by [OpenResty Inc.](#)



- <https://blog.openresty.com/en/ylang-intro-part2/>
- <https://openresty.com/en/xray/>
- ...

# XVI. Rethinking the future



## Overview

- 
-



# 1) Unikernel Overview

## ■ <https://en.wikipedia.org/wiki/Unikernel>

A **unikernel** is a specialised, [single address space](#) machine image constructed by using [library operating systems](#).<sup>[1]</sup> A developer selects, from a modular stack, the minimal set of libraries which correspond to the OS constructs required for the application to run. These libraries are then compiled with the application and configuration code to build sealed, fixed-purpose images (unikernels) which run directly on a [hypervisor](#) or [hardware](#) without an intervening OS such as Linux or Windows.

The first such systems were [Exokernel](#) and [Nemesis](#) in the late 1990s.

## ■ [https://en.wikipedia.org/wiki/Single\\_address\\_space\\_operating\\_system](https://en.wikipedia.org/wiki/Single_address_space_operating_system)

In computer science, a [single address space operating system](#) (or **SASOS**) is an [operating system](#) that provides only one globally shared [address space](#) for all [processes](#).

In a single address space operating system, numerically identical ([virtual memory](#)) [logical addresses](#) in different processes all refer to exactly the same byte of data.<sup>[1]</sup>

Single address-space operating systems offer many advantages. In a traditional OS with private per-process address space, memory protection is based on address space boundaries ("address space isolation"). Single address-space operating systems use a different approach for memory protection that is just as strong.<sup>[2][3]</sup>

One advantage is that the same virtual-to-physical map [page table](#) can be used with every process (and in some SASOS, the kernel as well). This makes context switches on a SASOS faster than on operating systems that must change the page table and flush the TLB caches on every context switch.

## ■ [https://en.wikipedia.org/wiki/Operating\\_system#Library](https://en.wikipedia.org/wiki/Operating_system#Library)

A library operating system is one in which the services that a typical operating system provides, such as networking, are provided in the form of [libraries](#) and composed with the application and configuration code to construct a [unikernel](#): a [specialized, single address space](#), machine image that can be deployed to cloud or embedded environments.

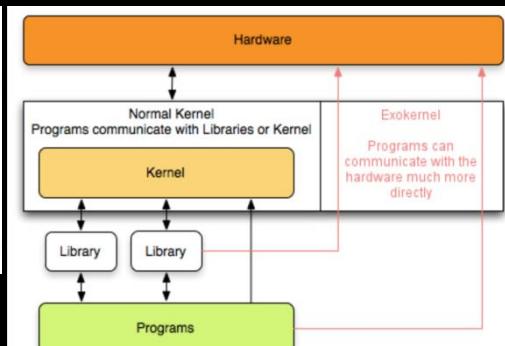
## ■ <https://en.wikipedia.org/wiki/Exokernel>

**Exokernel** is an [operating system kernel](#) developed by the [MIT Parallel and Distributed Operating Systems group](#),<sup>[1]</sup> and also a class of similar operating systems.

Operating systems generally present hardware resources to applications through high-level abstractions such as (virtual) file systems. The idea behind exokernels is to force as few abstractions as possible on application developers, enabling them to make as many decisions as possible about hardware abstractions.<sup>[2]</sup> Exokernels are tiny, since functionality is limited to ensuring protection and [multiplexing](#) of resources, which is considerably simpler than conventional microkernels' implementation of message passing and [monolithic kernels'](#) implementation of high-level abstractions.

Implemented applications are called library operating systems; they may request specific memory addresses, disk blocks, etc. The kernel only ensures that the requested resource is free, and the application is allowed to access it. This low-level hardware access allows the programmer to implement custom abstractions, and omit unnecessary ones, most commonly to improve a program's performance. It also allows programmers to choose what level of abstraction they want, high, or low.

Exokernels can be seen as an application of the [end-to-end principle](#) to operating systems, in that they do not force an application program to layer its abstractions on top of other abstractions that were designed with different requirements in mind. For example, in the MIT Exokernel project, the Cheetah [web server](#) stores preformatted Internet Protocol packets on the disk, the kernel provides safe access to the disk by preventing unauthorized reading and writing, but how the disk is abstracted is up to the application or the libraries the application uses.



Graphic overview of Exokernel. Exokernels are much smaller than a normal kernel ([monolithic kernel](#)). They give more direct access to the hardware, thus removing most abstractions



# Design

In a library operating system, protection boundaries are pushed to the lowest hardware layers, resulting in:

1. a set of libraries that implement mechanisms such as those needed to drive hardware or talk network protocols;
2. a set of policies that enforce access control and isolation in the application layer.

The library OS architecture has several advantages and disadvantages compared with conventional OS designs. One of the advantages is that since there is only a single address space, there is no need for repeated privilege transitions to move data between user space and kernel space. Therefore, a library OS can provide improved performance by allowing direct access to hardware without having to transition between user mode and kernel mode (on a traditional kernel this transition consists of a single TRAP instruction<sup>[2]</sup> and is not the same as a context switch<sup>[3]</sup>). Performance gains may be realised by elimination of the need to copy data between user space and kernel space, although this is also possible with Zero-copy device drivers in traditional operating systems.

A disadvantage is that because there is no separation, trying to run multiple applications side by side in a library OS, but with strong resource isolation, can become complex.<sup>[4]</sup> In addition, device drivers are required for the specific hardware the library OS runs on. Since hardware is rapidly changing this creates the burden of regularly rewriting drivers to remain up to date.

OS virtualization can overcome some of these drawbacks on commodity hardware. A modern [hypervisor](#) provides virtual machines with CPU time and strongly isolated virtual devices. A library OS running as a virtual machine only needs to implement drivers for these stable virtual hardware devices and can depend on the hypervisor to drive the real physical hardware. However, protocol libraries are still needed to replace the services of a traditional operating system. Creating these protocol libraries is where the bulk of the work lies when implementing a modern library OS.<sup>[1]</sup> Additionally, reliance on a hypervisor may reintroduce performance overheads when switching between the unikernel and hypervisor, and when passing data to and from hypervisor virtual devices.

By reducing the amount of code deployed, unikernels necessarily reduce the likely [attack surface](#) and therefore have improved security properties.<sup>[5][6]</sup>

An example unikernel-based messaging client has around 4% the size of the equivalent code bases using Linux.<sup>[7]</sup>

Due to the nature of their construction, it is possible to perform whole-system optimisation across device drivers and application logic, thus improving on the specialisation.<sup>[8][9]</sup> For example, off-the-shelf applications such as nginx, SQLite, and Redis running over a unikernel have shown a 1.7x-2.7x performance improvement.<sup>[10]</sup>

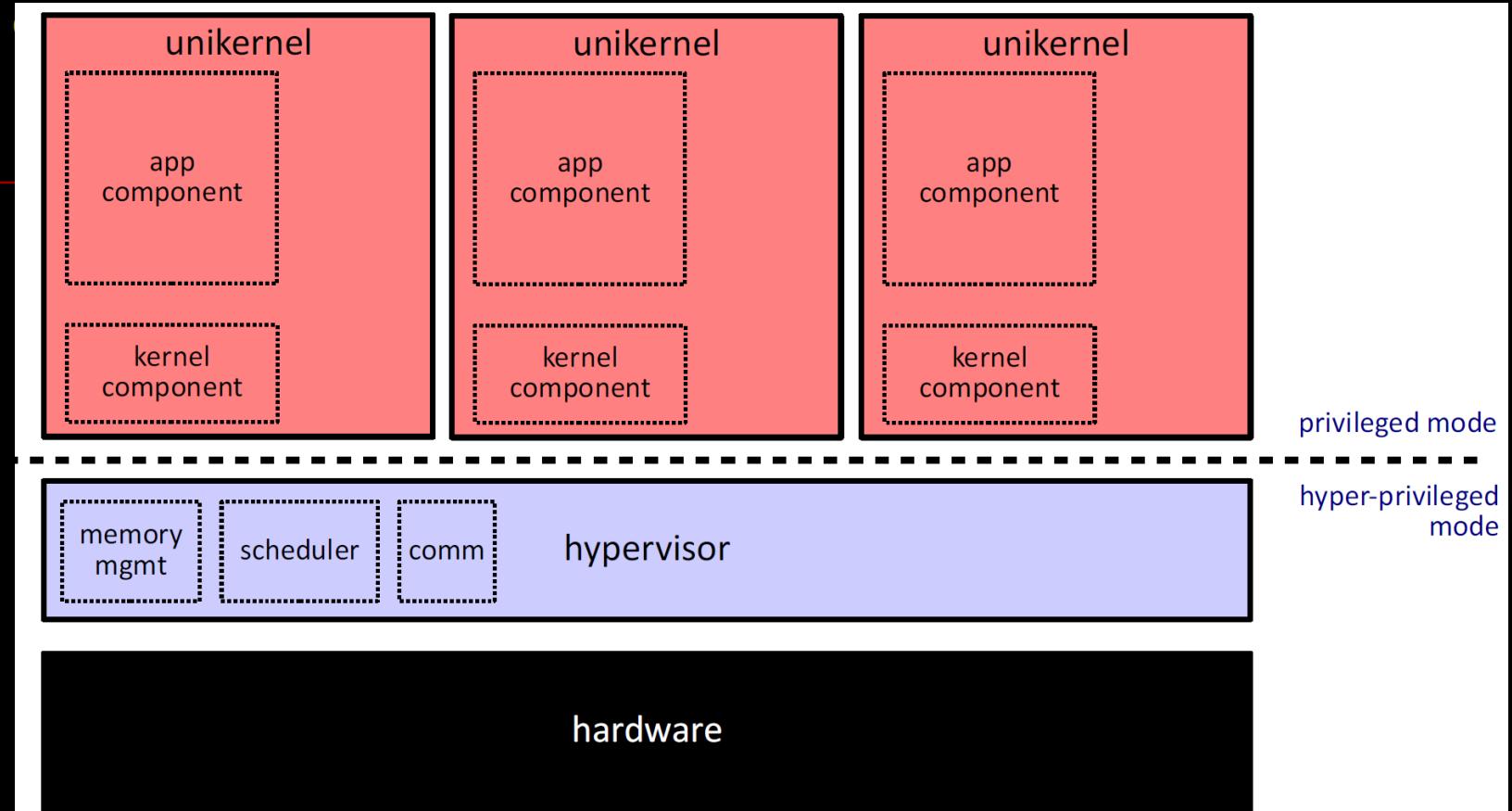
Unikernels have been regularly shown to boot extremely quickly, in time to respond to incoming requests before the requests time-out.<sup>[11][12][13]</sup>

Unikernels lend themselves to creating systems that follow the [service-oriented](#) or [microservices](#) software architectures.

The high degree of specialisation means that unikernels are unsuitable for the kind of general purpose, multi-user computing that traditional operating systems are used for. Adding additional functionality or altering a compiled unikernel is generally not possible and instead the approach is to compile and deploy a new unikernel with the desired changes.



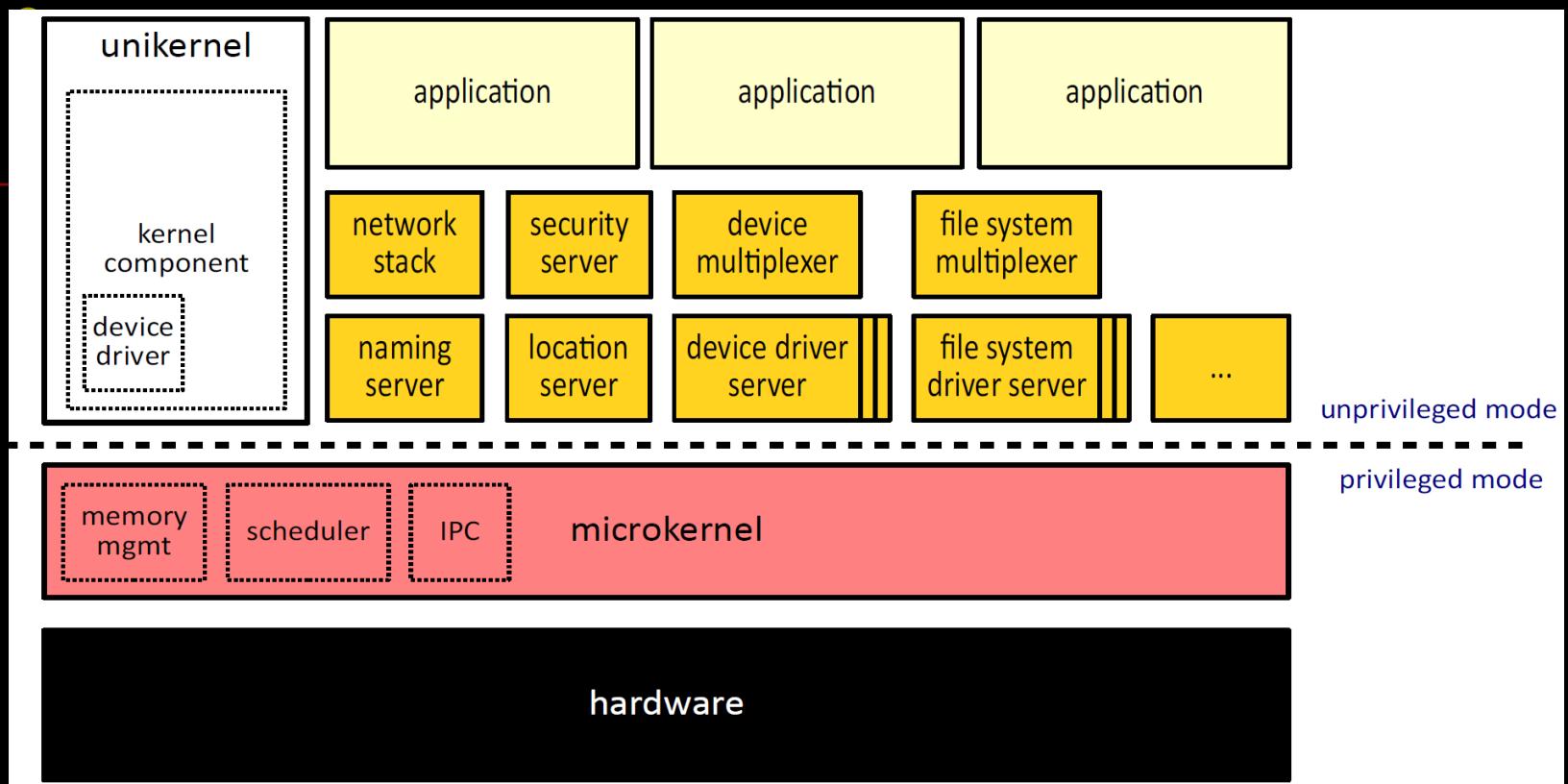
## Hypervisor with Unikernels



Source: <https://d3s.mff.cuni.cz/files/teaching/nswi161/martin-decky-microkernels-capabilities.pdf>



## Multiserver microkernel with unikernels for device drivers



Source: <https://d3s.mff.cuni.cz/files/teaching/nswi161/martin-decky-microkernels-capabilities.pdf>

## 2) Rust-based Cloud Infrastructure

### Overview

- <https://foundation.rust-lang.org/news/2021-11-16-news-announcing-cloud-compute-initiative/>
- <https://thenewstack.io/the-case-for-rust-as-the-future-of-javascript-infrastructure/>
- <https://rust-cloud-native.github.io/meetup/20-12-2021.html>
- <https://www.infoq.com/news/2021/08/linkerd-rust-cloud-native/>
- <https://events.linuxfoundation.org/cloud-native-rust-day/>
- ...

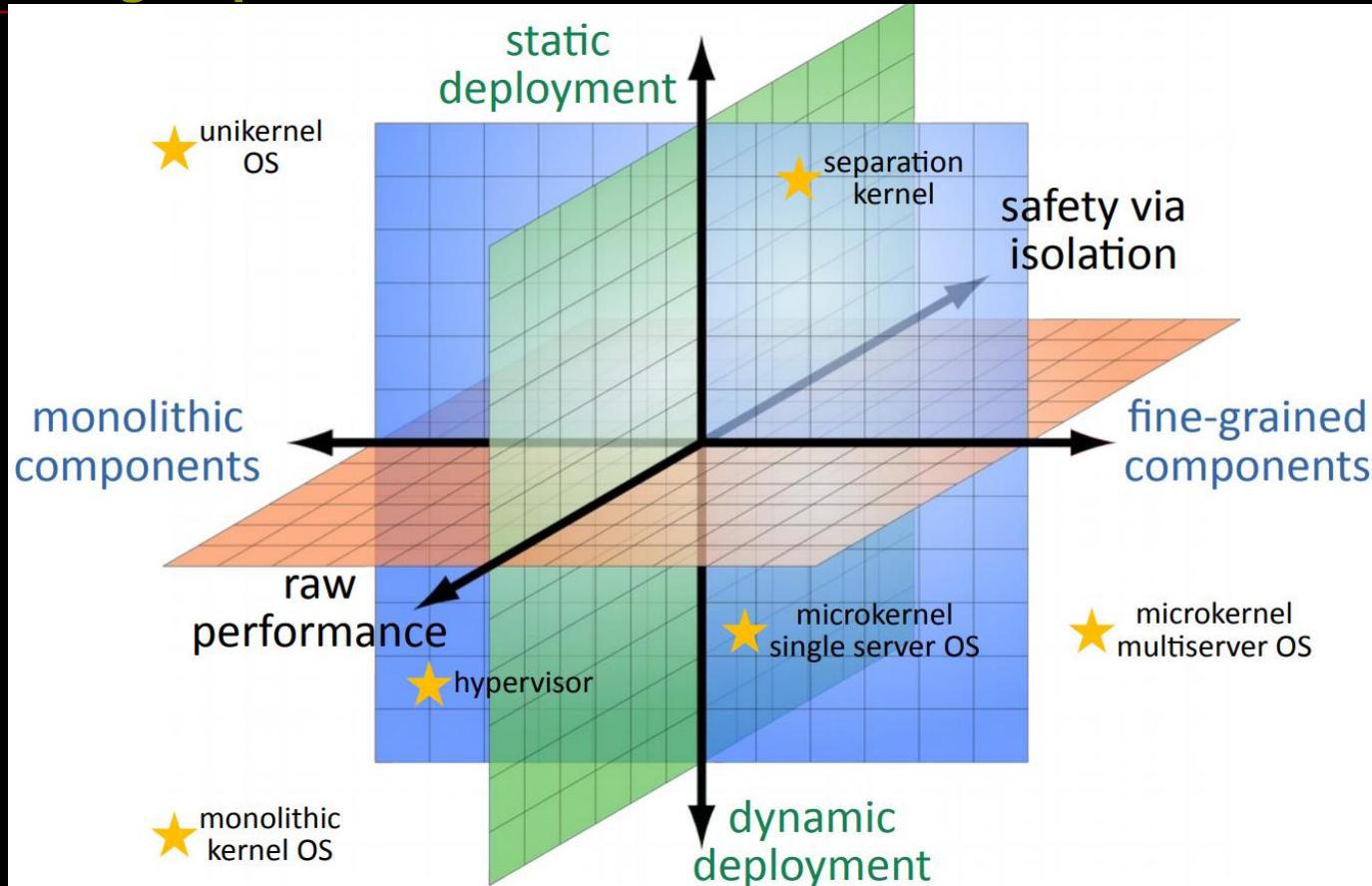


### 3) Innovations in Edge Infrastructure

#### 3.1 Novel Operating Systems

##### Overview

###### ■ Design Space of OSs



Source: <https://d3s.mff.cuni.cz/files/teaching/nswi161/martin-decky-microkernels-capabilities.pdf>





## 3.1.1 Pop!\_OS Overview

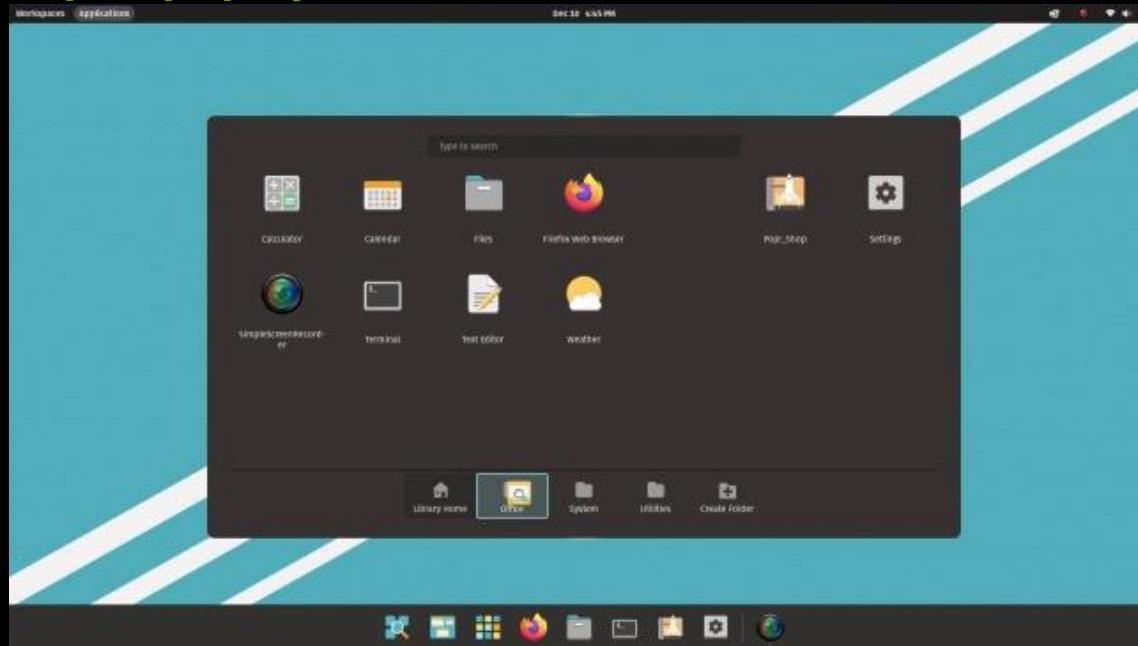
- [https://en.wikipedia.org/wiki/Pop!\\_OS](https://en.wikipedia.org/wiki/Pop!_OS)

Pop!\_OS is a free and open-source Linux distribution, based upon Ubuntu, and featuring a GTK-based desktop environment known as COSMIC, which is based on GNOME. The distribution is developed by American Linux computer manufacturer System76. Pop!\_OS is primarily built to be bundled with the computers built by System76, but can also be downloaded and installed on most computers.<sup>[3]</sup>

Pop!\_OS provides full out-of-the-box support for both AMD and Nvidia GPUs. It is regarded as an easy distribution to set up for gaming, mainly due to its built-in GPU support. Pop!\_OS provides default disk encryption, streamlined window and workspace management, keyboard shortcuts for navigation as well as built-in power management profiles. The latest releases also have packages that allow for easy setup for TensorFlow and CUDA.<sup>[4][5]</sup>

Pop!\_OS is maintained primarily by System76, with the release version source code hosted in a GitHub repository. Unlike many other Linux distributions, it is not community-driven, although outside programmers can contribute, view and modify the source code. They can also build custom ISO images and redistribute them under another name.<sup>[6][7]</sup>

- <https://pop.system76.com/>



- <https://itsfoss.com/why-use-pop-os/>
- ...



### 3.1.1.1 Rust in Pop!\_OS

#### Rust-Written Desktop

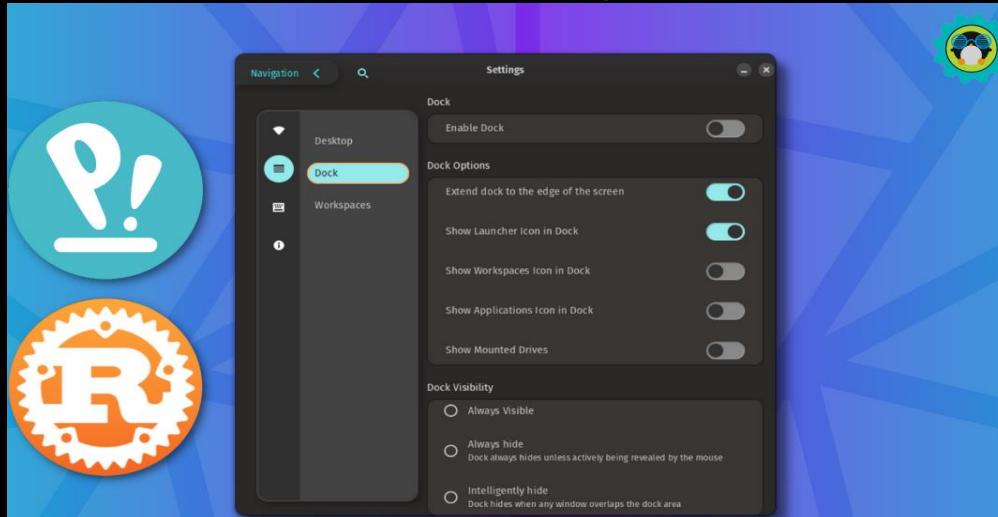
- [https://www.phoronix.com/scan.php?page=news\\_item&px=Pop-OS-New-Rust-Desktop](https://www.phoronix.com/scan.php?page=news_item&px=Pop-OS-New-Rust-Desktop)

Stemming from a [Reddit discussion](#) over the possibility of seeing a KDE flavor of Pop!\_OS, it was brought up by one of their own engineers they are working on their "own desktop".

System76 engineer and Pop!\_OS maintainer Michael Murphy "mmstick" commented that System76 will be its own desktop. When further poked about that whether that means a fork from GNOME, the [response](#) was "No it is its own thing written in [Rust](#)."

Word of System76 making their "own" desktop not based on GNOME does follow some recent friction between Pop!\_OS and GNOME developers over their approach to theming and customizations. Ultimately it will be interesting to see how it plays out. As well, aside from leveraging the [Rust](#) programming language, it will be interesting to see ultimately how this plays out and what features are pursued. Additionally, it remains to be seen how quickly they will be able to shift away from a GNOME base for their Linux desktop and whether they plan to use any GNOME components at all as part of their new desktop effort.

- <https://news.itsfoss.com/system76-rust-cosmic-desktop/>





## System76-Scheduler

- <https://github.com/pop-os/system76-scheduler>

### **Auto-configure CFS and process priorities for improved desktop responsiveness**

Scheduling service which optimizes Linux's CPU scheduler and automatically assigns process priorities for improved desktop responsiveness. Low latency CPU scheduling will be activated automatically when on AC, and the default scheduling latencies set on battery. Processes are regularly swept and assigned process priorities based on configuration files. When combined with [pop-shell](#), foreground processes and their sub-processes will be given higher process priority.

These changes result in a noticeable improvement in the experienced smoothness and performance of applications and games. The improved responsiveness of applications is most noticeable on older systems with budget hardware, whereas games will benefit from higher framerates and reduced jitter. This is because background applications and services will be given a smaller portion of leftover CPU budget after the active process has had the most time on the CPU.

■ ...

## 3.1.2 Fuchsia Overview

- [https://en.wikipedia.org/wiki/Fuchsia\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Fuchsia_(operating_system))

Fuchsia is an open-source capability-based operating system developed by Google. In contrast to prior Google-developed operating systems such as Chrome OS and Android, which are based on the Linux kernel, Fuchsia is based on a new kernel named Zircon. It first became known to the public when the project appeared on a self-hosted git repository in August 2016 without any official announcement. After years of development, Fuchsia was officially released to the public on the first-generation Google Nest Hub, replacing its original Cast OS.

[Kernel](#) [edit]

Fuchsia is based on a new message passing kernel named Zircon, after the mineral zircon. Zircon's codebase was derived from that of Little Kernel (LK), a kernel for embedded devices, aimed for low resource uses, to be used on a wide variety of devices.<sup>[27]</sup> LK was developed by Travis Geiselbrecht, who had also co-authored the NewOS kernel used by Haiku.

Zircon is written mostly in C++, with some parts in assembly language. It is composed of a kernel with a small set of user services, drivers, and libraries which are all necessary for the system to boot, communicate with the hardware, and load the user processes.<sup>[28]</sup> Its present features include handling threads, virtual memory, processes intercommunication, and waiting for changes in the state of objects.<sup>[29]</sup>

It is heavily inspired by Unix kernels but differs greatly. For example, it does not support Unix-like signals but incorporates event-driven programming and the observer pattern. Most system calls do not block the main thread. Resources are represented as objects rather than files, unlike traditional Unix systems.

- <https://fuchsia.dev/>

### Features:

#### Secure

All software that runs on Fuchsia receives the least privilege it needs to perform its job, and gains access only to information it needs to know.

#### Updatable

Much like the web, software on Fuchsia is designed to come and go as needed, and security patches can be pushed to all products on demand.

#### Inclusive

Fuchsia is an open source project that currently supports a variety of languages and runtimes, including C++, Web, Rust, Go, Flutter, and Dart.

#### Pragmatic

Fuchsia is not a science experiment, it's a production-grade operating system that must adhere to fundamentals, like performance.





# Chrome on Fuchsia

The screenshot shows the Fuchsia Emulator running a custom build of Chromium. The top part of the screen displays the Chromium interface with various system status icons (Network, CPU, Memory, Processes) and control buttons (Power, Dark Mode, Shortcuts, Timezone, Brightness, Volume). Below this is the 'About Version' window, which provides detailed information about the build, including the revision (101.0.4922.0), OS (Fuchsia), User Agent (Mozilla/5.0 (X11; Fuchsia) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4922.0 Safari/537.36), and command-line arguments used to run the browser.

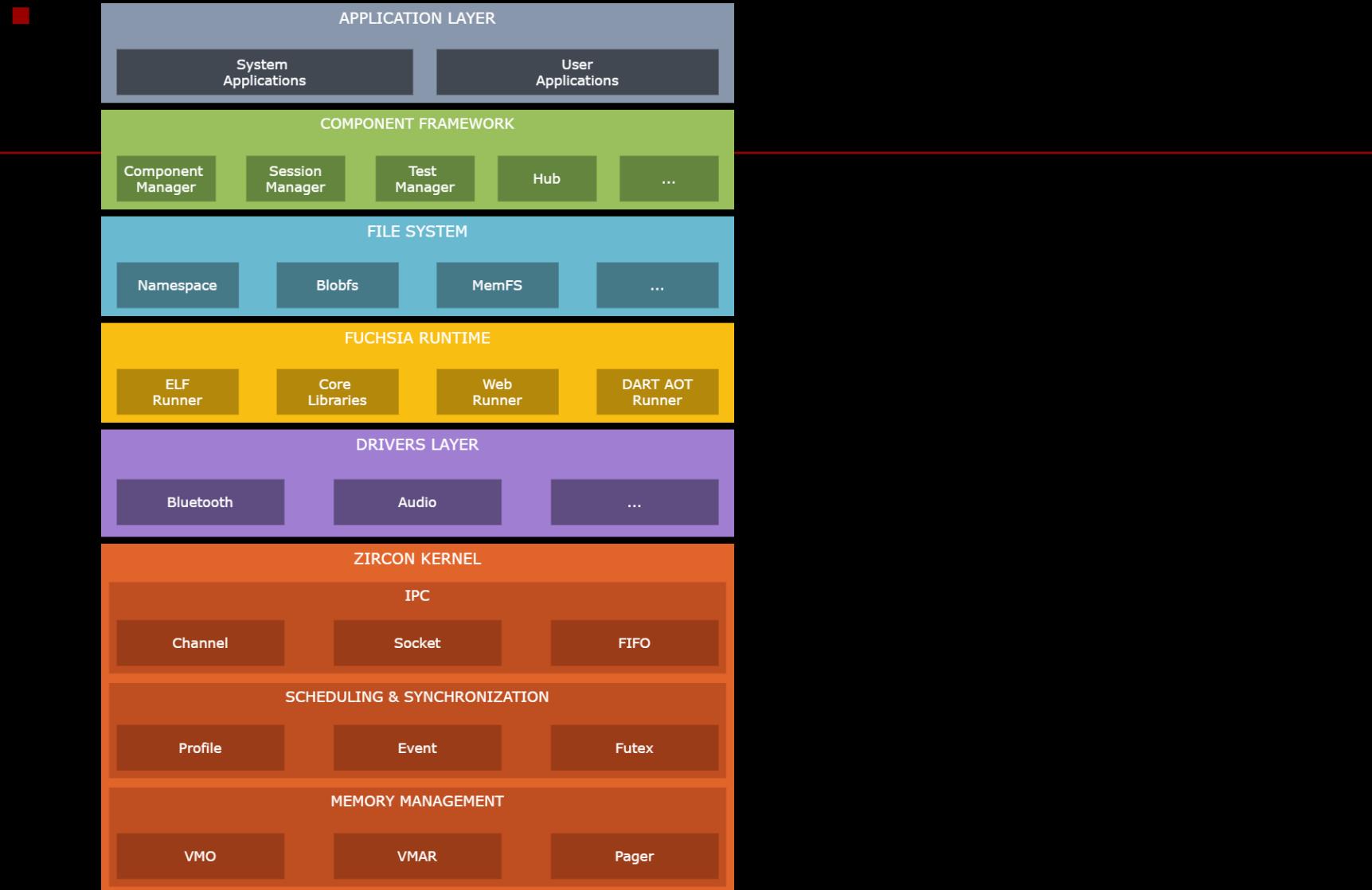
This screenshot shows the 'chrome://flags' page within the Chromium interface. It lists several experimental features that are temporarily unexpired. These include flags for M99 and M100 releases, as well as settings for software rendering, GPU acceleration, and 2D canvas rendering. Each flag has a dropdown menu to change its behavior (Default, Enabled, Disabled).

This screenshot shows the 'chrome://settings' page. It includes sections for 'You and Google' (Autofill, Privacy and security, Appearance, Search engine, Default browser, On startup), 'Advanced' (Extensions, About Chromium), and a 'Sync and Google services' section under 'You and Google'.

Source: <https://9to5google.com/2022/03/04/full-google-chrome-browser-running-on-fuchsia/>



# Architecture & Design



Source: <https://arxiv.org/pdf/2108.04183.pdf>



## Zircon

### <https://fuchsia.dev/fuchsia-src/concepts/kernel>

The Zircon Kernel provides syscalls to manage processes, threads, virtual memory, inter-process communication, waiting on object state changes, and locking (via futexes).

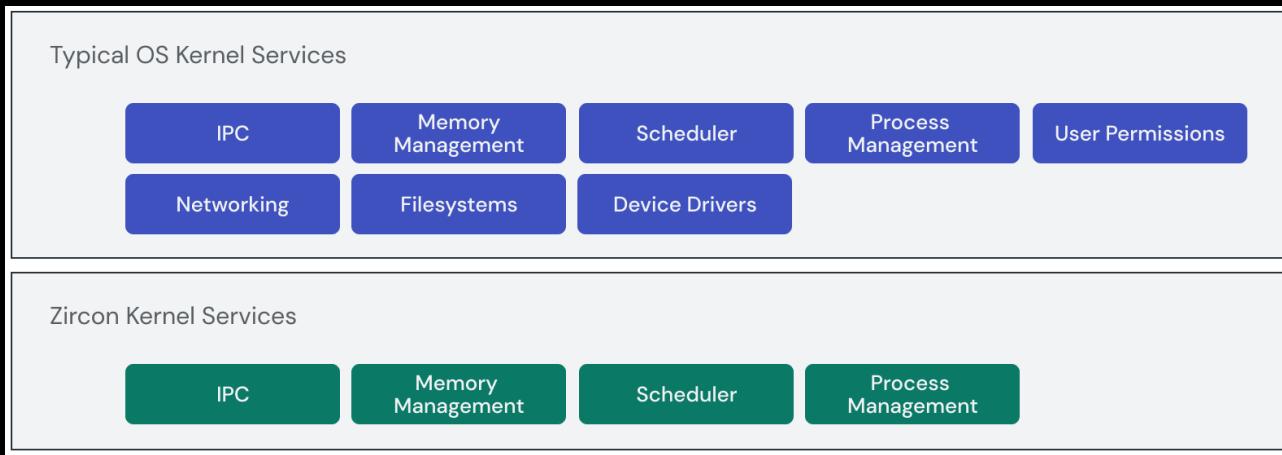
Currently there are some temporary syscalls that have been used for early bringup work, which will be going away in the future as the long term syscall API and ABI surface is finalized. The expectation is that there will be about 100 syscalls.

Zircon syscalls are generally non-blocking. The `wait_one`, `wait_many` `port_wait` and `thread sleep` being the notable exceptions.

### <https://fuchsia.dev/fuchsia-src/get-started/learn/intro/zircon>

[Zircon](#) is the core that powers Fuchsia. It is composed of a kernel and a small set of userspace services, drivers, and libraries necessary for core system functions such as booting.

Although [Zircon](#) applies many of the concepts popularized by microkernels, it does not strive to be minimal. Instead, the microkernel architecture of Zircon enables Fuchsia to reduce the amount of trusted code running in the system to a few core functions:



...



## ■ Component

<https://fuchsia.dev/fuchsia-src/concepts/components/v2>

### Overview

This document offers a brief conceptual overview of Components and the Component Framework.

---

In Fuchsia, [component](#) is the term for the common abstraction that defines how all software<sup>1</sup> (regardless of source, programming language, or runtime) is described, sandboxed, and executed on a Fuchsia system.

---

### What is sandboxing?

Sandboxing is a security mechanism to isolate programs from each other at runtime. In Fuchsia, all software is sandboxed. When a program is initially created, it does not have the ability to do anything – not even to allocate memory. The program relies on its creator to provide the capabilities needed for it to execute. This isolation property allows Fuchsia to employ the *principle of least privilege*: programs are provided only the minimal set of capabilities needed to execute.

### Component Framework

The Component Framework (CF) consists of the core concepts, tools, APIs, runtime, and libraries necessary to describe and run components and to coordinate communication and access to resources between components.

The Component Framework includes:

- CF concepts, including *component*, *component manifest*, *runner*, *realm*, *environment*, *capabilities*, and *resolver*.
- The [component\\_manager](#) process, which coordinates the communication and sharing of resources between components.
- [FIDL APIs](#) implemented by `component_manager`, or implemented by other components and used by `component_manager`, for the purposes of coordination.
- Developer tools to build, execute, and test components.
- Language-specific libraries for components to use to interact with the system. ([example](#))
- Testing tools and libraries to write unit and integration tests that exercise one or many components. ([example](#))

...



## Capabilities

<https://fuchsia.dev/fuchsia-src/concepts/components/v2/introduction>

Since Fuchsia is a capability-based operating system, components interact with each other through the use of [capabilities](#). A capability combines access to a resource and a set of rights, providing both a mechanism for access control and a means by which to interact with the resource.

To support the complex composition of software present in today's products, the Component Framework provides distinct [capability types](#) built upon Zircon [kernel objects](#). A common representation of a capability is a [channel](#) that speaks a particular [FIDL](#) protocol.

The Component Framework assembles the [namespace](#) for a component using [component declarations](#) that describe the capabilities the component requires to function. Components can discover the available capabilities in their namespace using the [fuchsia.io.Directory](#) protocol.

At runtime, every component receives its namespace as well as a handle to the server end of a [Directory](#) channel. This [Directory](#) channel is called the the [outgoing directory](#). Through the outgoing directory, the component's executable makes discoverable any capabilities that it provides.

The Component Framework brokers discovery from a providing component's outgoing directory to a consuming component's namespace through a process called [capability routing](#). While most capabilities are routed to component instances, *runner* and *resolver* capabilities are routed to [environments](#). Environments configure the behavior of the framework for the realms to which they are assigned.

★ **Note:** In the Fuchsia process layer, "having a capability" means the process holds a handle to the kernel object capability in its handle table. In the Component Framework, we often use "having a capability" to mean that the capability is discoverable through the component's namespace at runtime.

...



## IPC

The Zircon kernel enables the communication between different isolated processes through the following mechanisms:

- **Channels** are kernel objects that implement a bidirectional communication tunnel. A channel has two endpoints and each of them maintains a separate FIFO queue to store incoming messages. Components may use channels through the `libfdio` library that provides primitives to deliver and receive messages.
- **Sockets** are a bidirectional stream transports that enable processes to exchange data, using specific primitives. At the creation of the socket, the process can specify the in-bytes dimension of the socket.
- **FIFOs** are first-in-first-out interprocess mechanisms that exploit shared memory between processes to maintain a queue of messages. FIFOs are the most efficient mechanism in terms of I/O operations at the cost of a severely restricted buffer size.

Source: <https://arxiv.org/pdf/2108.04183.pdf>

...



# FIDL

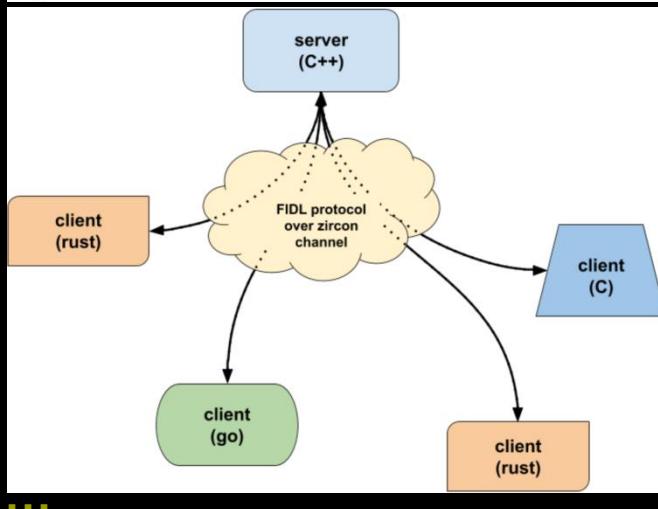
<https://fuchsia.dev/fuchsia-src/concepts/fidl/overview?hl=en>

This document is a high level overview of the Fuchsia Interface Definition Language (FIDL), which is the language used to describe interprocess communication (IPC) protocols used by programs running on Fuchsia. This overview introduces the concepts behind FIDL – developers familiar with these concepts already can start writing code by following the [tutorials](#), or dive deeper by reading the [language](#) or [bindings](#) references.

While "FIDL" stands for "Fuchsia Interface Definition Language," the word itself can be used to refer to a number of different concepts:

- **FIDL wire format:** the FIDL wire format specifies how FIDL messages are represented in memory for transmission over IPC
- **FIDL language:** the FIDL language is the syntax by which protocols are described in `.fidl` files
- **FIDL compiler:** the FIDL compiler generates code for programs to use and implement protocols
- **FIDL bindings:** the FIDL bindings are language-specific runtime support libraries and code generators that provide APIs for manipulating FIDL data structures and protocols.

The main job of FIDL is to allow diverse clients and services to interoperate. Client diversity is aided by decoupling the implementation of the IPC mechanism from its definition, and is simplified by automatic code generation.





## ***Good Resources***

- <https://www.bollyinside.com/articles/how-to-try-fuchsia-os-on-android-devices/>
- <https://androidkenya.com/2022/01/samsung-fuchsia/>
- ...



### 3.1.2.1 Src

■ <https://fuchsia.googlesource.com/>

Language	Files	Lines	Code	Comments	Blanks
GNU Style Assembly	372	249299	216735	6671	25893
Autoconf	11	865	815	21	29
BASH	237	23103	15356	4844	2903
Batch	1	23	20	0	3
C	1702	366189	257211	57422	51556
C Header	7648	935205	561025	216108	158080
CMake	4	394	227	123	44
C++	9343	2208612	1662781	199193	346638
C++ Header	14	10720	10292	211	217
CSS	6	213	169	13	31
Dart	606	73169	52536	10825	9808
Device Tree	6	205	133	48	24
Dockerfile	17	248	196	19	33
Emacs Lisp	1	71	45	12	14
Fish	2	145	85	43	17
FlatBuffers Schema	1	184	80	1	23
GLSL	84	26263	12663	9109	4551
Go	3028	849501	643493	123327	82681
Handlebars	32	745	663	8	74
INI	2	18	16	0	2
JavaScript	65	32998	30450	793	1755
JSON	1377	6013766	6013766	0	60
JSX	3	351	293	42	16
LD Script	2	122	108	10	4
Makefile	17	517	379	34	104
Meson	6	151	103	21	27
Module-Definition	1	53	46	0	7
Nix	1	7	6	0	1
Pan	6	77	42	11	24
Perl	41	48582	38835	4941	4806
Pest	5	350	280	35	35
Prolog	1	45	34	0	1
Protocol Buffers	34	185053	101522	1712	1679
Python	392	63358	48783	5394	9181
ReStructuredText	11	1922	1284	0	638
Shell	272	24215	16027	5025	3163
SVG	63	14957	14941	4	12
Plain Text	394	130098	0	115184	14914
TOML	654	25503	17477	4942	3084
Vim script	10	428	346	54	28
XML	41	5743	5350	156	237
YAML	329	23448	21137	1534	777
HTML	11	200	192	5	3
- CSS	4	20	20	0	0
- JavaScript	3	403	353	16	34
(Total)		623	565	21	37
Markdown	2982	333722	0	245581	88141
- C	63	967	920	7	69
- C++	230	3097	2459	245	393
- C++	39	2129	1652	264	213
- CSS	1	5	5	0	0
- Dart	35	701	575	55	71
- Go	45	1258	1077	56	125
- Handlebars	1	1	1	0	0
- HTML	4	173	154	2	17
- Java	2	224	149	34	41
- JSON	61	1454	1449	0	5
- Python	6	171	144	7	20
- Rust	428	12309	9504	1223	1582
- Shell	55	400	334	51	15
- TOML	192	622	541	52	29
- TypeScript	2	28	21	5	2
- XML	3	41	36	5	0
- YAML	5	86	72	4	10
(Total)		357388	19093	247591	90764
Rust	12131	3261276	2762601	158197	340478
- Markdown	7163	379894	17241	291269	70374
(Total)		3640260	2779942	449466	410852
Total	41966	15235176	12545439	1464960	1224777

[mydev@fedora fuchsia]\$ █

■ ■ ■

## 3.2 Project EVE(Edge Virtualization Engine)

■ <https://www.lfedge.org/projects/eve/>

### ■ Key Capabilities

The goal of Project EVE is to enable IoT edge computing deployments with the following capabilities:

- Access to hardware root of trust (e.g. TPM) when deployed on bare metal, supporting functions such as crypto-based ID (no device usernames and passwords), measured boot, remote attestation, signed updates, encryption, etc.
- “Secure by default” deployment profile
- High efficiency and usage of device resources including remote control of CPU, memory, networking and edge device I/O ports
- Hosting of any combination of apps in virtual machines, containers and Kubernetes clusters
- Hosting of any guest operating system deployable in a virtual machine
- Ability to assign CPU cores and co-processing (e.g. GPU) to specific apps
- Ability to block unused I/O ports to prevent physical tampering
- Remote updates of entire software stack with rollback capability to prevent bricking
- Automated patching for security updates
- Automated connectivity to one or more backends (cloud or on-premises)
- Distributed firewall to securely route data over networks per policy

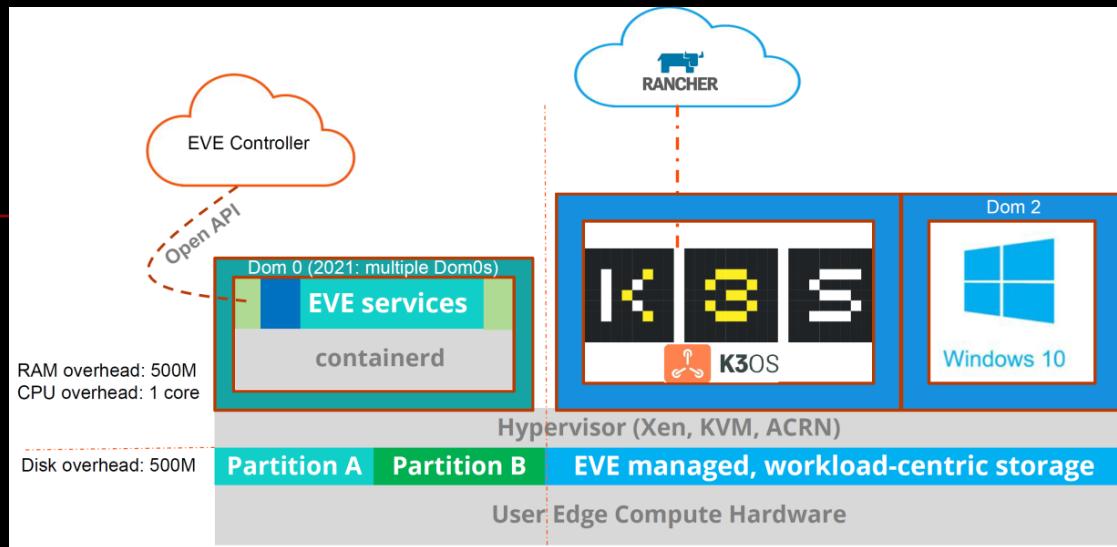
■ <https://github.com/lf-edge/eve>

EVE aims to develop an open, agnostic and standardized architecture unifying the approach to developing and orchestrating cloud-native applications across the enterprise on-premises edge. It offers users new levels of control through hardware-assisted virtualization of on-prem edge devices. Once installed, EVE has direct access to and control of underlying resources and provides standard APIs that allow more efficient use of resources and can effectively partition hardware to increase workload consolidation and application multi-tenancy.

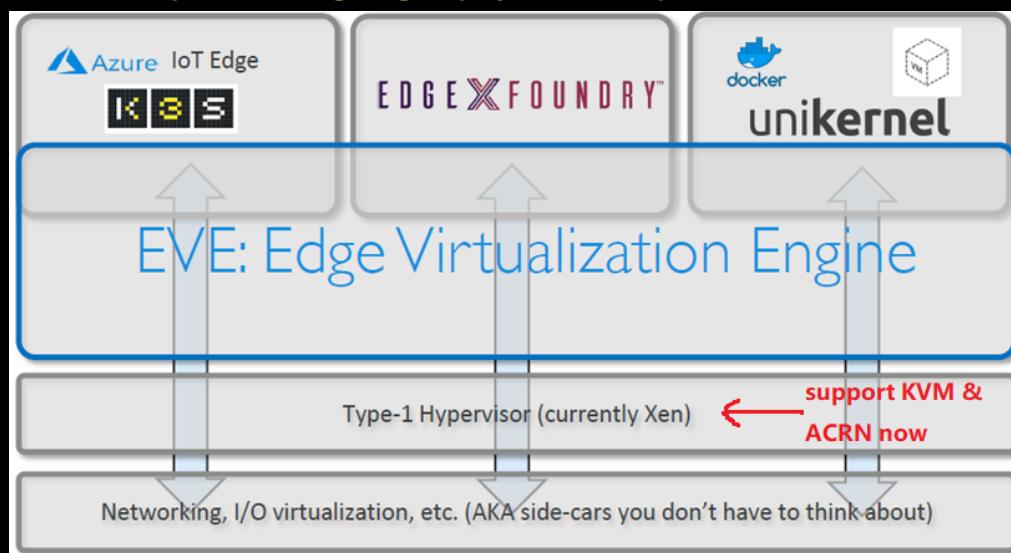
EVE supports both ARM and Intel architectures and requires hardware-assisted virtualization. While EVE can run on a board as small as a \$20 Orange Pi, the sweet spot for its deployment are IoT Gateways and Industrial PCs.

To get its job done, EVE leverages a lot of great open source projects: [Xen Project](#), [Linuxkit](#) and [Alpine Linux](#) just to name a few. All of that functionality is being orchestrated by the Go microservices available under [pkg/pillar](#). Why pillar? Well, because pillar is the kind of a monolith we need to break out into true, individual microservices under [pkg/](#).

# Architecture



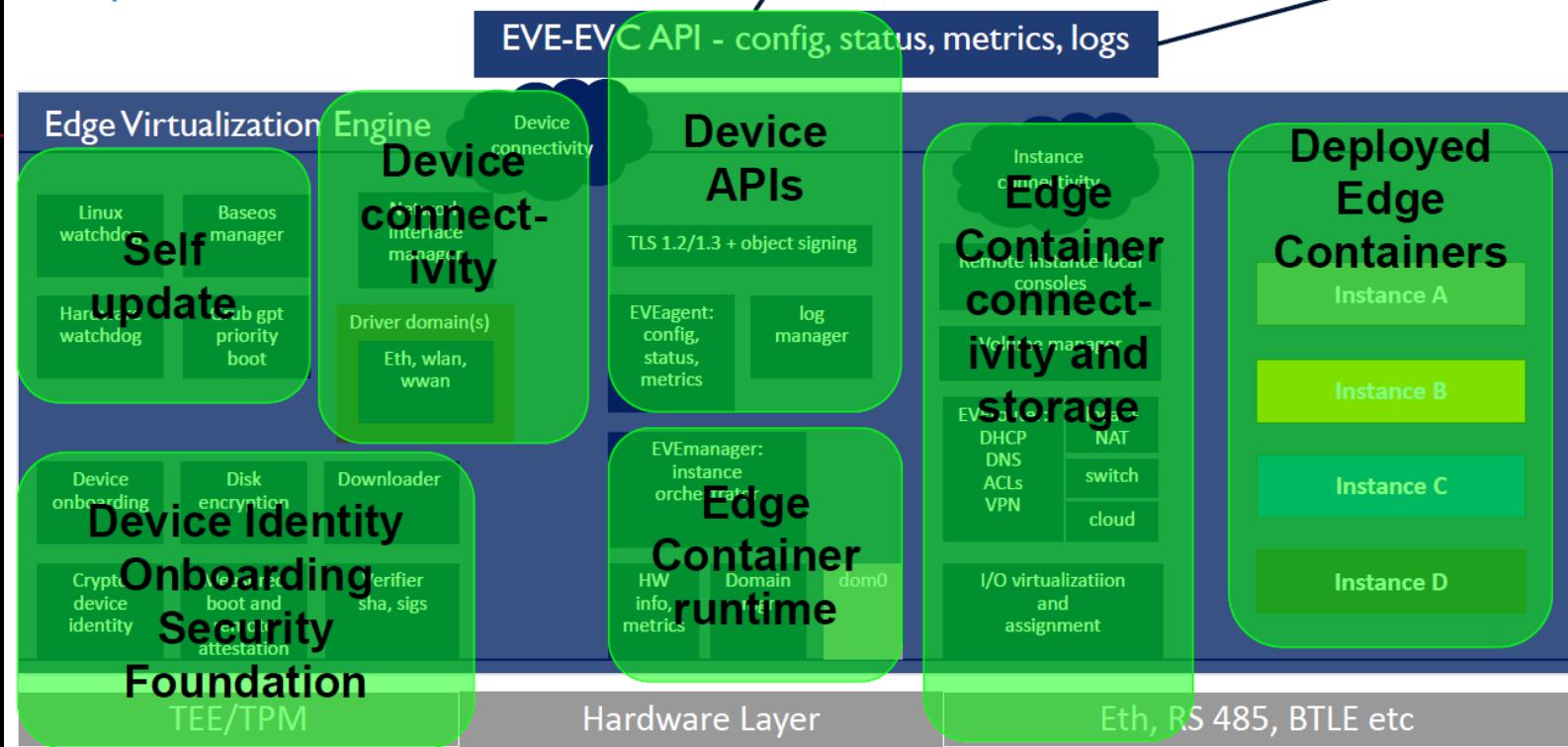
Source: <https://wiki.lfedge.org/display/EVE/Slide+presentations>



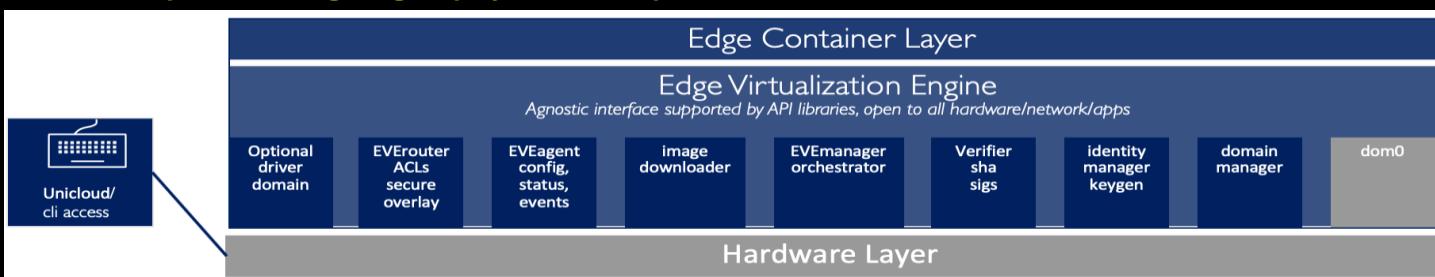
Source: “Linux Foundation’s Project EVE: a Cloud-Native Edge Computing Platform”, Roman Shaposhnik, QConSF2019



# Project EVE Architecture



Source: <https://wiki.lfedge.org/display/EVE/Slide+presentations>



Source: <https://wiki.lfedge.org/display/EVE/EVE>

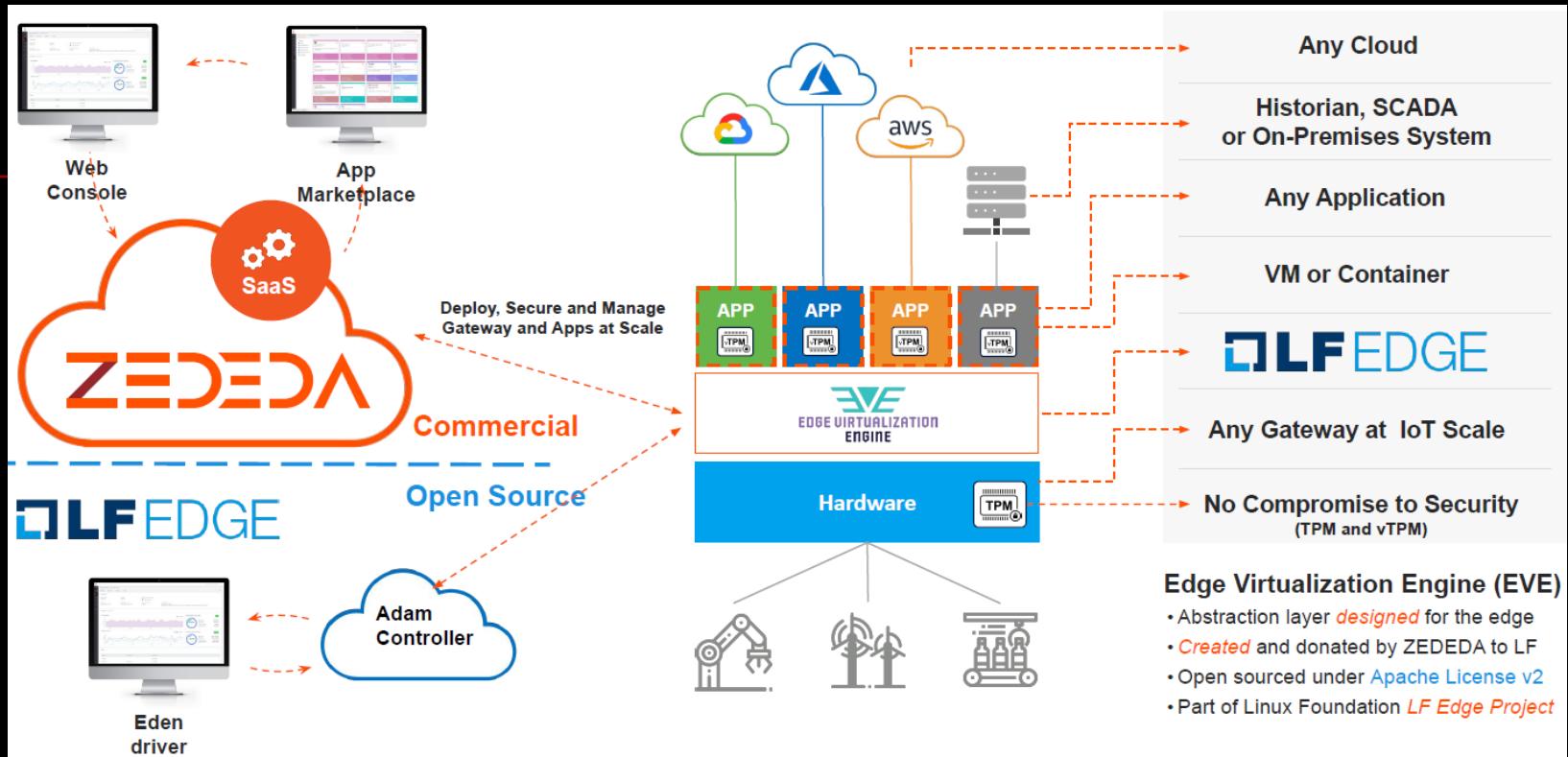
EVC sample: Adam

Commercial EVC:

**ZEDEDA**



## Challenges Solved with Edge Virtualization

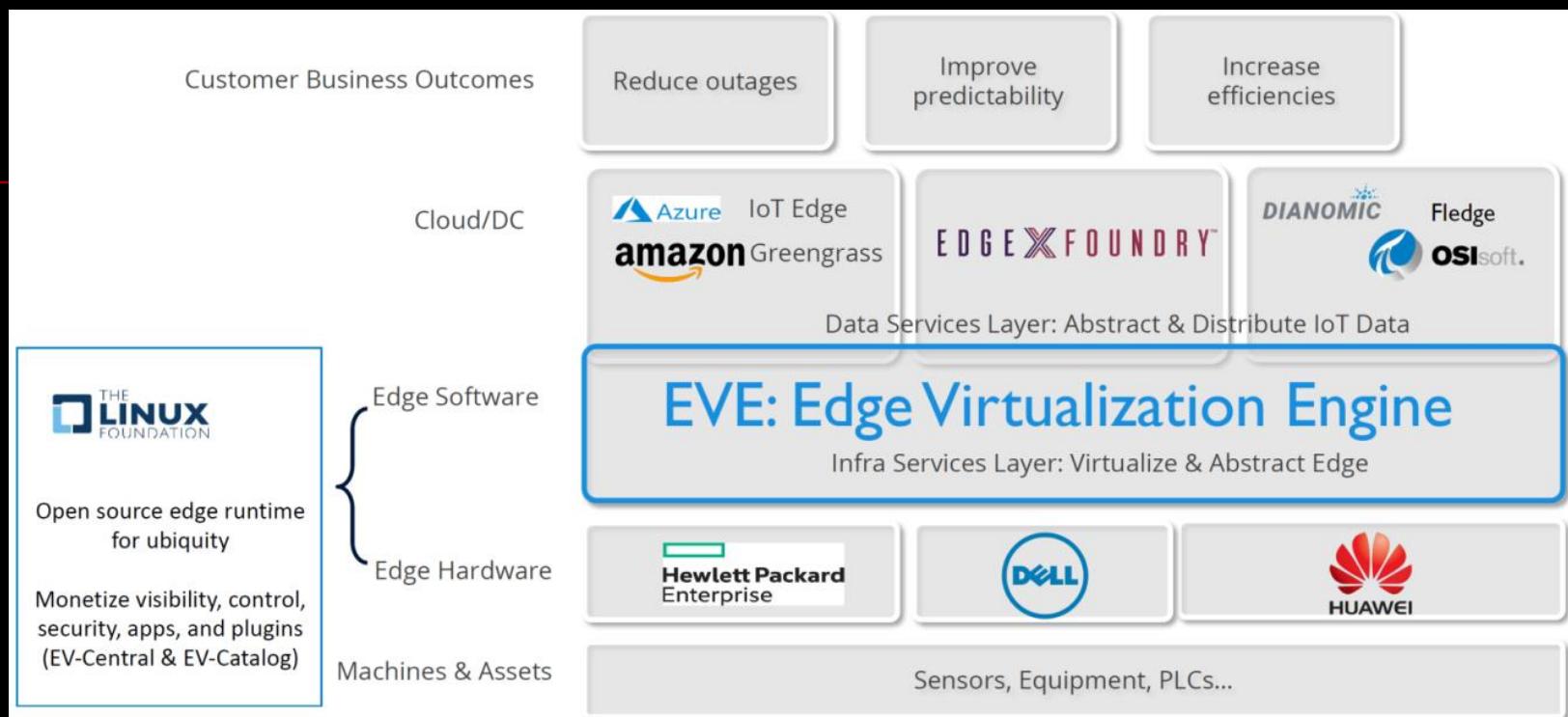


Source: <https://wiki.lfedge.org/display/EVE/Slide+presentations>



# EVE in The Enterprise Cyber-Physical Edge Stack

■



Source: <https://wiki.lfedge.org/display/EVE/Slide+presentations>



## EVE Tools

- <https://github.com/lf-edge/eve-tools>

### **Library and Tools to interact with Edge Virtualization Engine(EVE).**

This repository contains library and tools required to interact with the Edge Virtualization Engine(EVE), an operating system built for the Edge devices, a project under Linux Foundation Edge ([www.lfedge.org](http://www.lfedge.org))

In a quick summary, Edge Virtualization Engine provides a uniform virtualization layer on various Edge gateways, using which Edge workloads can be deployed as Containers or Virtual Machines, commonly called the Edge Containers. For more details about EVE, please visit <https://github.com/lf-edge/eve>.

Edge Containers run in virtualized environment. However, in some cases, it is useful for the applications running inside these Edge Containers to have a way to interact with the host operating system(i.e. EVE), to collaboratively implement certain functionality for better agility, security and accuracy. Some example use cases below:

- An application might want to pin its identity with an EK on the hardware TPM ASIC, to make sure its identity is sealed to the hardware platform it is running on.
- An application might want to have a way to communicate with its host operating system to establish a liveness check, where it can be restarted by the EVE layer, when the Edge Container turns unresponsive for whatever reason.
- An application might want to know about its underlying physical network ports going down, so that it can take any corrective action in a timely manner, like rerouting via backup link, or propagating the failure either upstream or downstream.
- An application might want to get a view of physical resources consumed by the Edge VM or Container, as seen by the host operating system, for accurate reporting of KPIs to its analytics services.



## Good Resources

- <https://wiki.lfedge.org/display/EVE/EVE>
- <https://wiki.lfedge.org/display/EVE/Feature+Roadmap>
- <https://wiki.lfedge.org/display/EVE/EVE+7.5.0+on+RPI4>
- [~~https://www.lfedge.org/event/eve-design-summit~~](https://www.lfedge.org/event/eve-design-summit)
- ...



### 3.2.1 EVE-OS

- <https://github.com/shantanoo-desai/EVE-OS-tutorials>

Getting Acquainted with EVE-OS, Eden and Adam Ecosystem for Edge Computing

---

Project EVE is building EVE-OS, a universal, open Linux-based operating system for distributed edge computing. EVE-OS aims to do for the distributed edge what Android did for mobile by creating an open foundation that simplifies development, orchestration and security of edge computing nodes deployed on-prem and in the field.

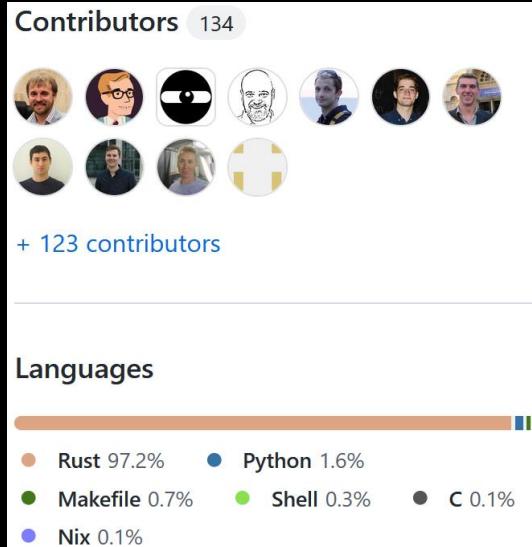


### 3.3 Tock

- <https://www.tockos.org/>
- **An embedded operating system designed for running multiple concurrent, mutually distrustful applications on low-memory and low-power microcontrollers.**
- <https://github.com/tock/tock>

Tock is a secure, embedded operating system for Cortex-M and RISC-V microcontrollers. Tock assumes the hardware includes a memory protection unit (MPU), as systems without an MPU cannot simultaneously support untrusted processes and retain Tock's safety and security properties. The Tock kernel and its extensions (called *capsules*) are written in Rust.

Tock can run multiple, independent untrusted processes written in any language. The number of processes Tock can simultaneously support is constrained by MCU flash and RAM. Tock can be configured to use different scheduling algorithms, but the default Tock scheduler is preemptive and uses a round-robin policy. Tock uses a microkernel architecture: complex drivers and services are often implemented as untrusted processes, which other processes, such as applications, can invoke through inter-process communication (IPC).





## ■ Features

<https://www.tockos.org/features>

### Safety

Tock takes advantage of hardware-protection mechanisms available on recent microcontrollers and the type-safety features of the Rust programming language to provide a multiprogramming environment that offers isolation of software faults.

Kernel components are isolated at compile-time using Rust's type and module systems. As a result, sensor drivers, virtualization layers, networking stacks and other components can only access resources they are allowed to, even if they operate on the same bus or share state with other components. For example, two drivers for peripherals on the same I2C bus can only talk to their respective peripherals.

### Reliability

Embedded applications, whether for sensor networks, IoT devices, or security focused, need to be highly reliable. If they crash, there is usually no way for a human to fix them in the field.

The Tock kernel uses an event driven execution model that uses no heap allocation, so the kernel won't run out of memory. Applications can manage their memory however they want, but are scheduled preemptively and decoupled from the kernel such that the system can keep going if an application crashes or restarts.

### Seamless Low-power

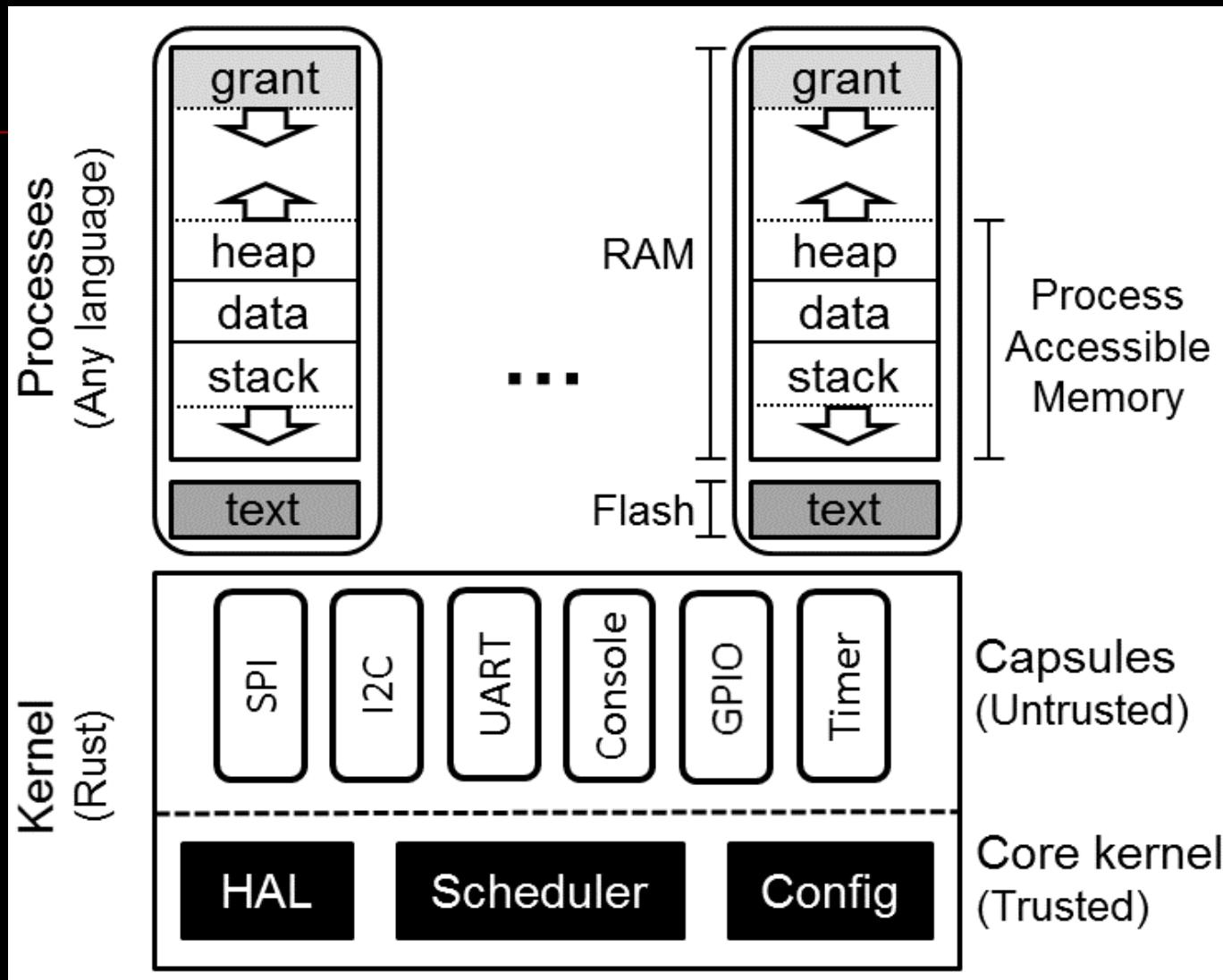
Tock-based systems can run on battery for months or years, or from energy harvesting sources like solar indefinitely. The Tock kernel and drivers seamlessly put the hardware into the lowest possible sleep state based on application requirements. No explicit power-management is required from the application! Even naive apps like this blink app sleep as low as 5 $\mu$ A :

```
int main() {
    while(1) {
        led_toggle(0);
        delay_ms(5000);
    }
}
```



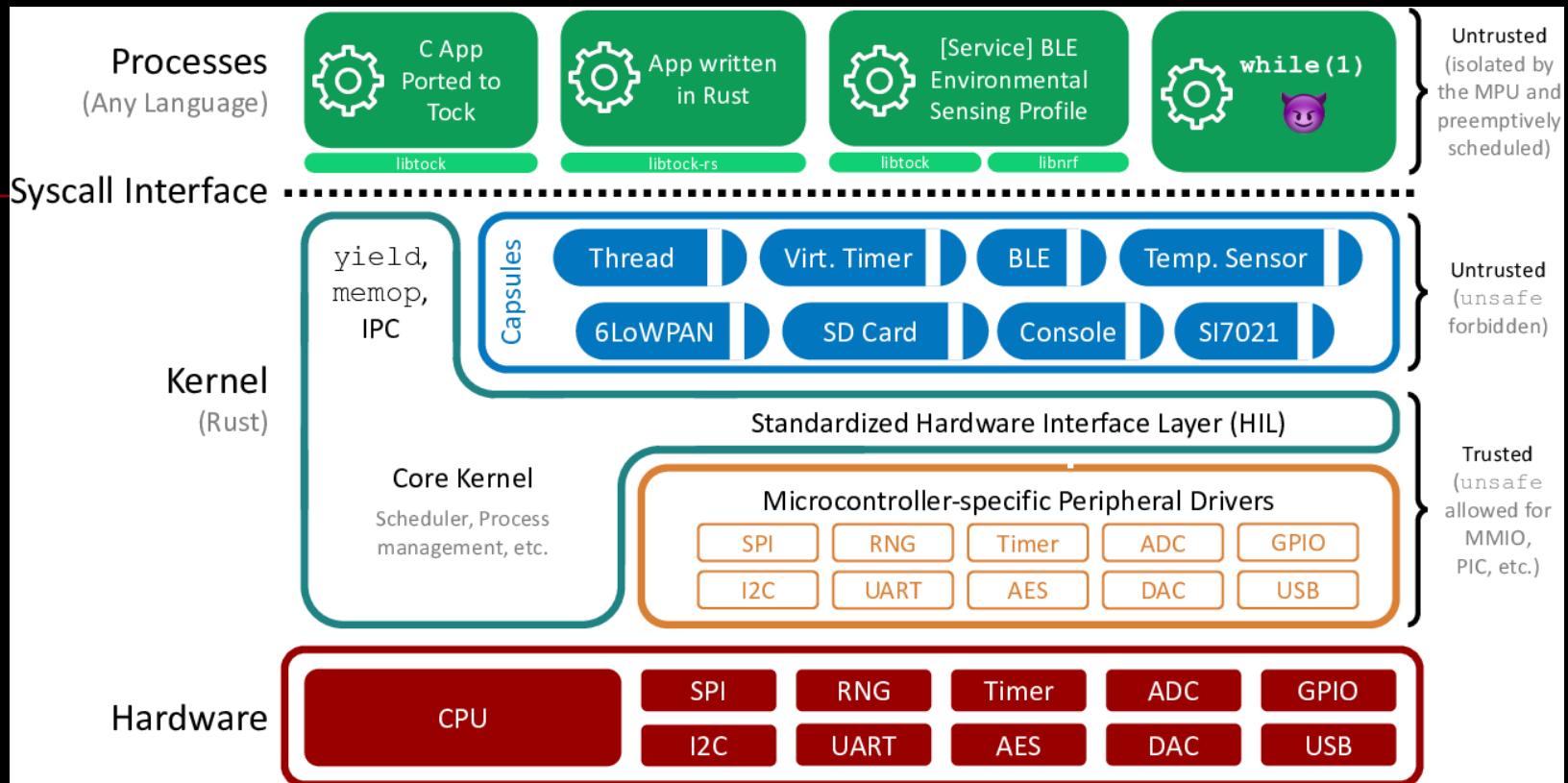
## Architecture and Design

- <https://www.tockos.org/documentation/design>





# Stack



Source: <https://github.com/tock/tock/blob/master/doc/tock-stack.png>



## Components

System components (an application, driver, virtualization layer, etc.) can be implemented in either a capsule or process, but each mechanism trades off concurrency and safety with memory consumption, performance, and granularity.

Category	Capsule	Process
Protection	Language	Hardware
Memory Overhead	None	Separate stack
Protection Granularity	Fine	Coarse
Concurrency	Cooperative	Preemptive
Update at Runtime	No	Yes

As a result, each is more appropriate for implementing different components. In general, drivers and virtualization layers are implemented as capsules, while applications and complex drivers using existing code/libraries, such as networking stacks, are implemented as processes.



## Capsules

A capsule is a Rust struct and associated functions. Capsules interact with each other directly, accessing exposed fields and calling functions in other capsules. Trusted platform configuration code initializes them, giving them access to any other capsules or kernel resources they need. Capsules can protect internal state by not exporting certain functions or fields.

Capsules run inside the kernel in privileged hardware mode, but Rust's type and module systems protect the core kernel from buggy or malicious capsules. Because type and memory safety are enforced at compile-time, there is no overhead associated with safety, and capsules require minimal error checking. For example, a capsule never has to check the validity of a reference. If the reference exists, it points to valid memory of the right type. This allows extremely fine-grained isolation since there is virtually no overhead to splitting up components.

Rust's language protection offers strong safety guarantees. Unless a capsule is able to subvert the Rust type system, it can only access resources explicitly granted to it, and only in ways permitted by the interfaces those resources expose. However, because capsules are cooperatively scheduled in the same single-threaded event loop as the kernel, they must be trusted for system liveness. If a capsule panics, or does not yield back to the event handler, the system can only recover by restarting.



## Processes

Processes are independent applications that are isolated from the kernel and run with reduced privileges in separate execution threads from the kernel. The kernel schedules processes preemptively, so processes have stronger system liveness guarantees than capsules. Moreover, Tock uses hardware protection to enforce process isolation at runtime. This allows processes to be written in any language and to be safely loaded at runtime.

## Memory Layout

Processes are isolated from each other, the kernel, and the underlying hardware explicitly by the hardware Memory Protection Unit (MPU). The MPU limits which memory addresses a process can access. Accesses outside of a process's permitted region result in a fault and trap to the kernel.

A process's code, stored in flash, is made accessible with a read-only memory protection region while its memory is allocated as a contiguous region of RAM. One novel aspect of a Tock process is the presence of a "grant" region at the top of the address space. This is memory allocated to the process but covered by a memory protection region that the process can neither read nor write. The grant region allows the kernel to borrow memory from a process in response to system calls.



## Grants

Capsules are not allowed to allocate memory dynamically since dynamic allocation in the kernel makes it hard to predict if memory will be exhausted. A single capsule with poor memory management could cause the rest of the kernel to fail. Moreover, since it uses a single stack, the kernel cannot easily recover from capsule failures.

---

However, capsules often need to dynamically allocate memory in response to process requests. For example, a virtual timer driver must allocate a structure to hold metadata for each new timer any process creates. Therefore, Tock allows capsules to dynamically allocate from the memory of a process making a request.

It is unsafe, though, for a capsule to directly hold a reference to process memory. Processes crash and can be dynamically loaded, so, without explicit checks throughout the kernel code, it would not be possible to ensure that a reference to process memory is still valid.

For a capsule to safely allocate memory from a process, the kernel must enforce three properties:

1. Allocated memory does not allow capsules to break the type system.
2. Capsules can only access pointers to process memory while the process is alive.
3. The kernel must be able to reclaim memory from a terminated process.

Tock provides a safe memory allocation mechanism that meets these three requirements through memory grants. Capsules can allocate data of arbitrary type from the memory of processes that interact with them. This memory is allocated from the grant segment.

Just as with buffers passed through `allow`, references to granted memory are wrapped in a type-safe struct that ensures the process is still alive before dereferencing. Unlike shared buffers, which can only be a buffer type in a capsule, granted memory can be defined as any type. Therefore, processes cannot access this memory since doing so might violate type-safety.



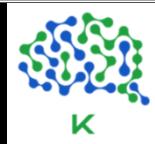
## Good Resources

- <https://github.com/tock/tock/tree/master/doc>
  - <https://www.tockos.org/hardware/>
  - ...
-

## 4) Hardware Acceleration

### 4.1 Overview

- ...
- 





## 4.2 Cost-effective SmartNIC for Edge Computing

### 4.2.1 Overview

- ...
-



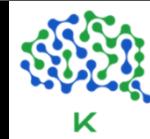
## 4.2.2 Our Target

### 4.2.2.1 Architecture & Design

#### In demand

- A FPGA-based SmartNIC that fits inside the USB port
- At least two interfaces: a USB port for host connection, and a RJ45 or Mini I/O connector for networking
- The main logic in FPGA is to implement the desired functionalities of the SmartNIC, the conversion of I/O signals, and so on.
- Support for USB 4.0 is mostly preferred(The host must also support USB 4.0)
- ...

The main reason that this design do not use PCIe interfaced SmartNIC is base on the consideration of add more flexibility to our overall solution for Edge Computing.



## 4.2.3 Reference FOSS Projects

### 4.2.2.1 Corundum

#### Overview

- <https://github.com/corundum/corundum/>

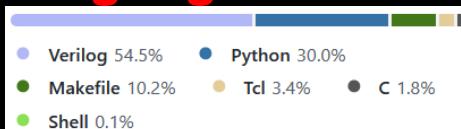
#### **Open source, high performance, FPGA-based NIC**

Corundum is an open-source, high-performance FPGA-based NIC. Features include a high performance datapath, 10G/25G/100G Ethernet, PCI express gen 3, a custom, high performance, tightly-integrated PCIe DMA engine, many (1000+) transmit, receive, completion, and event queues, scatter/gather DMA, MSI interrupts, multiple interfaces, multiple ports per interface, per-port transmit scheduling including high precision TDMA, flow hashing, RSS, checksum offloading, and native IEEE 1588 PTP timestamping. A Linux driver is included that integrates with the Linux networking stack. Development and debugging is facilitated by an extensive simulation framework that covers the entire system from a simulation model of the driver and PCI express interface on one side to the Ethernet interfaces on the other side.

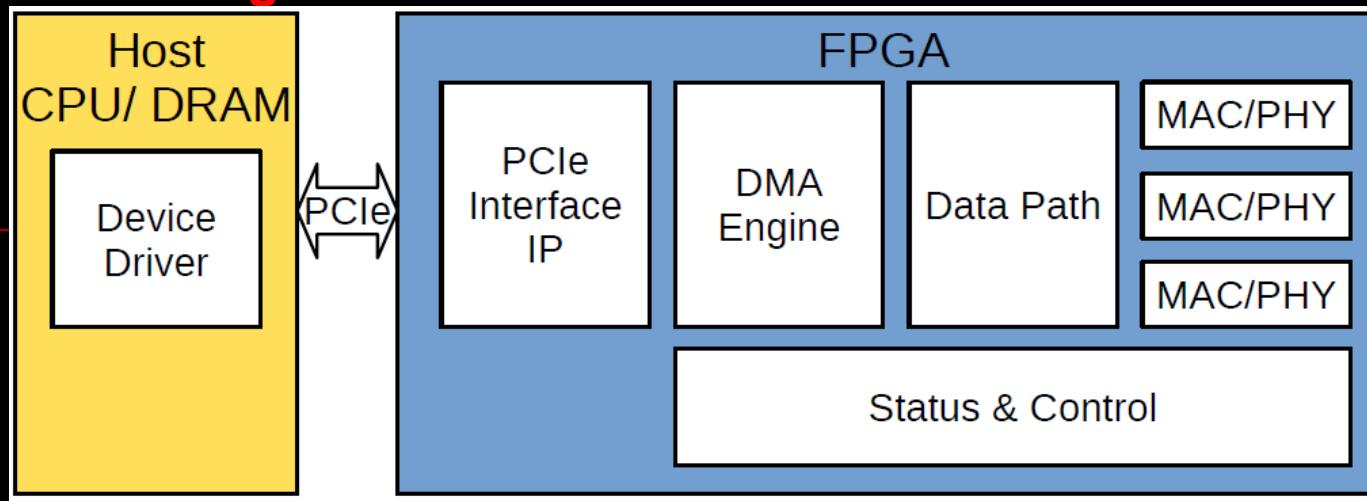
Corundum has several unique architectural features. First, transmit, receive, completion, and event queue states are stored efficiently in block RAM or ultra RAM, enabling support for thousands of individually-controllable queues. These queues are associated with interfaces, and each interface can have multiple ports, each with its own independent scheduler. This enables extremely fine-grained control over packet transmission. Coupled with PTP time synchronization, this enables high precision TDMA.

Corundum also provides an application section for implementing custom logic. The application section has a dedicated PCIe BAR for control and a number of interfaces that provide access to the core datapath and DMA infrastructure.

- **Languages**



## ■ Block Diagram



Source: “CORUNDUM: An open source FPGA based 100G NIC”, Alexander & Ulrich, Fosdem 2022.

## ■ Block Diagram

- PCIe attached NIC
- 1-8x (1/10/25/100 GbE physical Ports
- Driver for Linux
- > 16k rings supported for RX and TX
- Custom scheduling, processing logic addition in RTL and driver
- Full Xilinx UltraScale+ device support
- Initial Intel Stratix 10 (H-Tile) device support

Source: “CORUNDUM: An open source FPGA based 100G NIC”, Alexander & Ulrich, Fosdem 2022.

## ■ **Background**

Corundum is based on various (sub-) projects and components  
(partially also maintained by Alex)

- 
- [verilog-axi](#)
  - [verilog-axis](#)
  - [verilog-pcie](#)
  - [verilog-ethernet](#)
  - [cocotb](#)
  - [MyHDL](#)
  - Device Specific Vendor IP

Source: “CORUNDUM: An open source FPGA based 100G NIC”, Alexander & Ulrich, Fosdem 2022.

## ■ **Roadmap**

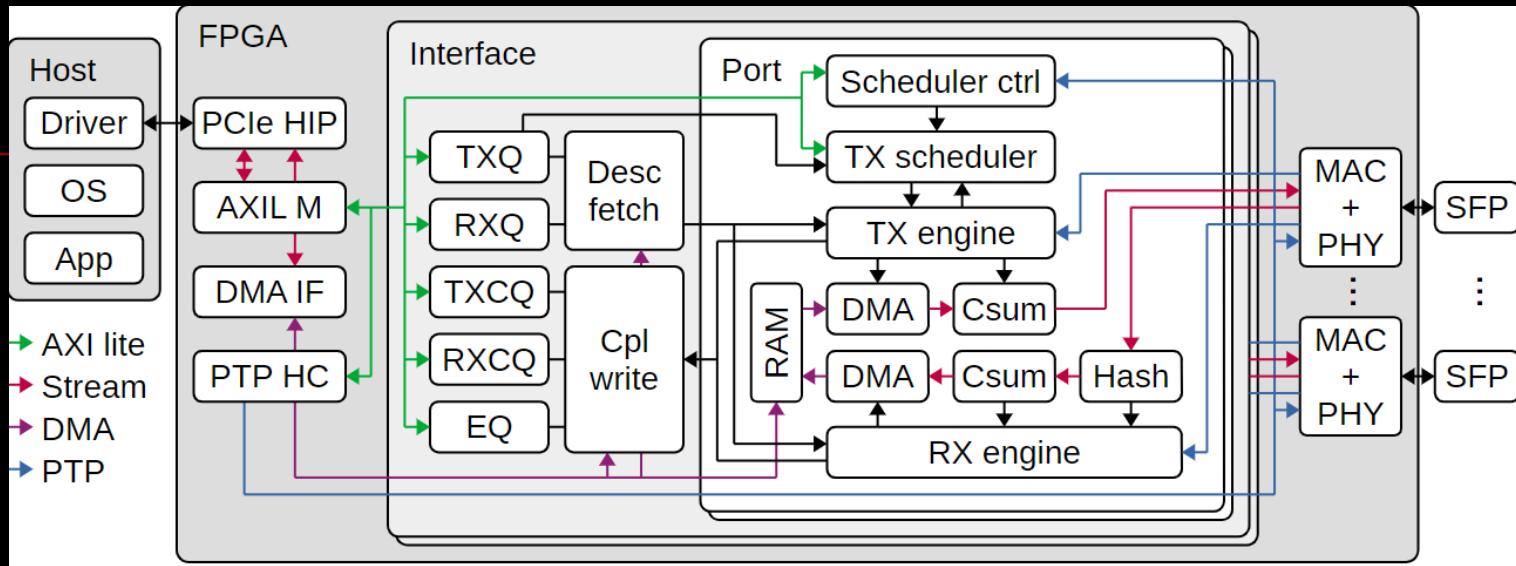
- Restructure RTL (shared datapath)
- SW/HW Interface Optimisation (Variable Length Descriptors)
- Unified DMA Address Space (Host + on-board DRAM)
- RDMA Support
  - RoCEv2
  - On-board DRAM Cache
- DPDK Support
- Large Send Offload (LSO)/ Large Receive Offload (LRO)

Source: “CORUNDUM: An open source FPGA based 100G NIC”, Alexander & Ulrich, Fosdem 2022.



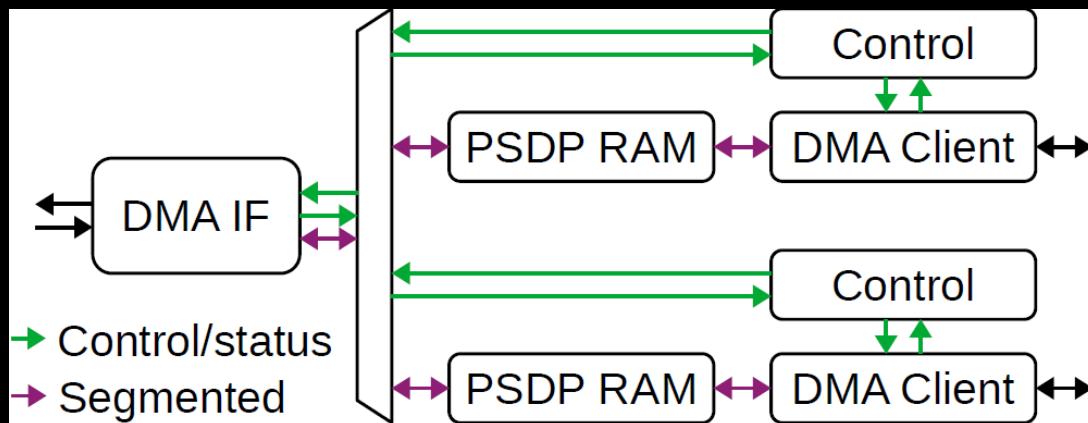
## Architecture & Design

### RTL Structure



Source: "CORUNDUM: An open source FPGA based 100G NIC", Alexander & Ulrich, Fosdem 2022.

### DMA RTL Structure



Source: "CORUNDUM: An open source FPGA based 100G NIC", Alexander & Ulrich, Fosdem 2022.



## ***Supported boards***

- Corundum currently supports Xilinx Virtex 7, UltraScale, and UltraScale+ series devices. Designs are included for the following FPGA boards:

- Alpha Data ADM-PCIE-9V3 (Xilinx Virtex UltraScale+ XCVU3P)
- Exablaze ExaNIC X10/Cisco Nexus K35-S (Xilinx Kintex UltraScale XCKU035)
- Exablaze ExaNIC X25/Cisco Nexus K3P-S (Xilinx Kintex UltraScale+ XCKU3P)
- Silicom fb2CG@KU15P (Xilinx Kintex UltraScale+ XCKU15P)
- NetFPGA SUME (Xilinx Virtex 7 XC7V690T)
- Intel Stratix 10 MX dev kit (Intel Stratix 10 MX 1SM21CHU1F53E1VG)
- Xilinx Alveo U50 (Xilinx Virtex UltraScale+ XCU50)
- Xilinx Alveo U200 (Xilinx Virtex UltraScale+ XCU200)
- Xilinx Alveo U250 (Xilinx Virtex UltraScale+ XCU250)
- Xilinx Alveo U280 (Xilinx Virtex UltraScale+ XCU280)
- Xilinx VCU108 (Xilinx Virtex UltraScale XCVU095)
- Xilinx VCU118 (Xilinx Virtex UltraScale+ XCVU9P)
- Xilinx VCU1525 (Xilinx Virtex UltraScale+ XCVU9P)
- Xilinx ZCU106 (Xilinx Zynq UltraScale+ XCZU7EV)

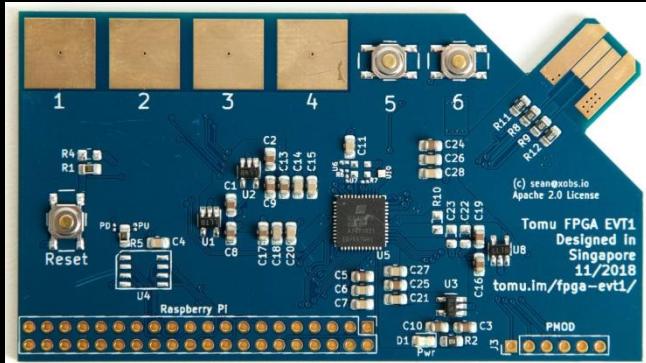
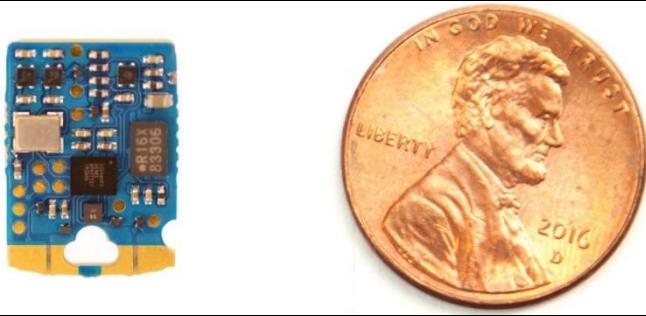
- For operation at 10G and 25G, Corundum uses the open source 10G/25G MAC and PHY modules from the verilog-ethernet repository, no extra licenses are required. However, it is possible to use other MAC and/or PHY modules.

Operation at 100G on Xilinx UltraScale+ devices currently requires using the Xilinx CMAC core with RS-FEC enabled, which is covered by the free CMAC license.



## 4.2.2.2 Fomu

- <https://www.crowdsupply.com/sutajio-kosagi/fomu>



- <https://tomu.im/fomu.html>

**Tomu FPGA (Fomu for short), a FPGA which fits inside your USB port!**

- <https://github.com/im-tomu/fomu-hardware>

- ...

Python, RISC-V, FPGA, All Open

### Fomu has Python

With 128 kilobytes of RAM and a large amount of storage, Fomu is powerful enough to run Python natively. And since it lives in your USB port, installation is super simple. FPGAs are complicated, but the latest Python tools make it easy to use Fomu without any specialized training.

### Fomu runs RISC-V

Underneath the Python interpreter lies a RISC-V softcore running on the FPGA fabric. RISC-V is an up-and-coming processor architecture that is poised to take over everything from deeply-embedded chips to high-performance computing. Fomu's RISC-V softcore is a great introduction to the processor architecture of the future.

### Fomu is an FPGA

An FPGA is a piece of reconfigurable silicon. The default Fomu firmware exposes a USB bootloader running a RISC-V softcore, but you can load whatever you want. Softcores are also available for LM32 and OpenRISC. You can practice adding instructions to the CPU, or add new blocks such as LED blink patterns or better captouch hardware blocks.

### Fomu is entirely open

Developing with Fomu is incredibly easy: just load code via USB and go. Whether you're writing RISC-V code, Python code, or HDL, it's all uploaded to Fomu in the same way. The ICE40UP5K FPGA is supported with a fully open toolchain, meaning you can start development without creating an account, signing an NDA, or downloading a multi-gigabyte installer.



## 5) Beyond Kubernetes

### Motivation

- Now **Kubernetes** is the de facto standard for today's production-grade **Container** orchestration, which is surrounded by an amazing huge and continuously growing ecosystem. It comes with the Container first design philosophy and is optimized for that.
- Kubernetes is really powerful, but it is getting more complex and accumulating more and more historical baggage at the same time...
- Lightweight Kubernetes solutions like **K3S** and **K0S** are suitable for **Edge** devices that have limited computing resources when compared with Cloud side, but may not still be a good fit for hardware platform with even less computing resources like **Microcontrollers** which is common on **IoT** devices.
- New workload like **Wasm** and **eBPF** is more lightweight than Container, and their ecosystem is booming.
- Current solutions like **Krustlet** or **Kata Containers/WasmEdge** supports the above new workload by extending Kubernetes or is implemented as **OCI**-compatible: the benefits are it can both support Container workload and the new workload, and make best of use the existed code and ecosystem of **Kubernetes**, **Docker**, and so on.

**the drawbacks are thus means it has to inherit some kind of "historical burden", and most of all, it is not burn for new workload like Wasm and eBPF.**

■ ...

---

## Goals

A re-designed orchestration system for new workload mainly.

For IoT devices like Microcontroller:

- For platforms support Wasm(high probability) or xBPF(low possibility), a completely new design inspired by Bento(will be introduced in next session) is preferred, and the built-in support for TinyML must be gave priority to.

For Edge devices:

- For platforms support Wasm, xBPF, or specialized Container technologies targets IoT Edge, a new lightweight solution will be carefully designed and implemented with modular and extensible architecture as Kubernetes, it will also leverage our universal distributed engine and a unified scheduler...
- Seamlessly work with our SuperVM for unified Wasm and eBPF runtime.
- Add the support for our Smart Runtime in the long term.

The discussions for the above topic will come in 2022...



# 5.1 Reference FOSS Projects

## 5.1.1 Bento

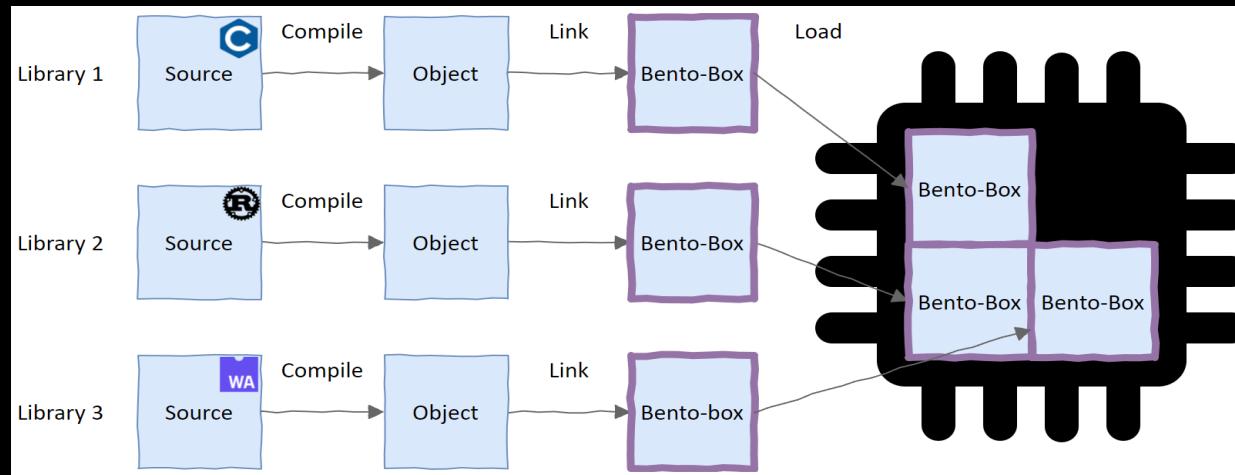
### Overview

- <https://github.com/ARM-software/bento-linker>

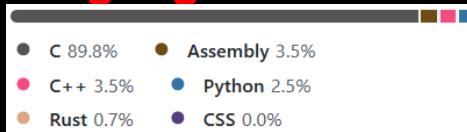
A light-weight alternative to processes for microcontrollers.

- **Bento-boxes**

Independently-linked, memory-isolated pieces of code that are designed to work together. You can think of them as a light-weight alternative to processes for Microcontrollers.



- **Languages**





# The glue

## Runtimes

The available runtimes can be listed with the `bento runtimes` command. These determine how each box is executed.

- **jumptable** - A native runtime that uses a table of function pointers for connecting imports/exports.

Note! No memory isolation is enforced with this runtime.

By default, setjmp/longjmp is also used in the case of explicit aborts or assert failures, but this can also be disabled by setting `runtime.jumptable.no_longjmp=true`.

- **arm{v7m,v8m}-mpu** - A native runtime that uses an Arm MPU to enforce memory isolation.
- **arm{v7m,v8m}-sys** - This is a bit of a special runtime. It's a native runtime without memory isolation.

However, instead of a jumptable, this runtime connects to the standard Arm ISR vector. This allows you to hook type-checked interrupt handlers as you would any other box export.

- **awsm** - awsm is an Ahead-of-Time compiler for [WebAssembly](#) that provides software enforced memory isolation at near-native performance as long as you can ensure the binary has not been tampered.

More info here:

<https://github.com/gwsystems/awsm>

- **wamr** - Wamr is a [WebAssembly](#) interpreter with an optional Ahead-of-Time compiler.

More info here:

<https://github.com/bytocodealliance/wasm-micro-runtime>

- **wasm3** - Wasm3 is a [WebAssembly](#) interpreter built on continuation passing, which allows for very few dependencies.

More info here:

<https://github.com/wasm3/wasm3>



## Loaders

The available loaders can be listed with the `bento loaders` command. These determine how each box is loaded before execution.

- **noop** - This is the default loader, which assumes the box can be placed in persistent executable memory.
- **glz** - This loader provides GLZ decompression at bringup time. GLZ is a compression algorithm designed for extremely lightweight decompression.
- **bd** - This loader loads boxes from a user provided block device.
- **fs** - This loader loads boxes from a user provided filesystem.

## 6) Better System Programming Language

### Motivation

- C is great, C++ is powerful, Go is the de facto TOP 1 language in today's Cloud Infrastructure, Rust is awesome and booming, and more...  
but do we still need "a better system programming language"!?



## A better system language with the following demands

- Comparable performance to **C/C++**
- Low learning curve and high productivity, especially when compared to **C++/Rust**, while close to the level of **Python/Java** is mostly preferred
- ~~Supports multi-paradigm as most of the modern programming languages~~
- Built-in support for concurrency that closer to **Go**
- Guarantees memory-safety and thread-safety as **Rust**
- Good **interoperability** for most of the popular programming languages
- Can be used for **Linux Kernel** development
- Natively support for **Heterogeneous Parallel Computing**
- Easier to implement new **DSLs** base on it
- With high-level abstraction ability for **HW/SW Co-Design, Co-Development, Co-Simulation, and Co-Debugging** etc
- Comes with a certain foundation of **ecosystem**

**Let's start a progressive enhancement for an existing language to try to meet all of the above in 2022!**

# XVII. Wrap-up



- eBPF-oriented programming is sure to play an more and more important role in Hyper-Converged Infrastructure.
- VM Cloud 1.0 → Container Cloud 2.0 → LightVM/MicroVM Cloud 2.5 → Unikernel Cloud 3.0?
  - Add Wasm here?
  - Add xBPF here?
- Serverless Architecture is a good fit for IoT Edge.
- Security is the first citizen of the Edge Cloud world, and seL4 is sure to play an essential rule in tomorrow's privacy computing.
- How about GaaS(GraalVM as a Service)?
- Embracing the ecosystem of Rust for Linux Kernel, Virtualization and Cloud-native.
- Leveraging Ray as a universal distributed engine for HCI.

- What shall we think about "Beyond Kubernetes"!?
- Upgrading to the next generation ARM, X86, RISC-V, FPGA etc...
- HW/SW co-designed system is the future.
- Do we need a better system programming language?





# THANKS



2021 K+

全球软件研发行业创新峰会

# Reference



Slides/materials from many and varied sources:

- <http://en.wikipedia.org/wiki/>
- <http://www.slideshare.net/>
- <https://codelani.com/lists/languages.html>
- <https://github.com/zoidbergwill/awesome-ebpf>
- <https://www.brendangregg.com/>
- <https://github.com/mikeroyal/eBPF-Guide>
- <https://github.com/lizrice/ebpf-beginners>
- <https://www.anandtech.com/show/17019/apple-announced-m1-pro-m1-max-giant-new-socs-with-allout-performance>
- <https://www.anandtech.com/show/16584/arm-announces-armv9-architecture>
- <https://www.netlox.io/>
- <https://openresty.com/en/edge/>
- <https://www.nextplatform.com/2021/09/09/the-edge-is-just-a-massive-geographically-distributed-cluster/>
- <https://www.zhangjiuzhi.com/2021/10/21/network-security-for-microservices-with-ebpf/>
- <https://thenewstack.io/ebpf-tools-an-overview-of-falco-inspektor-gadget-hubble-and-cilium/>



- <https://blog.px.dev/kubecon-na-2021/>
- <https://thenewstack.io/how-ebpf-streamlines-the-service-mesh/>
- <https://developers.redhat.com/articles/2021/12/16/secure-your-kubernetes-deployments-ebpf#>
- <https://www.containiq.com/post/using-ebpf-to-enhance-kubernetes-monitoring>
- <https://developpaper.com/kubecon-2021-%EF%BD%9C-using-ebpf-instead-of-iptables-to-optimize-the-performance-of-service-grid-data-plane/>
- [https://en.wikipedia.org/wiki/Loadable\\_kernel\\_module](https://en.wikipedia.org/wiki/Loadable_kernel_module)
- <https://zhuanlan.zhihu.com/p/409248433>
- <https://source.android.com/devices/architecture/kernel/bpf>
- <https://embeddedbits.org/introduction-to-trusted-execution-environment-tee-arm-trustzone/>
- <https://developer.arm.com/documentation/den0125/0100>
- <https://www.arm.com/solutions/infrastructure/edge-computing>
- <https://engineering.fb.com/2021/04/27/developer-tools/reverse-debugging/>
- <https://www.kernel.org/doc/Documentation/trace/>
- <https://riscv.org/announcements/2021/12/risc-v-celebrates-incredible-year-of-growth-and-progress-ratifying-multiple-technical-specifications-launching-new-education-programs-and-accelerating-broad-industry-adoption/>



- [https://en.wikipedia.org/wiki/P4\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/P4_(programming_language))
- [https://michael-f-bryan.github.io/cheat-sheets/Rust/rust\\_interop.html](https://michael-f-bryan.github.io/cheat-sheets/Rust/rust_interop.html)
- <https://github.com/jupyter-xeus/xeus>
- <https://github.com/jupyter-xeus/xeus-lua>
- <https://blog.jupyter.org/from-jupyter-to-the-moon-2e432df402c8>
- <https://plus.qconferences.com/plus2022/presentation/there-and-back-again-our-rust-adoption-journey>
- <https://medium.com/walmartglobaltech/introducing-walmarts-l3af-project-how-do-we-use-ebpf-to-provide-network-visibility-in-a-8b9ae4d26200>
- <https://www.graplsecurity.com/post/kernel-pwning-with-ebpf-a-love-story>
- <https://sourceforge.net/software/real-time-operating-systems-rtos/>
- <https://en.wikipedia.org/wiki/DDC-I>
- <https://en.wikipedia.org/wiki/DO-178B>
- <https://semiengineering.com/cxl-and-omi-competing-or-complementary/>
- <https://openmemoryinterface.org/>
- <https://www.producthunt.com/alternatives/codex-by-openai>
- [https://www.phoronix.com/scan.php?page=news\\_item&px=RusticL-Mesa-OpenCL-MR](https://www.phoronix.com/scan.php?page=news_item&px=RusticL-Mesa-OpenCL-MR)
- [https://www.phoronix.com/scan.php?page=news\\_item&px=Roadtest-Linux-Driver-Testing](https://www.phoronix.com/scan.php?page=news_item&px=Roadtest-Linux-Driver-Testing)

- <https://www.inapps.net/how-ebpf-turns-linux-into-a-programmable-kernel-inapps-2022/>
  - ...
-