OpenInfra Days
Shanghai 2019

Rethinking Hyper-Converged Infrastructure for Edge Computing

Feng Li (李枫)
hkli2013@126.com
Nov 5，2019

Agenda

I. Edge Computing

- Overview
- Outstanding Platforms
- Status, Trend, and HCI
- Summary

II. Overall Design

- Design Goals & Principles
- Hardware Platform
- Why is eBPF
- eBPF Development

III. Testbed

- Development Boards
- Software Platform
- Development Model

IV. Computing, Networking, Storage

- HPC
- Networking

- ntopng
- DRedis
- Rethink Lightweight Storage Solutions

V. Containerization, Services, and DevOps

- Kata Container
- Lightweight Kubernetes
- Cilium
- In-Kernel Services

VI. Distributed Framework

- Ray
- My Practice

VII. Messaging & RPC

- gRPC & Protobuf
- Rethink In-Kernel Messaging

VIII. Data Processing

- Apache Arrow
- Lightweight Solution

IX. Artificial Intelligence

- Trends
- TVM
- ARM NN
- My Practice

X. Monitoring, Tuning, and Debugging

- Extending LISA
- Extending ntopng
- My Practice

XI. Security

- Reference Design
- Hardware-Software Co-designed System Security

XII. Miscs

- New Python Runtime
- FPGA

XIII.eBPF-centric New Design

- Software Architecture

XIV.Wrap-up

I. Edge Computing

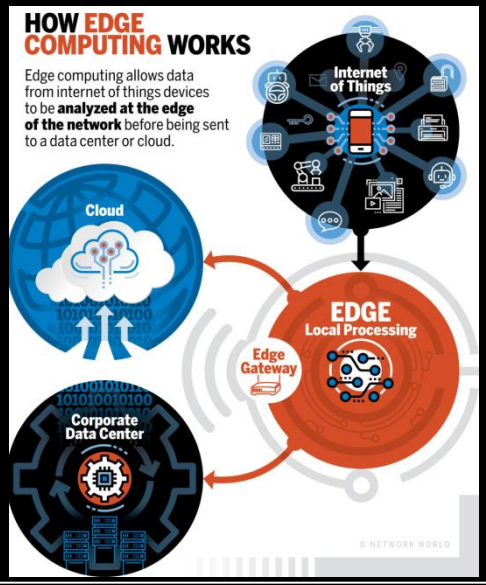
1) Overview

- https://en.wikipedia.org/wiki/Edge_computing

  a distributed computing paradigm which brings computation and data storage closer to the location where it is needed, to improve response times and save bandwidth….

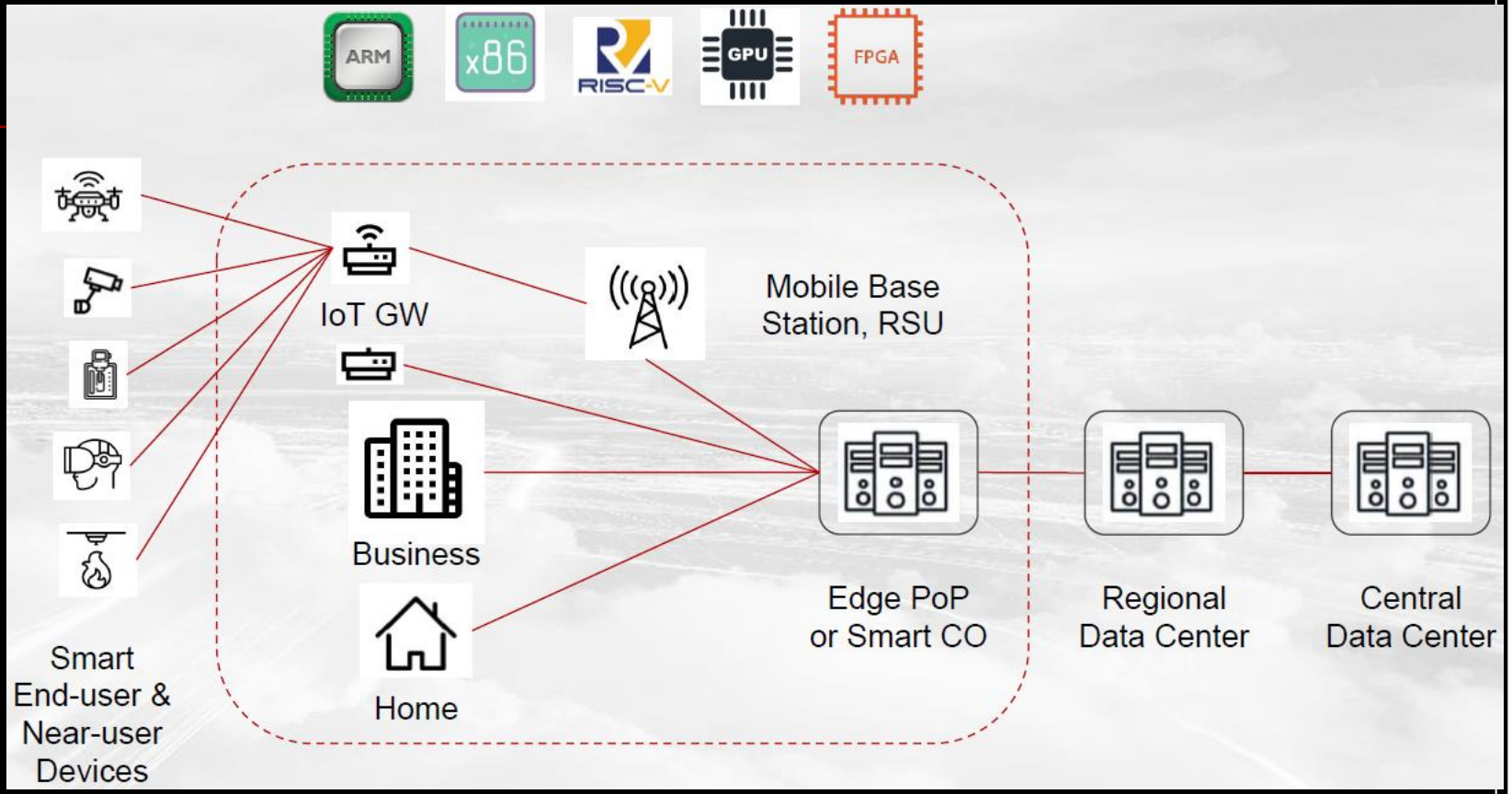
- https://www.networkworld.com/article/3224893/what-is-edge-computing-and-how-it-s-changing-the-network.html

  a way to streamline the flow of traffic from IoT devices and provide real-time local data analysis



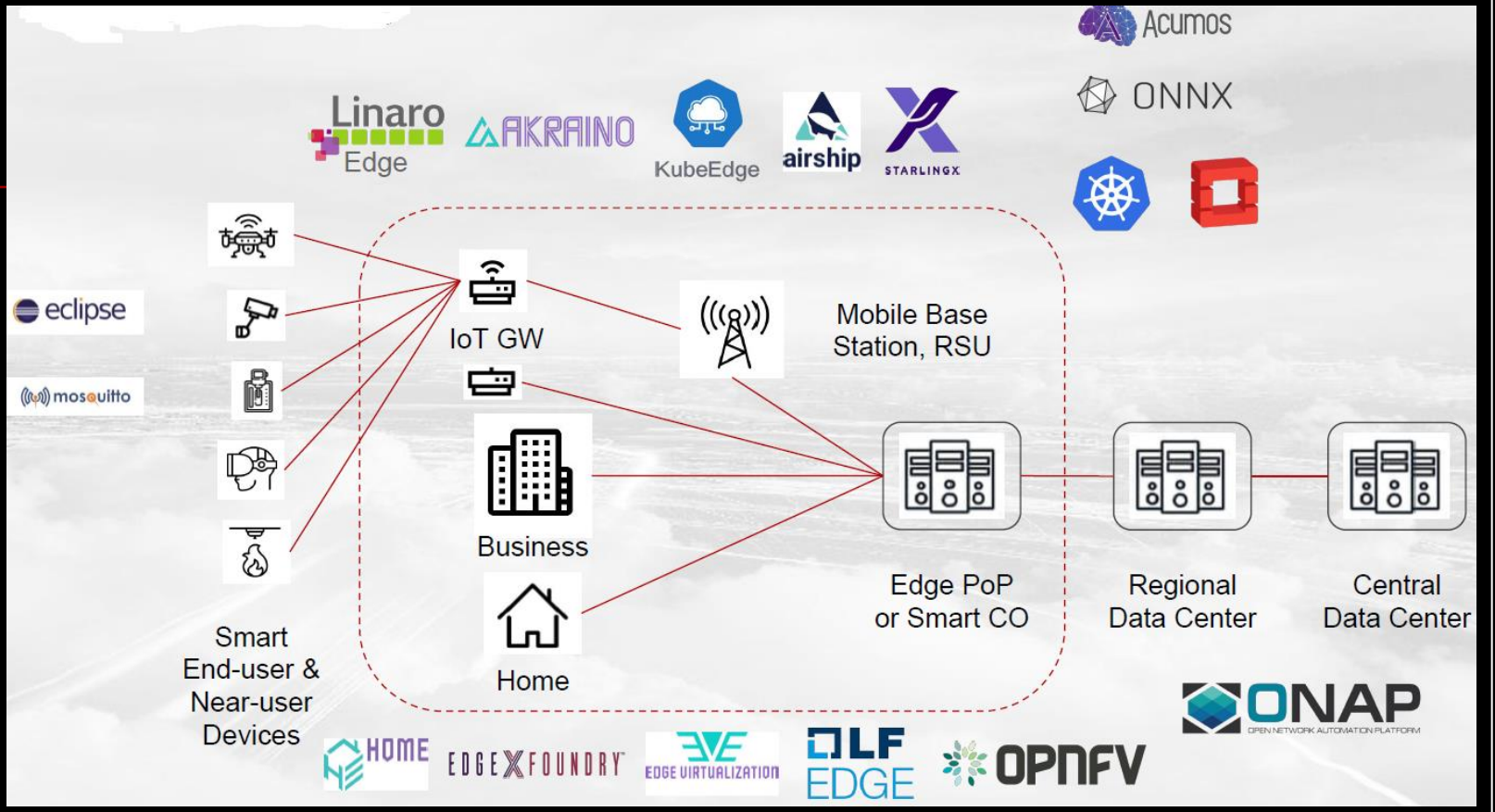
Intelligent Edge with Hardware, Software and Data

- **Hardware**



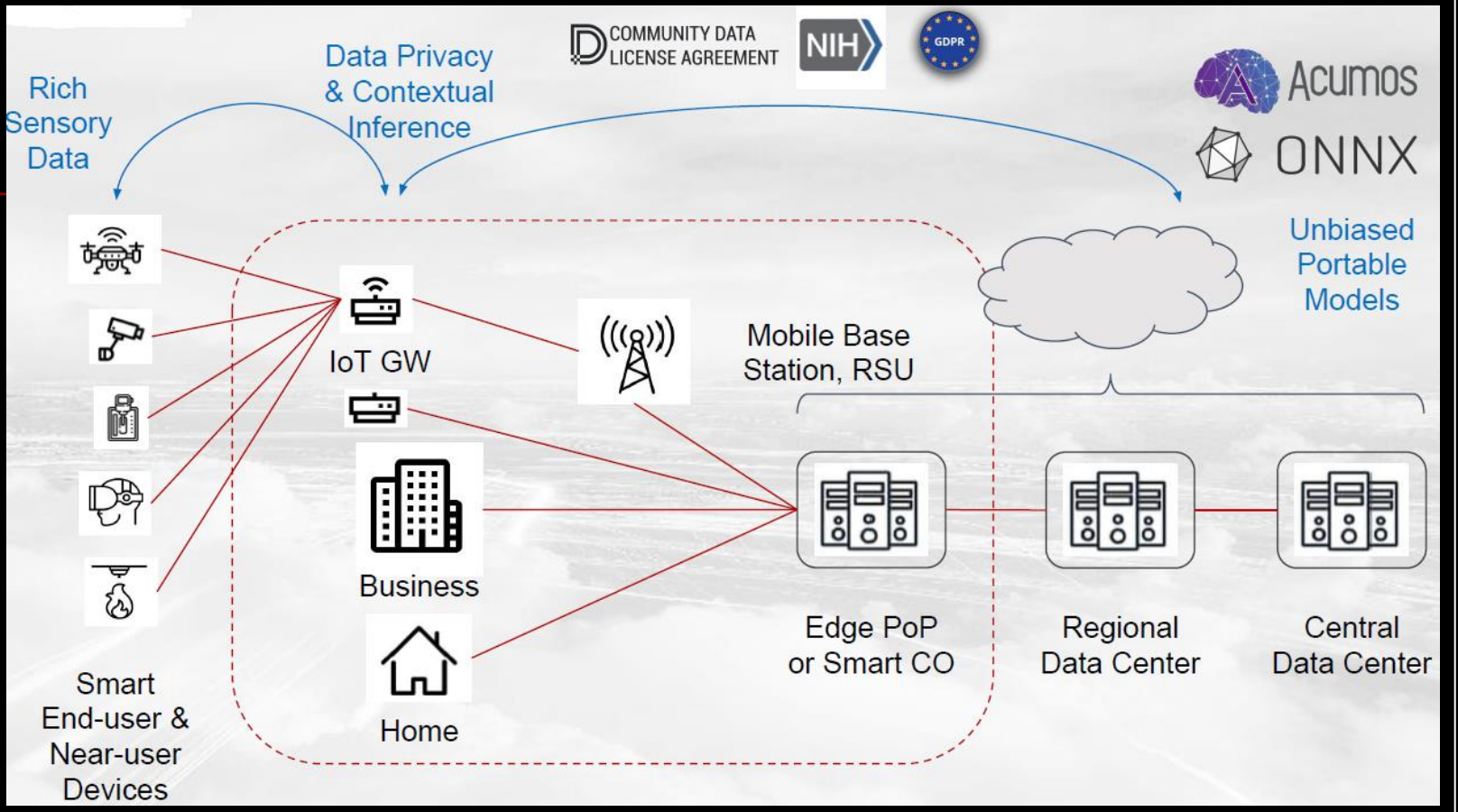
Source: "Deploying the New Intelligent Edge with Open Hardware, Software and Data", Wenjing Chu, ONS North America 2019

Software



Source: "Deploying the New Intelligent Edge with Open Hardware, Software and Data", Wenjing Chu, ONS North America 2019

- **Data**

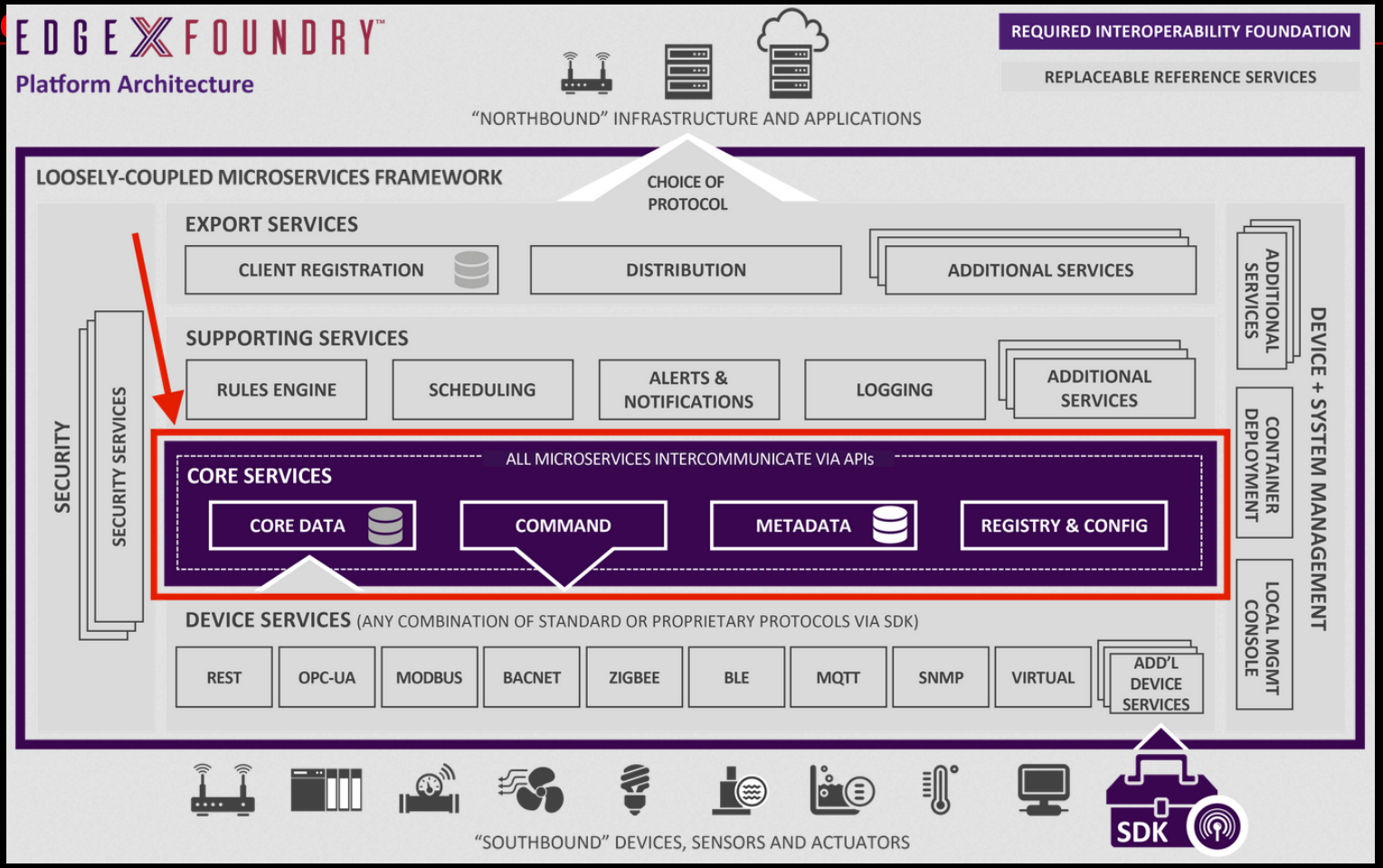


Source: "Deploying the New Intelligent Edge with Open Hardware, Software and Data", Wenjing Chu, ONS North America 2019

2) Outstanding Platforms

2.1 EdgeX Foundry

- https://www.edgexfoundry.org/
-



- Migrating from Java to Go since 1.0 release

2.2 StarlingX

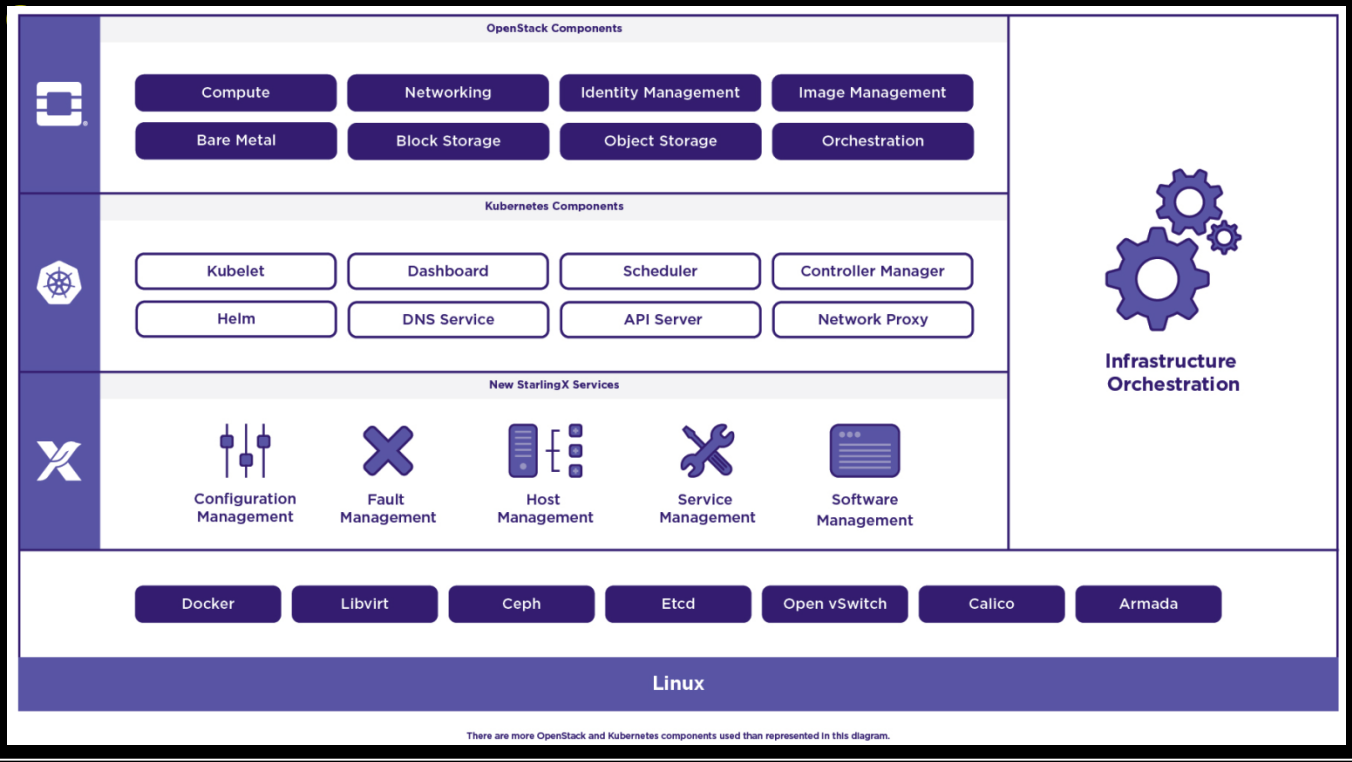
- https://www.starlingx.io/

StarlingX is a complete cloud infrastructure software stack for the edge used by the most demanding applications in industrial IOT, telecom, video delivery and other ultra-low latency use cases. With deterministic low latency required by edge applications, and tools that make distributed edge manageable, StarlingX provides a container-based infrastructure for edge implementations in scalable solutions that is ready for production now.

Headline Features of StarlingX 2.0

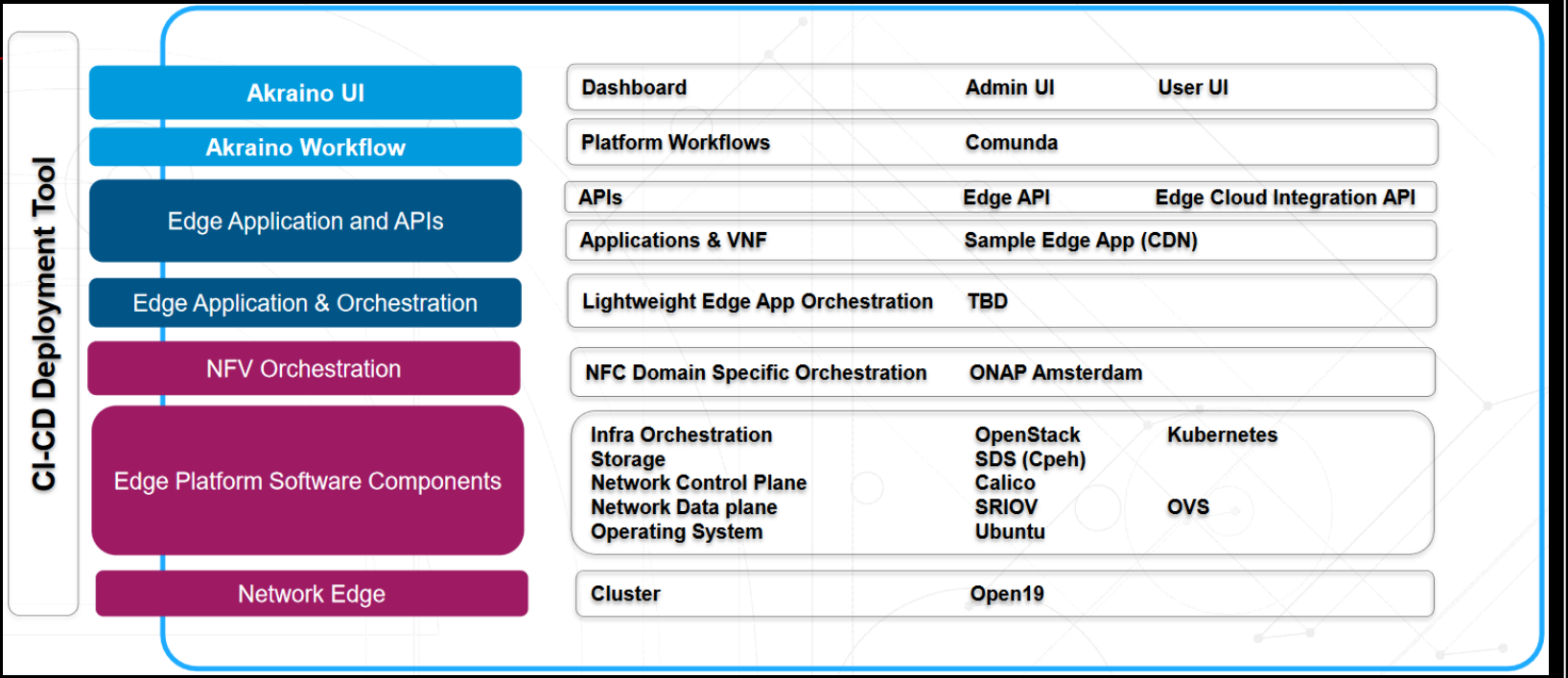
- Hardened cloud-native platform integrating Kubernetes and OpenStack on dedicated physical servers
- Containerized OpenStack based on the Stein release
- Kubernetes-based edge sites for containerized workloads

StarlingX is closely aligned with the OpenStack code base. Out-of-tree patches continue to decline with the new release of StarlingX, and plans are to eliminate them entirely with the Train release this fall.



2.3 Akraino

- https://www.lfedge.org/projects/akraino/
- Akraino Edge Cloud Stack

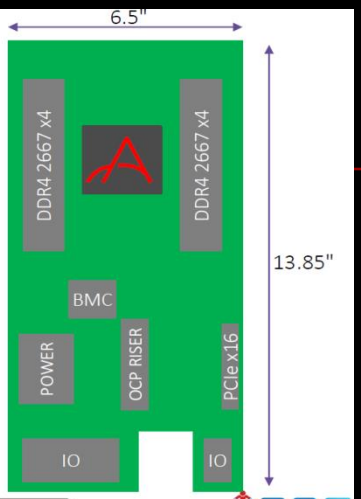


https://www.opennetworking.org/wp-content/uploads/2018/12/ONF_Connect_Open19_and_Akraino.pdf

Akraino on ARM

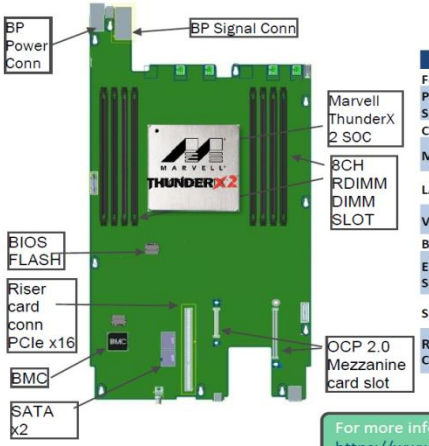
- **Ampere OpenEdge Compute Platform**

Overview	• Compatible with OCP OpenEdge Chassis CPU sled • 32 and 16 core SKUs • 32bit and 64bit Support
Processor	• 32 / 16 Ampere ARMv8 64-bit CPU cores 3.3 GHz Turbo • 32 KB L1 I-cache, 32 KB L1 D-cache per core • Shared 256 KB L2 cache per 2 cores • 32MB globally shared L3 cache • TSMC 16 nm FinFET
Memory	• 8x 72-bit DDR4-2667 channels • Up to 16 DIMMs and 1 TB/socket • ECC, ChipKill, and DDR4 RAS features
I/O	• OCP Mezzanine v2 (Conn. A/B) 10/40/100 GbE NIC • 1 x16 PCIe slot • 2 x M.2 x4 NVME • 4 x SATA3 • 2 x USB 2.0
Power	• 125W TDP 32 cores • 85W TDP 16 cores • Advanced Power Management
Performance	• SPECrate2017_int_peak: 68 • SPECint_rate2006 (peak): 502
Availability	• Sample Q419 • MP Q120



Contact sales@amperecomputing.com for further details

- **Marvell Open Edge ARM Server Board Detail**

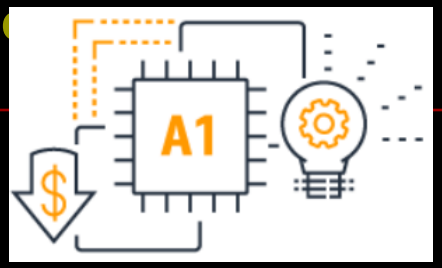


Feature	Specification
Form factor	Proprietary (407.95 x 205.8 mm)
Processor Support	Marvell ThunderX2 CPU with up to 32 cores, 128 threads 2.2GHz in nominal mode, 2.5Hz in Turbo mode.
Chipset	SoC
Memory	8 x DIMM slots support/8 channel DDR4 2666 MT/s @ R-DIMM with 1DPC configuration
LAN	1G Base-T to backplate 1 x Management LAN 10/100/1G
VGA / VRAM	Integrated in BMC
BMC	ASPEED AST2500
Expansion Slot	1 x PCIe x16 (@Gen 3 x16) 1x OCP mezzanine PCIe (@Gen 3 x16)(TYPE 1 P1,P2,P3,P4 NCSI support)
Storage	2 x SATA(6Gb/s) Optional PCIe M.2 on riser
Rear IO Connector	2 x USB3.0 1 x ID Button, System RST BTN; PWR BTN,

For more information please refer to
https://www.marvell.com/server-processors

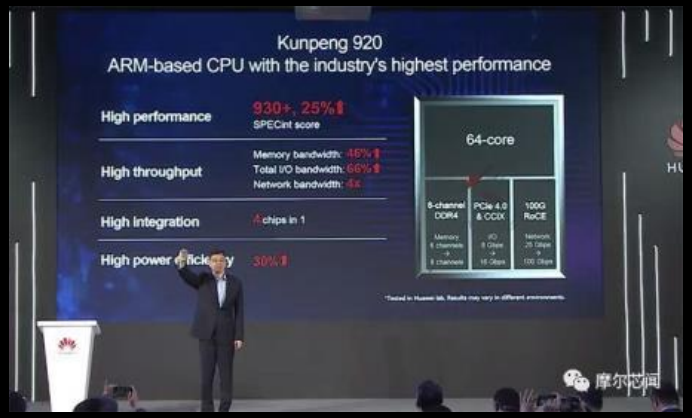
https://www.opennetworking.org/wp-content/uploads/2019/09/2pm-Tina-Tsou-Efficient-Computing-at-the-Edge-with-Arm-for-SEBA.pdf

3) Status, Trend, and HCI
3.1 ARM Ecosystem in 2019



Model	vCPUs	Memory (GiB)	Instance Storage	Network Bandwidth (Gbps)	EBS Bandwidth (Mbps)
a1.medium	1	2	EBS-Only	Up to 10	Up to 3,500
a1.large	2	4	EBS-Only	Up to 10	Up to 3,500
a1.xlarge	4	8	EBS-Only	Up to 10	Up to 3,500
a1.2xlarge	8	16	EBS-Only	Up to 10	Up to 3,500
a1.4xlarge	16	32	EBS-Only	Up to 10	3,500
a1.metal	16*	32	EBS-Only	Up to 10	3,500

* a1.metal provides 16 physical cores



ARM Neoverse

- Arm Announces Neoverse N1 & E1 Platforms & CPUs: Enabling A Huge Jump In Infrastructure Performance

96 Comments + Add A Comment

by Andrei Frumusanu on February 20, 2019 9:00 AM EST

Posted in Infrastructure | Arm | Servers | Cortex-A76 | Neoverse | CPU | Cortex A65AE | Neoverse N1 | Neoverse E1

ARM IN INFRASTRUCTURE

ARM announces Ares

By: **Linus Torvalds** (torvalds.delete@this.linux-foundation.org), February 20, 2019 9:36 am Room: Moderated Discussions

nobody in particular (nobody.delete@this.nowhe.re) on February 20, 2019 7:35 am wrote:
> Write-up at Anandtech
>
> The stated SPECint numbers are superb. I'm looking forward
> to spec2017, but this is a very, very promising start.

We'll see if it actually delivers, but ARM is certainly looking a whole lot better than it used to.

They've been nicely strengthening their memory model, to the point that these days it's actually one of the better ones. They even seem to have decided that I$ coherency matters. Good for them. Doing cache coherency in software is crazy, and people who think it's a good idea don't understand the complexities of reality. Maybe some day it will even become architectural.

And I have to say, I like the direction ARM is going with vector math a *lot* more than the AVX512 that Intel is pushing. I don't know how well it works in practice, but the whole "let's try to do something that works for different vector lengths" is laudable. I'm quite tired of the model where Intel introduces yet another incompatible model every few years.

I still will hold judgement until we actually see widely available hardware that people actually can use for development and deployment. I've just seen too many promises and "released" hardware that never went anywhere and nobody really had reasonably available.

In particular, I do wish they didn't push the "hyperscale" design so much. Maybe they got the scaling working, but honestly, I doubt it. It just takes time and effort and learning. Don't try to jump to 64-128 core targets until you've had a few years of just plain getting something like a "simple" 8-core one right. Which they haven't actually demonstrated, so far.

But hey, maybe they'll surprise me.

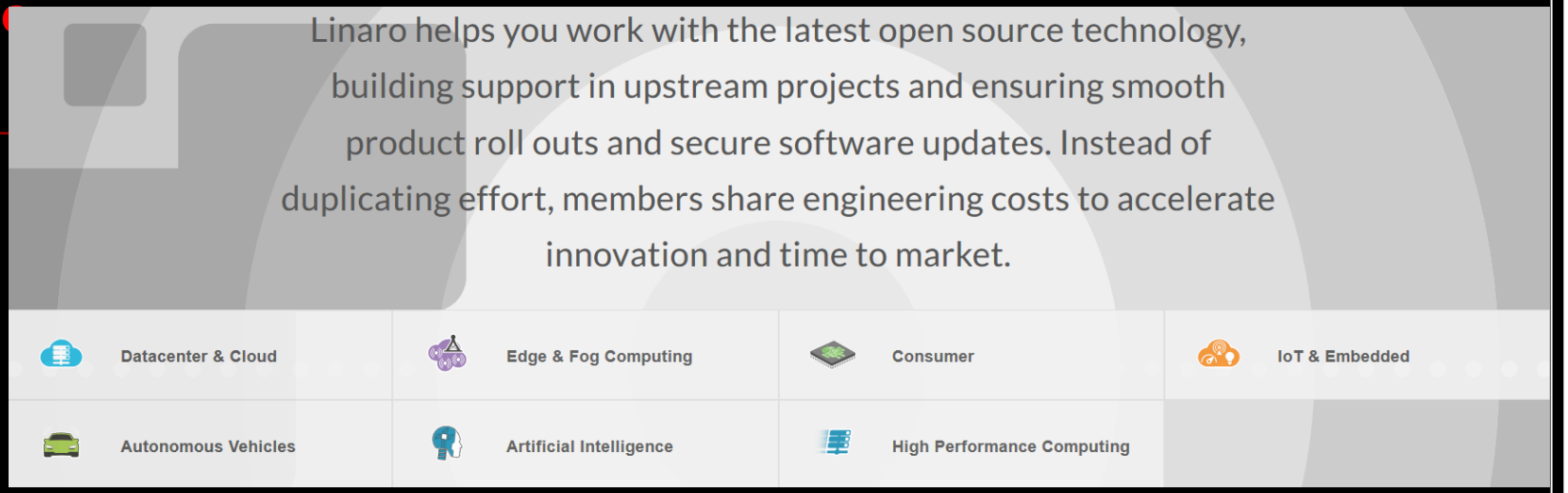
Linus

▲ antirez 20 hours ago | parent | favorite | on: Linus Torvalds on Why ARM Won't Win the Server Spa...

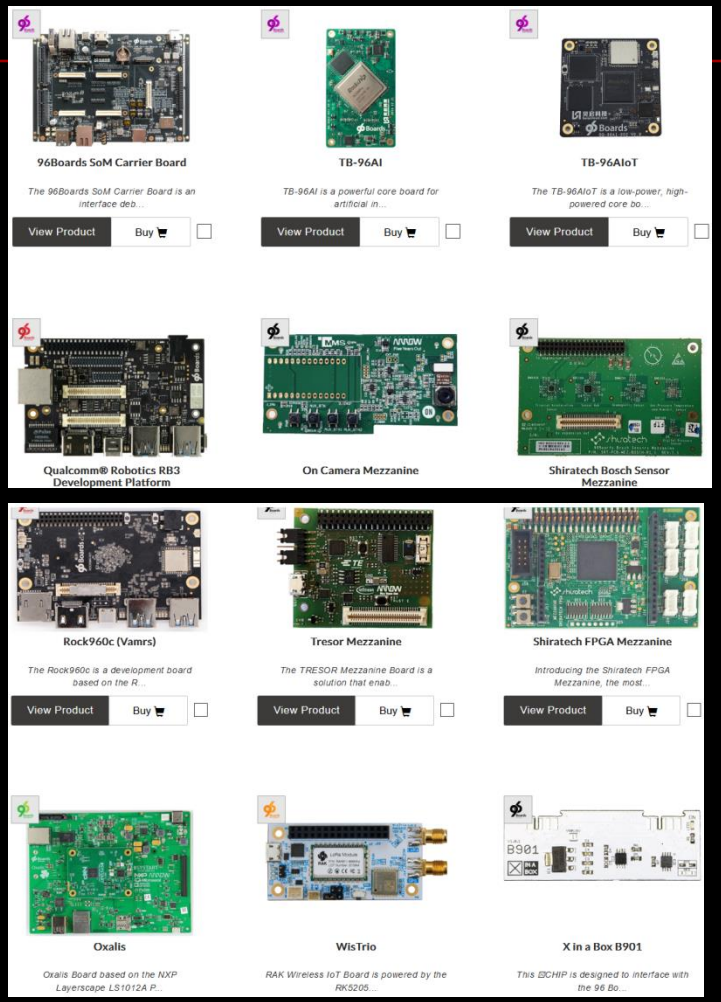
It's extremely hard to agree with Linus on that. One problem in his argument is that he believes that everybody has a kernel hacker mindset: most today's developers don't care about environment reproducibility at architecture level. The second problem is that he believes that every kind of development is as platform sensitive as kernel hacking, and he even makes the example of Perl scripts. The reality is that one year ago I started the effort to support ARM as a primary architecture for Redis, and all I had to do is to fix the unaligned accesses, that are anyway fixed in ARM64 almost entirely, and almost fixed also in ARM >= v7 if I remember correctly, but for a subset of instructions (double words loads/stores). Other than that, Redis, that happens to be a low level piece of code, just worked on ARM, with all the tests passing and no stability problems at all. I can't relate to what Linus says there. If a low level piece of code written in C, developed for many years without caring about ARM, just worked almost out of the box, I can't see the Ruby or Node application to fail once uploaded to an ARM server.

Linaro

- https://www.linaro.org/
-



- **96boards**
  https://www.96boards.org/
  https://www.96boards.org/specifications/

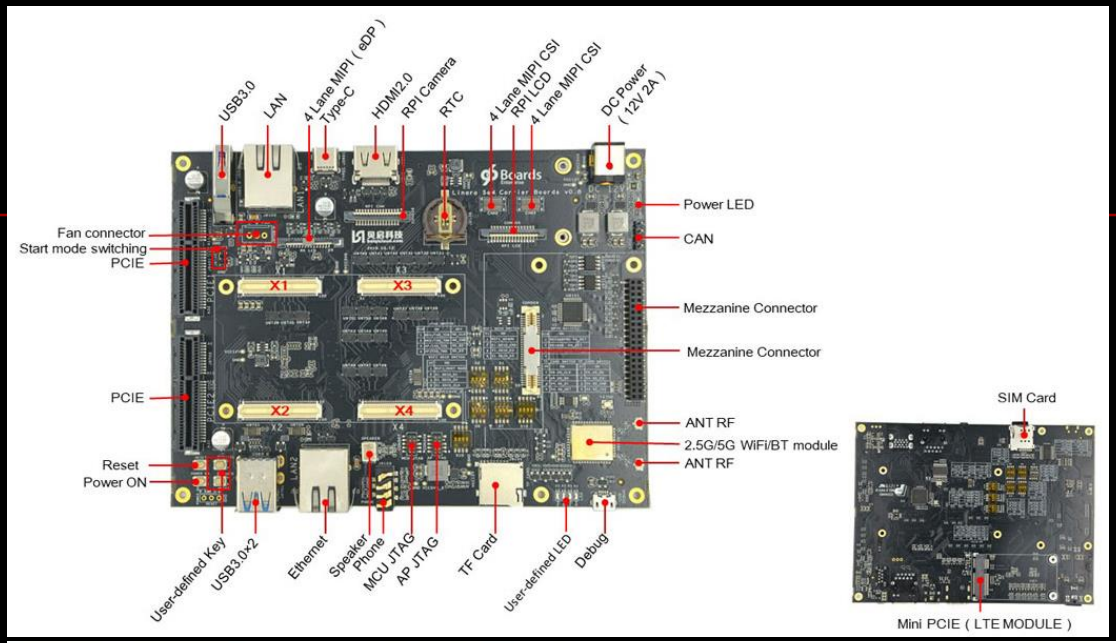


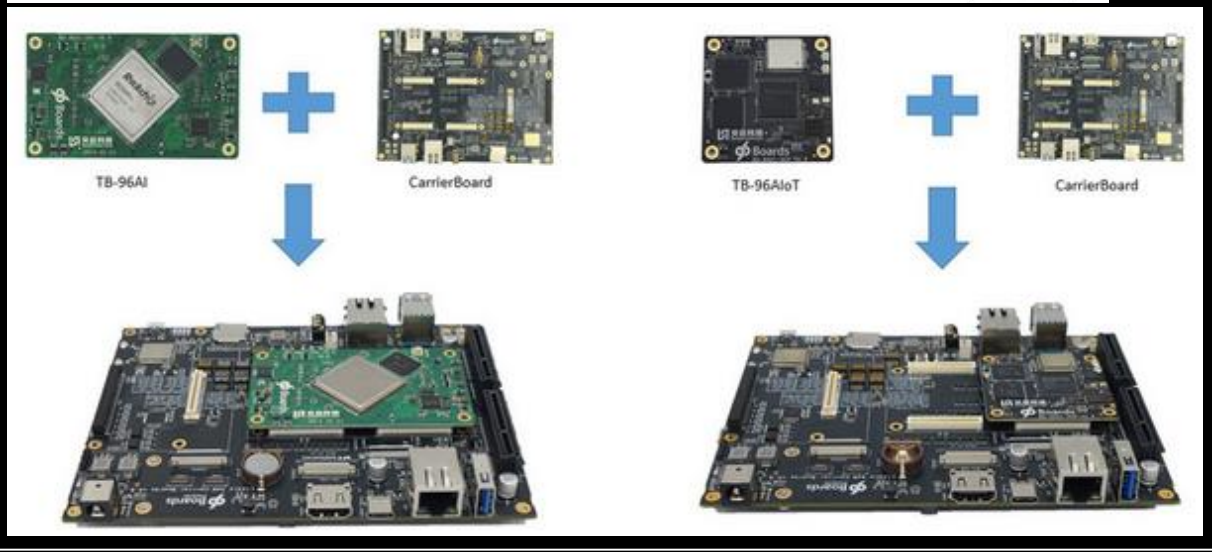
SOM/COM

- **https://www.linaro.org/news/linaro-announces-launch-of-96boards-system-on-module-som-specification/**
- **http:///linuxgizmos.com/linaro-launches-two-96boards-som-specifications/**
- **http://static.linaro.org/assets/specifications/96BoardsComputeSoMSpecificationV1.0.pdf**
- **https://static.linaro.org/assets/specifications/96BoardsWirelessSoMSpecificationV1.0.pdf**
- **C**

Newly released for ArmTechCon 2018, the miniNodes Raspberry Pi 3 Computer on Module (CoM) 5-node Carrier Board enables extreme edge computing and IoT workload processing in a small, easy to use platform. Final engineering is wrapping up, so we are opening up Preorders, with expected delivery in February / March 2019.

- http://www.beiqicloud.com/product_detail.html?pid=CarrierBoard



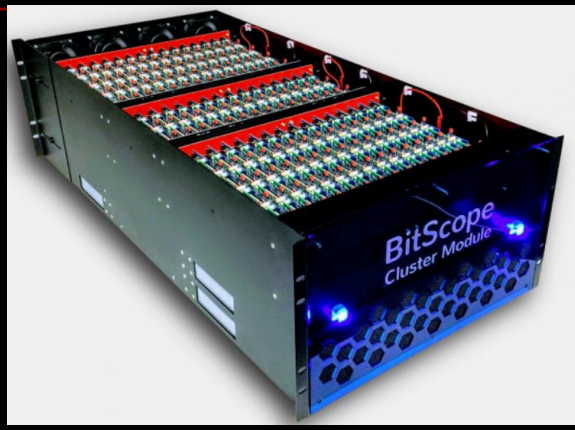


3.2 Clustering

BitScope Pi Cluster

- http://cluster.bitscope.com/

Scalable clusters make HPC R&D easy as Raspberry Pi



CLUSTER MODULE

The basic building block of scalable **BitScope Clusters**.

High Density, Low Cost

Each module packs **144 active nodes**, six spare nodes and one cluster manager node in a single 6U drawer. Build a **1000 node** cluster in **42U** for less than **$150/node**.

Low Power, Low Heat

At less than **5W/node** in typical operation you need only **6kW** to run **1000 nodes** including network fabric and air flow.

Highly Affordable

The world's most cost effective scalable solution. Inexpensive to build, operate and maintain.

Extremely Flexible

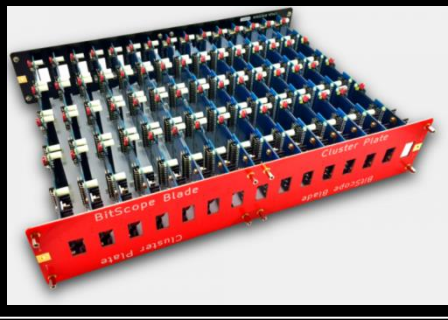
Built with the amazing ARM based Raspberry Pi. Many software options and a vast developer community.

Perfect for Research

Develop new cluster architectures that scale at very low cost before commiting to production designs.

Ideal for Education

The world's leading computing education platform can now be used to teach network, cluster and cloud computing. All open source.



CLUSTER PACK

The key building block of every **Cluster Module**.

Power & Mounting Solution

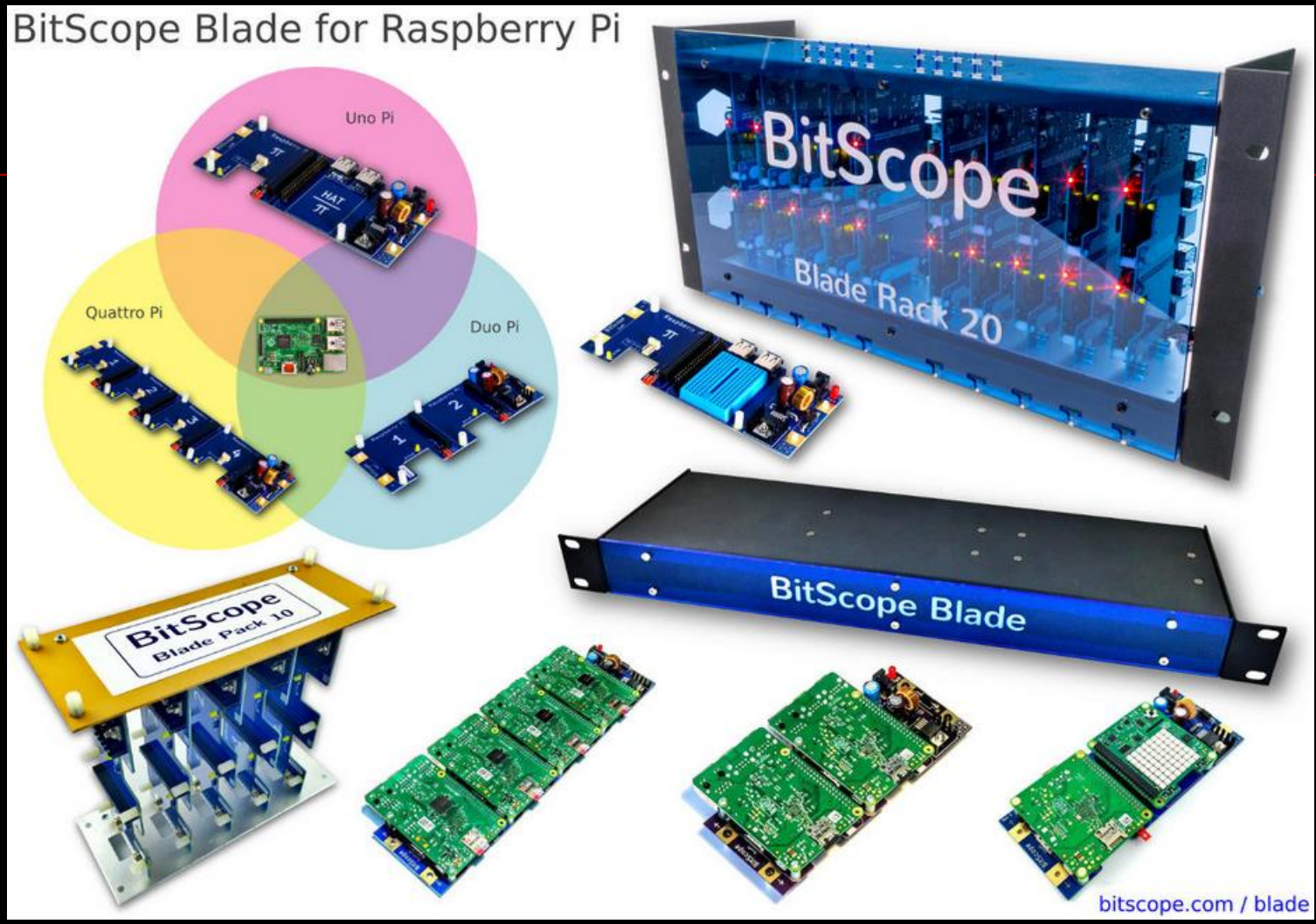
Cluster Packs simplify the mounting of thousands of nodes.

They route and regulate power to every node, locally.

No Wires, No Problems

Power the packs and you power the cluster.

- **http://bitscope.com/product/blade/?p=about**



Oracle RPi Supercomputer

- **https://www.servethehome.com/oracle-shows-1060-raspberry-pi-supercomputer-at-oow/**

Oracle Shows 1060 Raspberry Pi Supercomputer

What's the matter

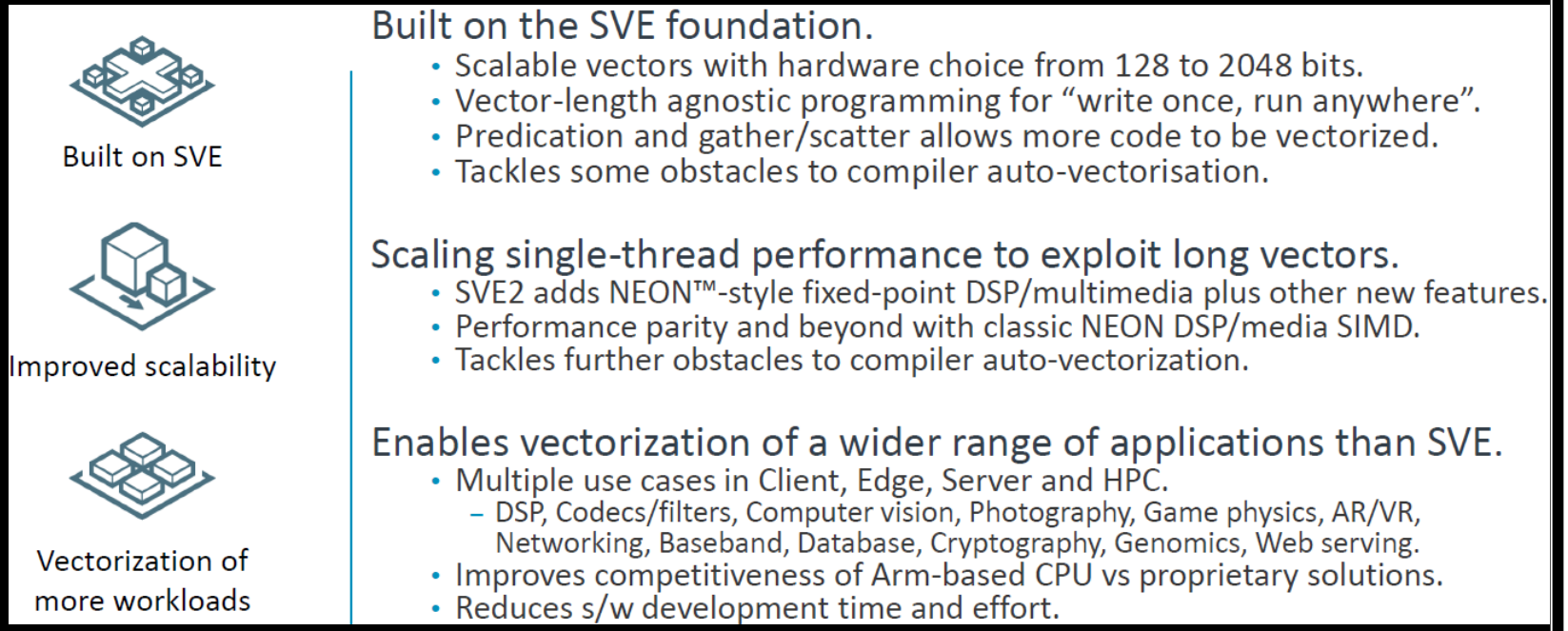
- **https://www.servethehome.com/aoa-analysis-marvell-thunderx2-equals-190-raspberry-pi-4/**

Raspberry Pi Units	
Raspberry Pi 4 4GB	$55.00
PoE Hat	$20.00
3M Ethernet Cable	$1.50
Case	$7.00
Total RPi Unit Cost	**$83.50**
Rack Hardware	
Rack Shelf	$78
1/48th Rack Shelf	**$1.63**
PoE Switches and Cables	
PoE Switch Mikrotik CRS328-24P-4S+-RM	$350
2x SFP+ Uplink Cables	$50
1/24th PoE Network	**$16.67**
Network Aggregation Layer	
Aggregation SFP+ Switch	$500
1/288th Aggregation SFP+ Switch	**$1.74**
Storage	
FreeNAS Mini XL+ 24TB Raw	$2,400
1/288th Shared Storage	**$8.33**
Total Per RPi 4 in CI/CD Cluster	
Total Cost Per RPi 4 4GB	**$111.86**

3.3 New Technologies in the Arm Architecture

SVE2 & TME

- https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/new-technologies-for-the-arm-a-profile-architecture
- Scalable Vector Extension v2



Source: "New Technologies in the Arm Architecture", Nigel Stephens(Fellow, Arm Ltd) Linaro Connect 2019(Bangkok)

Transactional Memory Extension

Hardware Transactional Memory

Improved scalability

Simpler software design

Hardware Transactional Memory (HTM) for the Arm architecture.

- Improved competitiveness with other architectures that support HTM.
- Strong isolation between threads.
- Failure atomicity.

Scaling multi-thread performance to exploit many-core designs.

- Database.
- Network dataplane.
- Dynamic web serving.

Simplifies software design for massively multi-threaded code.

- Supports Transactional Lock Elision (TLE) for existing locking code.
- Low-level concurrent access to shared data is easier to write and debug.

Enabling tools and documentation

LLVM

- Upstreaming of SVE2/TME assembly support to begin immediately.
- Goal of initial SVE2 auto-vectorization & ACLE upstream by end Q1 CY20.

GNU Tools

- Upstreaming of SVE2/TME assembly, initial SVE2 auto-vectorization & ACLE to begin immediately. (SVE autovec present since GCC8).
- Targeting GCC10 release at end Q1 CY20.

Glibc

- Aiming for Transactional Lock Elision support in upstream Glibc by Q3 CY19.

Arm Tools

- SVE2/TME support in Arm compiler, debugger and fast models planned for H2 CY2019.

Documentation

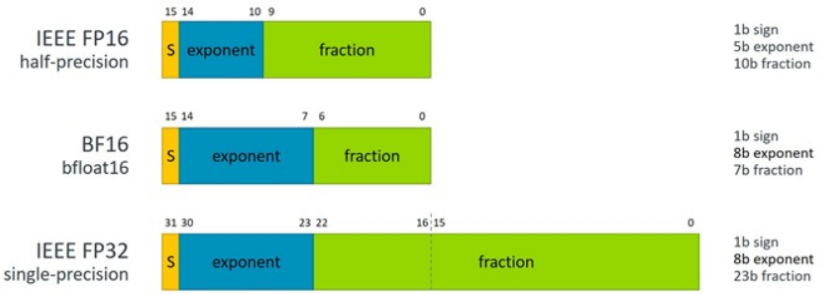
- SVE2/TME ISA XML available by 15 April 2019.
  - developer.arm.com/architectures.
- SVE literature at developer.arm.com/hpc.
  - No ABI changes required by SVE2.
- SVE2 literature – including VLA programmer's guide with code examples – available soon.

Source: "New Technologies in the Arm Architecture", Nigel Stephens(Fellow, Arm Ltd) Linaro Connect 2019(Bangkok)

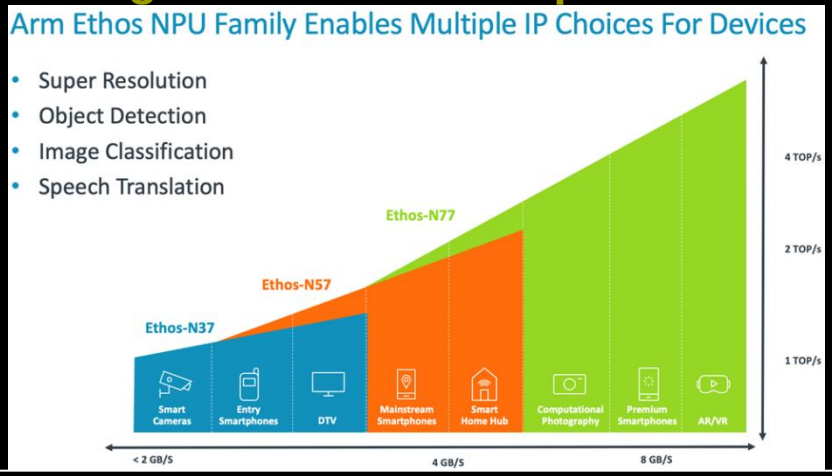
AI

- **https://community.arm.com/developer/ip-products/processors/b/ml-ip-blog/posts/bfloat16-processing-for-neural-networks-on-armv8_2d00_a**

The next revision of the Armv8-A architecture will introduce Neon and SVE vector instructions designed to accelerate certain computations using the BFloat16 (BF16) floating-point number format. BF16 has recently emerged as a format tailored specifically to high-performance processing of Neural Networks (NNs). BF16 is a truncated form of the IEEE 754 [ieee754-2008] single-precision representation (IEEE-FP32), which has only 7 fraction bits, instead of 23 (see **Figure 1**).



- **https://www.arm.com/company/news/2019/10/new-arm-ip-brings-intelligent-immersive-experiences-to-mainstream-markets**

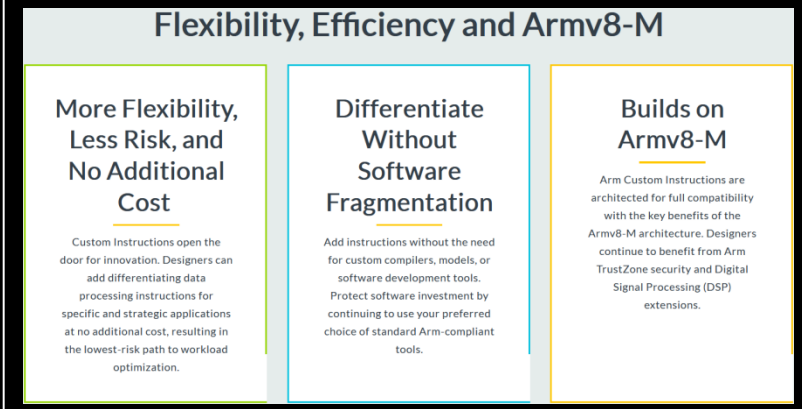


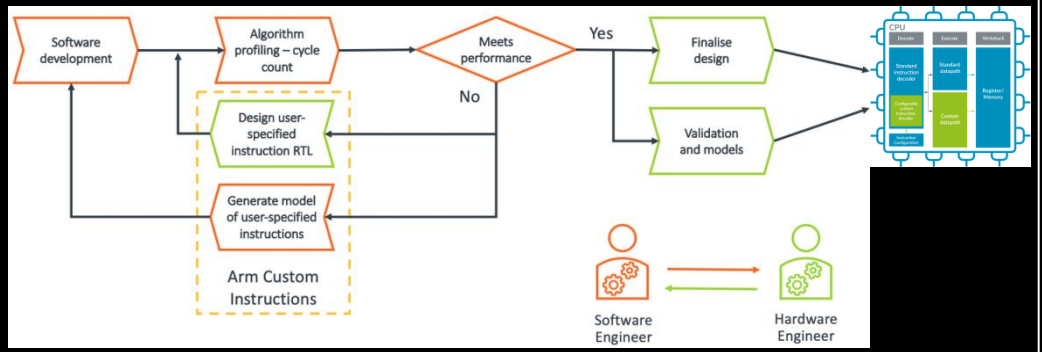
AI

- https://www.arm.com/company/news/2019/10/arm-enables-custom-instructions-for-embedded-cpus?utm_source=twitter&utm_medium=social&utm_campaign=2019_techcon-marketing_mk17_na-&utm_term=arm-enables-custom-instructions&utm_content=press-release
- https://www.arm.com/why-arm/technologies/custom-instructions
- https://developer.arm.com/architectures/instruction-sets/custom-instructions
- Arm Custom Instructions for the Armv8-M architecture enable you to push performance and efficiency further by adding application domain specific features in small embedded processors, while maintaining all the advantages of Arm's software ecosystem.

  Arm Custom Instructions allow you to add a customizable module, called configuration space, inside the Cortex-M33 processor. This module is driven by the pre-decoded instructions and shares the same interface as the standard arithmetic logic unit (ALU) of the CPU. Adding custom instructions to a customizable CPU requires two steps:

  1. Providing a configuration file that lists the regions you want to use for adding your own custom instructions.
  2. Building the data path for your own custom instructions and integrating it into the configuration space.

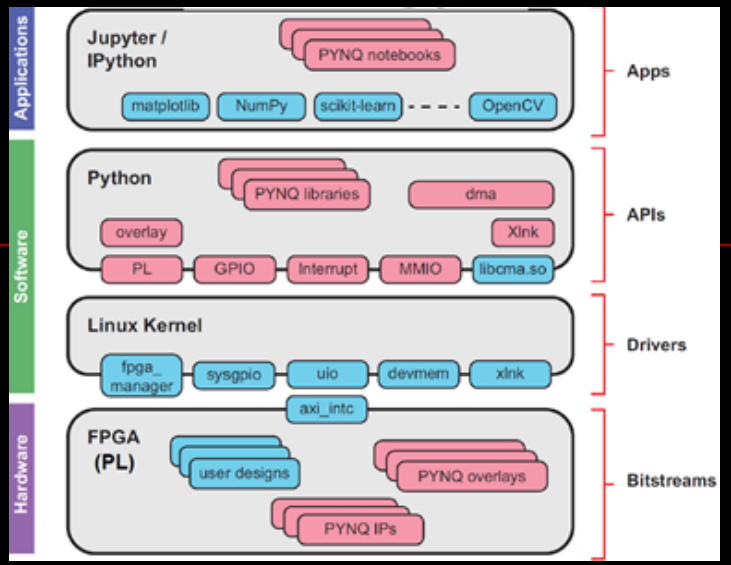




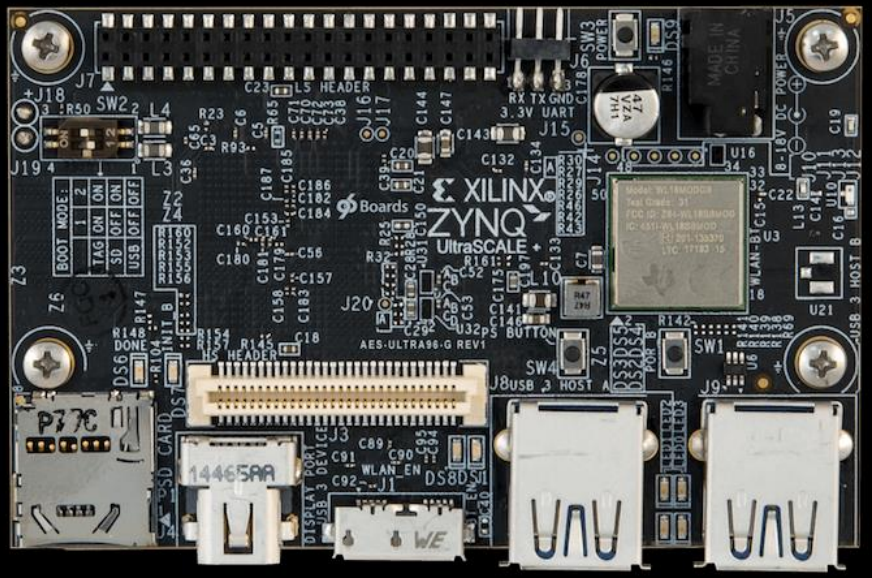
- **http://www.pynq.io/**

PYNQ is an open-source project from Xilinx® that makes it easy to design embedded systems with Xilinx Zynq® Systems on Chips (SoCs).
Using the Python language and libraries, designers can exploit the benefits of programmable logic and microprocessors in Zynq to build more capable and exciting embedded systems.
PYNQ users can now create high performance embedded applications with

- parallel hardware execution
- high frame-rate video processing
- hardware accelerated algorithms
- real-time signal processing
- high bandwidth IO
- low latency control



- **https://www.96boards.org/product/ultra96/ (~250$)**



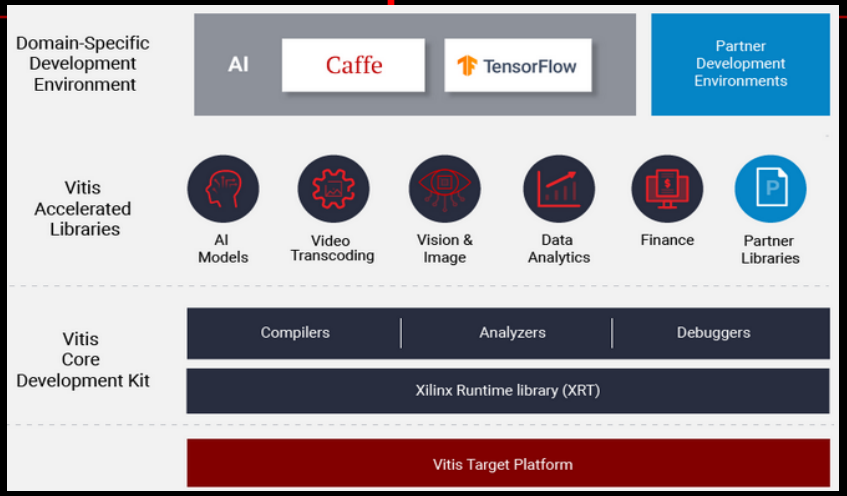
- Xilinx Zynq UltraScale+ MPSoC ZU3EG A484
- Micron 2 GB (512M x32) LPDDR4 Memory
- Delkin 16 GB microSD card + adapter
- PetaLinux environment available for download
- Microchip Wi-Fi / Bluetooth
- Mini DisplayPort (MiniDP or mDP)
- 1x USB 3.0 Type Micro-B upstream port
- 2x USB 3.0, 1x USB 2.0 Type A downstream ports
- 40-pin 96Boards Low-speed expansion header
- 60-pin 96Boards High-speed expansion header
- 85mm x 54mm form factor
- Linaro 96Boards Consumer Edition compatible

3.4 FPGA

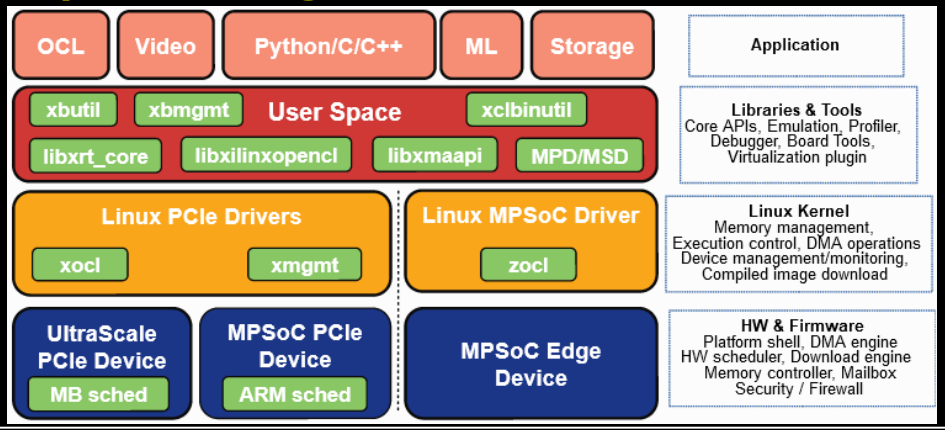
Xilinx Vitis & XRT

- https://www.xilinx.com/products/design-tools/vitis.html

  Unified software platform for all developers

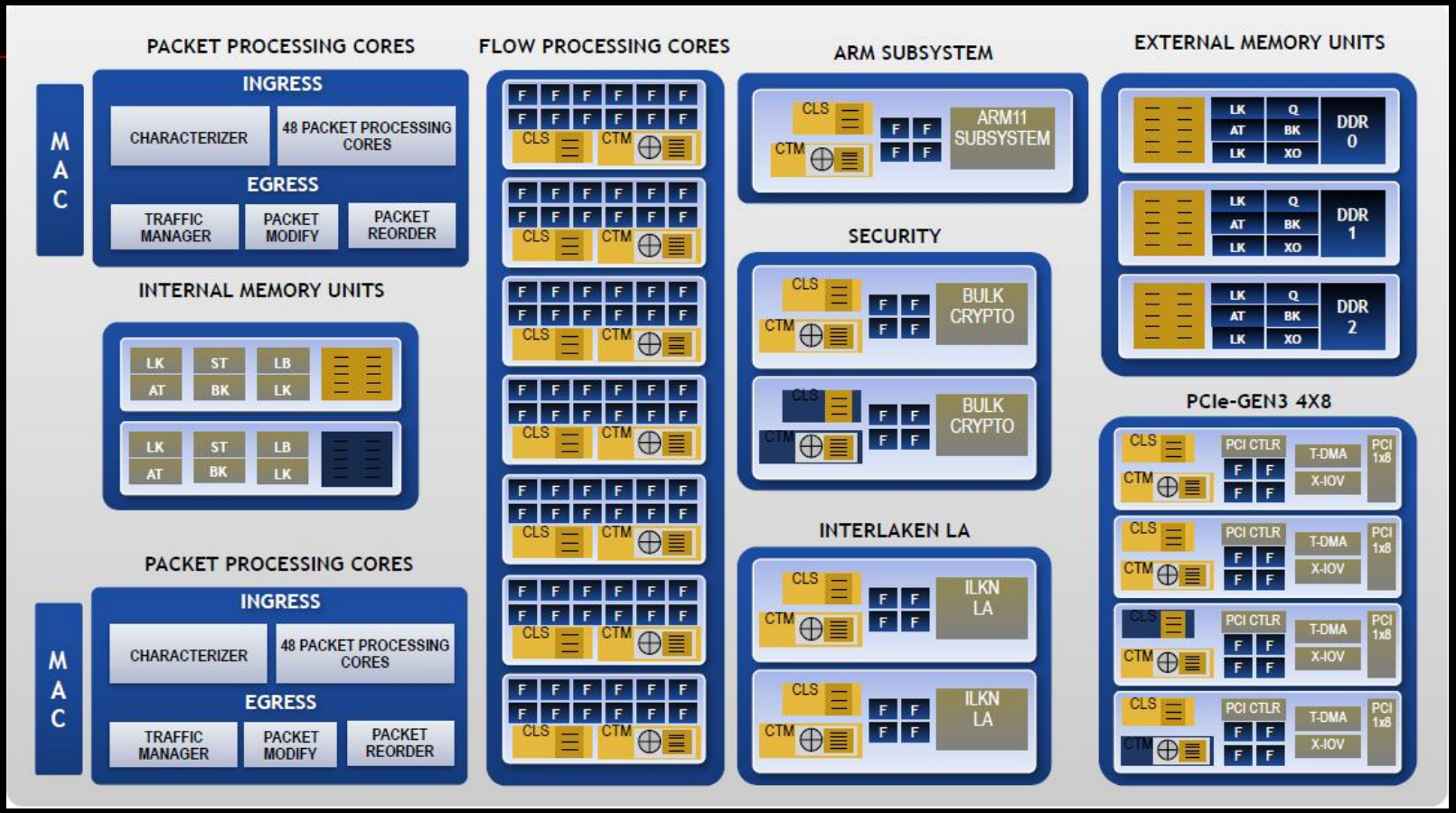


- https://xilinx.github.io/XRT/



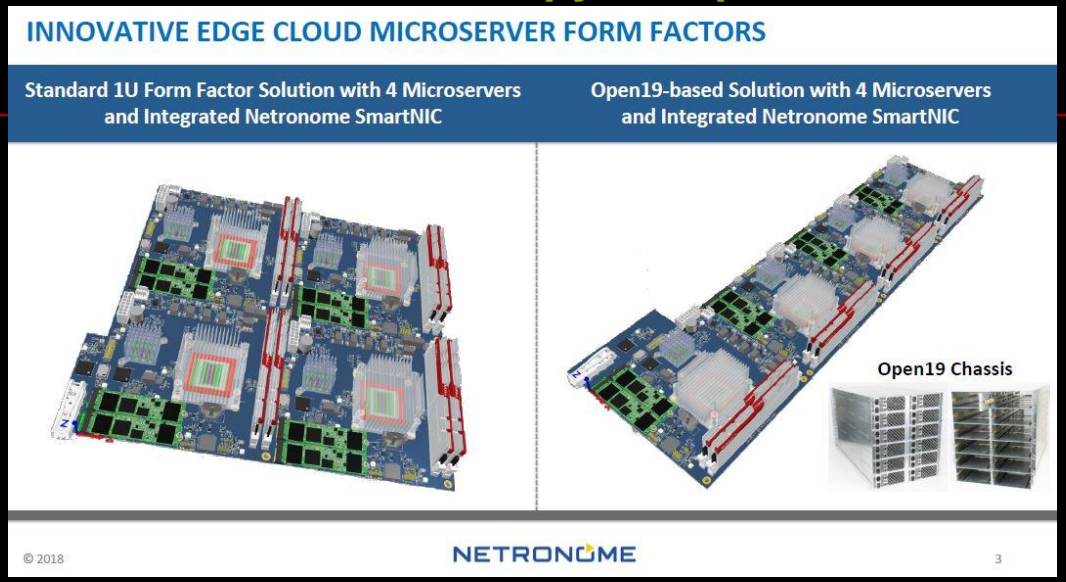
3.5 SmartNIC

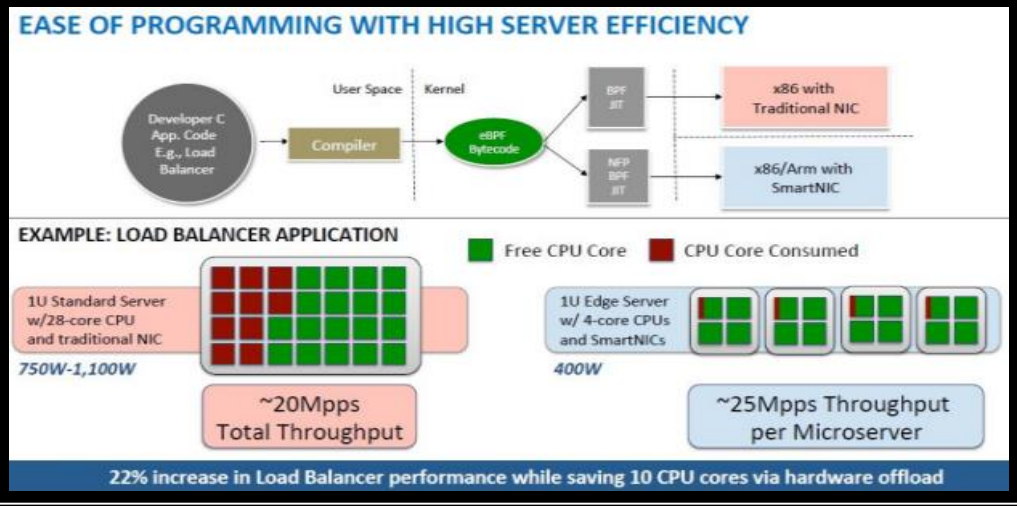
- https://www.nextplatform.com/2018/08/06/living-in-the-smartnic-future/
- Netronome NFP-6XXX DETAILED BLOCK DIAGRAM



Use Case

- https://www.servethehome.com/packet-launching-microservers-netronome-smartnics-epyc-surprise/

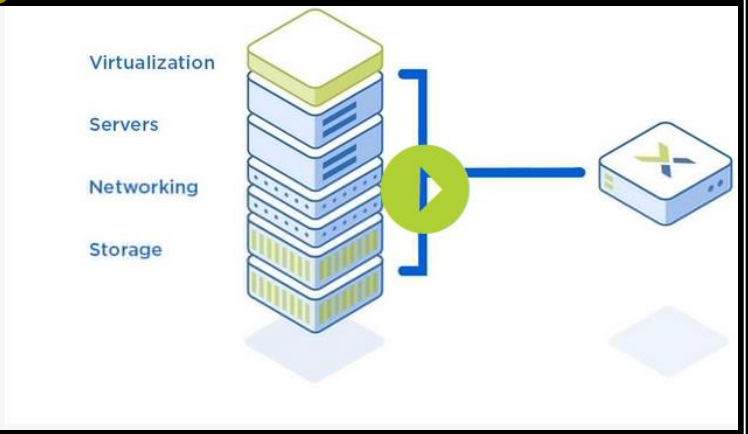




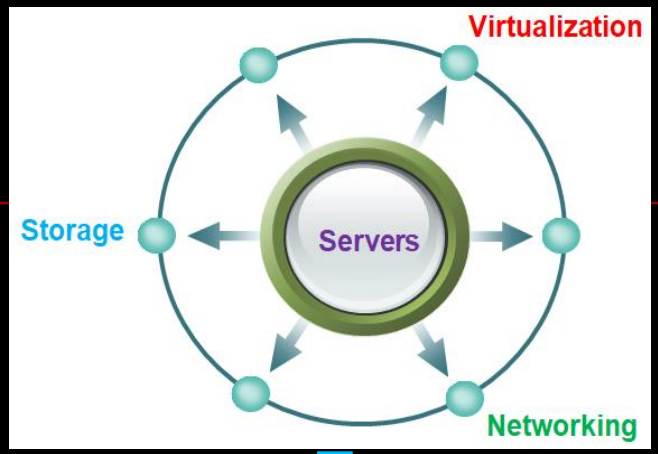
3.6 HCI (Hyper-Converged Infrastructure)

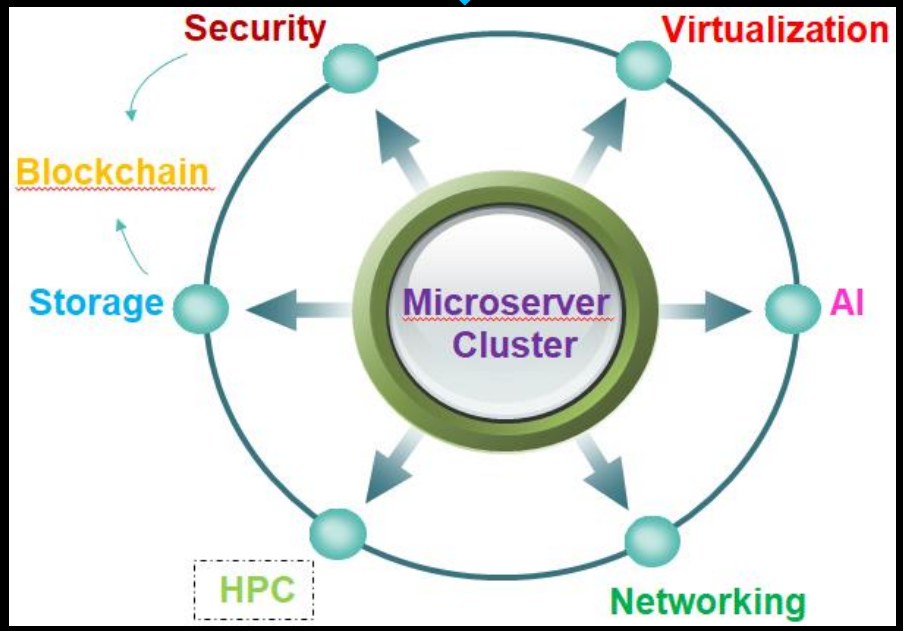
- https://en.wikipedia.org/wiki/Hyper-converged_infrastructure
- software-defined IT infrastructure that virtualizes all of the elements of conventional "hardware-defined" systems…
- https://www.nutanix.com/hyperconverged-infrastructure

Hyperconverged infrastructure (HCI) combines common datacenter hardware using locally attached storage resources with intelligent software to create flexible building blocks that replace legacy infrastructure consisting of separate servers, storage networks, and storage arrays. Benefits include lower TCO, increased performance, and greater productivity within IT teams.



- **Trend (from my point of view)**





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

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
- …

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

3) Why is eBPF

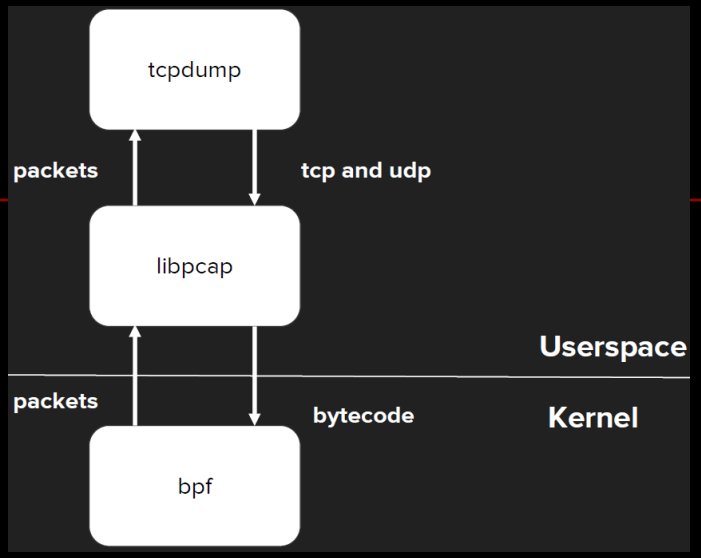
3.1 BPF (Berkeley Packet Filter, aka cBPF)

- https://en.wikipedia.org/wiki/Berkeley_Packet_Filter
- http://www.tcpdump.org/papers/bpf-usenix93.pdf
- History
  - Before BPF, each OS (Sun, DEC, SGI etc) had its own packet filtering API
  - In 1993: Steven McCanne & Van Jacobsen released a paper titled the *BSD Packet Filter (BPF)*
  - Implemented as “Linux Socket Filter” in kernel 2.2
  - While maintaining the BPF language (for describing filters), uses a different internal architecture

Source: ebpfbasics-190611051559.pdf

What is it

- Network packet filtering, Seccomp
- Filter Expressions → Bytecode → Interpret
- Small, in-kernel VM, Register based, switch dispatch interpreter, few instructions
- BPF uses a simple, non-shared buffer model made possible by today's larger address space



Source: ebpfbasics-190611051559.pdf

- Bytecode, register based VM, with a limited instruction set
- Runs in-kernel, designed for fast packet filtering
- 32-bit instructions (LOAD, STORE, ALU, BRANCH, RETURN)
- 2, 32-bit registers (A, X), hidden frame pointer

Source: understandingebpfinahurry-190611040804.pdf

https://blog.cloudflare.com/bpf-the-forgotten-bytecode/

…

3.2 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”

BPF Features by Linux Kernel Version

- https://github.com/iovisor/bcc/blob/master/docs/kernel-versions.md

Instruction Set

- https://github.com/iovisor/bpf-docs/blob/master/eBPF.md

Projects using eBPF

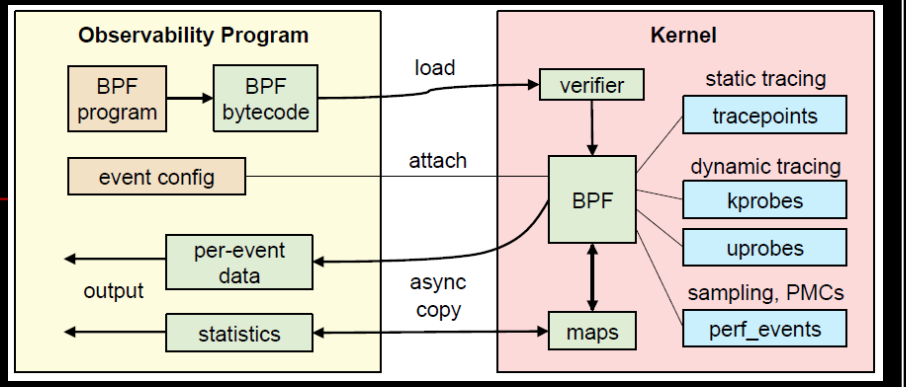
- http://cilium.readthedocs.io/en/latest/bpf/#projects-using-bpf

Good Resource

- https://github.com/zoidbergwill/awesome-ebpf

What is it

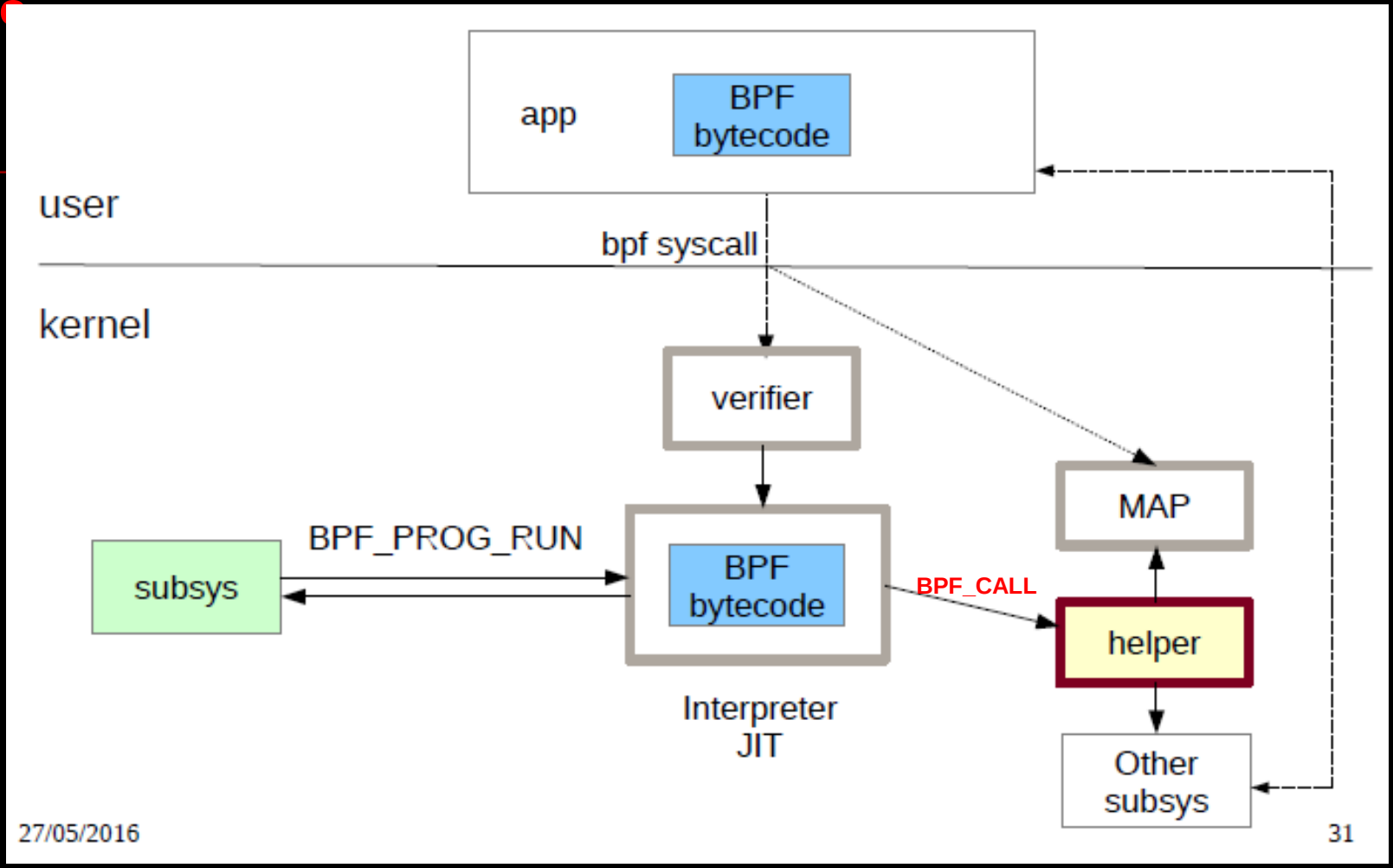
- User-defined, sandboxed bytecode executed by the kernel
- VM that implements a RISC-like assembly language in kernel space



- Similar to LSF, but with the following improvements:
  - More registers, JIT compiler (flexible/ faster), verifier
  - Attach on Tracepoint, Kprobe, Uprobe, USDT
  - In-kernel trace aggregation & filtering
  - Control via bpf()
  - Designed for general event processing within the kernel
  - All interactions between kernel/ user space are done through eBPF “maps”

Source: ebpfbasics-190611051559.pdf
Source: https://kernel-recipes.org/en/2017/talks/performance-analysis-with-bpf/

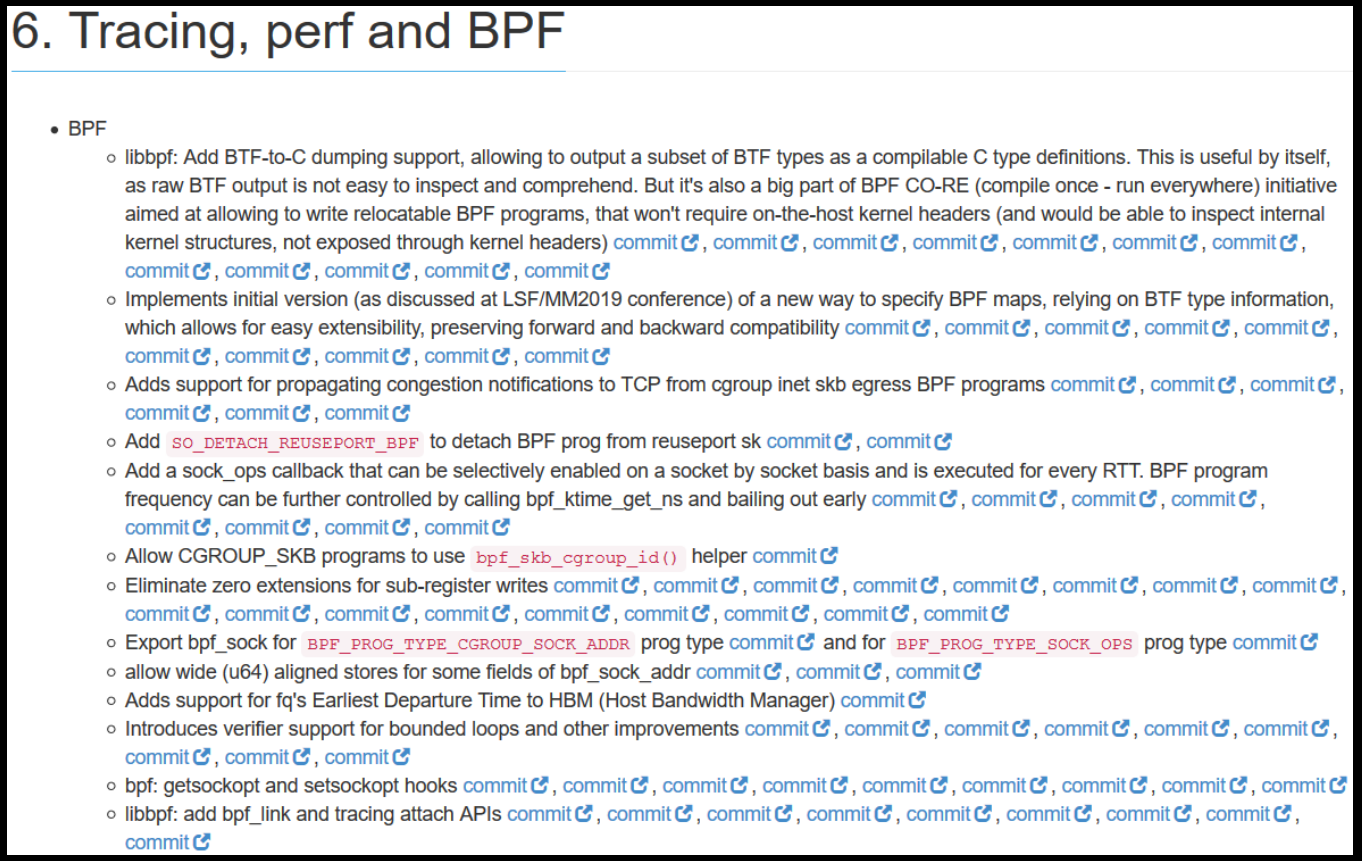
Workflow



Source: http://www.slideshare.net/vh21/meet-cutebetweenebpfandtracing

Internals

- **Pls refer to my presentation "eBPF in Action" at LC3 Beijing (on Jun 25, 2018)**
- **$KERNEL_SRC/Documentation/networking/filter.txt**
- **https://kernelnewbies.org/Linux_5.3**

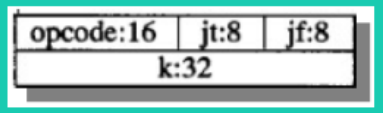
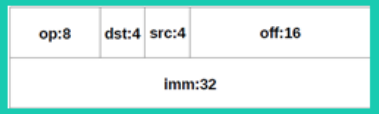


6. Tracing, perf and BPF

- BPF
  - libbpf: Add BTF-to-C dumping support, allowing to output a subset of BTF types as a compilable C type definitions. This is useful by itself, as raw BTF output is not easy to inspect and comprehend. But it's also a big part of BPF CO-RE (compile once - run everywhere) initiative aimed at allowing to write relocatable BPF programs, that won't require on-the-host kernel headers (and would be able to inspect internal kernel structures, not exposed through kernel headers) commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit
  - Implements initial version (as discussed at LSF/MM2019 conference) of a new way to specify BPF maps, relying on BTF type information, which allows for easy extensibility, preserving forward and backward compatibility commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit
  - Adds support for propagating congestion notifications to TCP from cgroup inet skb egress BPF programs commit, commit, commit, commit, commit, commit
  - Add `SO_DETACH_REUSEPORT_BPF` to detach BPF prog from reuseport sk commit, commit
  - Add a sock_ops callback that can be selectively enabled on a socket by socket basis and is executed for every RTT. BPF program frequency can be further controlled by calling bpf_ktime_get_ns and bailing out early commit, commit, commit, commit, commit, commit, commit, commit, commit
  - Allow CGROUP_SKB programs to use `bpf_skb_cgroup_id()` helper commit
  - Eliminate zero extensions for sub-register writes commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit, commit
  - Export bpf_sock for `BPF_PROG_TYPE_CGROUP_SOCK_ADDR` prog type commit and for `BPF_PROG_TYPE_SOCK_OPS` prog type commit
  - allow wide (u64) aligned stores for some fields of bpf_sock_addr commit, commit, commit
  - Adds support for fq's Earliest Departure Time to HBM (Host Bandwidth Manager) commit
  - Introduces verifier support for bounded loops and other improvements commit, commit, commit, commit, commit, commit, commit, commit, commit, commit
  - bpf: getsockopt and setsockopt hooks commit, commit, commit, commit, commit, commit, commit, commit, commit, commit
  - libbpf: add bpf_link and tracing attach APIs commit, commit, commit, commit, commit, commit, commit, commit, commit, commit

- **…**

Comparison

	cBPF	eBPF
Register	Two 32 bit registers: A: accumulator X: indexing	Eleven 64 bit registers: R0: return value/exit value R1-R5: arguments R6-R9: callee saved registers R10: read-only frame pointer
Instruction	~30 	~90 
JIT	Support	Support (better mapping with newer architectures for JITing)
Toolchain	GCC, tools/net	LLVM eBPF backend
Platform	x86_64, ARM, ARM64, SPARC, PowerPC, MIPS and s390	x86-64, aarch64, s390x...

System Call		`#include <linux/bpf.h>`
		`int bpf(int cmd, union bpf_attr *attr, unsigned int size);`
		(CALL, MAP, LOAD...)

Application	tcpdump... apply for seccomp filters, traffic control…	DDoS Mitigation, Intrusion Detection, Container Security, SDN Configuration, Observability...

- bpf() system call
  http://www.man7.org/linux/man-pages/man2/bpf.2.html

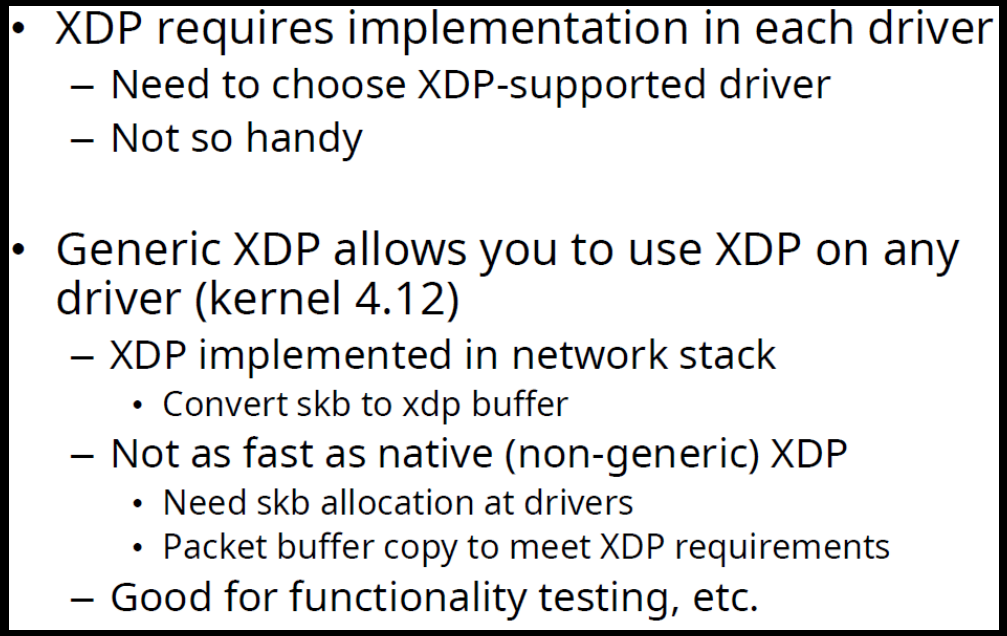
XDP (eXpress Data Path)

- https://www.iovisor.org/technology/xdp
- https://lwn.net/Articles/708087/ //Debating the value of XDP
- Generic hook

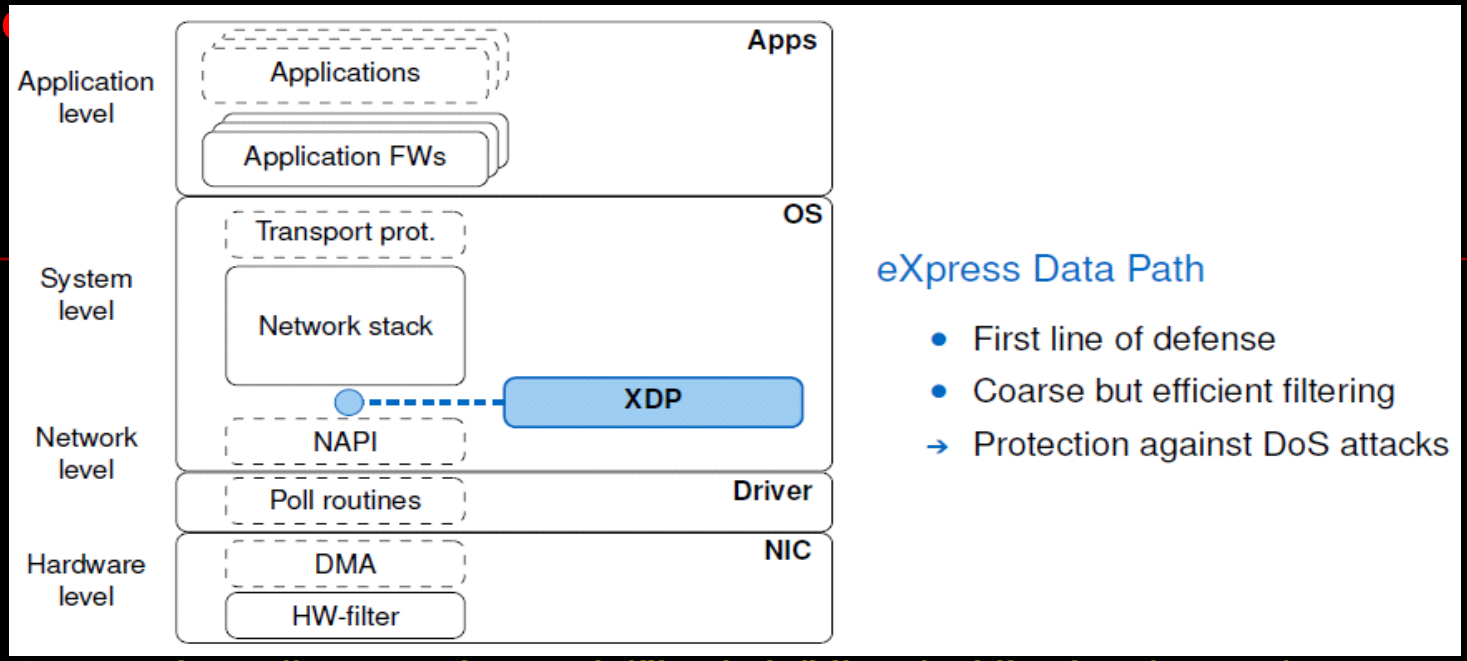
> XDP is a further step in evolution and enables to run a specific flavor of BPF programs from the network driver with direct access to the packet's DMA buffer. This is, by definition, the earliest possible point in the software stack, where programs can be attached to in order to allow for a programmable, high performance packet processor in the Linux kernel networking data path.

Source: https://github.com/cilium/cilium

- Native XDP & Generic XDP



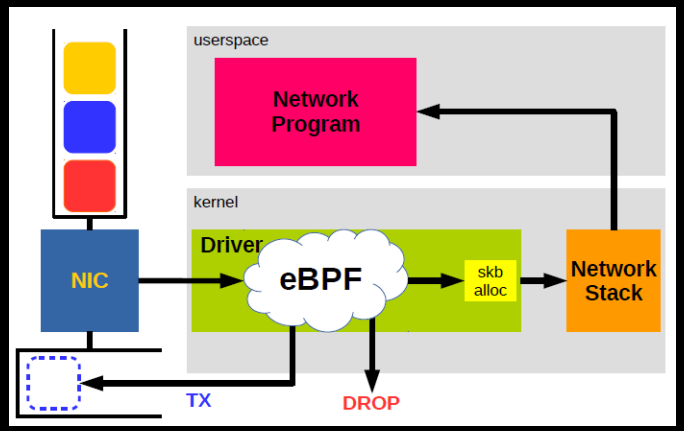
Source: "Veth XDP--XDP for containers", Toshiaki Makita & William Tu, NetDev 2019



eXpress Data Path

- First line of defense
- Coarse but efficient filtering
- → Protection against DoS attacks

Source: https://www.net.in.tum.de/fileadmin/bibtex/publications/papers/ITC30-Packet-Filtering-eBPF-XDP-slides.pdf



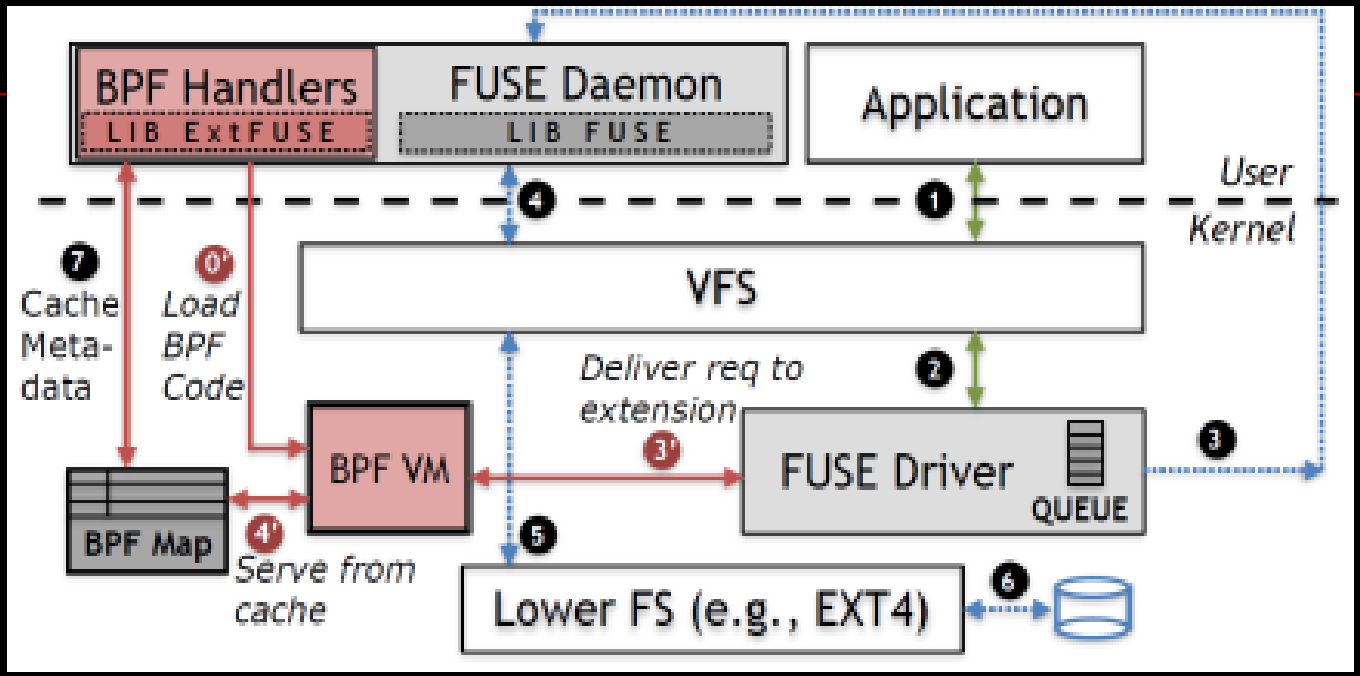
Source: https://www.slideshare.net/lcplcp1/xdp-and-ebpfmaps

- eBPF trigger actions based on return codes
  - **XDP_DROP** - very fast drop by recycling
    - DDoS mitigation
  - **XDP_PASS** – pass possibly modified packet to network stack
    - Handle and pop new unknown encap protocols
  - **XDP_TX** – Transmit packet back out same interface
    - Facebook use it for load-balancing, and DDoS scrubber
  - **XDP_ABORTED** – also drop, but indicate error condition
    - Tracepoint: xdp_exception
  - **XDP_REDIRECT** – Transmit out other NICs
    - Very new (est.4.14), (plan also use for steering packets CPUs + sockets)

Source: http://people.netfilter.org/hawk/presentations/theCamp2017/theCamp2017_XDP_eBPF_technology_Jesper_Brouer.pdf

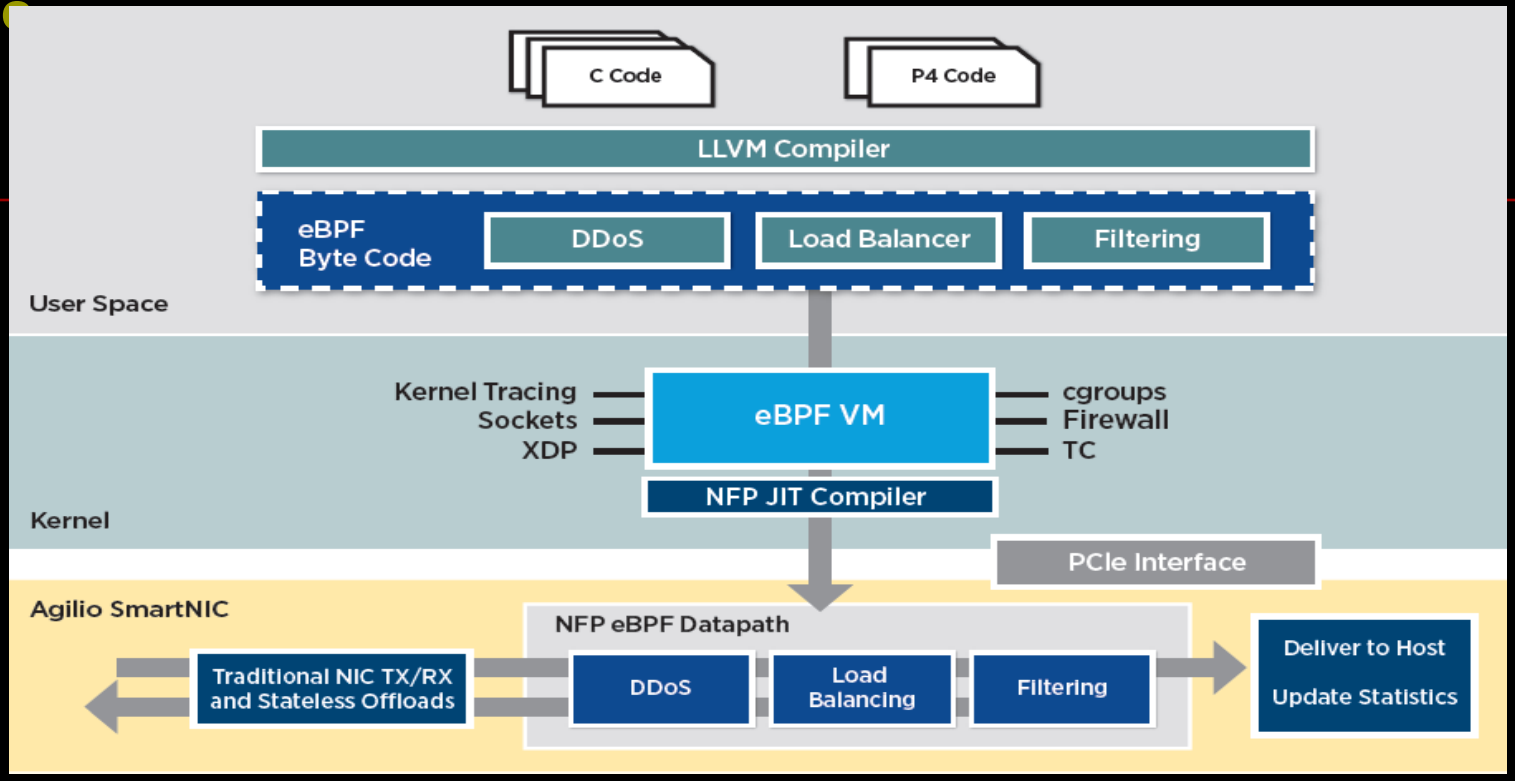
eBPF for Storage Subsystem

- eBPF can also be used for Storage, project ExtFuse (https://github.com/extfuse) shows its potential in this field.



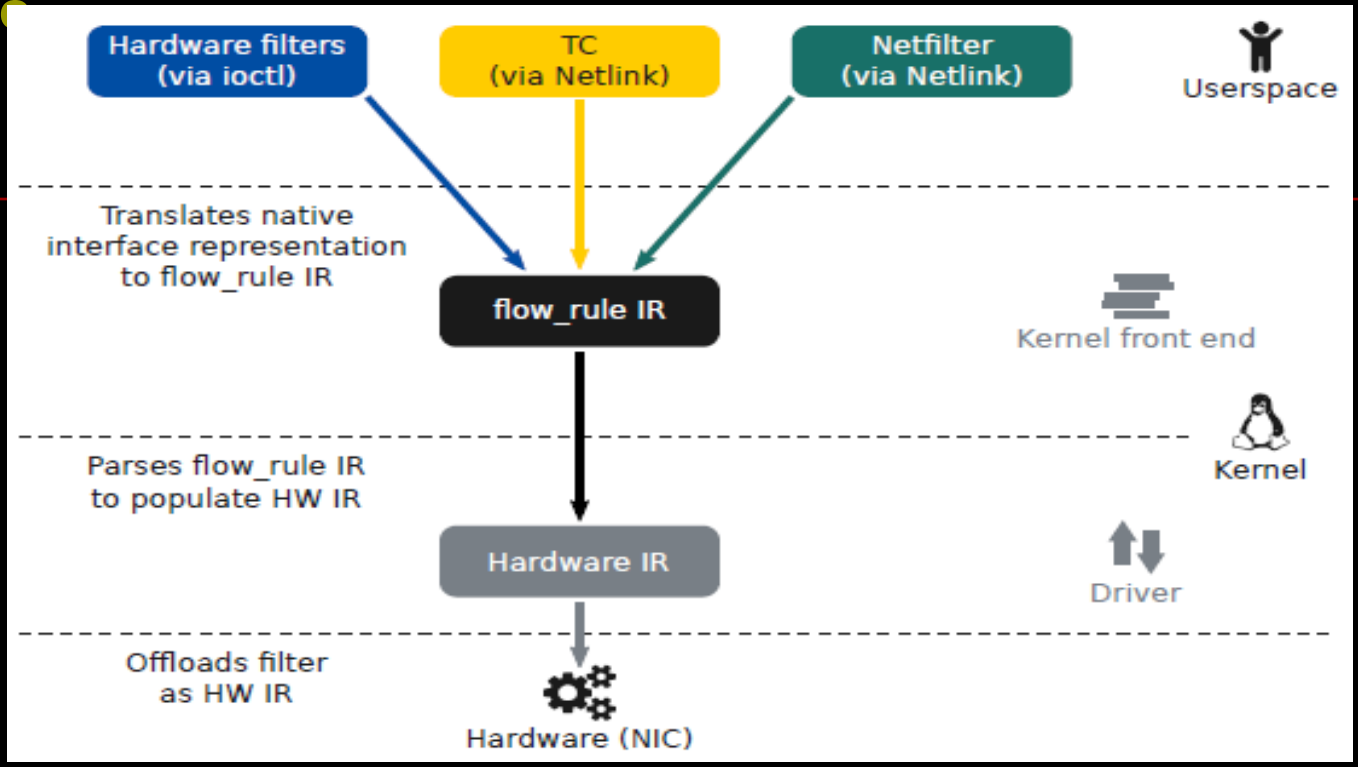
Source: https://www.phoronix.com/scan.php?page=news_item&px=ExtFUSE-Faster-FUSE-eBPF

eBPF for SmartNIC Development



Source: https://www.netronome.com/m/documents/PB_Agilio-eBPF.pdf

emerging project flow_rule



Source: https://fosdem.org/2019/schedule/event/network_filtering_with_bpf/
https://lwn.net/Articles/775046/

BPF development is "100% driven by use cases"

- **https://lwn.net/Articles/801871/**
- **BPF Contributors**

380 Daniel Borkmann (Cilium, Maintainer)
161 Alexei Starovoitov (Facebook, Maintainer)
160 Jakub Kicinski Netronome
110 John Fastabend (Cilium)
96 Yonghong Song (Facebook)
95 Martin KaFai Lau (Facebook)
94 Jesper Dangaard Brouer (Red Hat)
74 Quentin Monnet (Netronome)
45 Roman Gushchin (Facebook)
45 Andrey Ignatov (Facebook)

Top contributors of the total 186 contributors to BPF from January 2016 to November 2018.

Source: "BPF--Turning Linux into a Microservices-aware Operating System", Thomas Graf

Summary

- **Evaluation**

"eBPF is Linux's new superpower"

Gaurav Gupta

"Run code in the kernel without having to write a kernel module"

Liz Rice

"eBPF does to Linux what JavaScript does to HTML"

Brendan Gregg

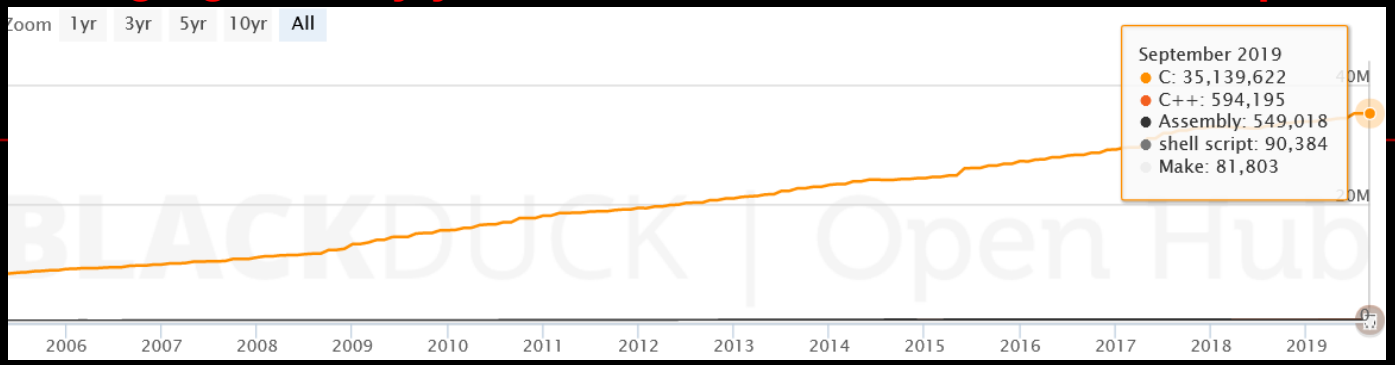
Source: ebpfbasics-190611051559.pdf

- **Kernel Space & User Space Instrumentation**

- **Dynamically extend Kernel funcationalities at runtime**

- **Polyglot VM**

  Changing the way you think about Linux Kernel development:



Language	Code Lines	Comment Lines	Comment Ratio	Blank Lines	Total Lines	Total Percentage
C	35,139,622	6,241,390	15.1%	6,333,153	47,714,165	95.7%
C++	594,195	262,892	30.7%	116,246	973,333	2.0%
Assembly	549,018	124,794	18.5%	96,032	769,844	1.5%

Source: https://www.openhub.net/p/linux/analyses/latest/languages_summary

- **The Next Linux Superpower:**

  User Space/Kernel Space Repartition & Unifying

  Reconstructing nearly every aspect of Linux Networking and Security subsystem

4) eBPF Development

4.1 Toolchain

LLVM

- eBPF backend firstly introduced in LLVM 3.7 release
- http://llvm.org/docs/CodeGenerator.html#the-extended-berkeley-packet-filter-ebpf-backend
- (
  - Enabled by default with all major distributions
    - Registered targets: `llc --version`
    - `llc`'s BPF `-march` options: bpf, bpfeb, bpfel
    - `llc`'s BPF `-mcpu` options: generic, v1, v2, probe
  - Assembler output through `-S` supported
  - `llvm-objdump` for disassembler and code annotations (via DWARF)
  - Annotations correlate directly with kernel verifier log
  - Outputs ELF file with maps as relocation entries
    - Processed by BPF loaders (e.g. iproute2) and pushed into kernel

  Source: https://ossna2017.sched.com/event/BCsg/making-the-kernels-networking-data-path-programmable-with-bpf-and-xdp-daniel-borkmann-covalent
- $LLVM_SRC/lib/Target/BPF
- http://cilium.readthedocs.io/en/latest/bpf/
- LLVM 9.0 Released With Ability To Build The Linux x86_64

GCC (GCC support for eBPF is on the way)

- https://www.phoronix.com/scan.php?page=news_item&px=GNU-Binutils-eBPF-Support
  //GNU Binutils Begins Landing eBPF Support
- https://www.phoronix.com/scan.php?page=news_item&px=GCC-10-eBPF-Backend-Plans
  //Oracle Is Aiming To Contribute An eBPF Backend To The GCC 10 Compiler
- https://www.phoronix.com/scan.php?page=news_item&px=Oracle-More-DTrace-Linux-eBPF
  //Oracle Is Working To Upstream More Of DTrace To The Linux Kernel & eBPF Implementation
- https://www.phoronix.com/scan.php?page=news_item&px=Oracle-GCC-10-eBPF-V2
  //2019-8-17::Oracle Continues Working On eBPF Support For GCC 10
- https://www.phoronix.com/scan.php?page=news_item&px=GCC-10-eBPF-Port-Lands
  //GCC 10 Lands The eBPF Port For Targeting The Linux In-Kernel VM

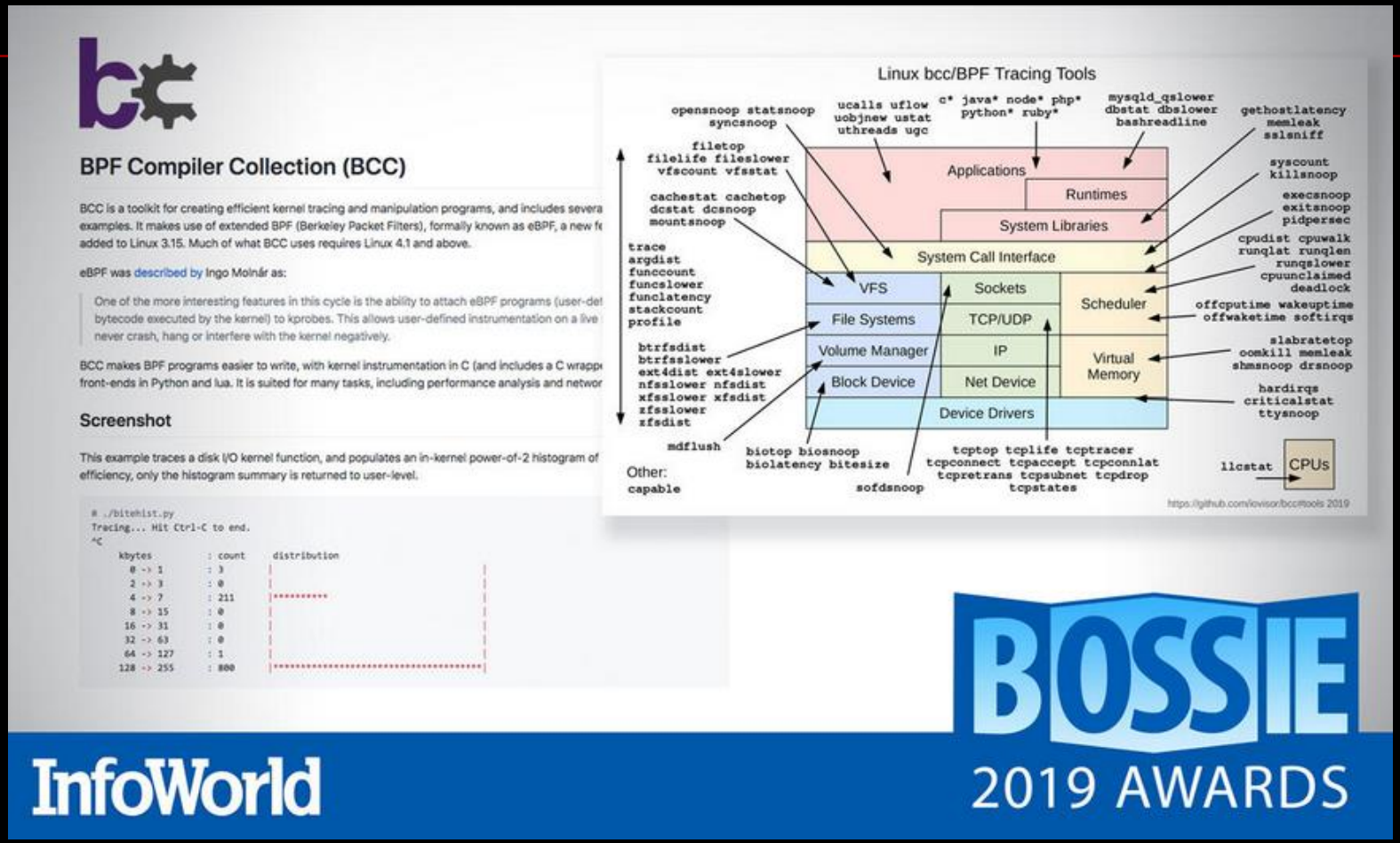
Status

- **Phase 1:** add eBPF target to the toolchain
  - `bpf-unknown-none`
  - binutils (upstream since May 2019)
  - GCC (upstream since September 2019)
- **Phase 2:** make the generated programs palatable for the kernel loaders and verifier, and **keep it that way**.
- **Phase 3:** provide development goodies for eBPF developers
  - GNU simulator
  - GDB
  - ...

Source: "eBPF support in the GNU Toolchain", Jose E. Marchesi (Oracle), Linux Plumbers Conference 2019

4.2 BCC (BPF Compiler Collection)

- https://www.infoworld.com/article/3444198/the-best-open-source-software-of-2019.html



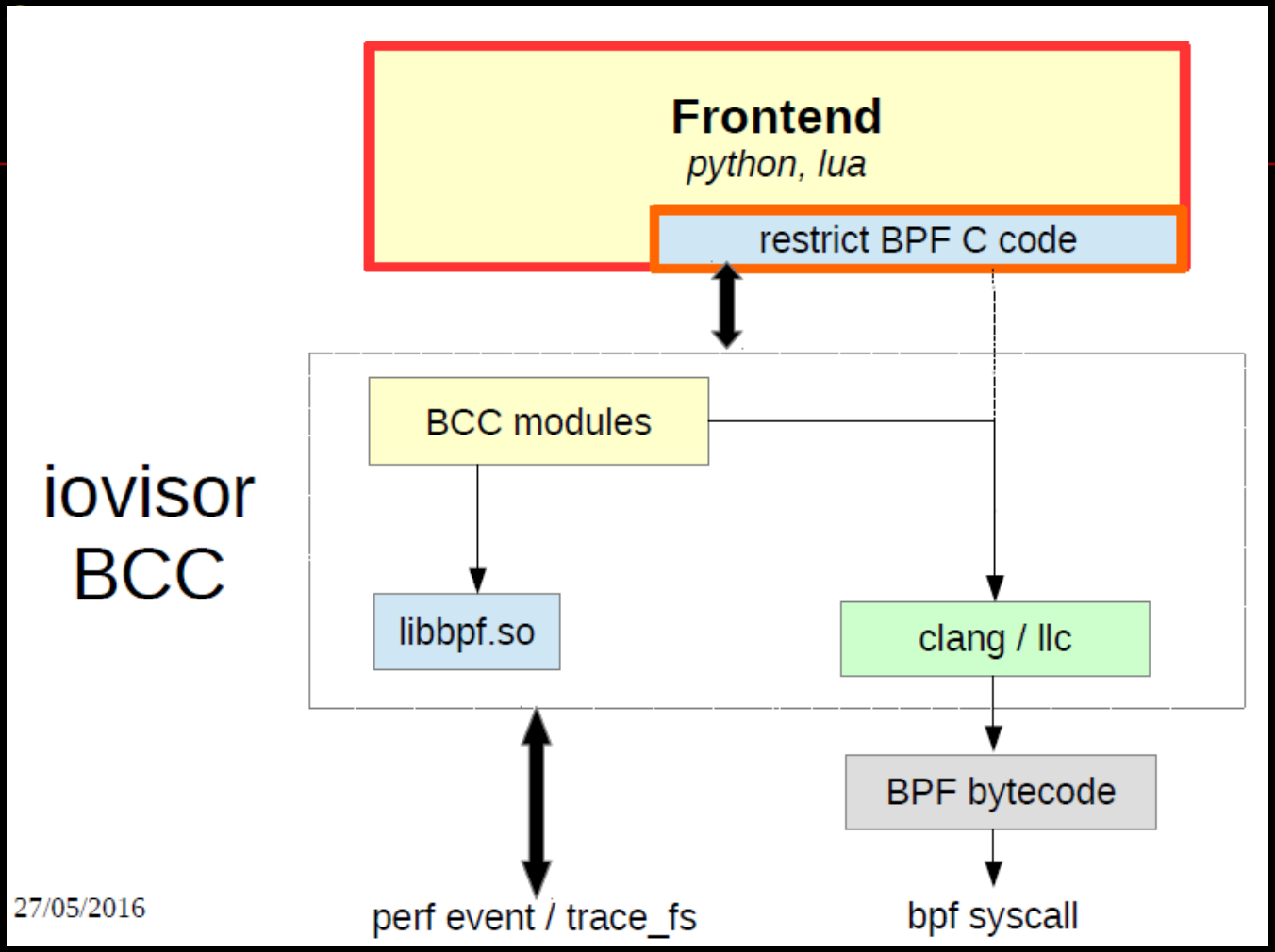
What is it

- **https://github.com/iovisor/bcc/**

a toolkit with Python/Lua frontend for compiling, loading, and executing BPF programs, which allows user-defined instrumentation on a live kernel image:

- **compile BPF program from C source**
- **attach BPF program to kprobe/uprobe/tracepoint/USDT/socket**
- **poll data from BPF program**
- **framework for building new tools or one-off scripts**
- **additional projects to support Go, Rust, and DTrace-style frontend**
- **…**

Architecture

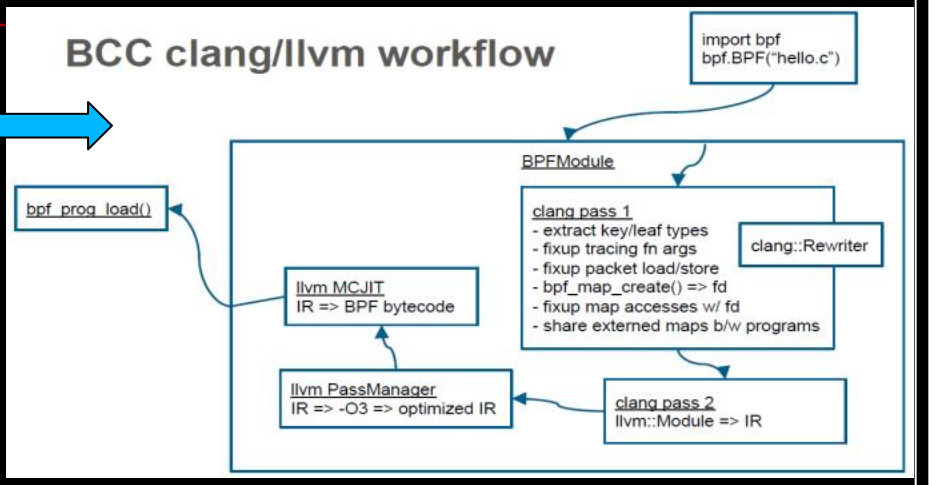


Source: http://www.slideshare.net/vh21/meet-cutebetweenebpfandtracing

A Sample

- https://lwn.net/Articles/747640/ //Some advanced BCC topics

```
#!/usr/bin/env python

from bcc import BPF
from time import sleep

program = """
    BPF_HASH(callers, u64, unsigned long);

    TRACEPOINT_PROBE(kmem, kmalloc) {
        u64 ip = args->call_site;
        unsigned long *count;
        unsigned long c = 1;

        count = callers.lookup((u64 *)&ip);
        if (count != 0)
            c += *count;

        callers.update(&ip, &c);

        return 0;
    }
"""
b = BPF(text=program)

while True:
    try:
        sleep(1)
        for k,v in sorted(b["callers"].items()):
            print ("%s %u" % (b.ksym(k.value), v.value))
        print
    except KeyboardInterrupt:
        exit()
```

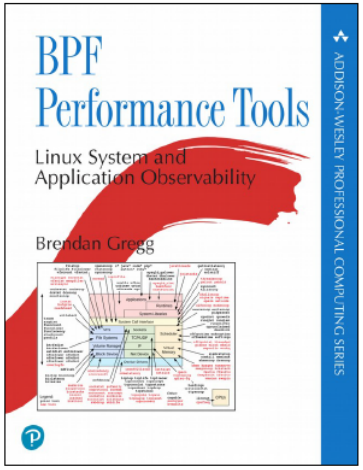


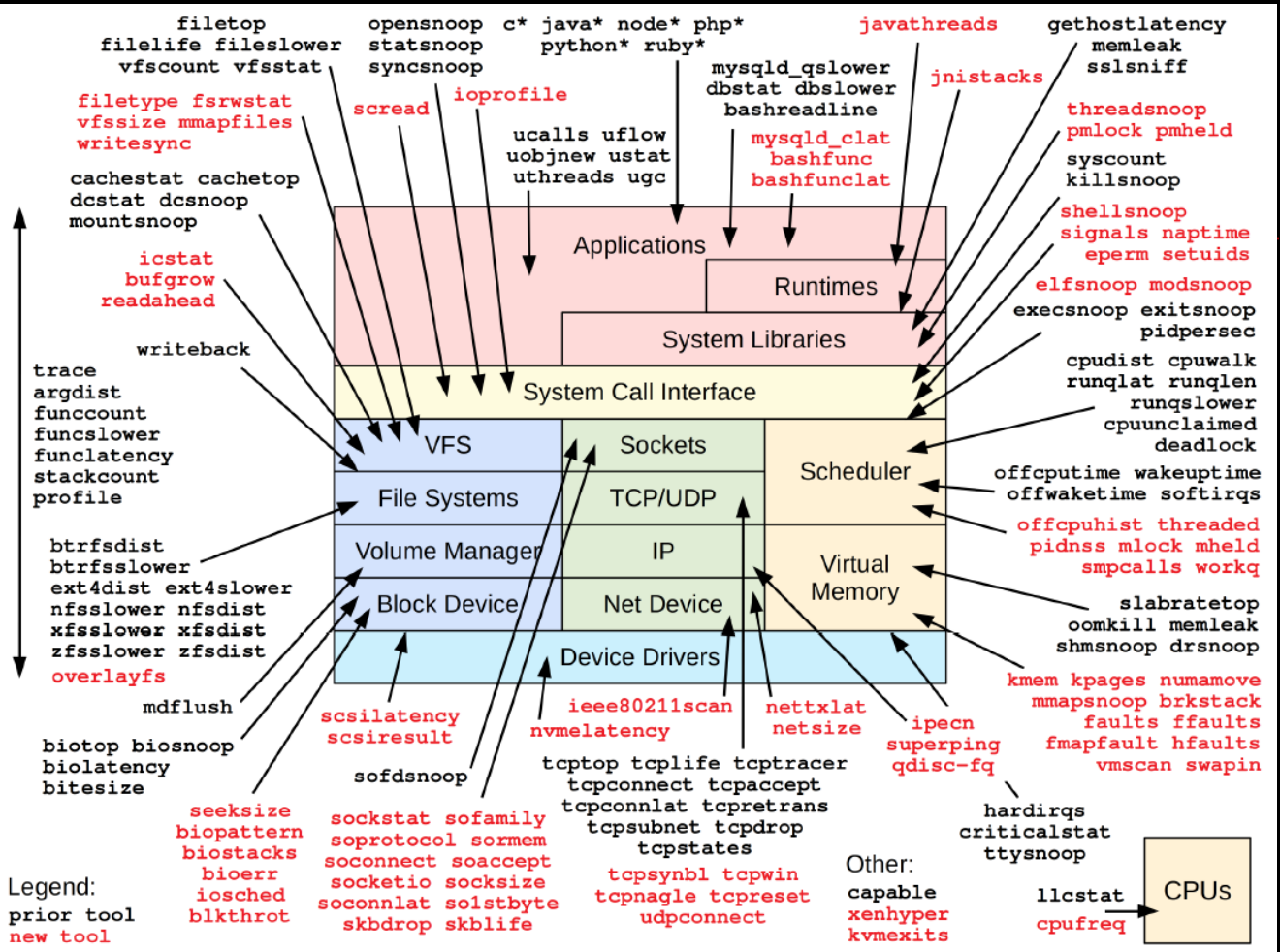
Source: http://linuxplumbersconf.org/2015/ocw//system/presentations/3249/original/bpf_llvm_2015aug19.pdf

The output from this little program looks like:

```
# ./example.py
i915_sw_fence_await_dma_fence 4
intel_crtc_duplicate_state 4
SyS_memfd_create 1
drm_atomic_state_init 4
sg_kmalloc 7
intel_atomic_state_alloc 4
seq_open 504
SyS_bpf 22
```

BPF Perf Tools





Source: "BPF Tracing Tools", Brendan Gregg, Linux Plumbers Conference 2019

development guide

- https://github.com/iovisor/bcc/blob/master/docs/tutorial.md
- https://github.com/iovisor/bcc/blob/master/docs/reference_guide.md
- https://github.com/iovisor/bcc/blob/master/docs/tutorial_bcc_python_developer.md
- …

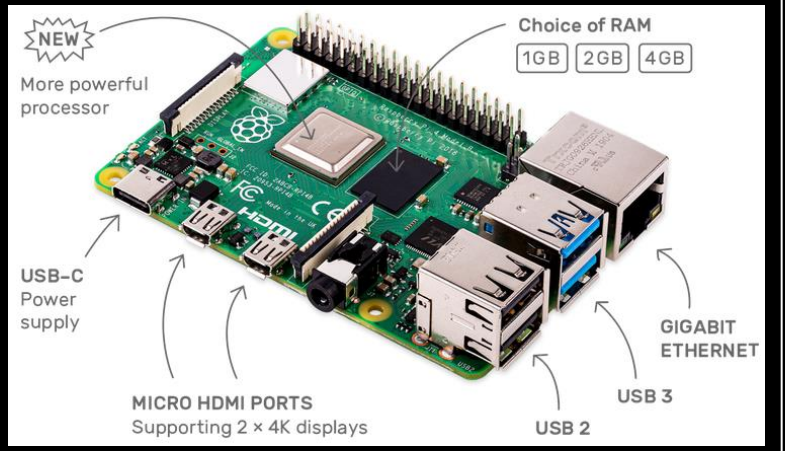
III. Testbed

1) Development Boards

1.1 Raspberry Pi 4

- https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
- https://www.cnx-software.com/2019/06/24/raspberry-pi-4-vs-pi-3-what-are-the-differences/

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	85×56 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)



Pros

- high cost–performance ratio (initial price $55 for RPi4 with 4GB RAM)
- Most active community & ecosystem

…

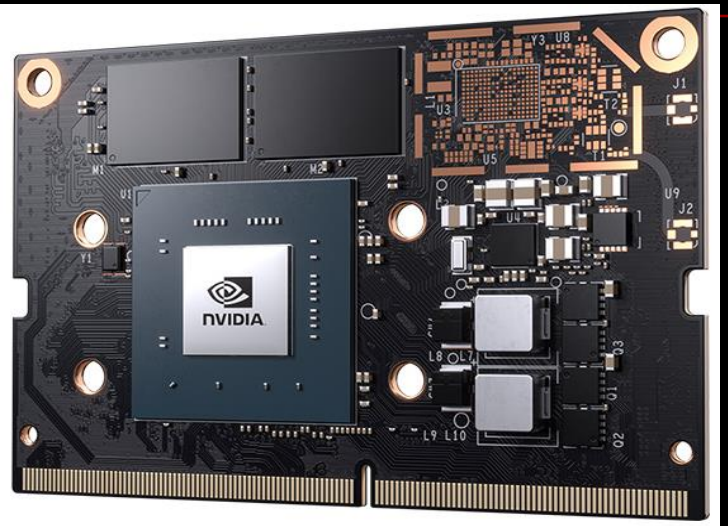
Cons

- More 64bit Linux distributions support is on the way
- Lack of detailed technical docs for CPU, GPU, etc from Broadcom
- …

1.2 Jetson Nano

- https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/
- Bringing the Power of Modern AI to Millions of Devices

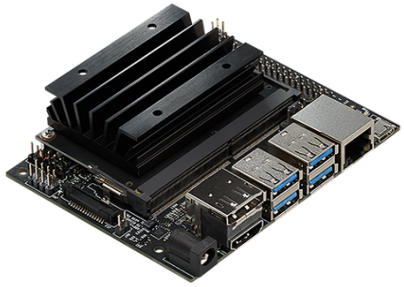
GPU	NVIDIA Maxwell™ architecture with 128 NVIDIA CUDA® cores
CPU	Quad-core ARM® Cortex®-A57 MPCore processor
Memory	4 GB 64-bit LPDDR4
Storage	16 GB eMMC 5.1 Flash
Video Encode	4K @ 30 (H.264/H.265)
Video Decode	4K @ 60 (H.264/H.265)
Camera	12 lanes (3x4 or 4x2) MIPI CSI-2 DPHY 1.1 (1.5 Gbps)
Connectivity	Gigabit Ethernet
Display	HDMI 2.0 or DP1.2 | eDP 1.4 | DSI (1 x2) 2 simultaneous
UPHY	1 x1/2/4 PCIE, 1x USB 3.0, 3x USB 2.0
I/O	1x SDIO / 2x SPI / 6x I2C / 2x I2S / GPIOs
Size	69.6 mm x 45 mm
Mechanical	260-pin edge connector



- https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit

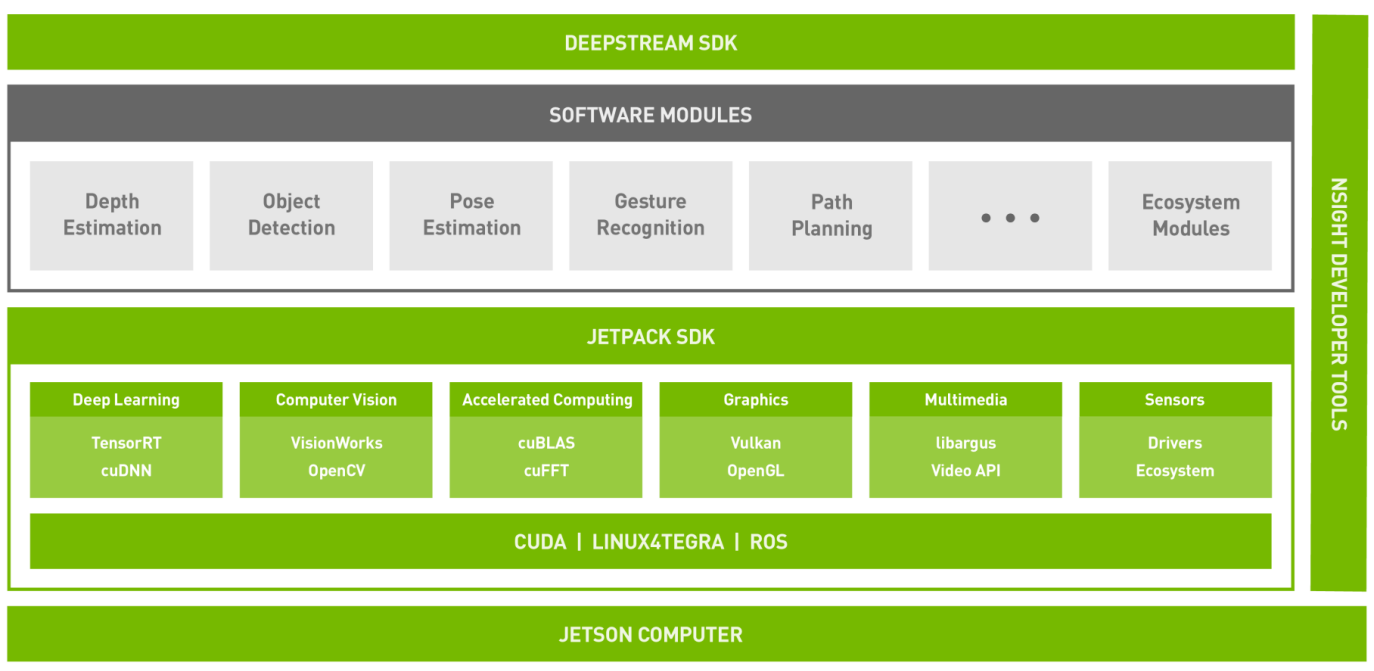
Get started today with the Jetson Nano Developer Kit that brings the power of modern artificial intelligence to a small, easy-to-use platform. Get started fast with the Jetpack SDK and a collection of ready-to-use projects.

DEVELOPER KIT I/OS	
USB	4x USB 3.0, USB 2.0 Micro-B
Camera Connector	1x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Storage	microSD (not included)
Display	HDMI 2.0 and eDP 1.4
Others	GPIO, I²C, I²S, SPI, UART



Pros

- high cost–performance ratio (initial price $99 for devkit)
  https://tryolabs.com/blog/machine-learning-on-edge-devices-benchmark-report/
- Complete Software Stack



- TensorRT is Open Sourced now

Cons

- The kernel version is still 4.9
- Not hacker friendly

1.3 VIM3 Pro

- https://www.khadas.com/vim3

Amlogic A311D SBC with 5.0 TOPS NPU

Model	Basic	Pro
SoC	Amlogic A311D 2.2GHz Quad core ARM Cortex-A73 and 1.8GHz dual core Cortex-A53 CPU ARM G52 MP4 GPU up to 800MHz HW UHD 4K H.265 75fps 10-bit video decoder & low latency 1080p H.265/H.264 60fps encoder Support multi-video decoder up to 4Kx2K@60fps+1x1080P@60fps Dolby Vision and HDR10, HDR10+, HLG and PRIME HDR video processing Build-in Cortex-M4 core for always on processing TrustZone based security for DRM video streaming	
NPU	5 TOPS Performance NPU INT8 inference up to 1536 MAC Supports all major deep learning frameworks including TensorFlow and Caffe	
MCU [1]	STM8S003 with Programmable EEPROM	
SPI Flash	16MB	
LPDDR4/4X [2]	2GB	4GB
EMMC 5.1	16GB	32GB

Pros

- Cost–performance ratio (initial price $139.99)
- Active community & ecosystem

  ROM support:

  Android

  Android TV

  Armbian

  Manjaro ARM

  Ubuntu XFCE

  LibreELEC

  CoreELEC

  OpenWRT

  …

Cons

- Does not provide a complete software stack
- Lack of development documents for NPU
- Complex Modes with various buttons in a confined space
- …

1.4 PYNQ-Z2

- http://www.tul.com.tw/ProductsPYNQ-Z2.html

ZYNQ XC7Z020-1CLG400C
- 650MHz dual-core Cortex-A9 processor
- DDR3 memory controller with 8 DMA channels and 4 High Performance AXI3 Slave ports
- High-bandwidth peripheral controllers: 1G Ethernet, USB 2.0, SDIO
- Low-bandwidth peripheral controller: SPI, UART, CAN, I2C
- Programmable from JTAG, Quad-SPI flash, and MicroSD card
- Programmable logic equivalent to Artix-7 FPGA
  - 13,300 logic slices, each with four 6-input LUTs and 8 flip-flops
  - 630 KB of fast block RAM
  - 4 clock management tiles, each with a phase locked loop (PLL) and mixed-mode clock manager (MMCM)
  - 220 DSP slices
  - On-chip analog-to-digital converter (XADC)

Memory
- 512MB DDR3 with 16-bit bus @ 1050Mbps
- 16MB Quad-SPI Flash with factory programmed 48-bit globally unique EUI-48/64™ compatible identifier
- MicroSD slot

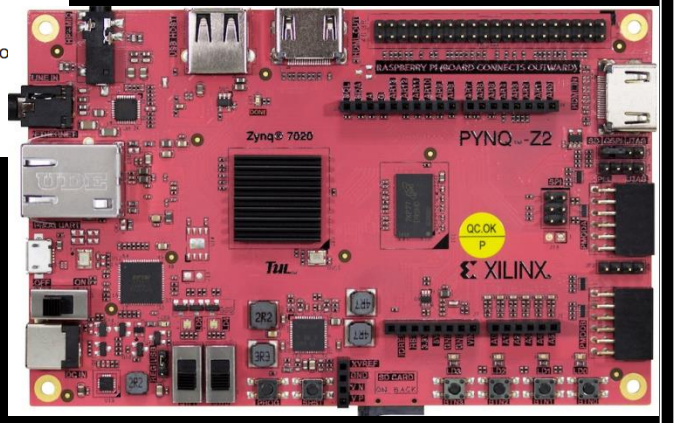
Power
- Powered from USB or 7V-15V external power source

USB and Ethernet
- Gigabit Ethernet PHY
- Micro USB-JTAG Programming circuitry
- Micro USB-UART bridge
- USB 2.0 OTG PHY (supports host only)

Audio and Video
- HDMI sink port (input)
- HDMI source port (output)
- I2S interface with 24bit DAC with 3.5mm TRRS jack
- Line-in with 3.5mm jack

Switches, Push-buttons and LEDs
- 4 push-buttons
- 2 slide switches
- 4 LEDs
- 2 RGB LEDs

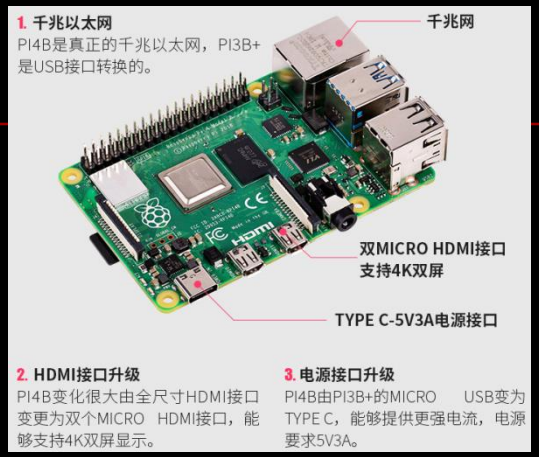
Expansion Connectors
- Two standard Pmod ports
  - 16 Total FPGA I/O (8 shared pins with Raspberry Pi connector)
- Arduino Shield connector
  - 24 Total FPGA I/O
  - 6 Single-ended 0-3.3V Analog inputs to XADC
- Raspberry Pi connector
  - 28 Total FPGA I/O (8 shared pins with Pmo A port)

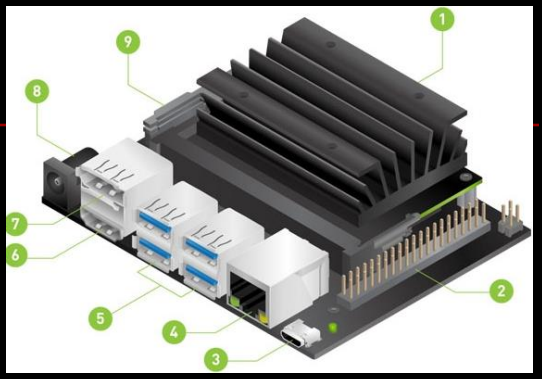


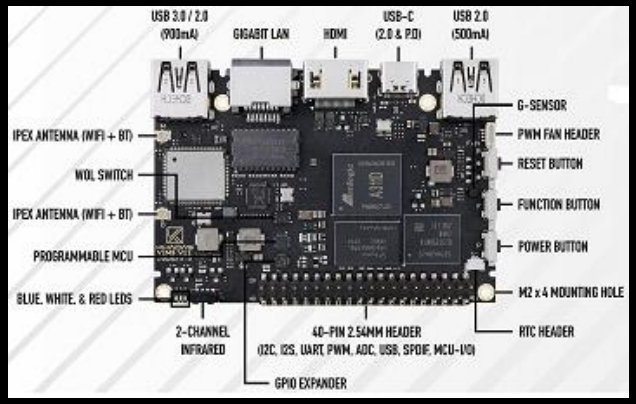
- Just as a learning platform for PYNQ

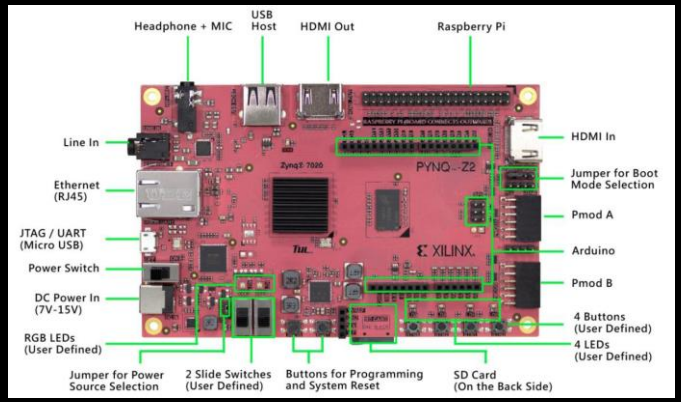
1.5 A development boards cluster

Current (as a small PoC platform)









Next Year

- **imagining a new development cluster in next yerar with:**
  more powerful Cortex-A76/A77 or even Neoverse development board with >= 8GB LPDDR5 (may plus UFS 3.0 memory card)
  more powerful GPU development board
  more powerful AI development board
  more powerful FPGA development board
  …

2) Software Platform

2.1 Manjaro

- https://manjaro.org
- an open-source Linux distribution based on the Arch Linux OS

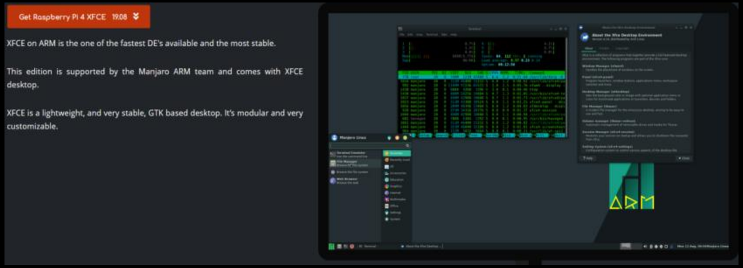
Rank	Distribution	H.P.D*
1	Mint	2880
2	Debian	1668
3	Ubuntu	1347
4	openSUSE	1234
5	elementary	1085
6	Manjaro	1039
7	Fedora	982
8	Zorin	914
9	CentOS	773
10	deepin	736

2016-->2019

https://www.distrowatch.com

Rank	Distribution	HPD*
1	MX Linux	4895
2	Manjaro	2597
3	Mint	2048
4	Debian	1543
5	Ubuntu	1398
6	elementary	1302
7	Solus	1087
8	Fedora	992
9	deepin	847
10	Zorin	830

- one of the first ARM64 Linux distribution that support RPi 4



- Advantages (from my point of view)
  - update packages quickly
  - group install

    https://www.archlinux.org/groups/
  - Python 3 is the default Python version now

    https://hg.python.org/peps/rev/76d43e52d978
  - much more stable than expected

  …

- https://www.howtogeek.com/430556/why-i-switched-from-ubuntu-to-manjaro-linux/
- https://wiki.archlinux.org/index.php/Pacman
- Preparation

upgrade original Kernel on Manjaro RPi4 from v4.19.x to v5.3.x with all the necessary options (like that for eBPF, Virtualization, Debugging, and so on) enabled

```
#
# eBPF
#
CONFIG_CGROUP_BPF=y
CONFIG_BPF=y
CONFIG_BPF_SYSCALL=y
CONFIG_BPF_JIT_ALWAYS_ON=y
CONFIG_IPV6_SEG6_BPF=y
CONFIG_NETFILTER_XT_MATCH_BPF=m
# CONFIG_BPFILTER is not set
CONFIG_NET_CLS_BPF=m
CONFIG_NET_ACT_BPF=m
CONFIG_BPF_JIT=y
CONFIG_BPF_STREAM_PARSER=y
CONFIG_LWTUNNEL_BPF=y
CONFIG_HAVE_EBPF_JIT=y
CONFIG_BPF_LIRC_MODE2=y
CONFIG_BPF_EVENTS=y
# CONFIG_TEST_BPF is not set
# XDP
CONFIG_XDP_SOCKETS=y
CONFIG_XDP_SOCKETS_DIAG=m
```

```
#
# Virtualization
#
CONFIG_VIRTUALIZATION=y
CONFIG_PARAVIRT=y
CONFIG_PARAVIRT_TIME_ACCOUNTING=y
CONFIG_VIRTIO=y
CONFIG_HAVE_VIRT_CPU_ACCOUNTING_GEN=y
CONFIG_BLK_MQ_VIRTIO=y
CONFIG_VIRTIO_VSOCKETS=m
CONFIG_VIRTIO_VSOCKETS_COMMON=m
CONFIG_NET_9P_VIRTIO=m
CONFIG_VIRTIO_BLK=m
CONFIG_SCSI_VIRTIO=m
CONFIG_VIRTIO_NET=m
CONFIG_VIRT_WIFI=m
CONFIG_VIRTIO_CONSOLE=m
CONFIG_HW_RANDOM_VIRTIO=m
CONFIG_DRM_VIRTIO_GPU=m
CONFIG_SND_VIRTUOSO=m
CONFIG_VIRTIO_MENU=y
CONFIG_VIRTIO_PCI=y
CONFIG_VIRTIO_PCI_LEGACY=y
CONFIG_VIRTIO_BALLOON=m
CONFIG_VIRTIO_INPUT=m
CONFIG_VIRTIO_MMIO=m
CONFIG_RPMSG_VIRTIO=m
CONFIG_CRYPTO_DEV_VIRTIO=m
CONFIG_DMA_VIRT_OPS=y
CONFIG_REGULATOR_VIRTUAL_CONSUMER=m
CONFIG_FB_VIRTUAL=m
CONFIG_DMA_VIRTUAL_CHANNELS=y
#CONFIG_VIRTIO_PMEM=m
#CONFIG_VIRTIO_IOMMU=m
```

```
#
# KVM
#
CONFIG_KVM=y
CONFIG_HAVE_KVM_IRQCHIP=y
CONFIG_HAVE_KVM_IRQFD=y
CONFIG_HAVE_KVM_IRQ_ROUTING=y
CONFIG_HAVE_KVM_EVENTFD=y
CONFIG_KVM_MMIO=y
CONFIG_HAVE_KVM_MSI=y
CONFIG_HAVE_KVM_CPU_RELAX_INTERCEPT=y
CONFIG_KVM_VFIO=y
CONFIG_HAVE_KVM_ARCH_TLB_FLUSH_ALL=y
CONFIG_KVM_GENERIC_DIRTYLOG_READ_PROTECT=y
CONFIG_HAVE_KVM_IRQ_BYPASS=y
CONFIG_HAVE_KVM_VCPU_RUN_PID_CHANGE=y
CONFIG_KVM_ARM_HOST=y
CONFIG_KVM_ARM_PMU=y
CONFIG_KVM_INDIRECT_VECTORS=y
```

```
#
# Network
#
CONFIG_IPV6=y
CONFIG_IP_VS_IPV6=y
CONFIG_IPV6_SEG6_LWTUNNEL=y
CONFIG_IPV6_SEG6_HMAC=y
CONFIG_IPV6_OPTIMISTIC_DAD=y
CONFIG_IPV6_MIP6=y
CONFIG_IPV6_GRE=m
CONFIG_IPV6_ILA=m
CONFIG_IPV6_VTI=m
#CONFIG_IPV6_FOU=m
#CONFIG_IPV6_FOU_TUNNEL=m
#CONFIG_AF_RXRPC_IPV6=y


#
# Miscs
#
CONFIG_KSM=y
CONFIG_PROC_EVENTS=y
#CONFIG_IKHEADERS=y
CONFIG_REMOTEPROC=m
CONFIG_REISERFS_PROC_INFO=y
CONFIG_PROC_CHILDREN=y
```

```
#
# Debug
#
CONFIG_ARCH_HAS_DEBUG_VIRTUAL=y
CONFIG_STRIP_ASM_SYMS=y
CONFIG_UNUSED_SYMBOLS=y
CONFIG_PM_DEBUG=y
CONFIG_CMA_DEBUGFS=y
CONFIG_L2TP_DEBUGFS=m
CONFIG_6LOWPAN_DEBUGFS=y
CONFIG_CFG80211_DEBUGFS=y
CONFIG_MAC80211_DEBUGFS=y
CONFIG_DEBUG_DEVRES=y
CONFIG_SCSI_DEBUG=m
CONFIG_DM_DEBUG=y
CONFIG_DM_DEBUG_BLOCK_MANAGER_LOCKING=y
CONFIG_ATH9K_DEBUGFS=y
CONFIG_ATH6KL_DEBUG=y
CONFIG_SUNRPC_DEBUG=y
CONFIG_DLM_DEBUG=y
CONFIG_DYNAMIC_DEBUG=y
CONFIG_DEBUG_INFO=y
#CONFIG_DEBUG_INFO_DWARF4=y
CONFIG_DEBUG_SECTION_MISMATCH=y
CONFIG_DEBUG_RODATA_TEST=y
CONFIG_DEBUG_VM=y
CONFIG_DEBUG_SHIRQ=y
CONFIG_DEBUG_LIST=y
CONFIG_ARM64_PTDUMP_DEBUGFS=y
CONFIG_DEBUG_WX=y
CONFIG_PROC_KCORE=y
CONFIG_PROC_VMCORE=y
CONFIG_PROC_VMCORE_DEVICE_DUMP=y
CONFIG_KEXEC=y
# CONFIG_KEXEC_FILE is not set
CONFIG_KEXEC_CORE=y
CONFIG_CRASH_DUMP=y
CONFIG_CRASH_CORE=y
CONFIG_CRASH=m
CONFIG_FTRACE_SYSCALLS=y
CONFIG_HWLAT_TRACER=y
```

- Pls refer to my presentation "Python for Linux Kernel Debugging" at PyCon China Hangzhou (Oct 19, 2019) for details

2.2 Ubuntu

- https://jamesachambers.com/raspberry-pi-ubuntu-server-18-04-2-installation-guide/
- https://ubuntu.com/blog/roadmap-for-official-support-for-the-raspberry-pi-4 //Nov 3, 2019

Raspberry Pi 4	Ubuntu Server 19.10	Ubuntu Server 18.04	Ubuntu Core
1GB			
2GB			
4GB	*Upcoming kernel fix*		

Current Next Future

2.3 Fedora

- official support for Raspberry Pi 4 may coming in Fedora 32☹
- developer friendly
- https://iot.fedoraproject.org

3) Development Model

- Fully In-Device development, no cross compilation

IV. Computing, Networking, Storage

1) HPC (Heterogeneous Parallel Computing)

1.1 CUDA

- https://developer.nvidia.com/cuda-zone
- most are closed source
- https://nvidianews.nvidia.com/news/nvidia-brings-cuda-to-arm-enabling-new-path-to-exascale-supercomputing

1.2 ROCm

- https://rocm.github.io/
- most are open sourced
- not support ARM yet☹

1.3 OpenCL

- https://www.khronos.org/opencl/
- Implementation is depend on vendor
- https://www.khronos.org/assets/uploads/developers/library/2019-iwocl/IWOCL%20Keynote%20May19.pdf

1.4 CAPI

- https://developer.ibm.com/linuxonpower/capi/

Coherent Accelerator Processor Interface (CAPI)

Features	Benefits
Customizable accelerator	Application developers can create an accelerator specifically tailored to their application using an FPGA platform. This specialization can improve performance.
Virtual addressing	An accelerator acts as a peer to the POWER8 cores, creating a truly shared memory space between the application and FPGA accelerator. The accelerator essentially becomes another thread of the application. Address translation is managed by the POWER8 processor.
Coherency	Application developers use a hardware managed 256KB cache in the FPGA for improved latency and synchronization with the main application threads. The FPGA cache is fully coherent with the caches in the POWER8 cores.
Simple API	Application developers can easily start and control the accelerator. The API features an intuitive programming interface similar to any multi-threaded application.
Flexible programming model	Application developers determine the best programming model for their application. The application can either dispatch work for the accelerator or the accelerator can act independently. This flexibility enables devices such as edge-of-network accelerators to process incoming work independently and to notify the main application when data is ready.
Extendable architecture	Application developers can choose an OpenPOWER Foundation partner device, such as Nallatech's CAPI Developer Kit. Extendable to other FPGA platforms, the architecture uses card features such as Ethernet and DRAM. In addition, the architecture supports prepackaged CAPI solutions such as the PDF IBM Data Engine for NoSQL.

1.5 Intel oneAPI

- https://newsroom.intel.com/news/intels-one-api-project-delivers-unified-programming-model-across-diverse-architectures/#gs.dx5h9x

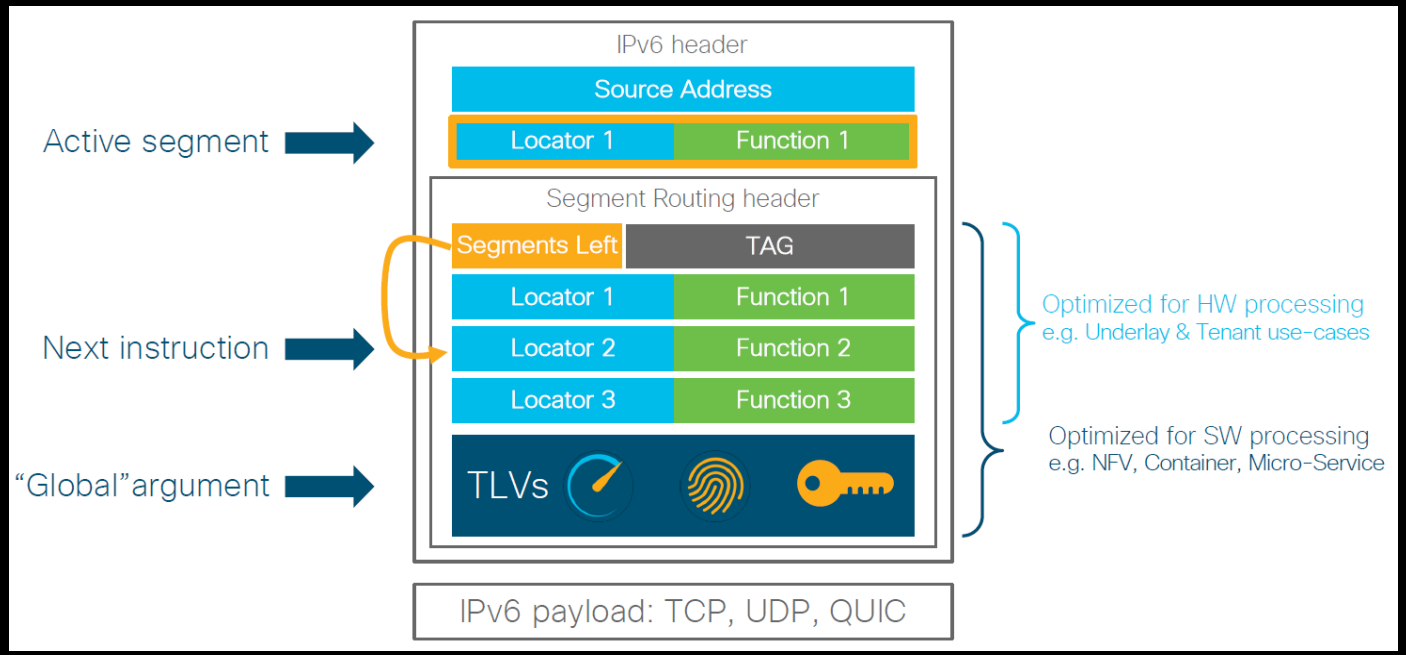
How It Works: One API supports direct programming and API programming, and will deliver a unified language and libraries that offer full native code performance across a range of hardware, including CPUs, GPUs, FPGAs and AI accelerators.

- **Direct programming:** One API contains a new direct programming language, Data Parallel C++ (DPC++), an open, cross-industry alternative to single architecture proprietary languages. DPC++ delivers parallel programming productivity and performance using a programming model familiar to developers. DPC++ is based on C++, incorporates SYCL* from The Khronos Group and includes language extensions developed in an open community process.
- **API-based programming:** One API's powerful libraries span several workload domains that benefit from acceleration. Library functions are custom-coded for each target architecture.
- **Analysis and debug tools:** Building on leading analysis tools, Intel will deliver enhanced versions of analysis and debug tools to support DPC++ and the range of SVMS architectures.

What Developers Should Expect: Intel will release a developer beta and additional details on the One API project in 2019's fourth quarter.

2) Networking
2.1 SRv6

- Segment Routing over IPv6 dataplane
- http://www.segment-routing.net/tutorials/2017-12-05-srv6-introduction/
- it uses a source-routing concept that the topological and service (NFV) path is encoded in the packet header. Today, 5G is the main driver for fast-paced development of SRv6 standards to simplify the user plane and network plane.
-



Source: http://www.segment-routing.net/images/20190130-bcn-cl-BRKRST-3122-rev7f-km2.pdf

Linux Implementation

- http://www.segment-routing.net/open-software/linux/

Linux kernel 4.10

The IPv6 dataplane functionalities enabling Segment Routing packet generation and forwarding are available in the latest Linux kernel releases (4.10 and later). This implementation is provided by the IP Networking Lab of Université Catholique de Louvain, Louvain-la-Neuve, Belgium.

Linux kernel 4.14

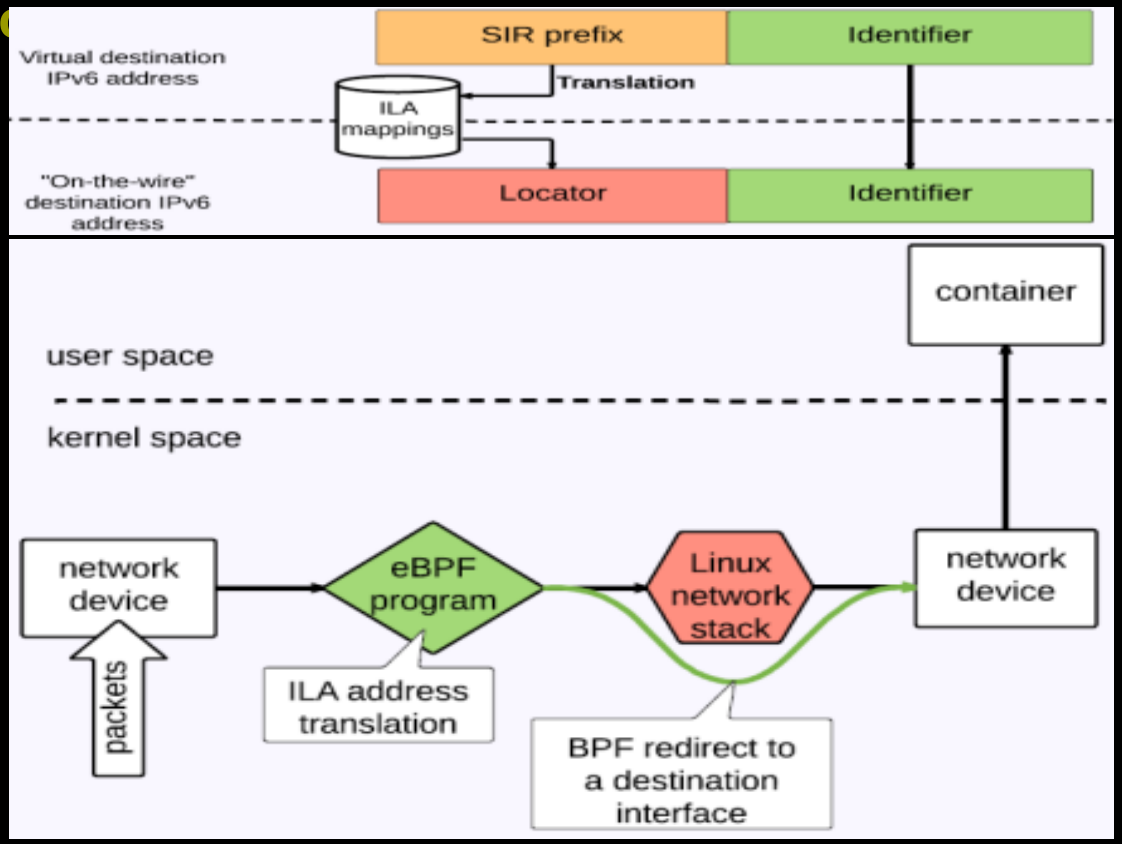
Kernel 4.14 is another milestone in SRv6 support in Linux. A new set of SRv6 behaviors has been added to the kernel (see table below).

Name	Description	Release
End	Endpoint function	**4.10** (February 2017), srext
End.X	Endpoint function with Layer-3 cross-connect	**4.10** (February 2017), srext
End.T	Endpoint function with specific IPv6 table lookup	**4.14** (November 2017)
End.DX2	Endpoint with decapsulation and Layer-2 cross-connect	**4.14** (November 2017), srext
End.DX6	Endpoint with decapsulation and IPv6 cross-connect	**4.14** (November 2017), srext
End.DX4	Endpoint with decapsulation and IPv4 cross-connect	**4.14** (November 2017), srext
End.DT6	Endpoint with decapsulation and IPv6 table lookup	**4.14** (November 2017)
End.DT4	Endpoint with decapsulation and IPv4 table lookup	In development
End.B6	Endpoint bound to an SRv6 policy	**4.14** (November 2017)
End.B6.Encaps	Endpoint bound to an SRv6 encapsulation Policy	**4.14** (November 2017)
End.BM	Endpoint bound to an SR-MPLS Policy	In development
End.S	Endpoint in search of a target in table T	In development
End.AD	Endpoint to SR-unaware APP via dynamic proxy	srext
End.AM	Endpoint to SR-unaware APP via masquerading	srext

...

2.2 ILA (Identifier-Locator Addressing)

- https://lwn.net/Articles/657012/
- https://datatracker.ietf.org/doc/draft-herbert-intarea-ila/
- $KERNEL_SRC/Documentation/networking/ila.txt
-



Source: https://www.delaat.net/sc/sc17/posters/ila-poster-SC17.pdf

2.3 ILNP (Identifier-Locator Network Protocol)

- https://ilnp.cs.st-andrews.ac.uk/

This project is enhancing the Internet Architecture by enriching the set of namespaces. The basic approach to this is to deprecate the concept of an *Address* and replace it with separate *Locator* and *Identifier* values. Although the architectural concept is independent of any particular network protocol, our research demonstration will be based on IPv6. Prototypes for FreeBSD and Linux are planned.

- https://ilnp.github.io/ilnp-public-1/
- Internet RFC documents

RFC6740 Identifier-Locator Network Protocol (ILNP) Architectural Description (Nov 2012)
RFC6741 Identifier-Locator Network Protocol (ILNP) Engineering Considerations (Nov 2012)
RFC6742 DNS Resource Records for the Identifier-Locator Network Protocol (ILNP) (Nov 2012)
RFC6743 ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv6 (ILNPv6) (Nov 2012)
RFC6744 IPv6 Nonce Destination Option for the Identifier-Locator Network Protocol for IPv6 (ILNPv6) (Nov 2012)
RFC6745 ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv4 (ILNPv4) (Nov 2012)
RFC6746 IPv4 Options for the Identifier-Locator Network Protocol (ILNP) (Nov 2012)
RFC6747 Address Resolution Protocol (ARP) for the Identifier-Locator Network Protocol for IPv4 (ILNPv4) (Nov 2012)
RFC6748 Optional Advanced Deployment Scenarios for the Identifier-Locator Network Protocol (ILNP) (Nov 2012)

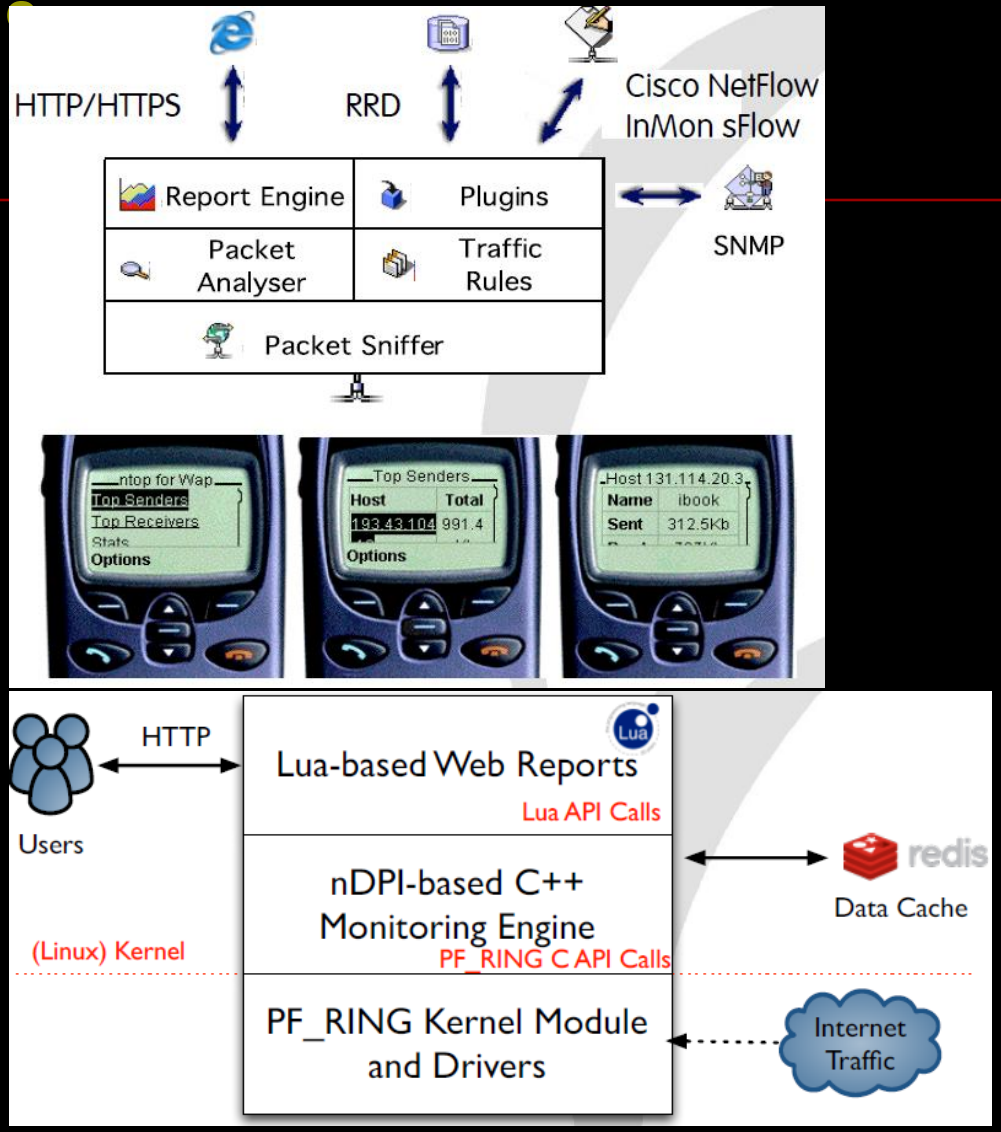
3) ntopng

- https://www.ntop.org/
- the next generation version of the original ntop, a high-speed web-based network traffic analysis and monitoring toolkit
- Features

…

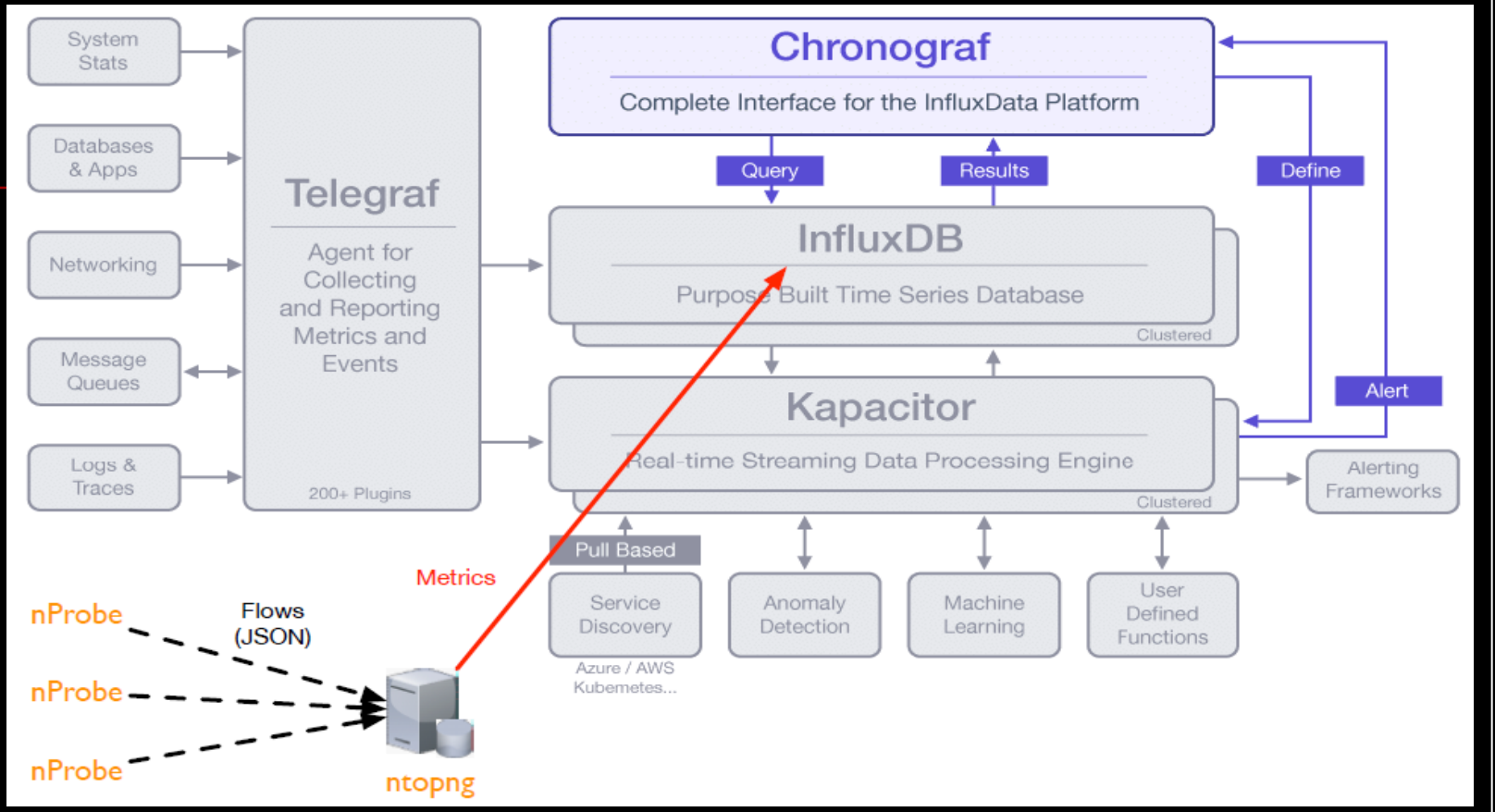
- https://www.ntop.org/support/documentation/documentation/

Architecture



Source: "Network Troubleshooting Using ntopng", Luca Deri, SharkFest 2015

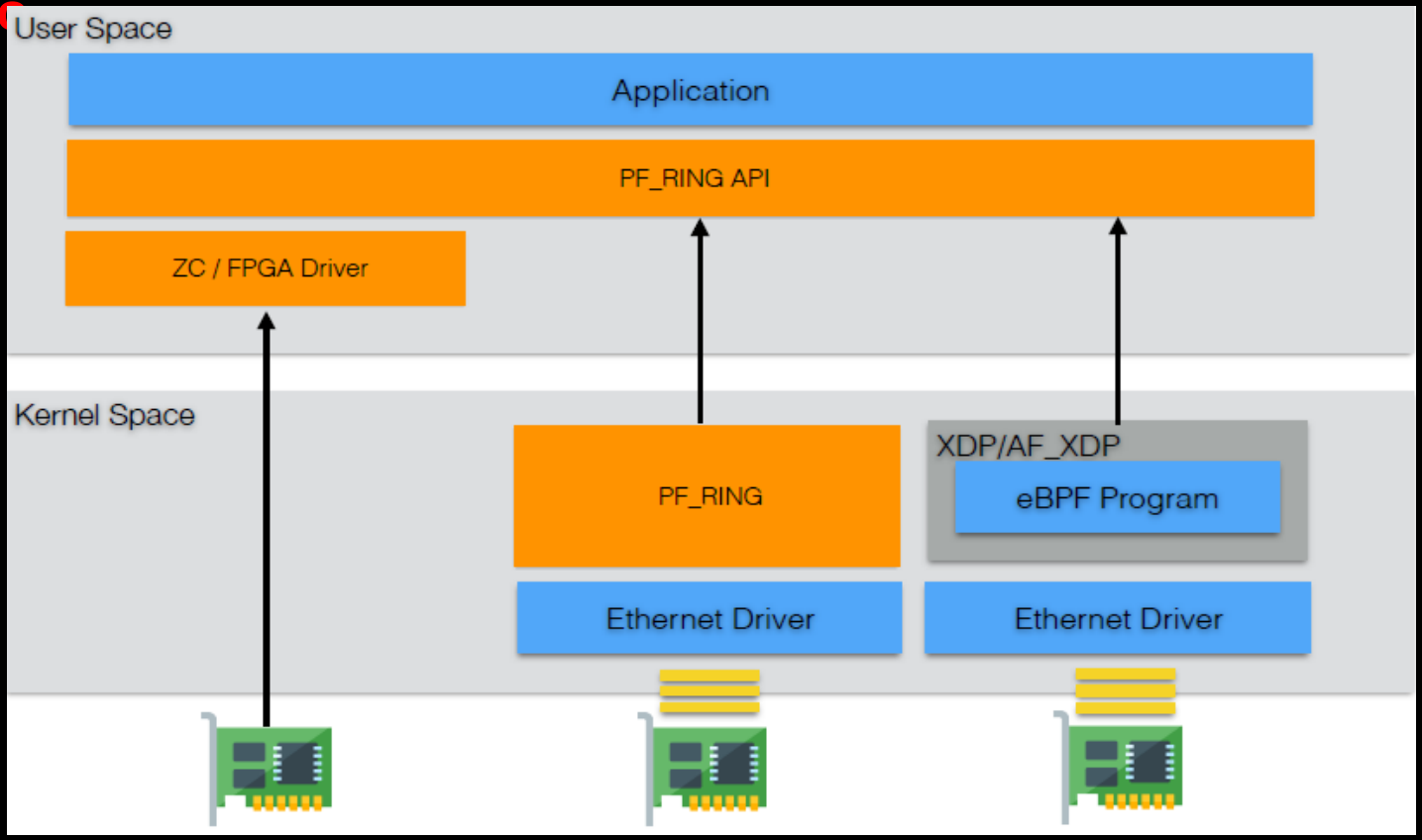
Data Collection Architecture



Source: https://www.ntop.org/wp-content/uploads/2019/05/InfluxData_Webinar_2019.pdf

PF_RING

- https://www.ntop.org/products/packet-capture/pf_ring
- a Linux kernel module and user-space framework that allows you to process packets at high-rates while providing a consistent API for packet processing applications
- C

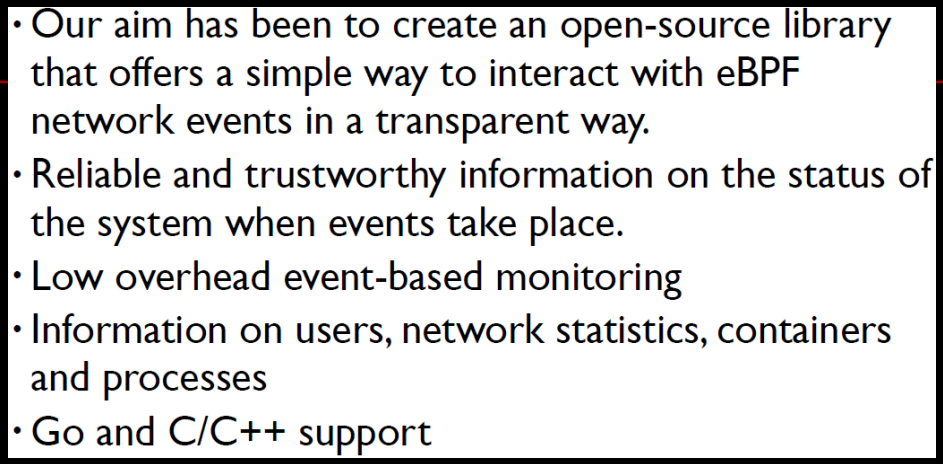


Source: "Joining Forces: PF_RING and XDP", Alfredo Cardigliano, ntopConf 2019

3.1 eBPF support

libebpfflow

- **https://github.com/ntop/libebpfflow**



Source: https://www.ntop.org/wp-content/uploads/2019/05/InfluxData_Webinar_2019.pdf

nBPF

- **https://www.ntop.org/guides/pf_ring/nbpf/nbpf.html**
  a filtering engine/SDK supporting the BPF syntax and can be used as alternative to the implementation that can be found in libpcap and inside the kernel

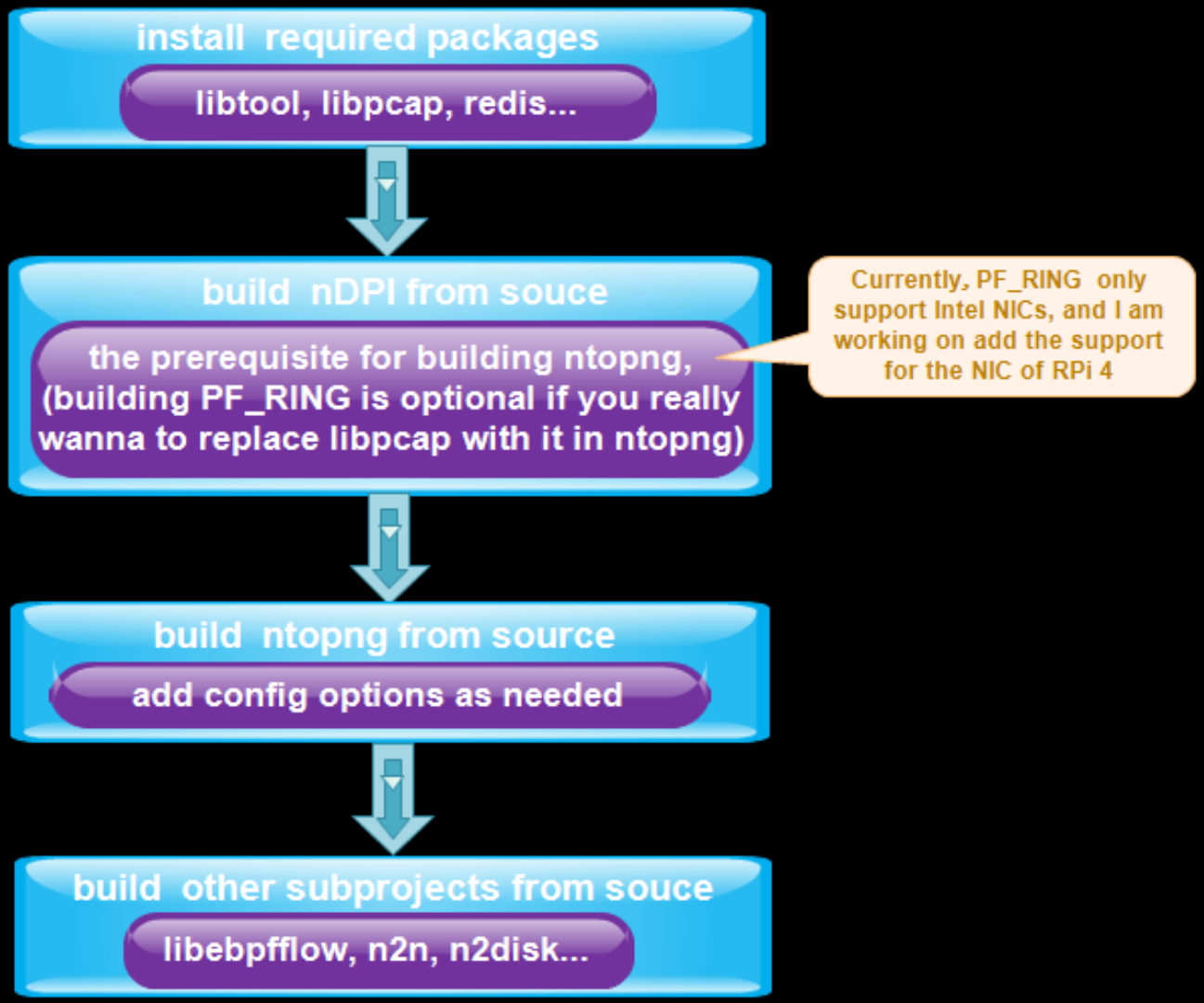
ntopng 4.x

- **ntopng 4.x will integrate system visibility with eBPF**

…

3.2 My Practice

- successfully build ntopng on RPi 4 with libpcap



4) DRedis

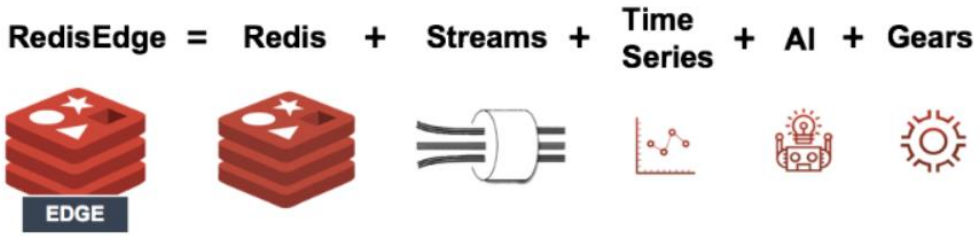
4.1 Redis

- https://redis.io/
- Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.
- Module
  https://redis.io/topics/modules-intro
  https://redis.io/topics/modules-api-ref

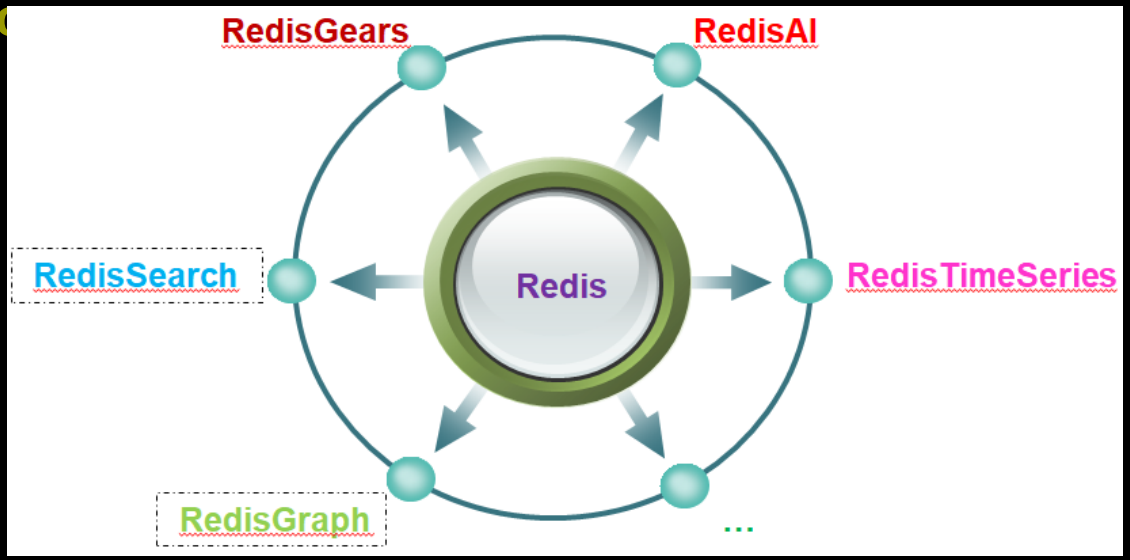
RedisEdge

- **https://redislabs.com/solutions/redisedge/**
- **The Edge Computing Database for the IoT Edge**

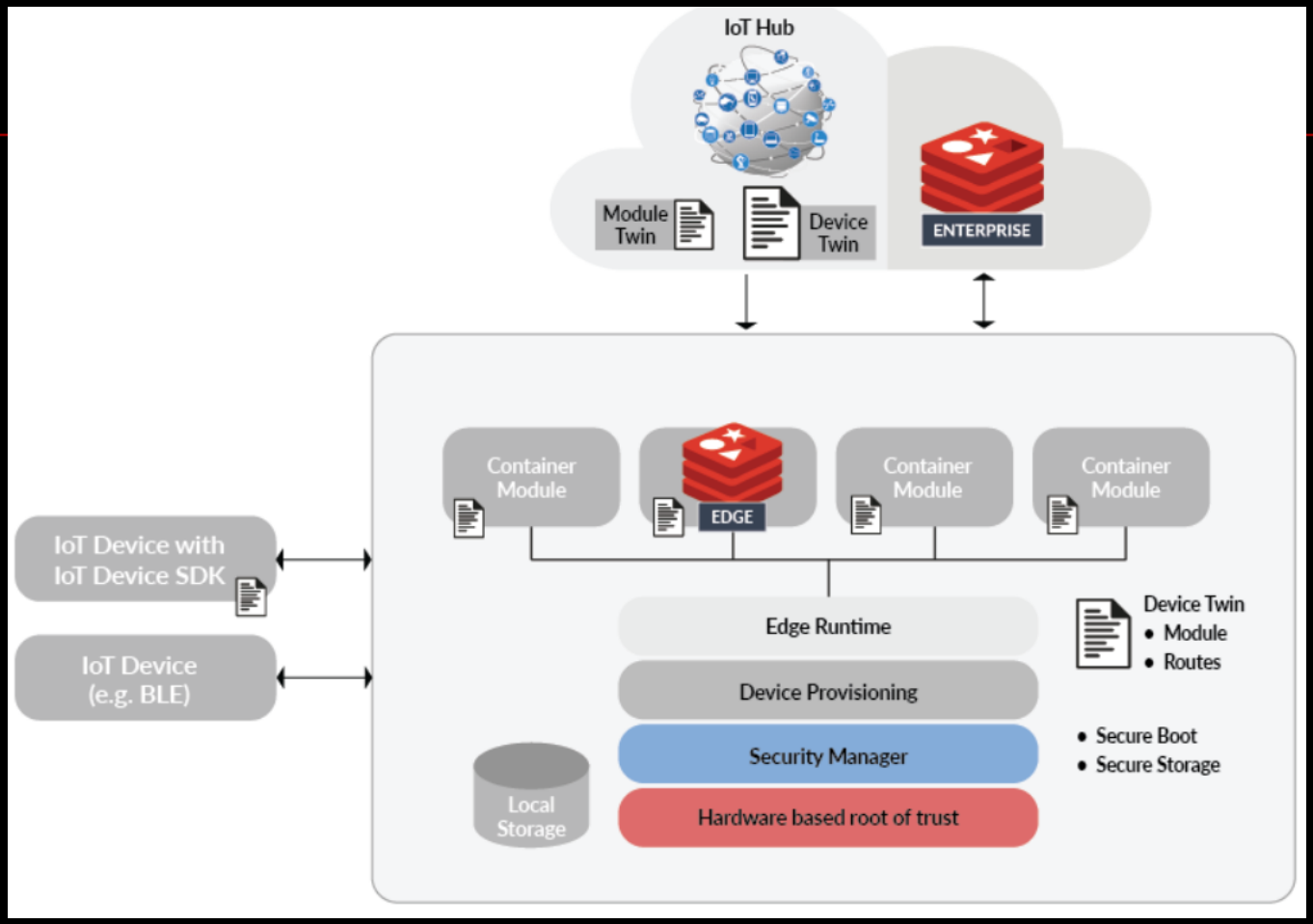
RedisEdge from Redis Labs is a purpose-built, multi-model database for the demanding conditions at the Internet of Things (IoT) edge. It can ingest millions of writes per second with <1ms latency and a very small footprint (<5MB), so it easily resides in constrained compute environments. It can run on a variety of edge devices and sensors ranging from ARM32 to x64-based hardware. RedisEdge bundles open source Redis (version 5 with Redis Streams) with the RedisAI and RedisTimeSeries modules, along with RedisGears for inter-module communication.



-



RedisEdge is also available as a module for Azure IoT Edge, making it easy for IoT application developers using Azure IoT services to leverage the power of Redis. Azure IoT Edge with RedisEdge helps businesses focus on insights instead of data management. Developers can configure and deploy their solutions via standard containers and monitor them from the cloud.



4.2 KeyDB

- https://keydb.dev/
- A Multithreaded Fork of Redis
- an ultra-fast, open source Key Value Store Database fully compatible with Redis API, modules, and protocols
-

	KeyDB	Redis
Key-Value Store Database	✓	✓
In Memory Database	✓	✓
Multithreaded IO	✓	✗
Advanced Multithreading	✓	✗
Flash Support	✓	$ $ $
Multi-Master Support	✓	$ $ $
Active-Active Support	✓	$ $ $
RAM/SSD Persistence	✓	✓
Database Not Limited by Size	✓	✗
Simple to Use at Scale	✓	✗
High Availability	✓	✓
Cloud Optimized	✓	✓
ARM Support for IOT	✓	✓

4.3 Tarantool

- https://tarantool.io/en/
-

Key features of the application server:

- 100% compatible drop-in replacement for Lua 5.1, based on LuaJIT 2.1. Simply use `#!/usr/bin/tarantool` instead of `#!/usr/bin/lua` in your script.
- full support for Lua modules and a rich set of own modules, including cooperative multitasking, non-blocking I/O, access to external databases, etc

Key features of the database:

- ANSI SQL, including views, joins, referential and check constraints
- MsgPack data format and MsgPack based client-server protocol
- two data engines: 100% in-memory with optional persistence and an own implementation of LSM-tree, to use with large data sets
- multiple index types: HASH, TREE, RTREE, BITSET
- asynchronous master-master replication
- authentication and access control
- the database is just a C extension to the application server and can be turned off

Supported platforms are Linux/x86 and FreeBSD/x86, Mac OS X.

Tarantool is ideal for data-enriched components of scalable Web architecture: queue servers, caches, stateful Web applications.

- incorporates the LuaJIT "Just In Time" compiler

4.4 DRedis Design Goals & Principles

- a re-implementation of Redis by combining the advantage of both KeyDB and Tarantool
- compatible with Redis 6 and RESP3
- all the existing Redis modules should work well
- Better solution for Redis persistence
- make fully use of LuaJIT
- …

5) Rethink Lightweight Storage Solutions

Ceph

- https://ceph.io/
- a unified, distributed storage system designed for excellent performance, reliability and scalability
- used as the default storage solution in OpenStack, and is popular in production environment
- but it is not a lightweight solution

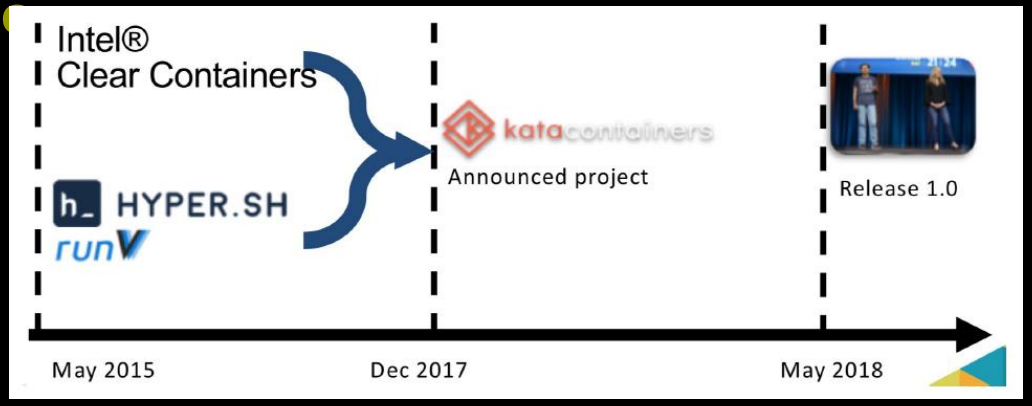
Other Interesting Projects

Kudu

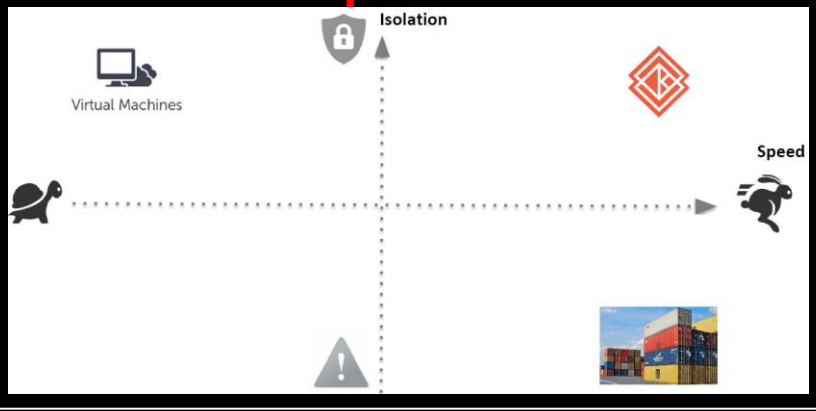
- https://kudu.apache.org/
- but with many limitations…

V. Containerization, Services, & DevOps

1) Kata Container

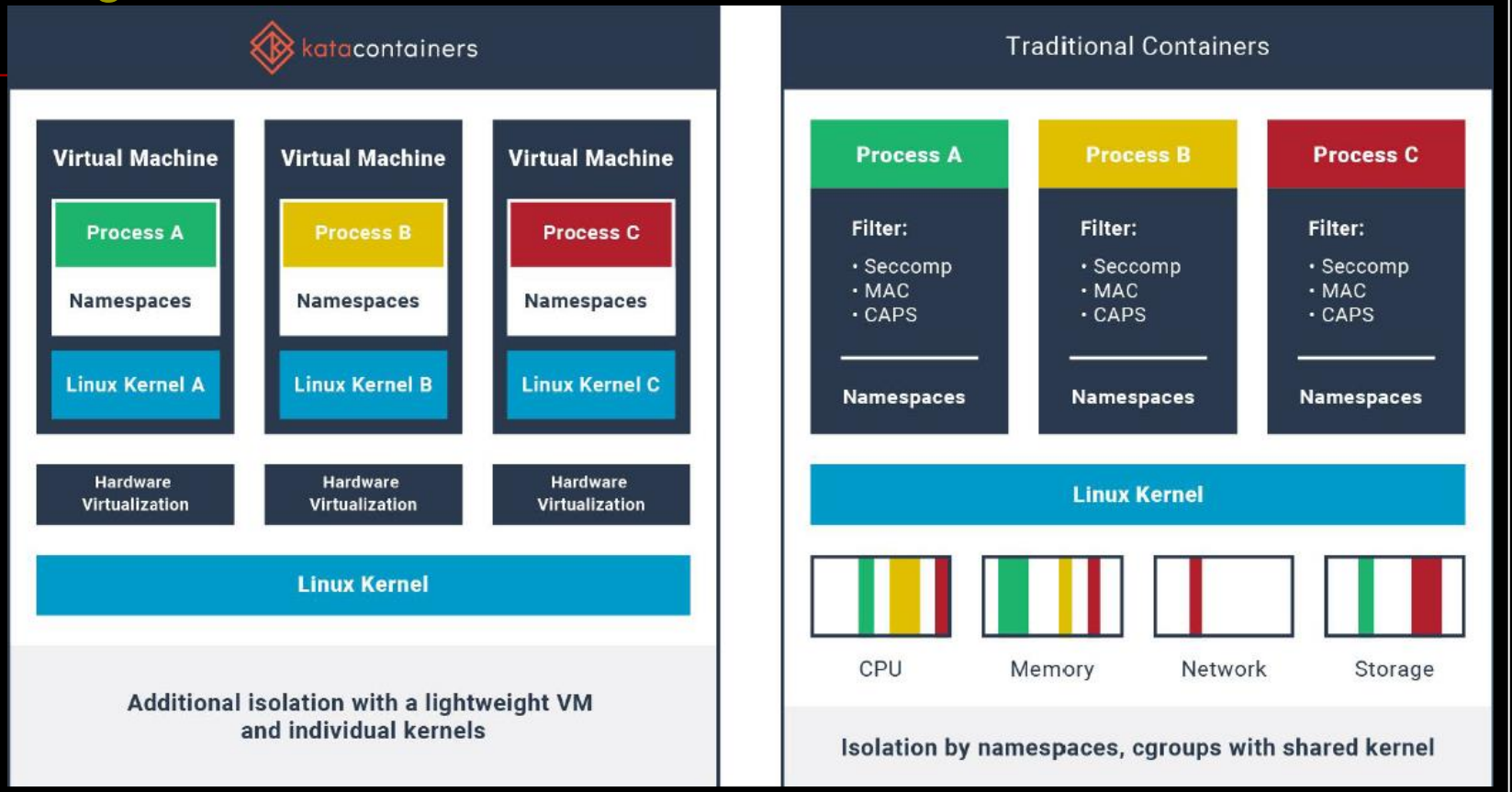
- https://katacontainers.io/
- https://github.com/kata-containers/
- 

- features the speed of containers, the security of VMs



Architecture

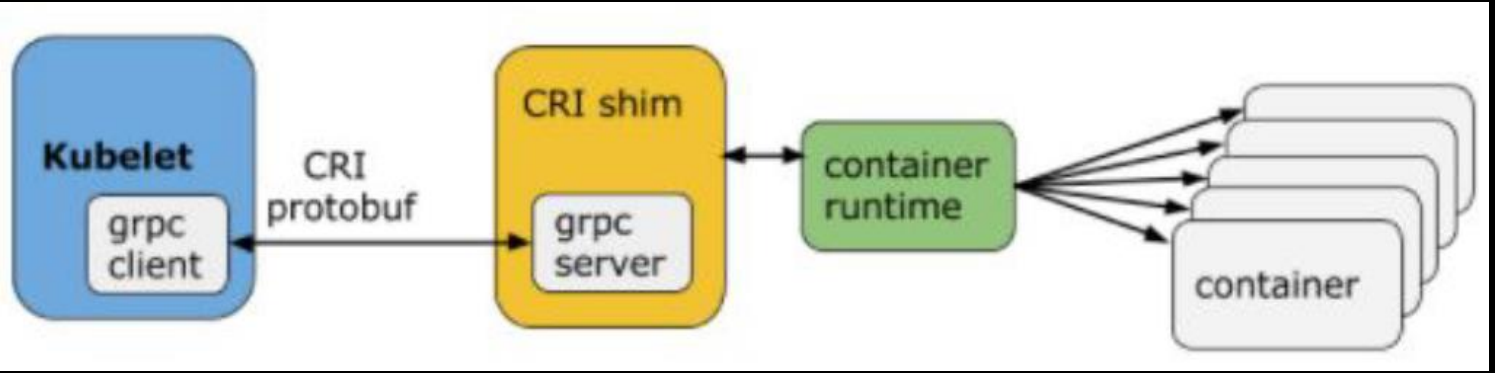
- **https://github.com/kata-containers/documentation/blob/master/design/architecture.md**



Source: https://www.openstack.org/summit/denver-2019/summit-schedule/events/23342/tracing-the-life-of-a-packet-with-kata-containers

Runtime

- The runtime is OCI-compatible, CRI-O-compatible, and Containerd-compatible, allowing it to work seamlessly with both both Docker and Kubernetes respectively
- Kubernetes CRI



- https://github.com/kata-containers/documentation/blob/master/how-to/how-to-use-k8s-with-cri-containerd-and-kata.md

1.2 My Practice

- Pls also refer to my presentation "OpenStack on ARM" at OpenInfra Days 2018(Beijing)
- Try to build Kata Containers on my RPi4

```
kata-runtime - version 1.9.0-rc0 (commit da981919400612bc09f9cee403c2b9fd269a4798-dirty)

• architecture:
        Host:
        golang: arm64
        Build: arm64

• golang:
        go version go1.13.3 linux/arm64

• hypervisors:
        Known: acrn firecracker nemu qemu qemu-virtiofs
        Available for this architecture: qemu

• Summary:

        destination install path (DESTDIR) : /
        binary installation path (BINDIR) : /usr/local/bin
        binaries to install :
         - /usr/local/bin/kata-runtime
         - /usr/local/bin/containerd-shim-kata-v2
         - /usr/libexec/kata-containers/kata-netmon
         - /usr/local/bin/data/kata-collect-data.sh
        configs to install (CONFIGS) :
         - cli/config/configuration-qemu.toml
        install paths (CONFIG_PATHS) :
         - /usr/share/defaults/kata-containers/configuration-qemu.toml
        alternate config paths (SYSCONFIG_PATHS) :
         - /etc/kata-containers/configuration-qemu.toml
        default install path for qemu (CONFIG_PATH) : /usr/share/defaults/kata-containers/configuration.toml
        default alternate config path (SYSCONFIG) : /etc/kata-containers/configuration.toml
        qemu hypervisor path (QEMUPATH) : /usr/bin/qemu-system-aarch64
        assets path (PKGDATADIR) : /usr/share/kata-containers
        proxy+shim path (PKGLIBEXECDIR) : /usr/libexec/kata-containers

    INSTALL  install-scripts
    INSTALL  install-completions
    INSTALL  install-configs
    INSTALL  install-configs
    INSTALL  install-bin
    INSTALL  install-containerd-shim-v2
    INSTALL  install-bin-libexec
```

- **But**

```
[myrpi4@myrpi4h1 KataContainer]$ sudo kata-runtime kata-check
[sudo] password for myrpi4:
ERRO[0000] /usr/share/defaults/kata-containers/configuration-qemu.toml: file /usr/share/kata-containers/vmlinuz.container does not exist  a
rch=arm64 name=kata-runtime pid=39801 source=runtime
/usr/share/defaults/kata-containers/configuration-qemu.toml: file /usr/share/kata-containers/vmlinuz.container does not exist
```

- **https://github.com/kata-containers/documentation/blob/master/install/README.md**

Supported Distributions

Kata is packaged by the Kata community for:

Distribution (link to installation guide)	Versions
CentOS	7
Debian	9
Fedora	28, 29, 30
openSUSE	Leap (15, 15.1) Tumbleweed
Red Hat Enterprise Linux (RHEL)	7
SUSE Linux Enterprise Server (SLES)	SLES 12 SP3
Ubuntu	16.04, 18.04

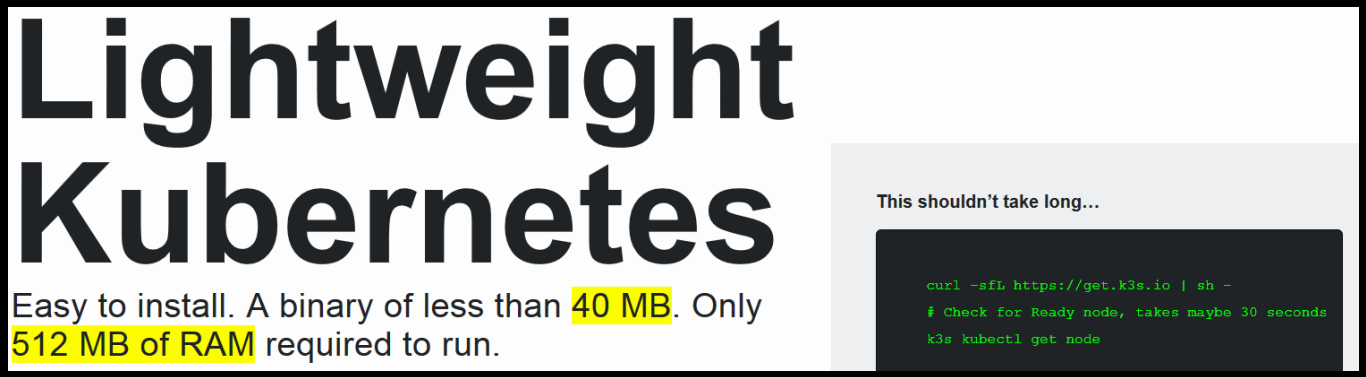
- **may due to the 5.3.x kernel that we built for Manjaro on RPi4, working on find the root cause of this issue…**
- **https://snapcraft.io/install/kata-containers/manjaro**

2) Lightweight Kubernetes

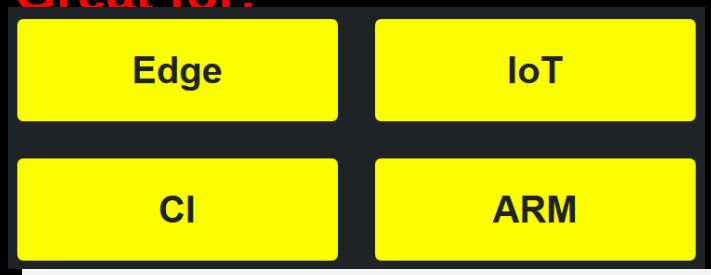
2.1 K3S

Overview

- https://k3s.io/



- Great for:



Optimized for ARM

Both ARM64 and ARMv7 are supported with binaries and multiarch images available for both. k3s works great from something as small as a Raspberry Pi or as large as an AWS a1.4xlarge 32GiB server.

Simplified Operations

k3s is wrapped in a simple package that reduces the dependencies and steps needed to run a production Kubernetes cluster. Packaged as a single binary, k3s makes installation and upgrade as simple as copying a file. TLS certificates are automatically generated to ensure that all communication is secure by default.

Perfect for Edge

K3s is a Certified Kubernetes distribution designed for production workloads in unattended, resource-constrained, remote locations or inside IoT appliances.

Internals

k3s is a fully compliant production-grade Kubernetes distribution with the following changes:

- Legacy, alpha, non-default features are removed. Many of these features are not available in most Kubernetes clusters already.
- Removed in-tree plugins (cloud providers and storage plugins) which can be replaced with out-of-tree add-ons.
- Added sqlite3 as the default storage mechanism. etcd3 is still available, but not the default.
- Wrapped in a simple launcher that handles a lot of the complexity of TLS and options.

Minimum System Requirements

Linux 3.10+
512 MB of ram per server
75 MB of ram per node
200 MB of disk space
x86_64, ARMv7, ARM64

Removes

Legacy and non-default features
Alpha features
In-tree cloud providers
In-tree storage drivers
Docker (optional)

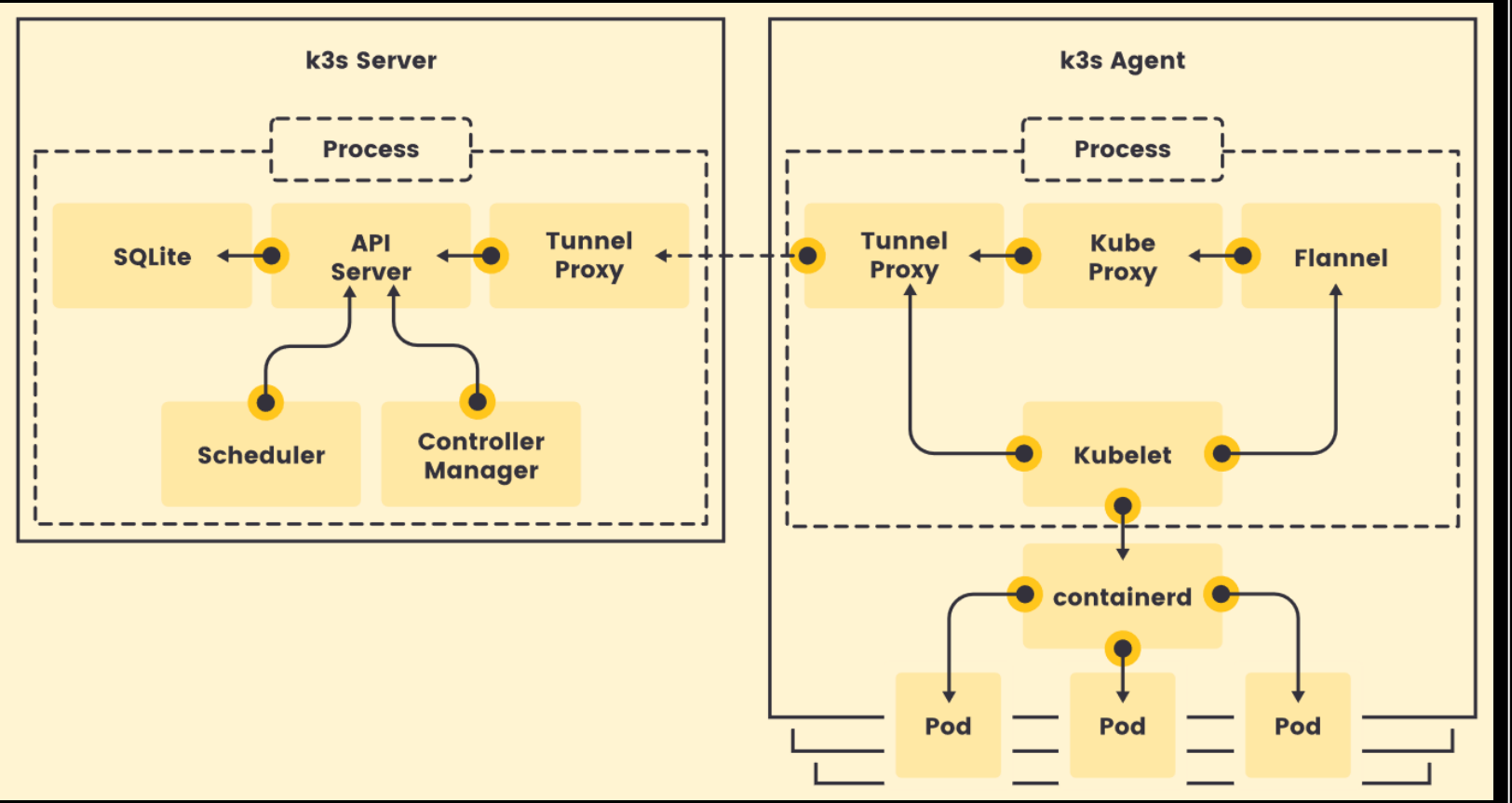
Adds

Simplified installation
SQLite3 support in addition to etcd
TLS management
Automatic Manifest and Helm Chart management
containerd, CoreDNS, Flannel

Minimal to no OS dependencies (just a sane kernel and cgroup mounts needed). k3s packages required dependencies

- containerd
- Flannel
- CoreDNS
- CNI
- Host utilities (iptables, socat, etc)

How it works



- for practicing K3S on ARM64, pls also refer to my presentation "In-Kernel Virtual Machine & Service" at OSDT Beijing (Nov 9, 2019)

3) Cilium

Overview

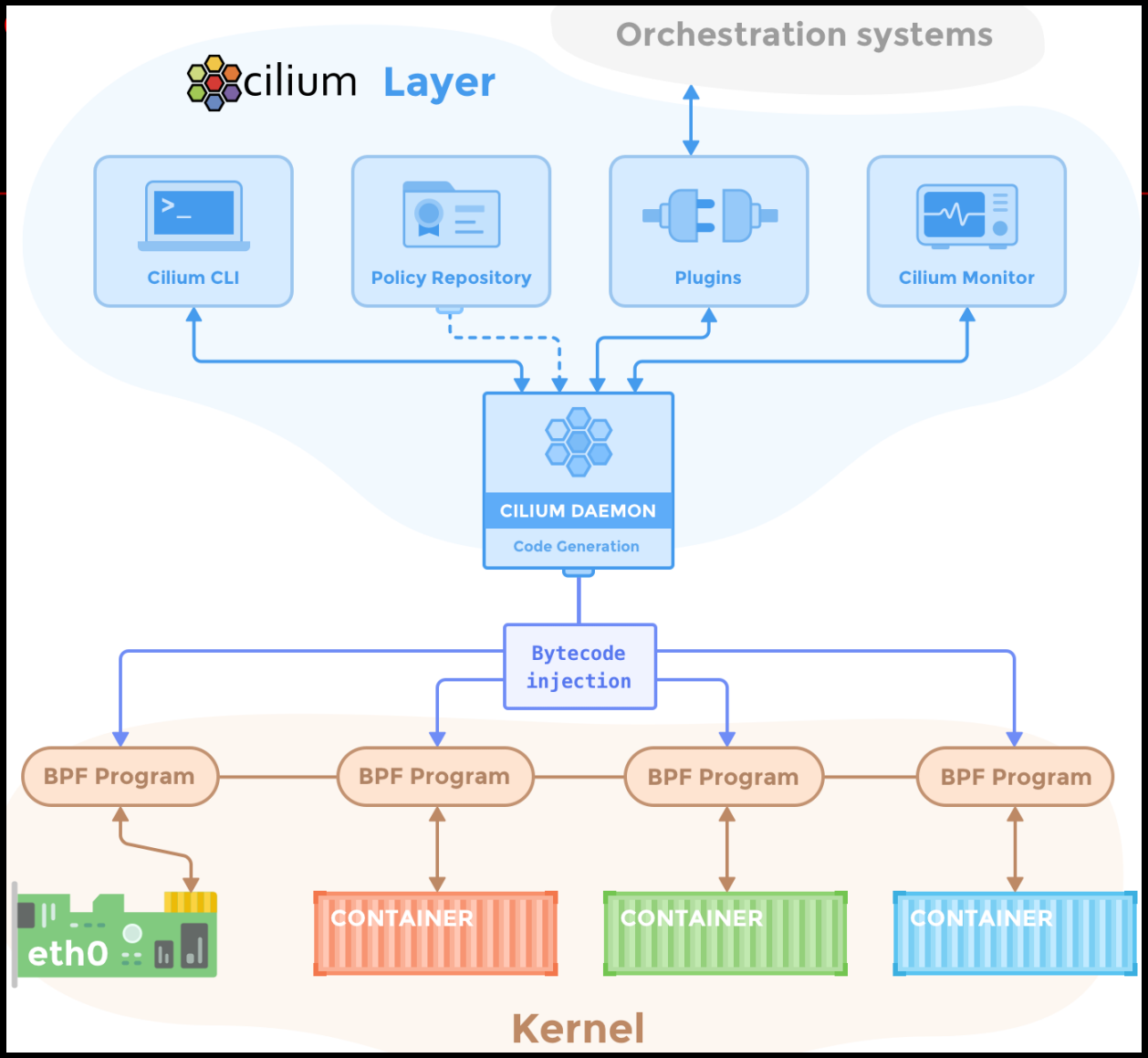
- https://cilium.io/
- https://github.com/cilium/

API Aware Networking and Security using BPF and XDP

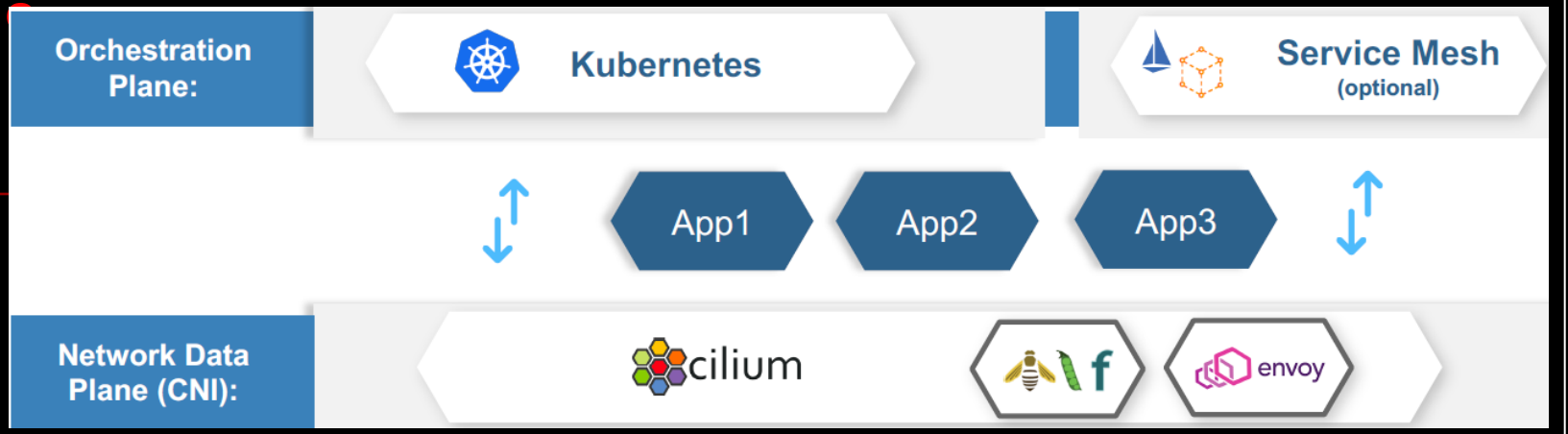
- Cilium is open source software for providing and transparently securing network connectivity and loadbalancing between application workloads such as application containers or processes. Cilium operates at Layer 3/4 to provide traditional networking and security services as well as Layer 7 to protect and secure use of modern application protocols such as HTTP, gRPC and Kafka. Cilium is integrated into common orchestration frameworks such as Kubernetes and Mesos.

Source: "How Cilium uses BPF to Supercharge Kubernetes Networking & Security", Mark Darnell, Roger Klorese, and Dan Wendlandt

How it works

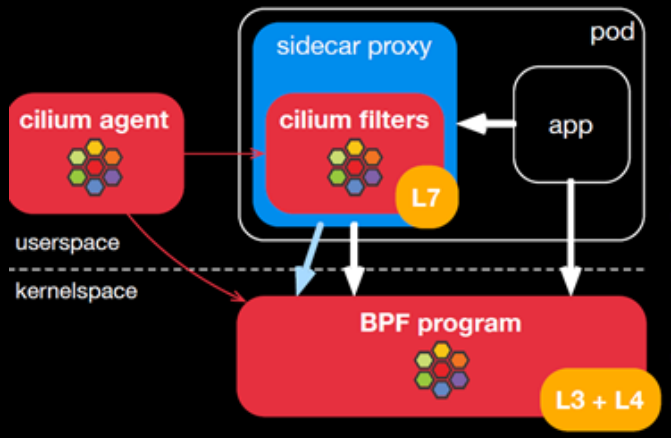

Orchestration systems
cilium Layer
Cilium CLI
Policy Repository
Plugins
Cilium Monitor
CILIUM DAEMON
Code Generation
Bytecode injection
BPF Program
BPF Program
BPF Program
BPF Program
eth0
CONTAINER
CONTAINER
CONTAINER
Kernel

Bringing the Power of BPF to Kubernetes & Service Mesh



Source: "How Cilium uses BPF to Supercharge Kubernetes Networking & Security", Mark Darnell, Roger Klorese, and Dan Wendlandt

Integration architecture

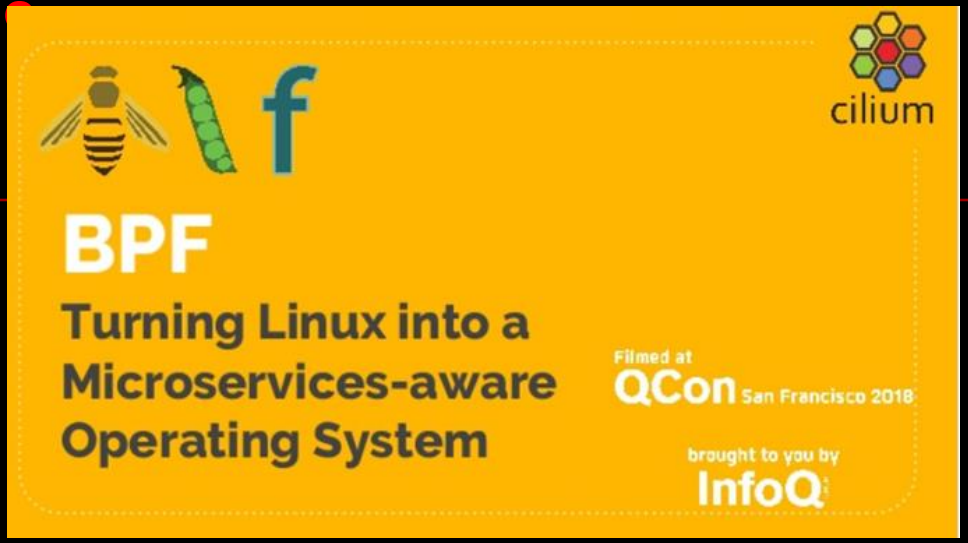


- Comprehensive L3-L7 policy language
- L7 enforcement using shared Envoy proxy
- Cilium-specific filters
- Applies to clear traffic (mTLS support)
- L3/L4 enforcement using BPF in-kernel
- Applies to all traffic, incl. Istio control plane & egress traffic

Source: "Kernel advantages for Istio realized with Cilium", Cynthia Thomas, OSCON 2018

an ambitious project

-



4) In-Kernel Services

4.1 Polycube

Overview

- **https://github.com/polycube-network/**

`Polycube` is an **open source** software framework for Linux that enables the creation of **virtual networks** and provides **fast** and **lightweight network functions**, such as *bridge*, *router*, *nat*, *load balancer*, *firewall*, *DDoS mitigator*, and more.

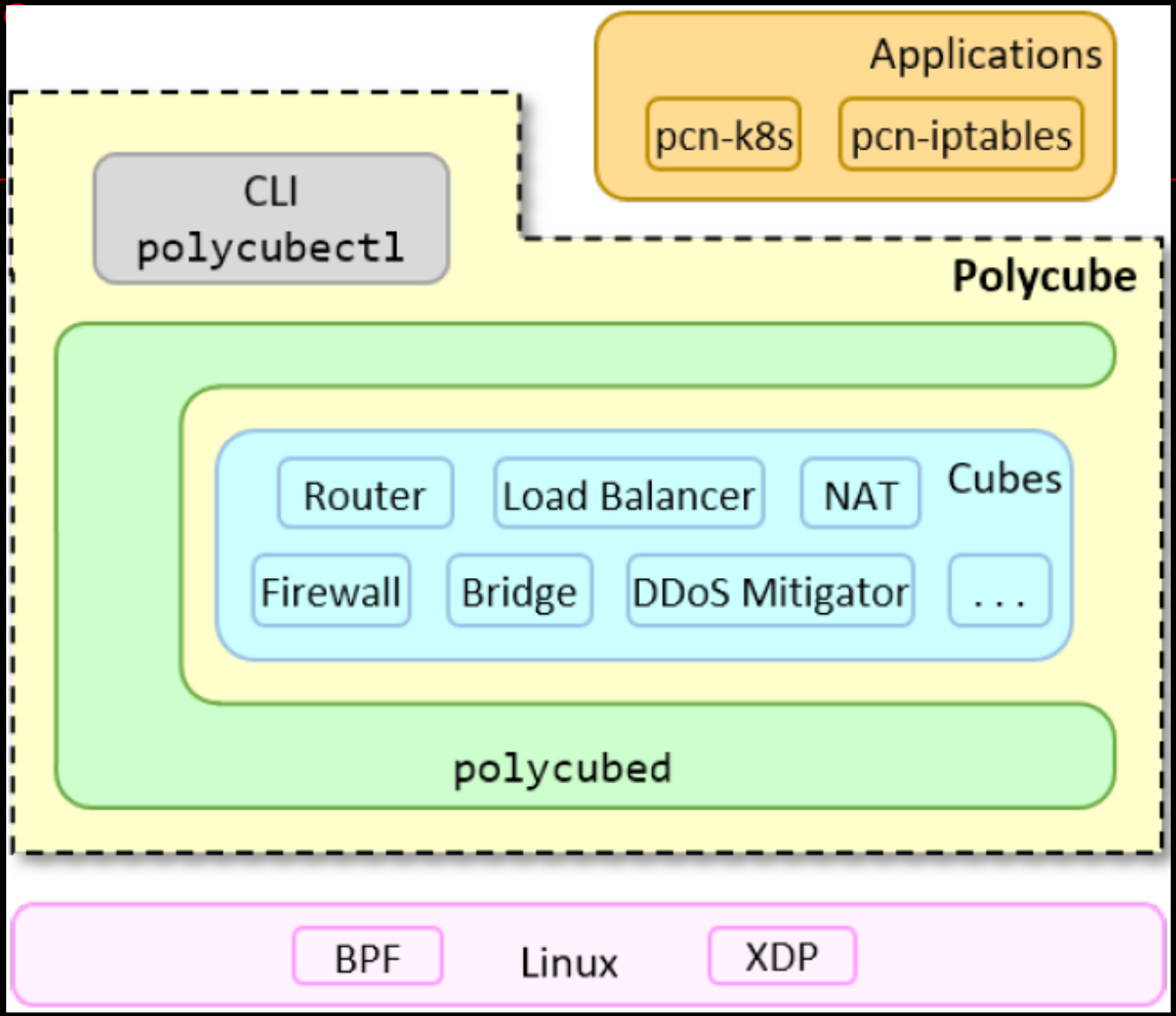
Within each virtual network, individual network functions can be composed to build arbitrary **service chains** and provide custom network connectivity to **namespaces**, **containers**, **virtual machines**, and **physical hosts**.

Virtual functions, called *cubes*, are extremely **efficient** because are based on the recent *BPF* and *XDP* Linux kernel technologies. In addition, cubes are easily **extensible** and **customizable**.

Polycube can control its entire virtual topology and all the network services with a simple and coherent command line, available through the *polycubectl* tool. A set of equivalent commands can be issued diretly to *polycubed*, the Polycube REST-based daemon, for better machine-to-machine interaction.

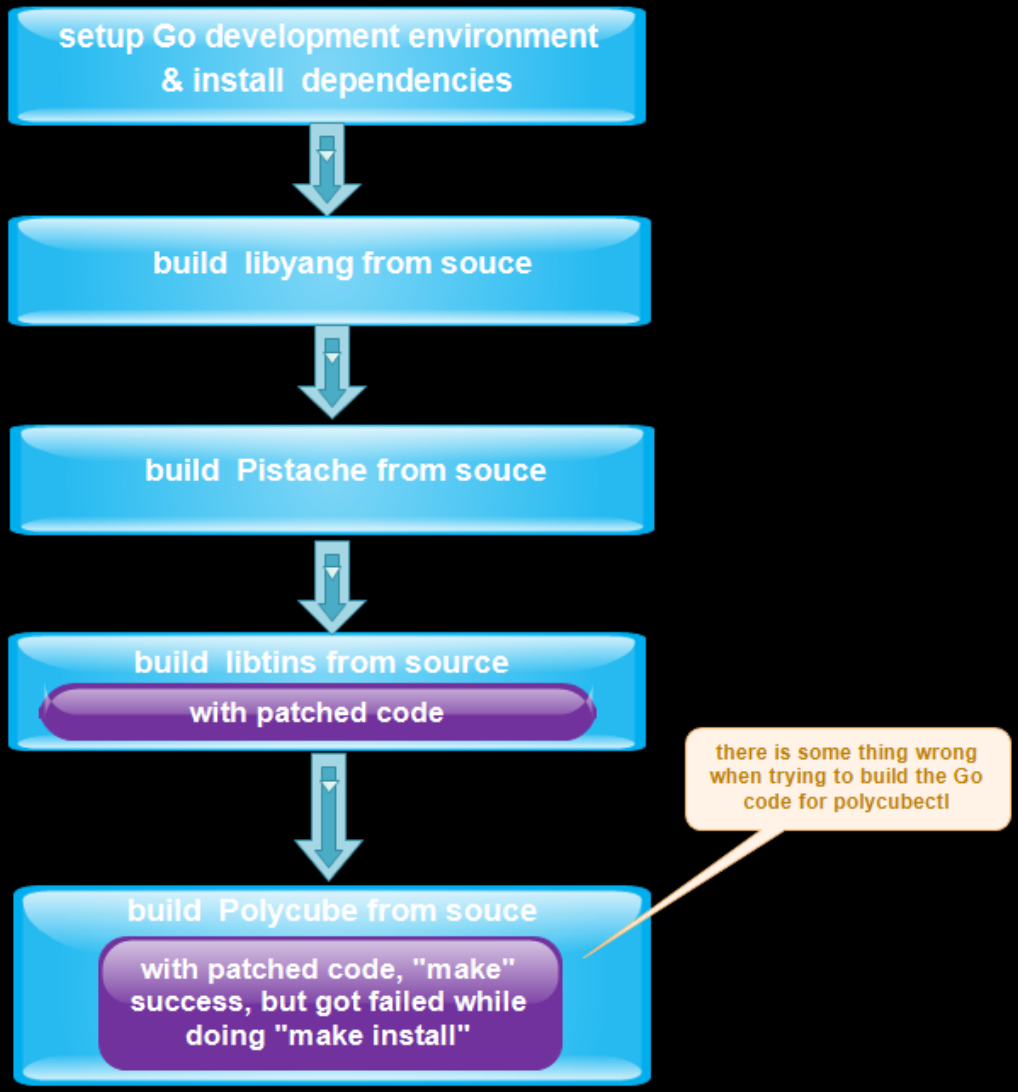
Polycube also provides two working **standalone applications** built up using this framework. *pcn-K8s* is a Polycube-based CNI plug-in for *Kubernetes*, which can handle the network of an entire data center. It also delivers better throughput as compared with some of the existing CNI plug-ins. *pcn-iptables* is a more efficient and scalable clone of the existing Linux *iptables*.

Architecure



My Practice

- building Polycube on ARM64



VI. Distributed Framework

1) Ray

The Machine Learning Ecosystem

Distributed System	Distributed System	Distributed System	Distributed System	Distributed System	Distributed System
Distributed Training	Model Serving	Streaming	Distributed RL	Data Processing	Hyperparameter Search
Horovod, Distributed TF, Parameter Server	Clipper, TensorFlow Serving	Flink, many others	Baselines, RLlab, ELF, Coach, TensorForce, ChainerRL	MapReduce, Hadoop, Spark	Vizier, many internal systems at companies

Source: "Scaling Emerging AI Applications with Ray", Peter Schafhalter, QCon London 2019

What is it

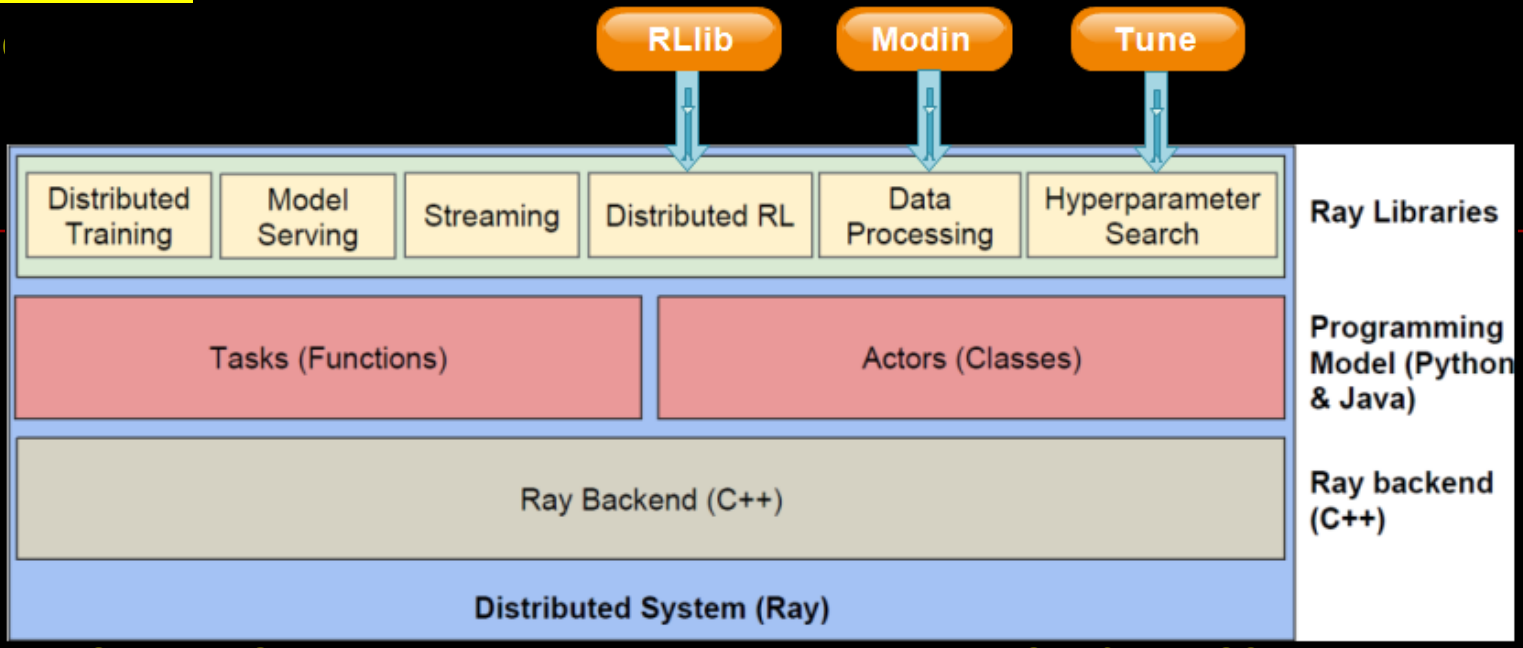
- https://rise.cs.berkeley.edu/projects/ray/

- https://github.com/ray-project/
- a system for distributed Python that unifies the ML ecosystem
- a fast and simple framework for building and running distributed applications

Execute Python functions in parallel.

```
import ray
ray.init()

@ray.remote
def f(x):
    return x * x

futures = [f.remote(i) for i in range(4)]
print(ray.get(futures))
```

Architecture



Source: "Scaling Emerging AI Applications with Ray", Peter Schafhalter, QCon London 2019

Tune: Scalable Hyperparameter Tuning

RLlib: Scalable Reinforcement Learning

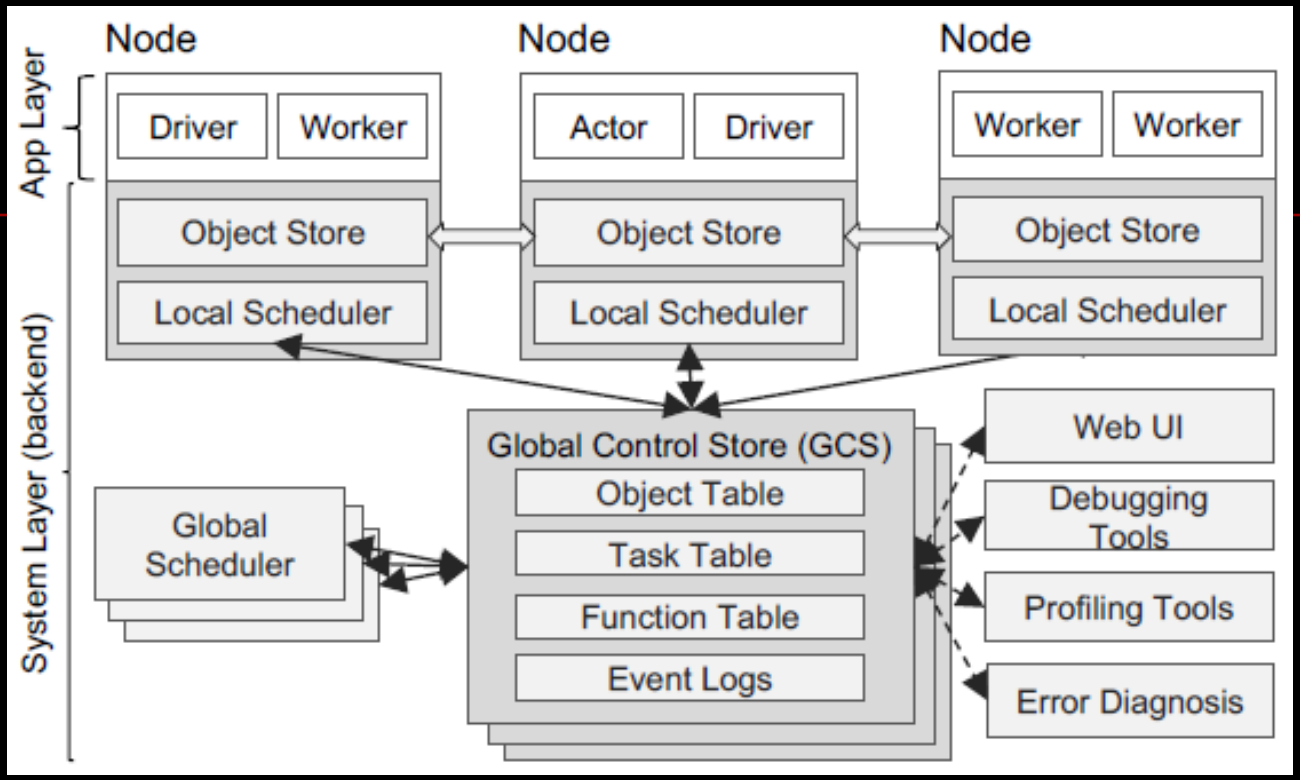
Experimental:

- Distributed Training (Experimental)
- TF Distributed Training
- Pandas on Ray
- Ray Projects (Experimental)
- Signal API (Experimental)
- Async API (Experimental)
- Ray Serve (Experimental)

has moved to Modin (https://github.com/modin-project/)

```
# import pandas as pd
import modin.pandas as pd
```

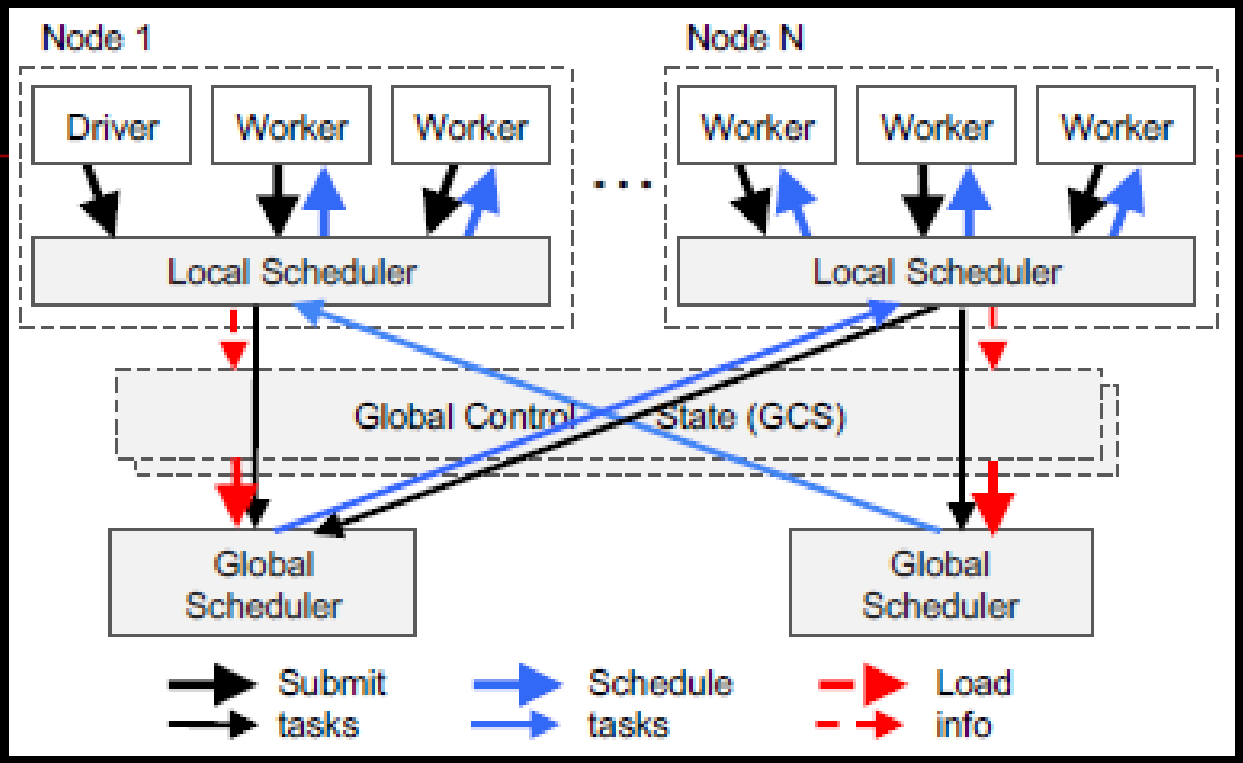
- **layered architecture**



Source: "Ray: A Distributed Framework for Emerging AI Applications", Philipp Moritz, Robert Nishihara, etc

Scheduling

- bottom-up distributed scheduler



Source: "Ray: A Distributed Framework for Emerging AI Applications", Philipp Moritz, Robert Nishihara, etc

Notes

- **Serialization**

  Plasma is a high-performance shared memory object store originally developed in Ray and now being developed in Apache Arrow.

- **Data Format**

  Ray optimizes for NumPy arrays by using the Apache Arrow.

- **Redis**

  GCS uses one Redis key-value store per shard, with entirely single-key operations.

- **AutoScaler**

  GCS uses one Redis key-value store per shard, with entirely single-key operations.

```
[myrpi4@myrpi4h1 ray-master]$ tree -C -L 1 python/ray/autoscaler
python/ray/autoscaler
├── autoscaler.py
├── aws
├── commands.py
├── docker.py
├── gcp
├── __init__.py
├── kubernetes
├── local
├── log_timer.py
├── node_provider.py
├── tags.py
└── updater.py
```

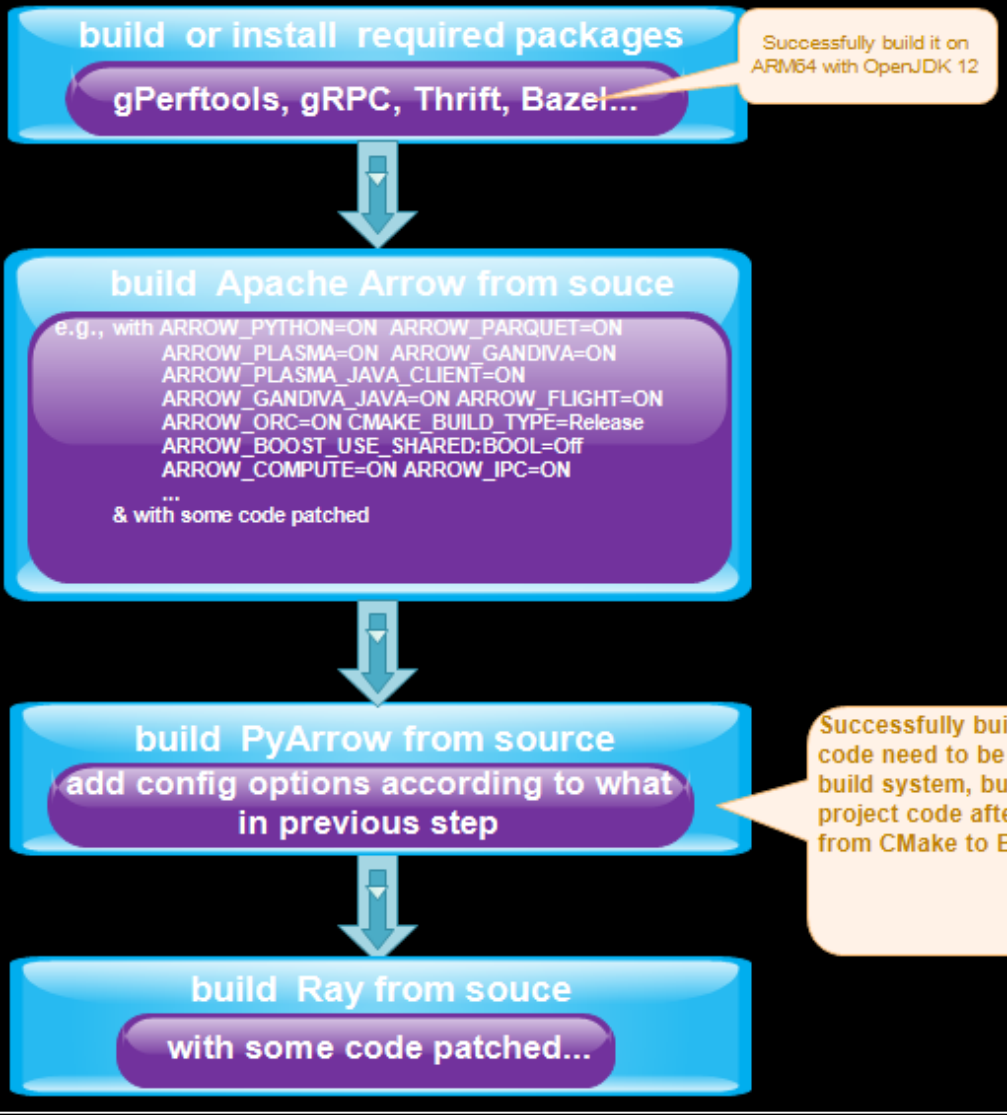
- **Deploying on Kubernetes**

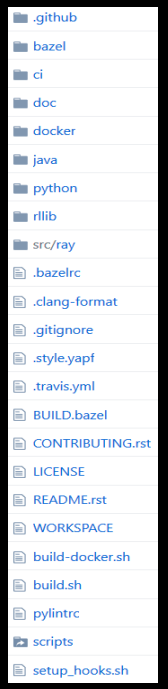
  The easiest way to run a Ray cluster is by using the built-in Autoscaler, which has support for running on top of Kubernetes.

  Warning: running Ray on Kubernetes is still a work in progress.

2) My Practice

- porting Ray for ARM is still on the way





Successfully build Ray (some 3rd-party external projects code need to be patched)on ARM64 with the old CMake build system, but still got failed to build the latest Ray project code after the build system of Ray switching from CMake to Bazel(since Ray v0.6.4)...

2.1 Redesign & Reimplementation of Ray

- extending Ray as a general-purse distributed App framework
- extending Ray as a generic scheduler for HPC tasks
- …

VII. Messaging & RPC

1) gRPC & Protobuf

- https://github.com/grpc
- https://github.com/protocolbuffers/protobuf
- they are the most popular Messaging & RPC projects

Future

Reconsider nanomsg & msgpack when implementing DRay, DRPC, and ntopng2:

- https://github.com/nanomsg/
- https://github.com/msgpack

2) Rethink In-Kernel Messaging

- **IKBUS (In-Kernel Bus)**
  for messaging among In-Kernel services & Kernel subsystems

Also refer to

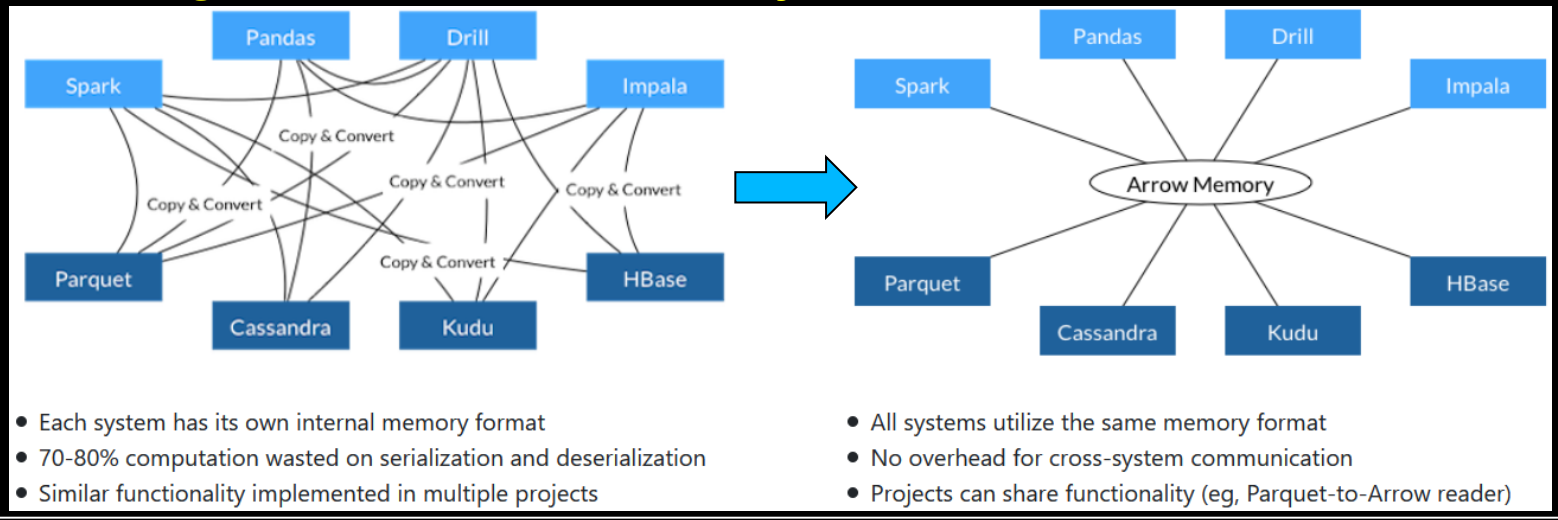
- **https://www.freedesktop.org/wiki/Software/systemd/kdbus/**
  Linux kernel D-Bus implementation, but not limit to it…
- **https://github.com/bus1**
  Capability-based IPC for Linux

VIII. Data Processing

1) Apache Arrow

- https://arrow.apache.org/
- https://github.com/apache/arrow
- a cross-language development platform for in-memory data. It specifies a standardized language-independent columnar memory format for flat and hierarchical data, organized for efficient analytic operations on modern hardware. It also provides computational libraries and zero-copy streaming messaging and interprocess communication…

Advantages of a Common Data Layer



Components

- **Major components of the project include:**

- The Arrow Columnar In-Memory Format
- 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

- **https://arrow.apache.org/powered_by/**

- **Apache Parquet:** A columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language. The C++ and Java implementation provide vectorized reads and write to/from Arrow data structures.
- **Apache Spark:** Apache Spark™ is a fast and general engine for large-scale data processing. Spark uses Apache Arrow to
  1. improve performance of conversion between Spark DataFrame and pandas DataFrame
  2. enable a set of vectorized user-defined functions (`pandas_udf`) in PySpark.
- **Dask:** Python library for parallel and distributed execution of dynamic task graphs. Dask supports using pyarrow for accessing Parquet files
- **Data Preview:** Data Preview is a Visual Studio Code extension for viewing text and binary data files. Data Preview uses Arrow JS API for loading, transforming and saving Arrow data files and schemas.
- **Dremio:** A self-service data platform. Dremio makes it easy for users to discover, curate, accelerate, and share data from any source. It includes a distributed SQL execution engine based on Apache Arrow. Dremio reads data from any source (RDBMS, HDFS, S3, NoSQL) into Arrow buffers, and provides fast SQL access via ODBC, JDBC, and REST for BI, Python, R, and more (all backed by Apache Arrow).
- **Fletcher:** Fletcher is a framework that can integrate FPGA accelerators with tools and frameworks that use the Apache Arrow in-memory format. From a set of Arrow Schemas, Fletcher generates highly optimized hardware structures that allow accelerator kernels to read and write RecordBatches at system bandwidth through easy-to-use interfaces.

- **…**

2) Lightweight Solution

- As Spark and Flink are not considered, and streaming data processing is important in IoT

Interesting Projects

TDengine

- https://github.com/taosdata/TDengine

  An open-source big data platform designed and optimized for the IoT

Stuart

- https://github.com/nubix-io/stuart
- a pure Lua rewrite of Apache Spark 2.2, designed for embedding and edge computing
- Design

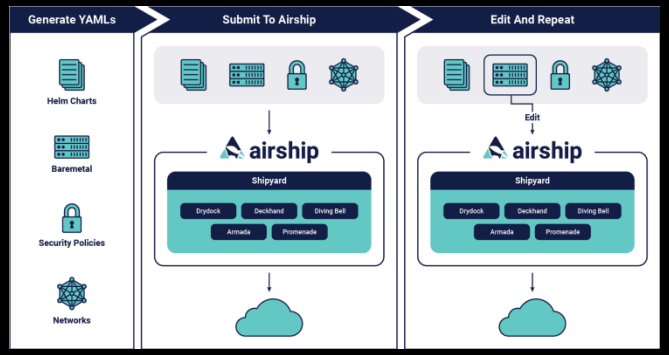
Stuart is designed for real-time and embedding, and so it follows some rules:

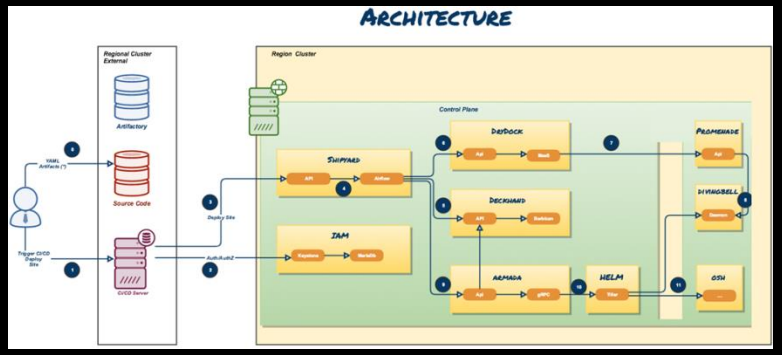
- It does not perform deferred evaluation of anything; all compute costs are paid upfront for predictable throughput.
- It uses pure Lua and does not include native C code. This maximizes portability and opportunity to be cross-compiled. Any potential C code optimizations are externally sourced through the module loader. For example, Stuart links to `lunajson`, but it also detects and uses `cjson` when that native module is present.
- It does not execute programs (like `ls` or `dir` to list files), because there may not even be an OS.
- It does not make use of coroutines, in order to ensure easy transpiling to C.
- It does not use upvalues or metatables in module scripts, so that module tables can be burned into ROM and chipsets (see eLua LTR)
- It should be able to eventually do everything that Apache Spark does.

IX. Artificial Intelligence

1) Trends

- Project Management, Pipeline, and Visualization
  https://mlflow.org/
  https://pipeline.ai
  …
- AIoT(Artificial Intelligence of Things)
  https://www.forbes.com/sites/janakirammsv/2019/08/12/why-aiot-is-emerging-as-the-future-of-industry-40/#3fa88278619b
  https://aiotworkshop.github.io/
  …
- AIOps(Artificial Intelligence for IT Operations)
  https://www.airshipit.org/





…

2) TVM

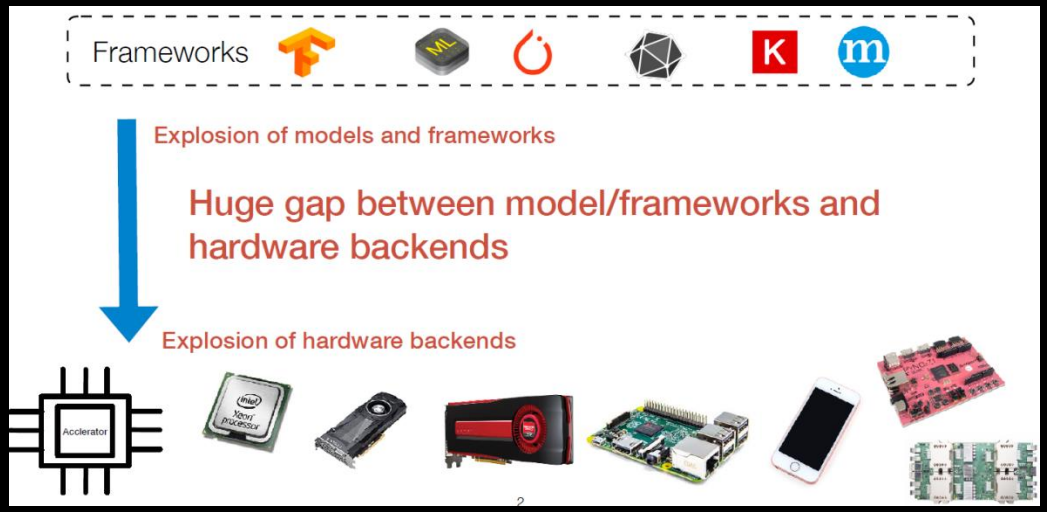
- https://tvm.ai/
- End to End Deep Learning Compiler Stack for CPUs, GPUs and specialized accelerators

TVM is an open deep learning compiler stack for CPUs, GPUs, and specialized accelerators. It aims to close the gap between the productivity-focused deep learning frameworks, and the performance- or efficiency-oriented hardware backends. TVM provides the following main features:

- Compilation of deep learning models in Keras, MXNet, PyTorch, Tensorflow, CoreML, DarkNet into minimum deployable modules on diverse hardware backends.
- Infrastructure to automatic generate and optimize tensor operators on more backend with better performance.

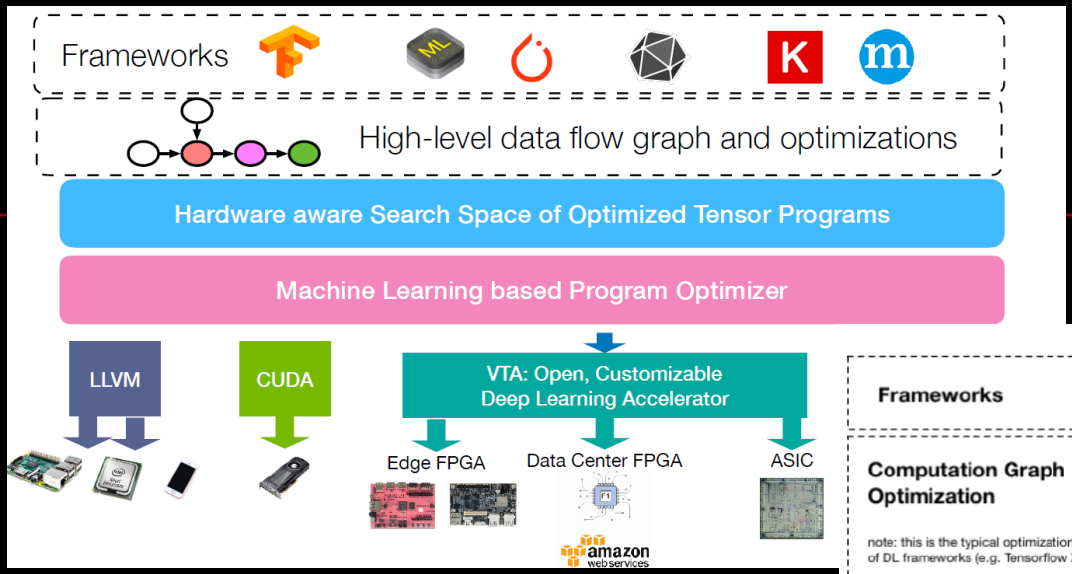
TVM stack began as a research project at the SAMPL group of Paul G. Allen School of Computer Science & Engineering, University of Washington. The project is now driven by an open source community involving multiple industry and academic institutions. The project adopts Apache-style merit based governace model.

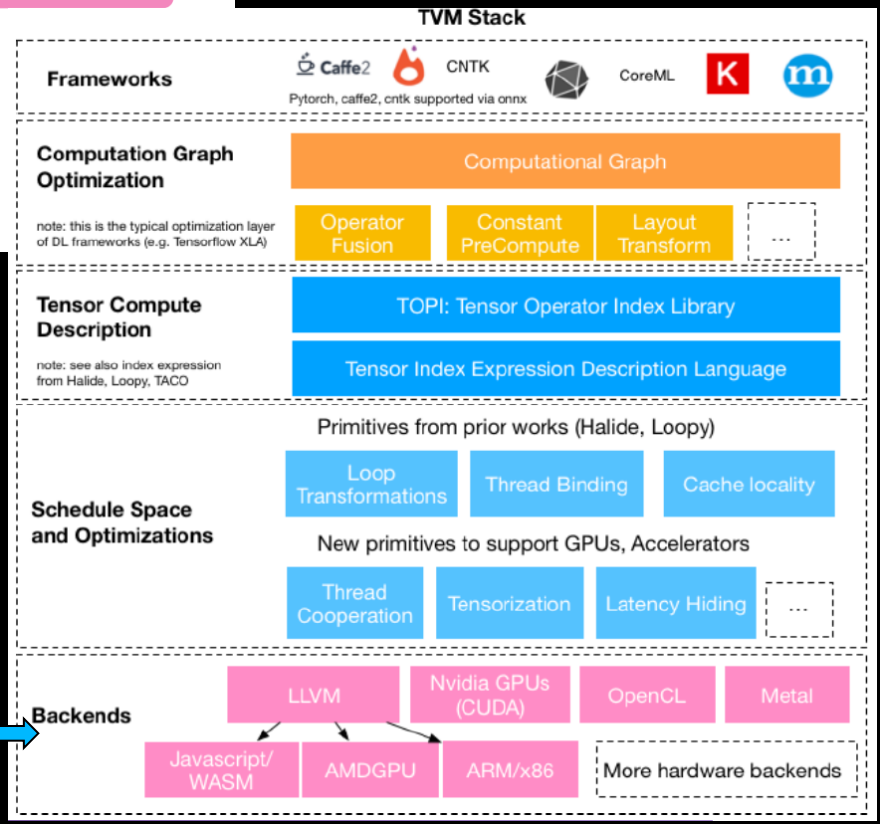
TVM provides two level optimizations show in the following figure. Computational graph optimization to perform tasks such as high-level operator fusion, layout transformation, and memory management. Then a tensor operator optimization and code generation layer that optimizes tensor operators. More details can be found at the techreport.

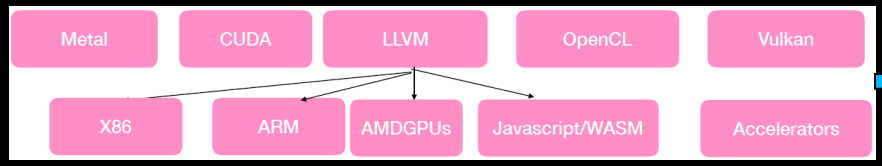


Source: "TVM: An Automated End-to-End Optimizing Compiler for Deep Learning", Tianqi Chen…

Architecture & Design





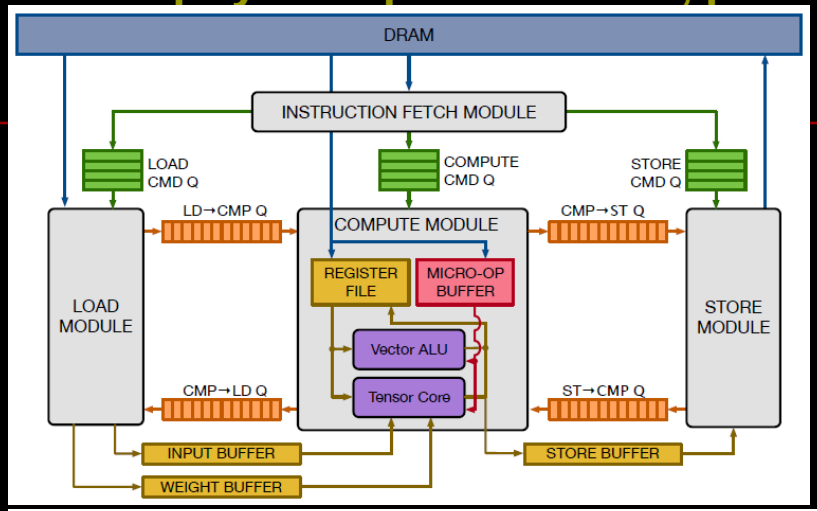


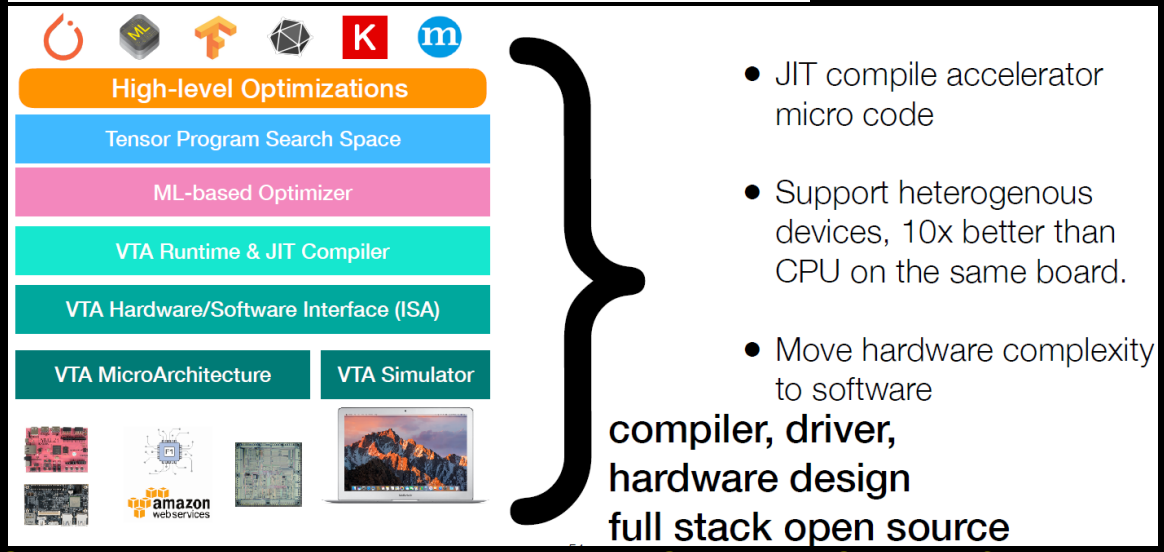
Source: “支援TVM AI Compiler on RISC V with P Instructions”, Jenq-KuenLee, RISC-V Conf TW 2019

VTA Hardware Architecture

https://docs.tvm.ai/vta/index.html

Philosophy: simple hardware, provide software-defined flexibility

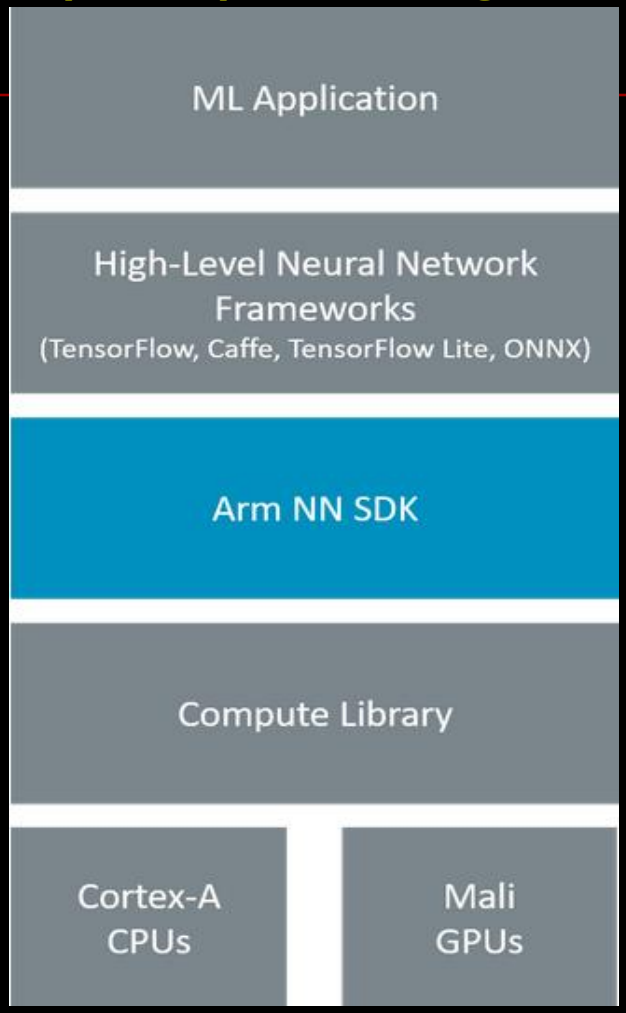


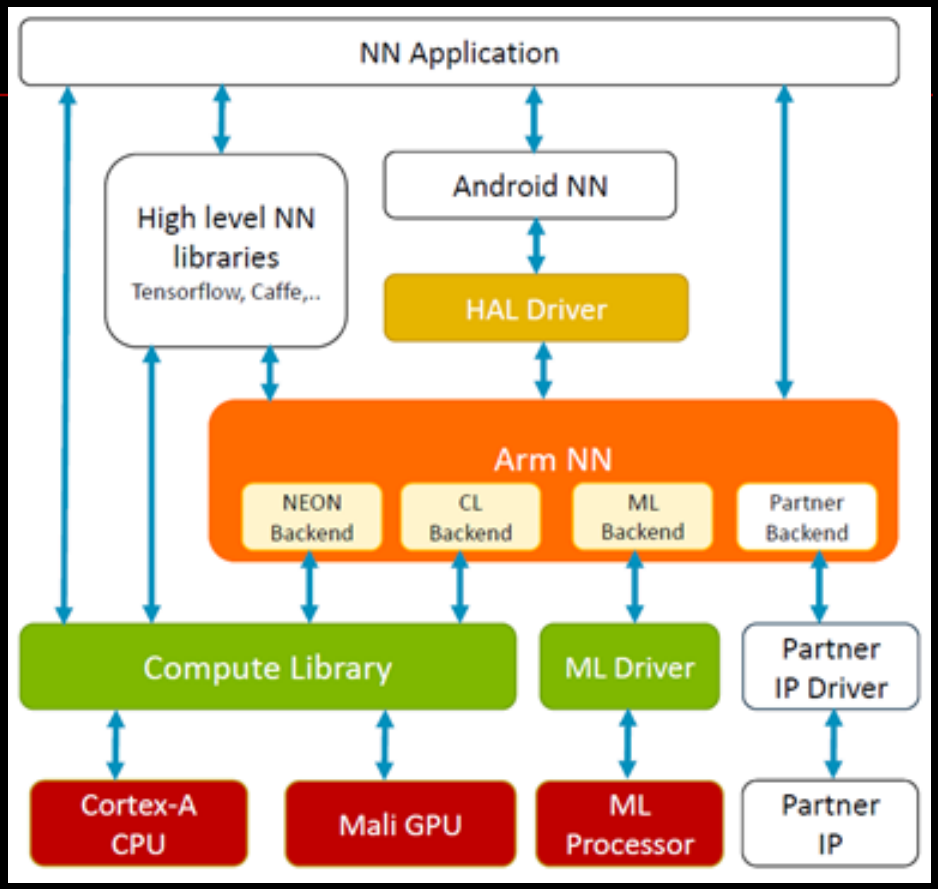


Source: "TVM: An Automated End-to-End Optimizing Compiler for Deep Learning", Tianqi Chen…

3) ARM NN

- https://www.arm.com/products/silicon-ip-cpu/machine-learning/arm-nn
- https://mlplatform.org



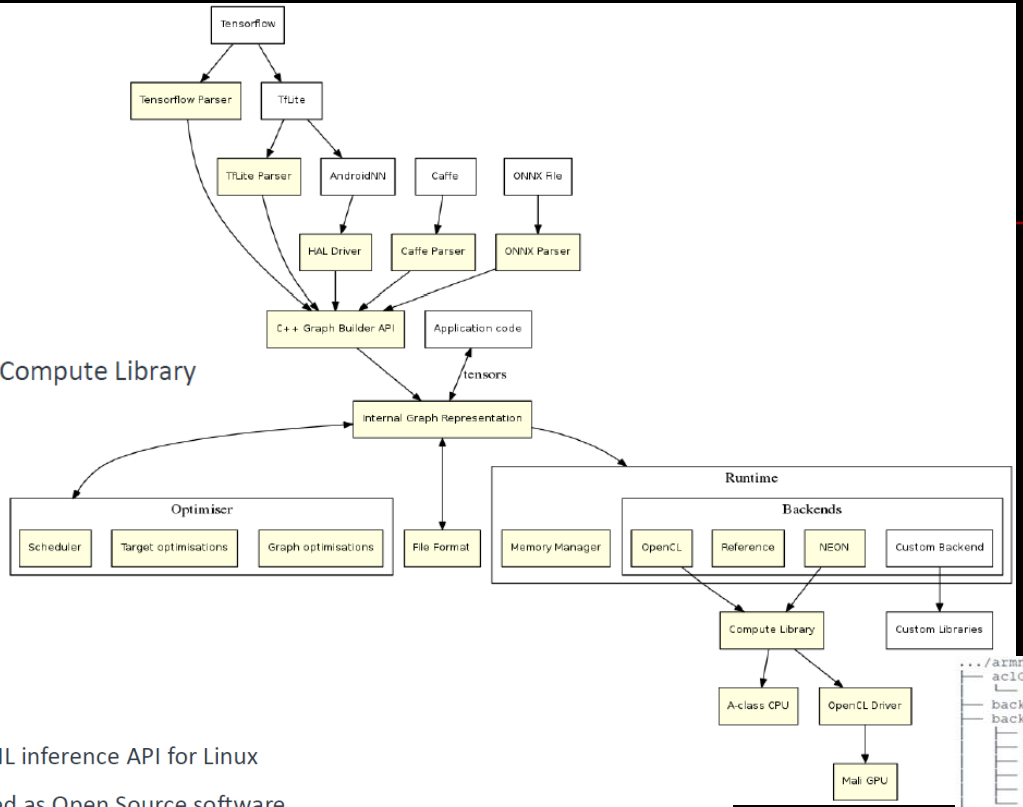


Source: "Arm NN 19.08 Improvements", Sadik Armagan & Jim Flynn, Linaro Connect 2019(San Diego)

- https://www.arm.com/why-arm/technologies/compute-library

Components & Design

- Arm NN Core
  - Graph builder API
  - Optimizer
  - Runtime
    - Reference and Neon/Cl via Compute Library
    - New backend planned
- Parsers
  - Tensorflow
  - Tensorflow Lite
  - Caffe
  - ONNX
- Android NNAPI Driver
  - C++ 14 ML inference API for Linux
  - Developed as Open Source software
    - Contributions are welcome from anyone and are reviewed before acceptance
  - Synchronised release with Compute Library and Android NNAPI driver libraries
    - Compute Library
      - Arm CPU with NEON acceleration (ARMv7 and v8x)
      - Arm Mali GPU with OpenCL acceleration (Midgard and Bifrost architectures)
    - Android NNAPI driver
      - Forwards Android NNAPI HAL calls to the Arm NN API




```
.../armnn/src/backends
├── aclCommon
│   └── ...
├── backends.cmake
├── backendsCommon
│   ├── BackendRegistry.hpp
│   ├── IBackendContext.hpp
│   ├── IBackendInternal.hpp
│   ├── IMemoryManager.hpp
│   ├── ...
│   └── test
│       ├── TestDynamicBackend.hpp
│       ├── TestDynamicBackend.cpp
│       └── ...
├── dynamic
│   └── reference
│       ├── CMakeLists.txt
│       ├── RefDynamicBackend.cpp
│       └── RefDynamicBackend.hpp
├── cl
│   ├── backend.cmake
│   ├── backend.mk
│   ├── CMakeLists.txt
│   └── ...
├── neon
│   ├── backend.cmake
│   ├── backend.mk
│   ├── CMakeLists.txt
│   └── ...
├── my_backend
│   ├── backend.cmake
│   ├── backend.mk
│   ├── CMakeLists.txt
│   └── ...
├── README.md
└── reference
    └── ...
```

Source: "Arm NN 19.08 Improvements", Sadik Armagan & Jim Flynn, Linaro Connect 2019(San Diego)

Source: "Arm NN plug-in framework", Les Bell, Matteo Martincigh, Narumol Prangnawarat , Linaro Connect 2019(Bangkok)

4) My Practice

- successfully built TVM and PyTorch on RPi4 natively
- successfully built ComputeLibrary on RPi4 natively by changing one line in its Scons build file
- successfully built ARMNN on RPi4 natively with Caffe disabled

- Android NDK: How to use Android NDK to build Arm NN
- Cross compilation from x86_64 Ubuntu to arm64 Linux: Arm NN Cross Compilation
- Native compilation under aarch64 Debian 9

- Next Steps

1. add support PyTorch support in ARM NN

   https://pytorch.org/blog/pytorch-1-dot-3-adds-mobile-privacy-quantization-and-named-tensors/

Component	Description
torch	a Tensor library like NumPy, with strong GPU support
torch.autograd	a tape-based automatic differentiation library that supports all differentiable Tensor operations in torch
torch.jit	a compilation stack (TorchScript) to create serializable and optimizable models from PyTorch code
torch.nn	a neural networks library deeply integrated with autograd designed for maximum flexibility
torch.multiprocessing	Python multiprocessing, but with magical memory sharing of torch Tensors across processes. Useful for data loading and Hogwild training
torch.utils	DataLoader and other utility functions for convenience

- Breaking Changes
- Highlights
  - [Experimental]: Mobile Support
  - [Experimental]: Named Tensor Support
  - [Experimental]: Quantization support
  - Type Promotion
  - Deprecations
- New Features
  - TensorBoard: 3D Mesh and Hyperparameter Support
  - Distributed
  - Libtorch Binaries with C++11 ABI
  - New TorchScript features
- Improvements

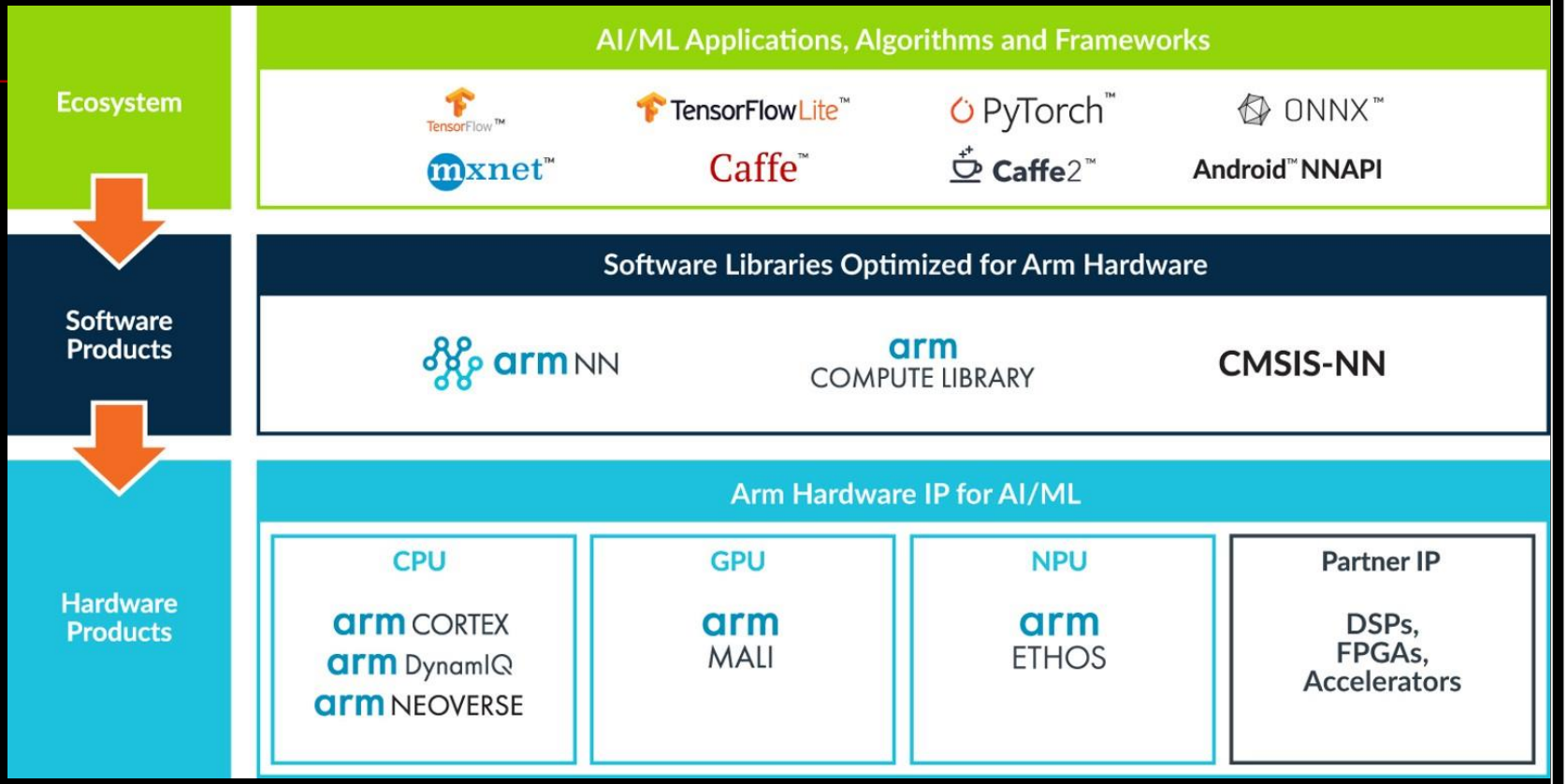
…

- android
- aten
- benchmarks
- binaries
- c10
- caffe2
- cmake
- docker
- docs
- ios
- modules
- scripts
- submodules
- test
- third_party
- tools
- torch

2. useful components such like Caffe2, QNNPACK, and Captum from PyTorch
3. practice TVM FPGA backend

- **ARM NN future roadmap**

  https://developer.arm.com/ip-products/processors/machine-learning/arm-nn



X. Monitoring, Tuning, and Debugging

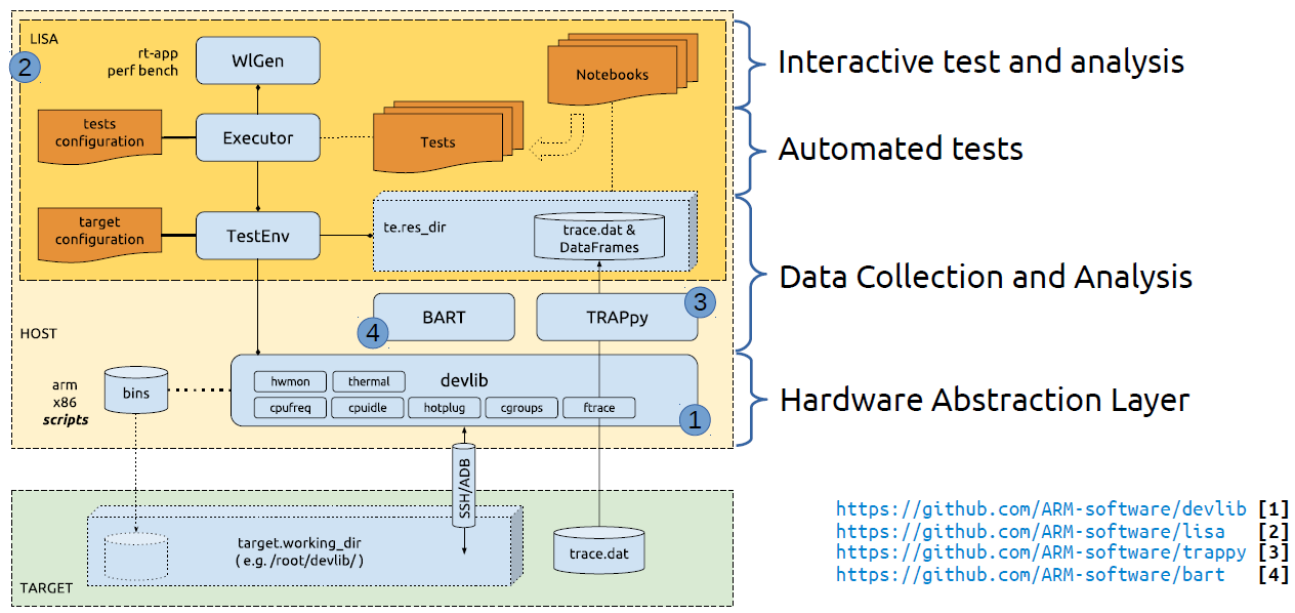
Design Goal

- the Swiss Army Knife for our development work

1) Extending LISA

Project LISA

- Linux Integrated System Analysis
- https://github.com/ARM-software/lisa



Source: "Linux Integrated System Analysis (LISA) & Friends ", Patrick Bellasi, ELC 2016

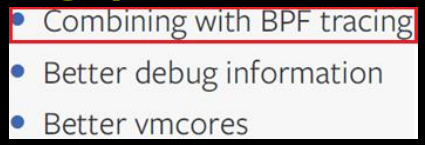
2) Extending ntopng

- rethinking ntopng as a good basic framework which is extended as a system monitoring toolkit. In particular, it is moving towards eBPF-based solutions.
- fully replace InfluxDB with our new DRedis database
- pls also refer to my presentation "In-Kernel Virtual Machine & Service" at OSDT Beijing (Nov 9, 2019)
- …

3) My Practice

- working on a new eBPF-centric unified system(both kernel space & user space) monitoring, tuning, and debugging toolkit base on LISA
- but reconstruct it from the following aspects:

1. in addition to Ftrace, combining with eBPF support
2. architecture redesign for better support existing Linux Kernel analysis and test utilities
3. integrate eBPF-oriented tools like BCC, Drgn(github.com/osandov/drgn), and kubectl-trace(github.com/iovisor/kubectl-trace)



Source: "drgn: Programmable Debugging", Omar Sandoval (Facebook), Linux Plumbers Conference 2019

4. support hardware debug interface like JTAG and SWD, integrate good on-chip debugger like OpenOCD(http://openocd.org/), and gradually add full CoreSight (https://developer.arm.com/architectures/cpu-architecture/debug-visibility-and-trace/coresight-architecture) support in the future

...

- Pls refer to my presentation "Python for Linux Kernel Debugging" at PyCon China Hangzhou (Oct 19, 2019) for details & preliminary work

XI. Security

Overview

- https://en.wikipedia.org/wiki/Computer_security
- https://en.wikipedia.org/wiki/Information_security
- https://en.wikipedia.org/wiki/Data_security
- …

1) Reference Design

AWS Nitro System

- https://aws.amazon.com/ec2/nitro/

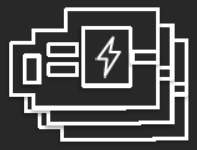
The AWS Nitro System is the underlying platform for our next generation of EC2 instances that enables AWS to innovate faster, further reduce cost for our customers, and deliver added benefits like increased security and new instance types.

AWS has completely re-imagined our virtualization infrastructure. Traditionally, hypervisors protect the physical hardware and bios, virtualize the CPU, storage, networking, and provide a rich set of management capabilities. With the Nitro System, we are able to break apart those functions, offload them to dedicated hardware and software, and reduce costs by delivering all of the resources of a server to your instances.

With the Nitro System, we shipped nearly 3x as many new instances in 2018 versus the prior year.

Nitro in three parts

Nitro Cards



VPC Networking
Amazon Elastic Block Store (Amazon EBS)
Instance Storage
System Controller

Nitro Security Chip

Integrated into motherboard
Protects hardware resources
Hardware Root of Trust

Nitro Hypervisor

Lightweight hypervisor
Memory and CPU allocation
Bare Metal-like performance

ENA PCIe Controller

VPC Data Plane

NVMe PCIe Controller

EBS Data Plane

NVMe PCIe Controller

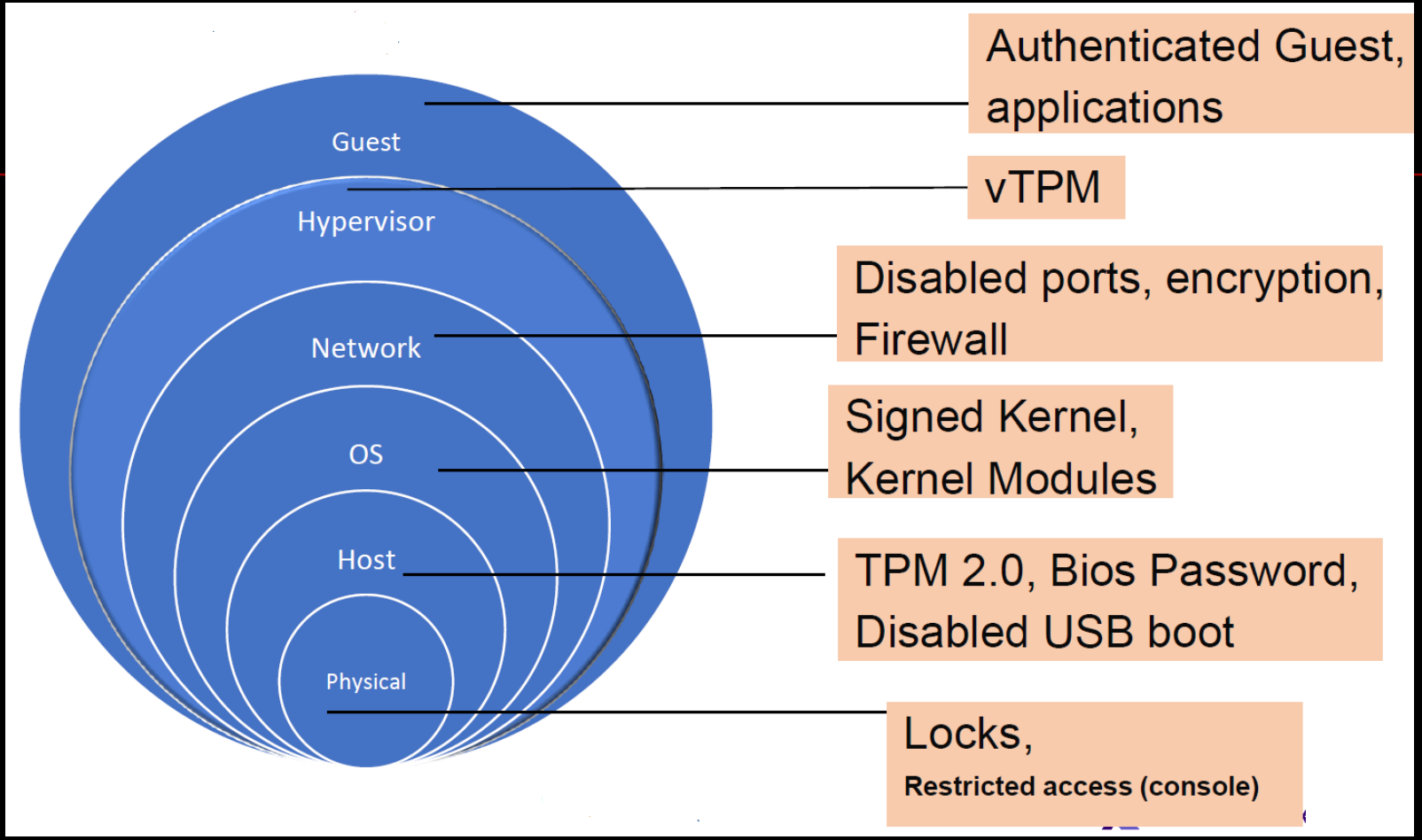
Transparent Encryption

System Control

Root of Trust

Source: "Powering Next Gen EC2 Instances--Deep Dive into the Nitro System", Anthony Liguori, AWS re:Invent 2018

Security at Edge



Source: "StarlingX Introduction ", EdgeX Foundry, Akraino Edge Stack Summit 2018

2) Hardware-Software Co-designed System Security

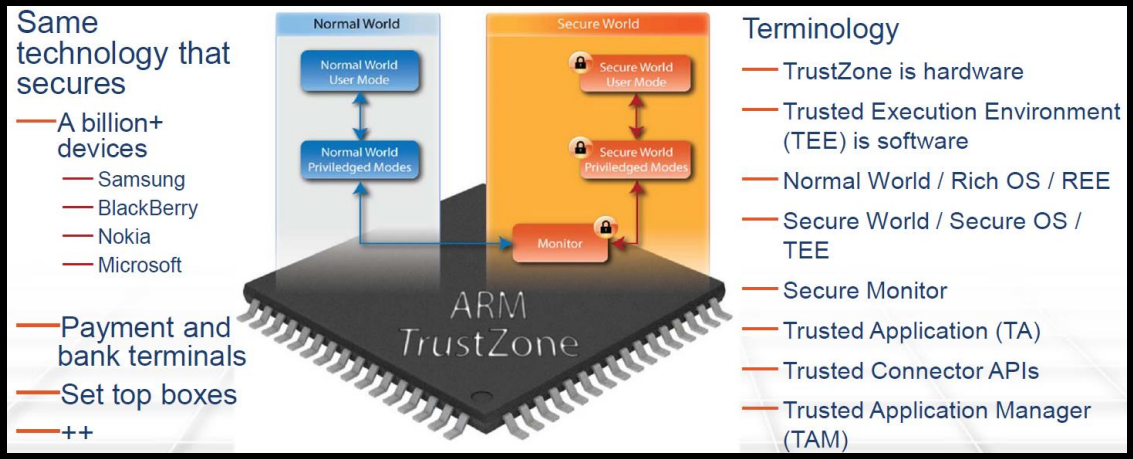
- Hardware-Software Co-designed System and System Security is the trend in the real world

2.1 ARM for Security

- https://www.arm.com/products/silicon-ip-security

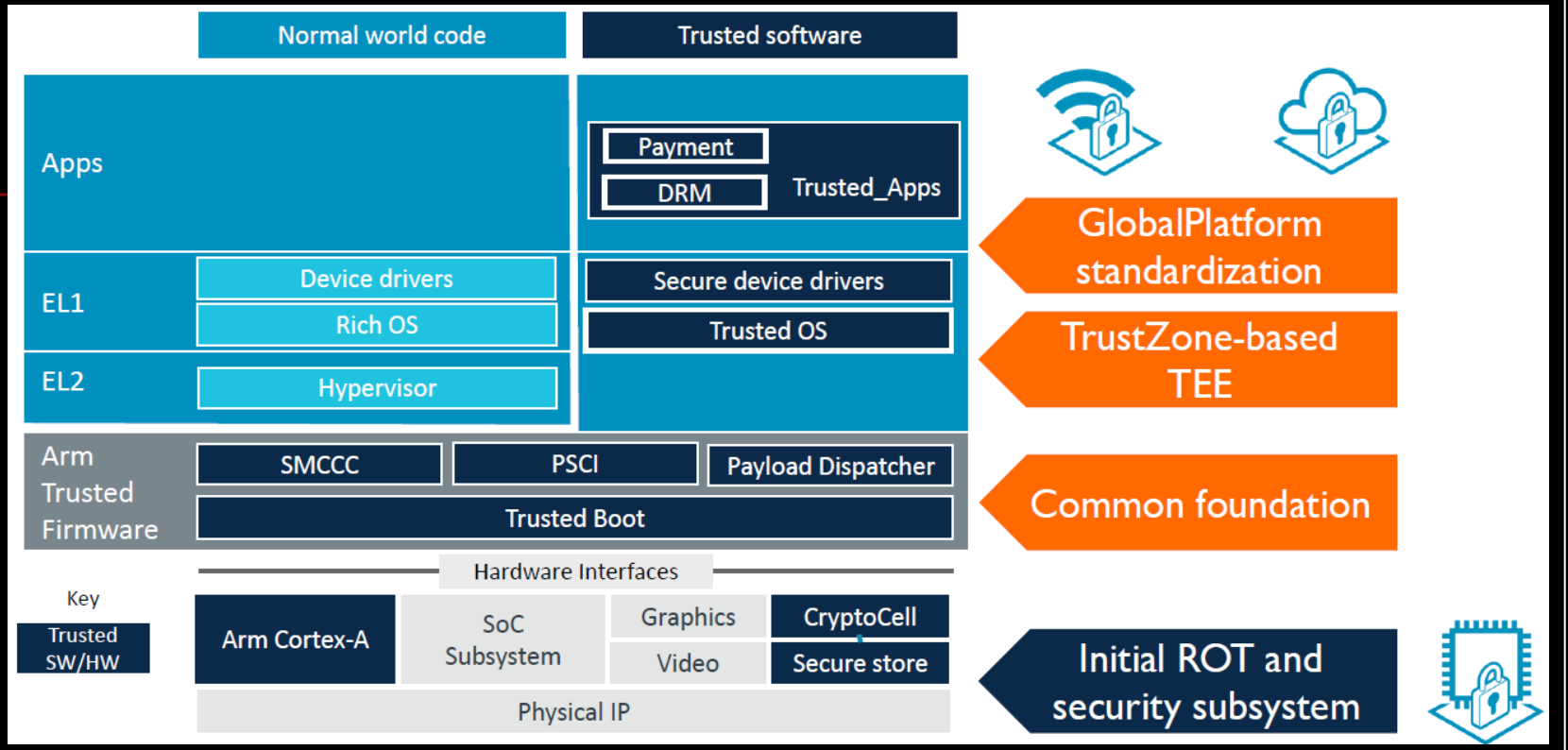
Trust Zone

- https://developer.arm.com/ip-products/security-ip/trustzone
  On-chip security enclave that provides hardware isolation and protection for sensitive material such as cryptographic keys, algorithms and data



- Source: "EASING ACCESS TO ARM TRUSTZONE -- OP-TEE AND RASPBERRY PI 3", Sequitur Labs Inc , Linsro Connect LAS2016
- ...

ARM TrustZone based TEE Architecture



Source: "Beyond TrustZone", Rob Coombs, ARM Tech Symposia 2017(Taipei)

PSA (Platform Security Architecture)

- https://www.arm.com/why-arm/architecture/platform-security-architecture
- Beyond TrustZone - Security Enclaves

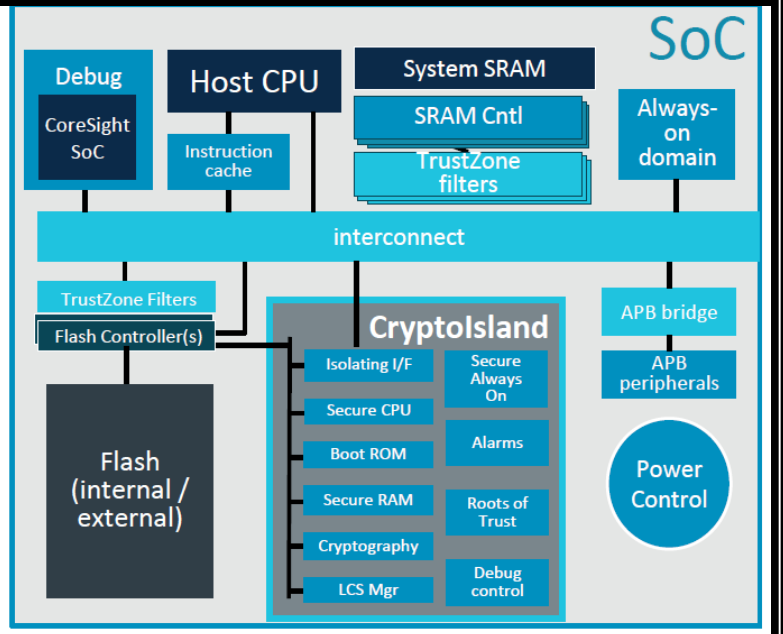
A programmable security enclave to extend fixed function CryptoCell family.

TrustZone CryptoIslands - an additional family of security solutions by Arm.

Aimed at providing on-die security services, in a physically isolated manner (host CPU agnostic).

Axiom: less sharing of resources leads to smaller attack surface and fewer vulnerabilities.

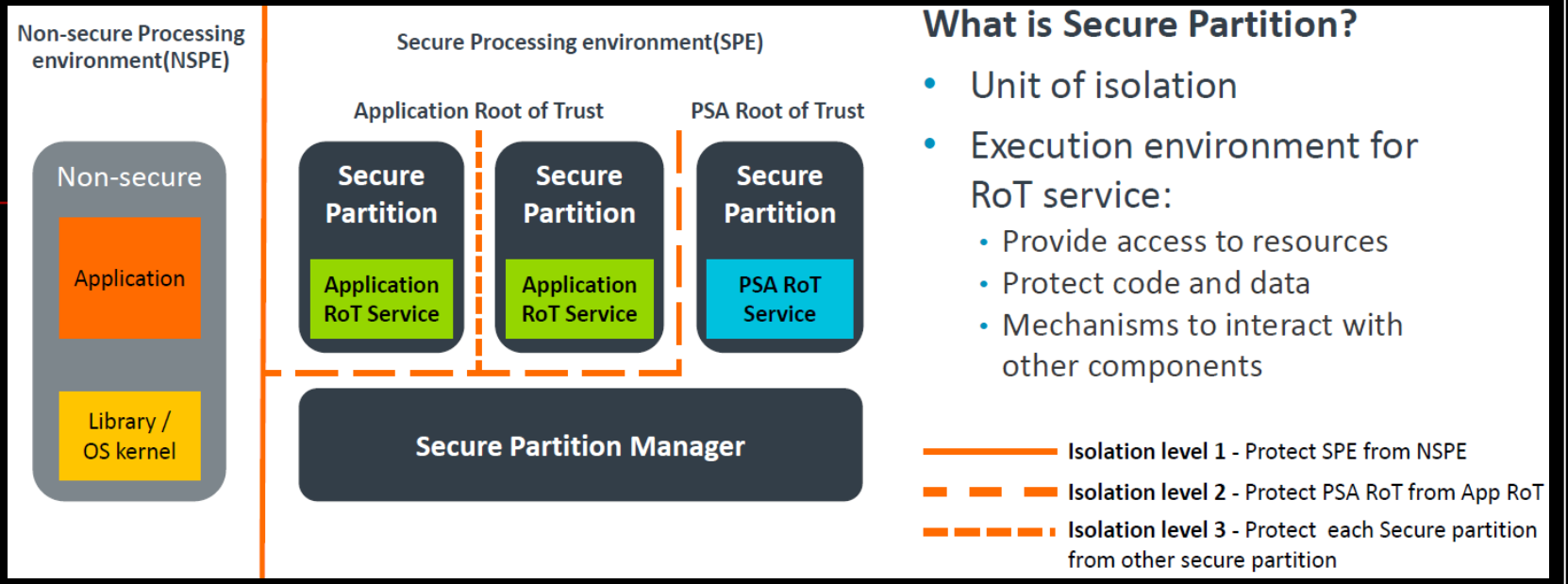
Certification, at a reasonable cost (i.e. reuse).



Source: "Beyond TrustZone", Rob Coombs, ARM Tech Symposia 2017(Taipei)

- https://www.arm.com/products/silicon-ip-security/cryptoisland
- …

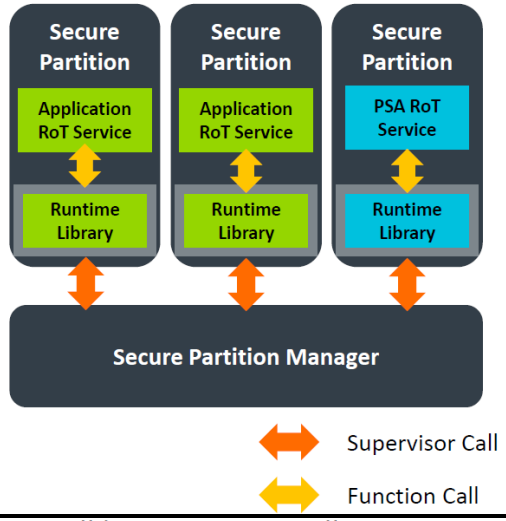
PSA Firmware Framework



Source: "Secure Partition Runtime Library on IoT Device", Ken Liu & Edison Ai, Linaro Connect 2019(San Diego)

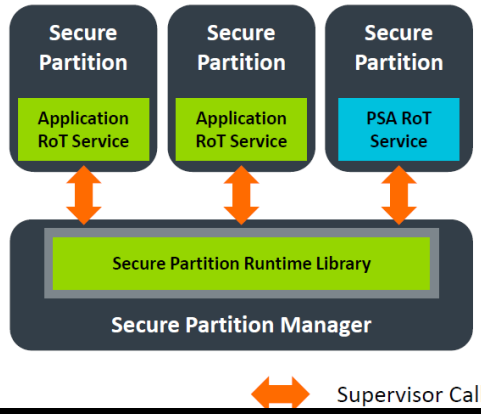
Placement of the Secure Partition Runtime Library

Option 1: Per-Partition Library



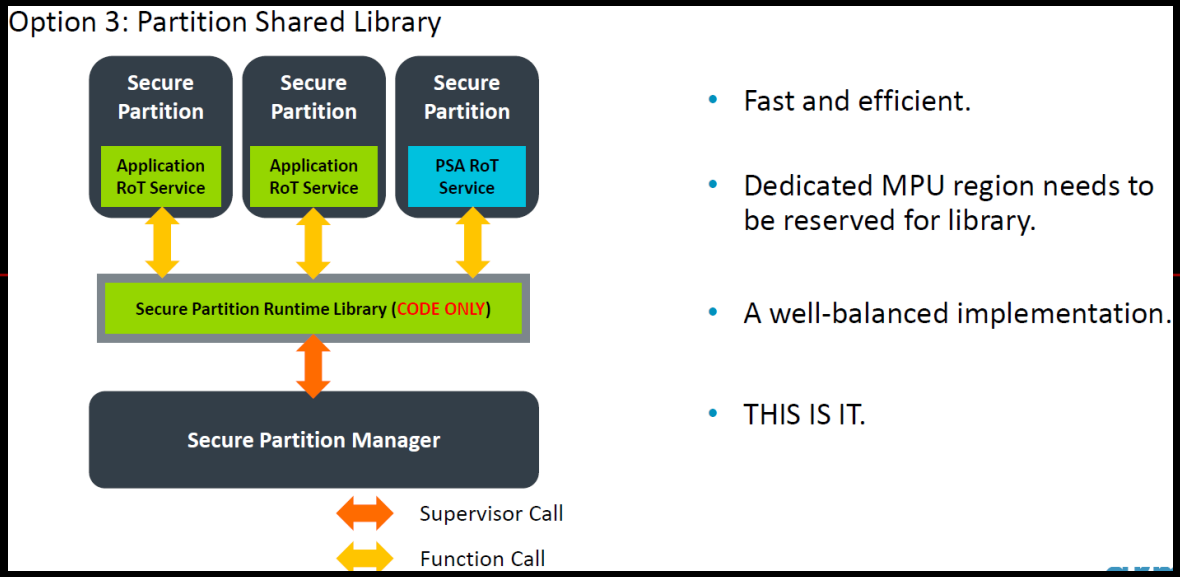
- Image size increases much.
- Hard to put into a single loaded Image.

Option 2: All by Supervisor Call



- SPM consume more execution time.
- Library execution cannot be preempted.
- Library code runs under unnecessary privileged level.

Source: "Secure Partition Runtime Library on IoT Device", Ken Liu & Edison Ai, Linaro Connect 2019(San Diego)



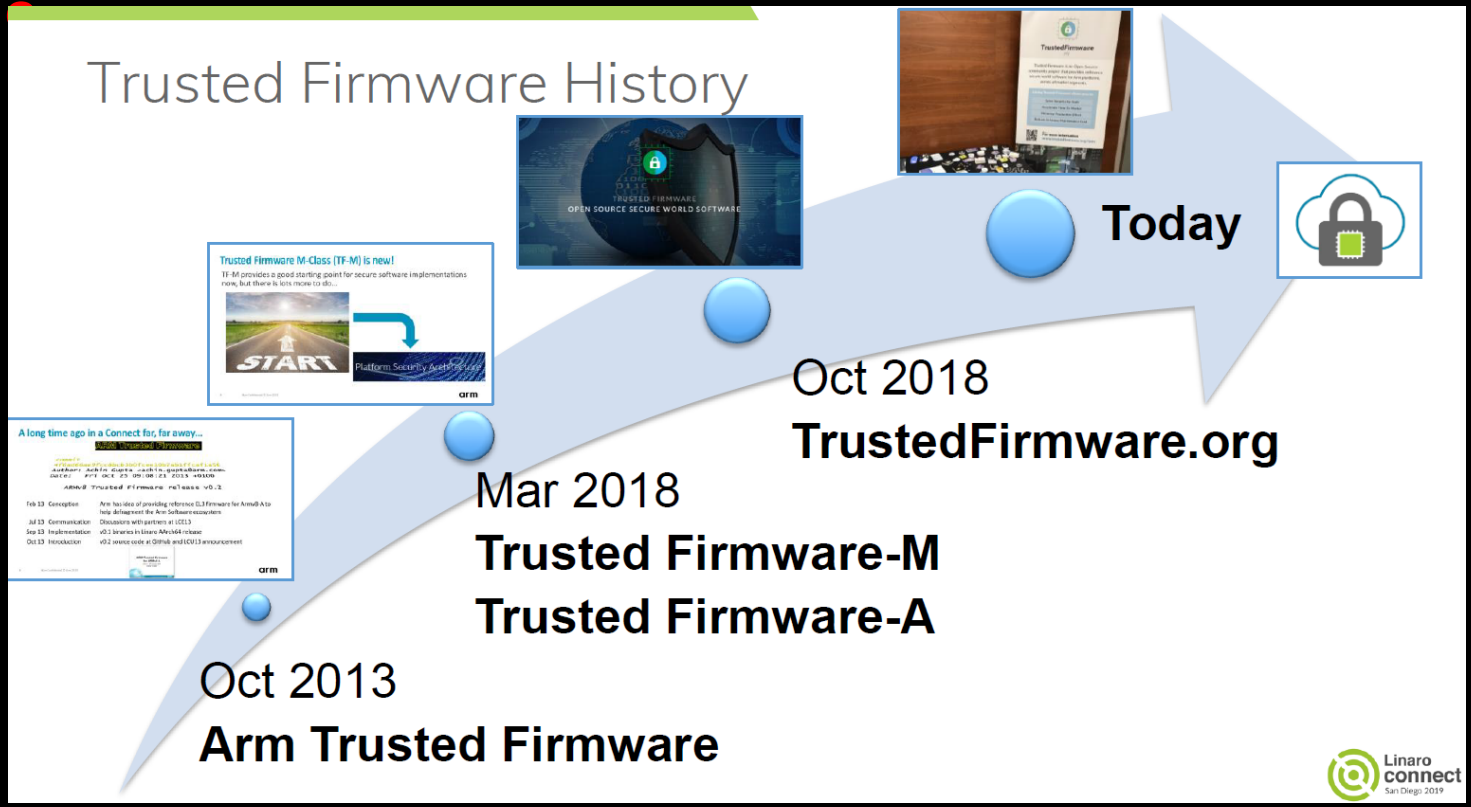
Source: "Secure Partition Runtime Library on IoT Device", Ken Liu & Edison Ai, Linaro Connect 2019(San Diego)

- https://git.trustedfirmware.org/trusted-firmwarem.git/tree/docs/design_documents/tfm_secure_partition_runtime_library.rst
- https://git.trustedfirmware.org/trusted-firmware-m.git/

Trusted Firmware

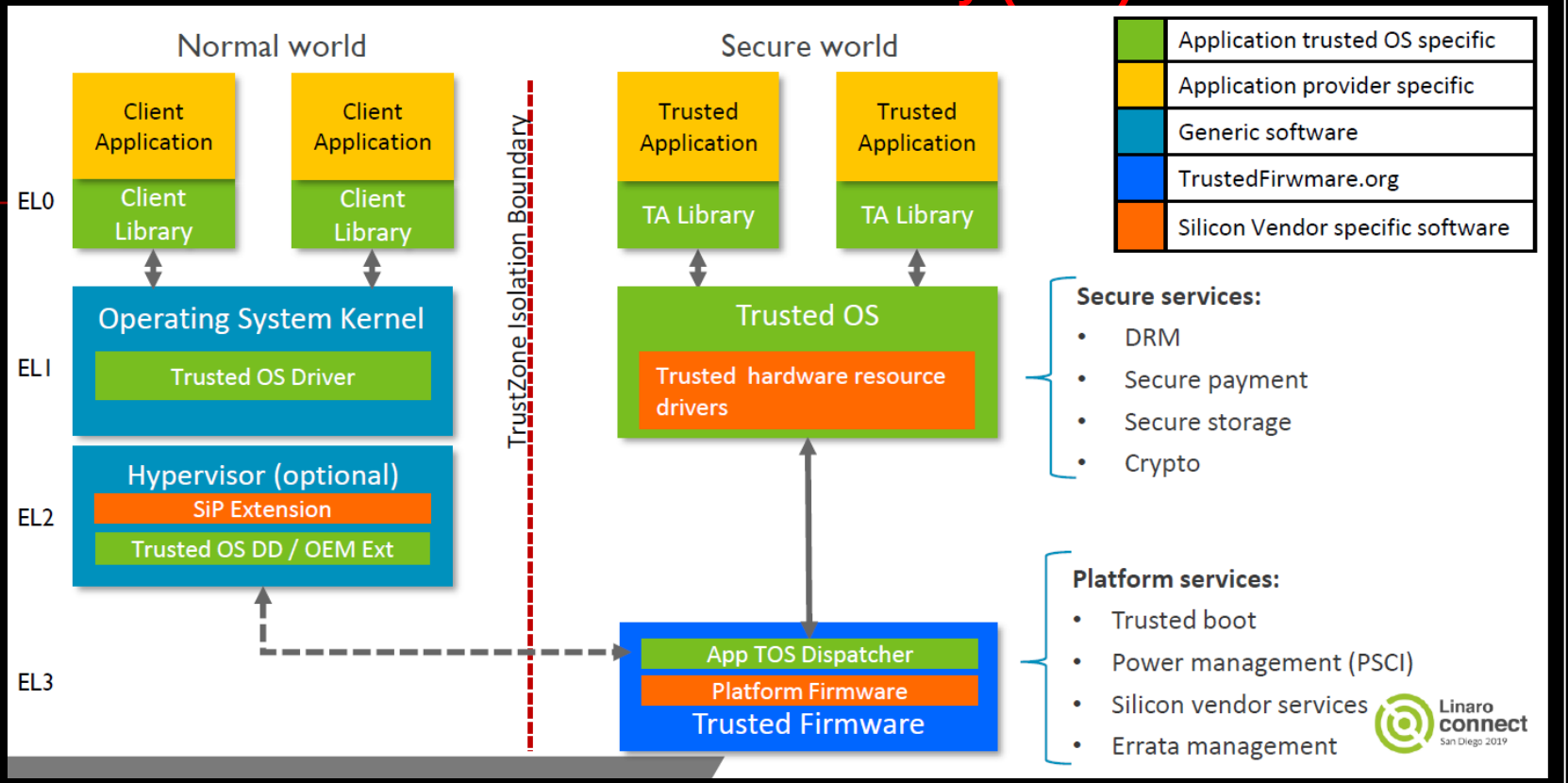
- https://www.trustedfirmware.org/

Trusted Firmware provides a reference implementation of secure world software for Armv8-A and Armv8-M. It provides SoC developers and OEMs with a reference trusted code base complying with the relevant Arm specifications. The code on this website is the preferred implementation of Arm specifications, allowing quick and easy porting to modern chips and platforms. This forms the foundations of a Trusted Execution Environment (TEE) on application processors, or the Secure Processing Environment (SPE) of microcontrollers.



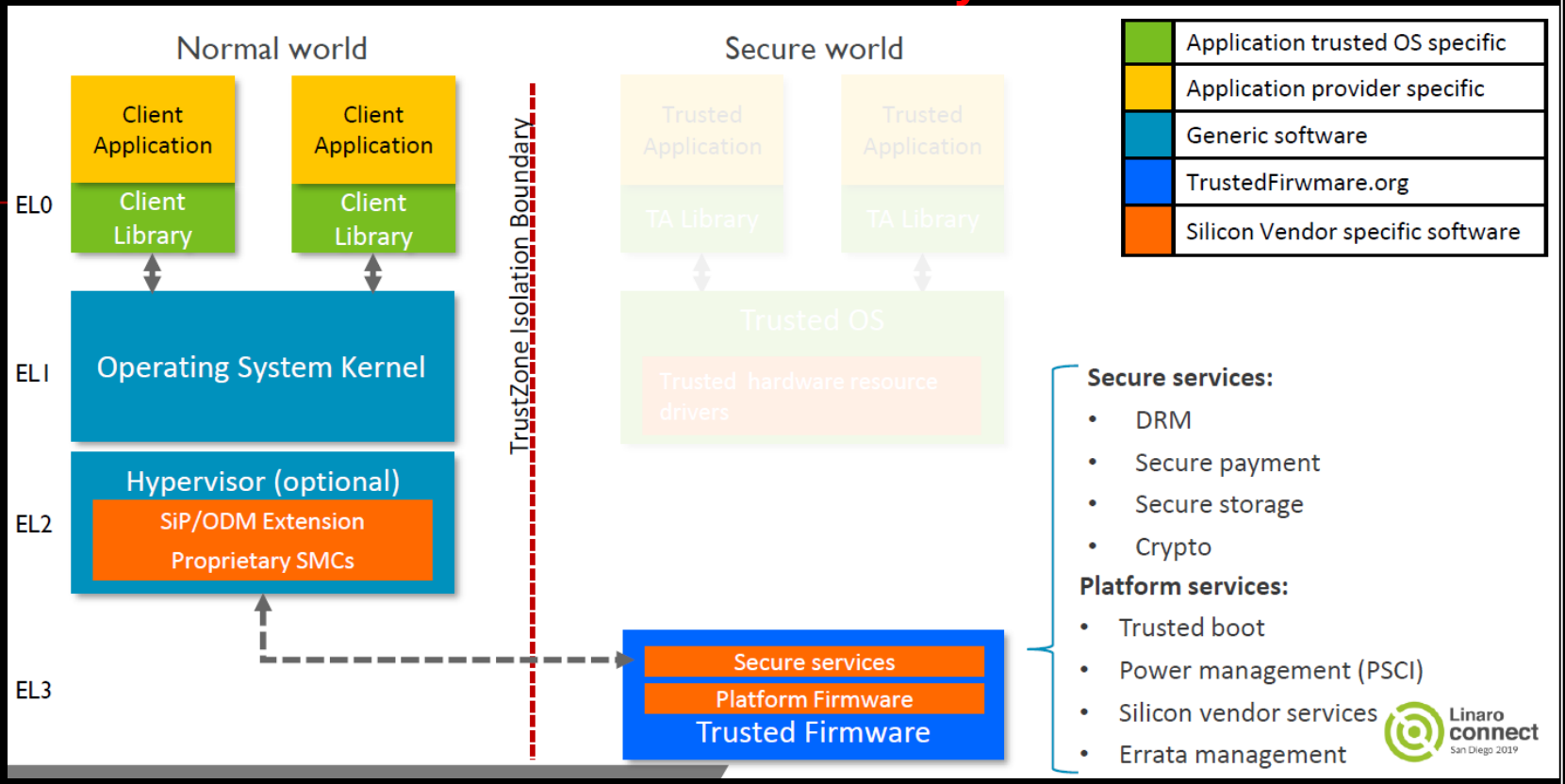
Source: "TrustedFirmware.org Project update", Matteo Carlini & Shebu V. Kuriakose, Linaro Connect 2019(San Diego)

Secure world software architecture today (TOS)



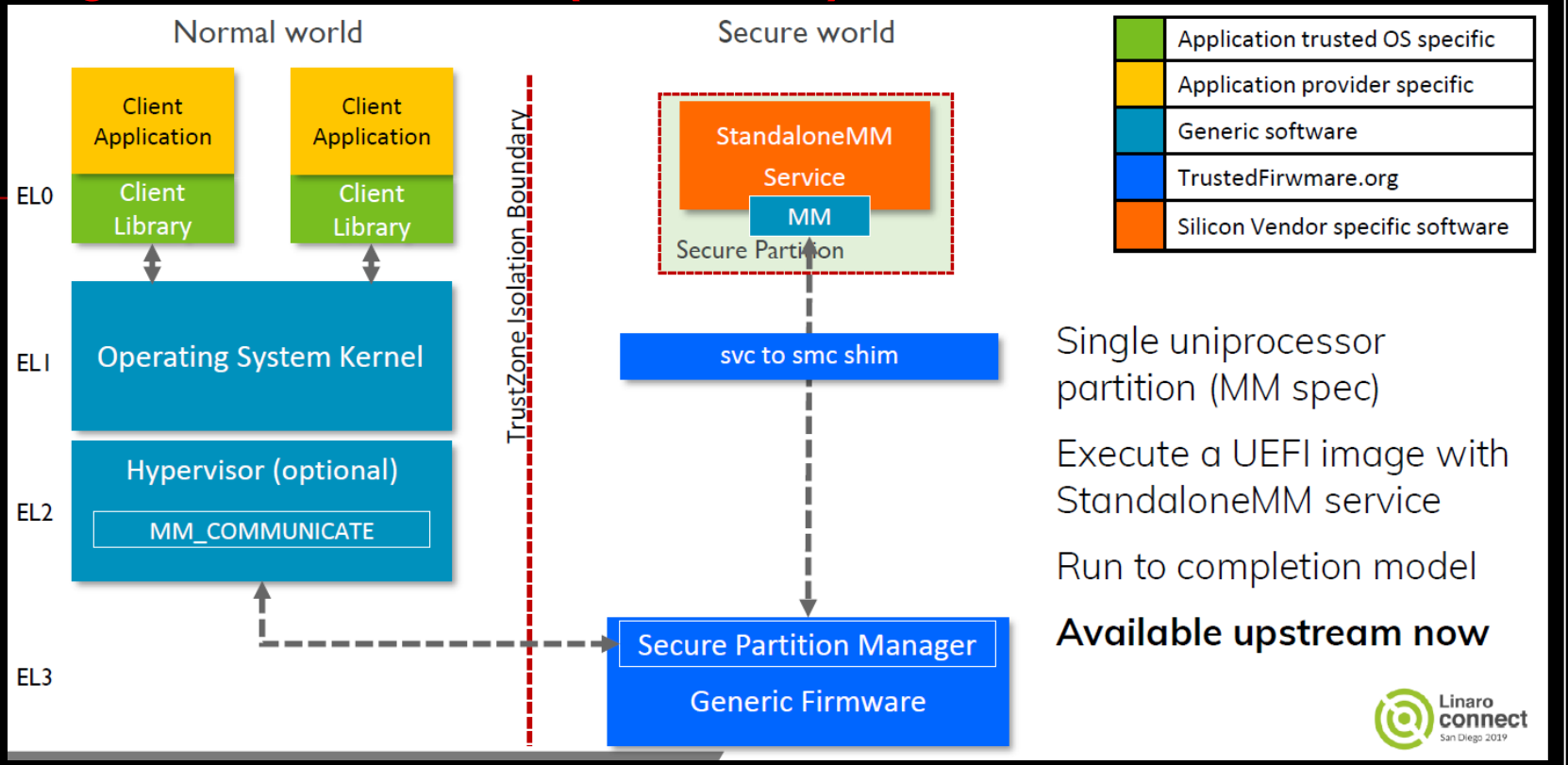
Source: "TrustedFirmware.org Project update", Matteo Carlini & Shebu V. Kuriakose, Linaro Connect 2019(San Diego)

Secure world software architecture today



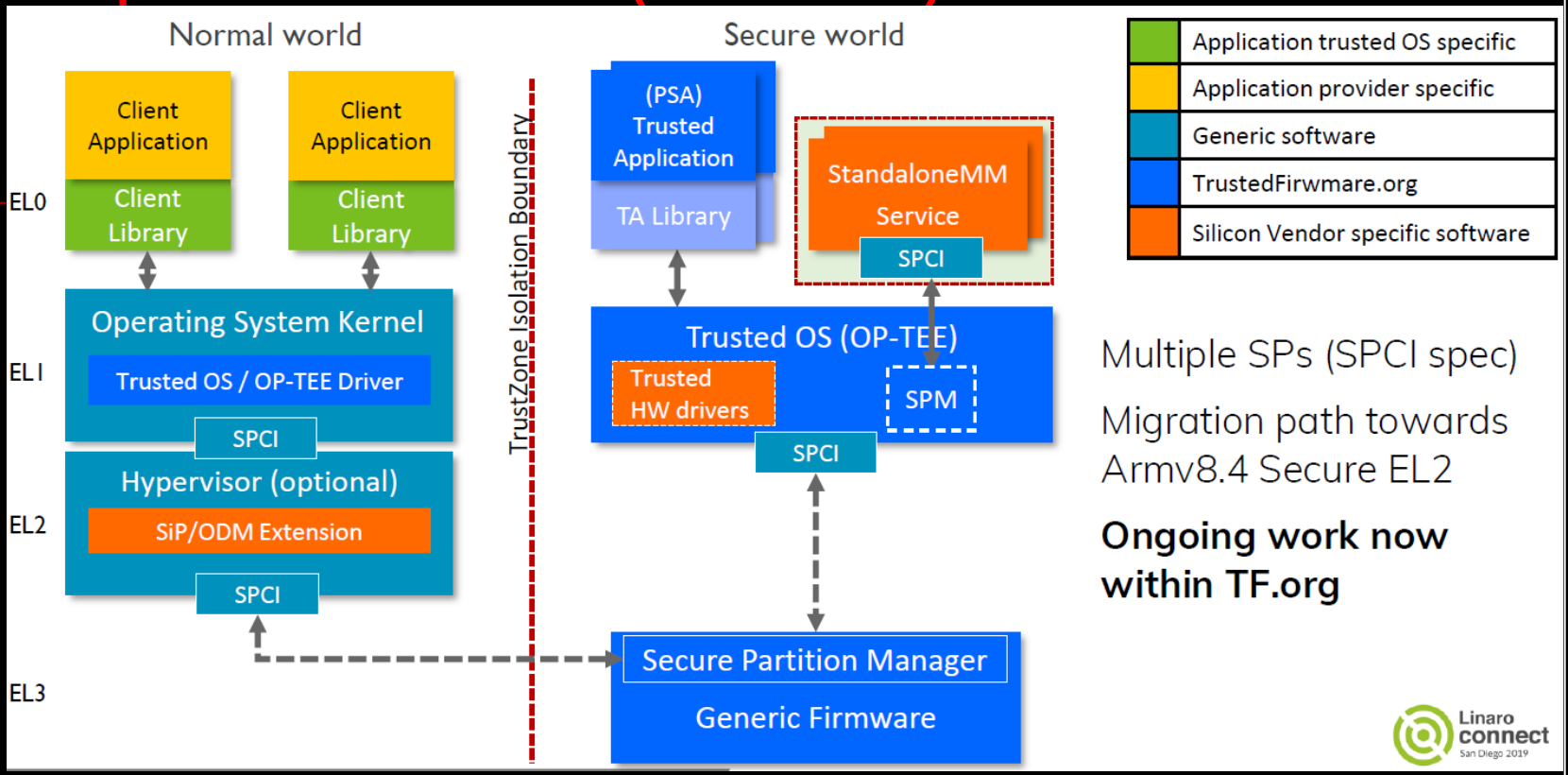
Source: "TrustedFirmware.org Project update", Matteo Carlini & Shebu V. Kuriakose, Linaro Connect 2019(San Diego)

Single Secure Partition (MM-based)



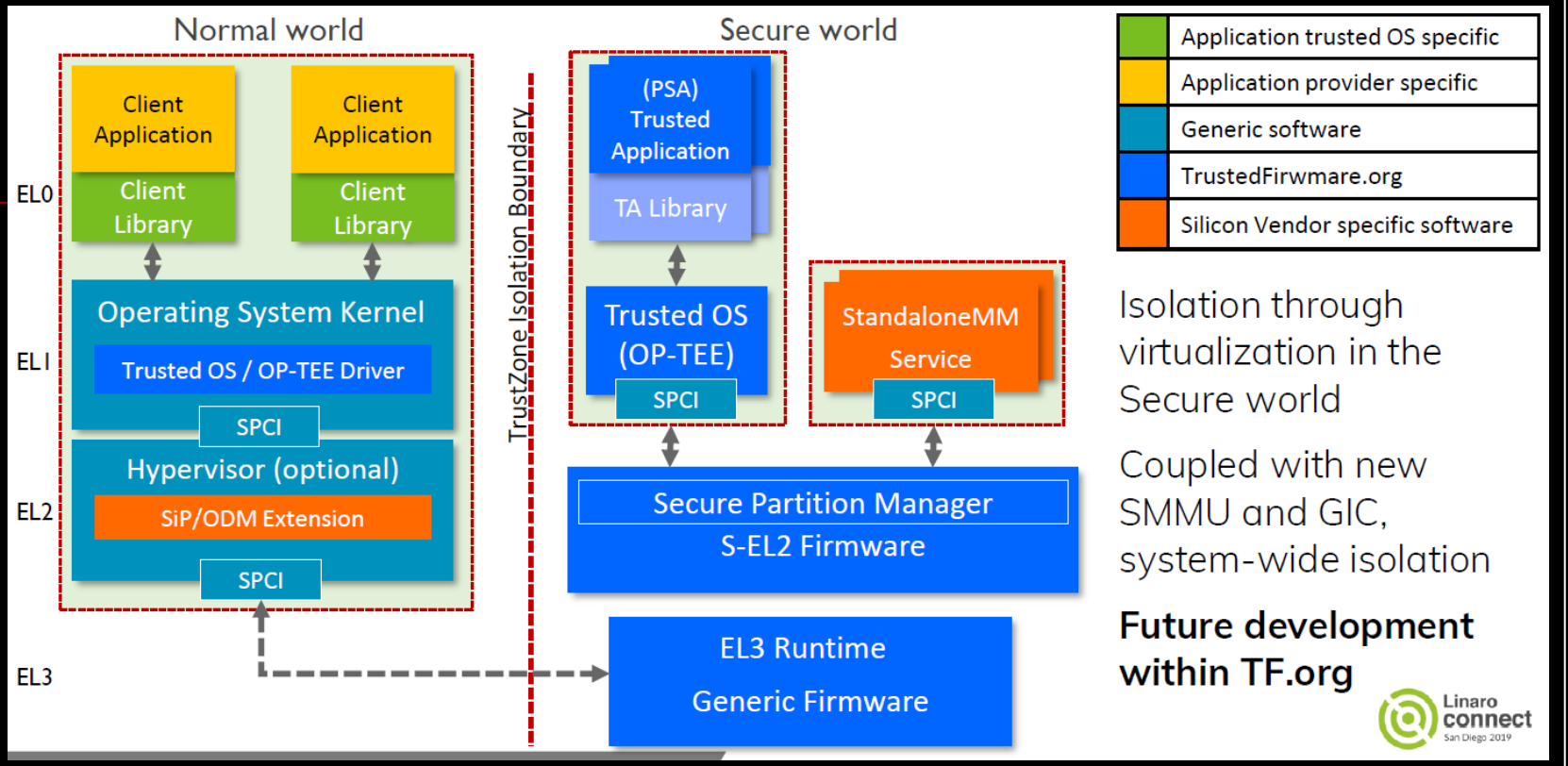
Source: "TrustedFirmware.org Project update", Matteo Carlini & Shebu V. Kuriakose, Linaro Connect 2019(San Diego)

Multiple Secure Partitions (SPCI-based)

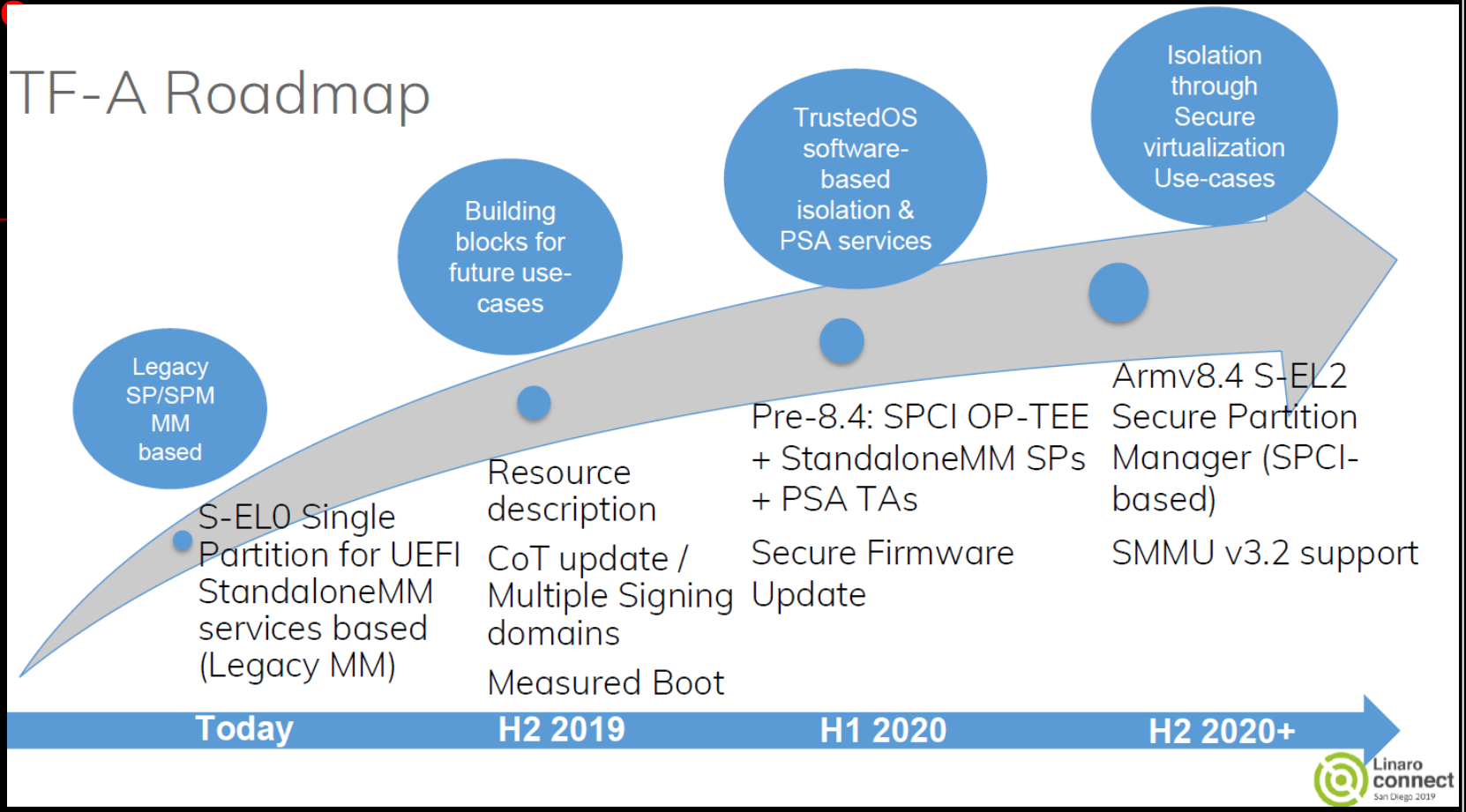


Source: "TrustedFirmware.org Project update", Matteo Carlini & Shebu V. Kuriakose, Linaro Connect 2019(San Diego)

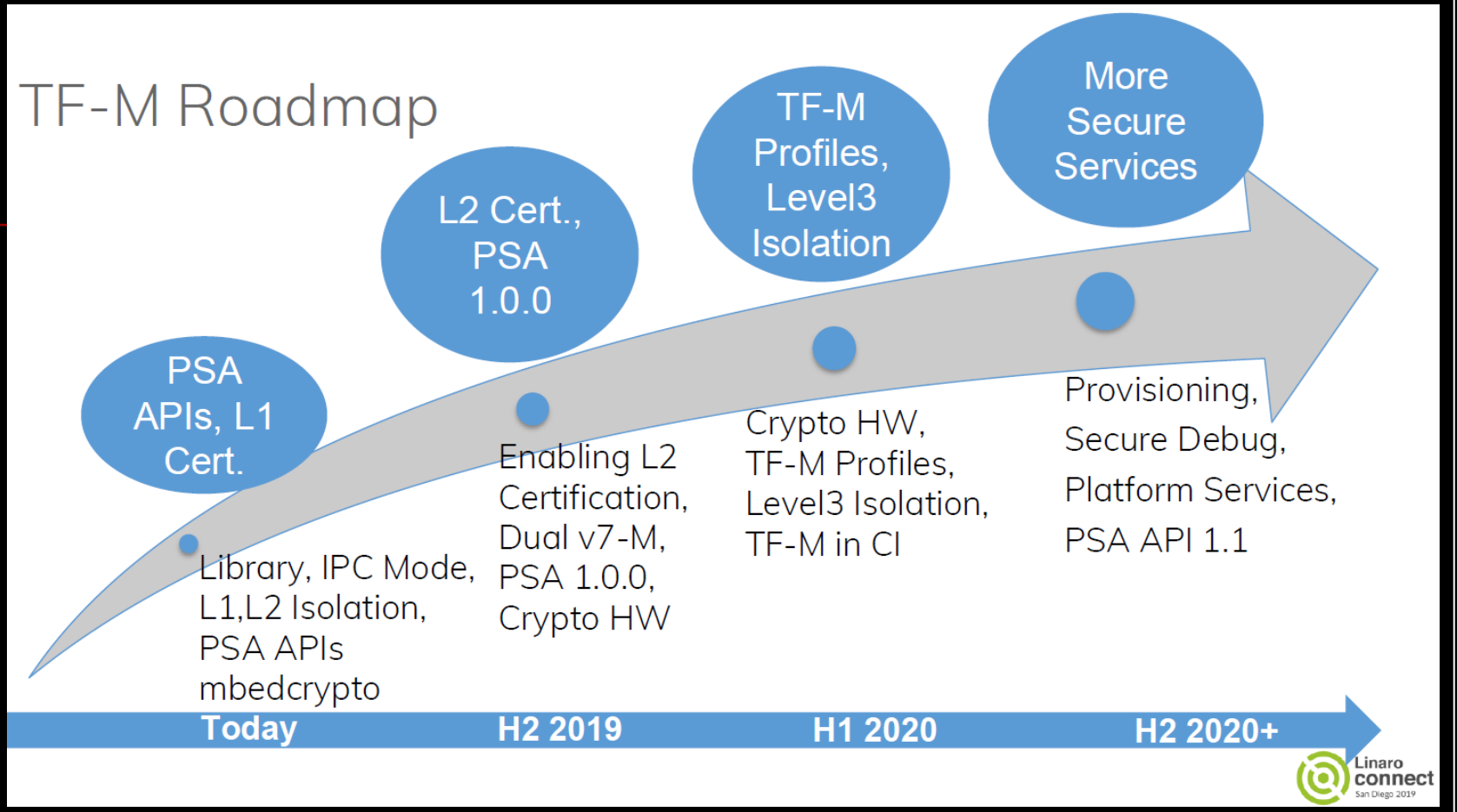
Armv8.4 Secure EL2 Virtualization extension



Source: "TrustedFirmware.org Project update", Matteo Carlini & Shebu V. Kuriakose, Linaro Connect 2019(San Diego)



Source: "TrustedFirmware.org Project update", Matteo Carlini & Shebu V. Kuriakose, Linaro Connect 2019(San Diego)



Source: "TrustedFirmware.org Project update", Matteo Carlini & Shebu V. Kuriakose, Linaro Connect 2019(San Diego)

Source: "TrustedFirmware.org Project update", Matteo Carlini & Shebu V. Kuriakose, Linaro Connect 2019(San Diego)

2.2 OpenEnclave

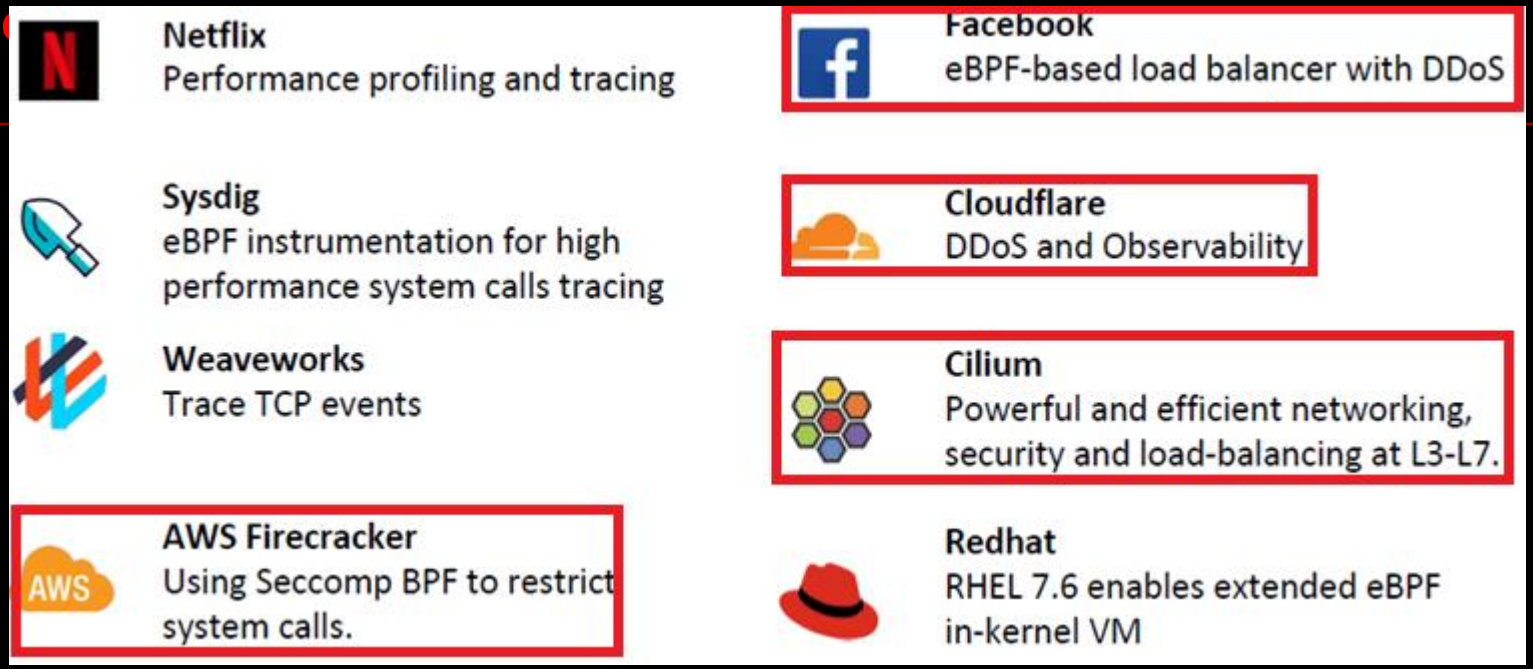
- https://openenclave.io/sdk/
- https://github.com/openenclave
- Open Enclave (OE) is an SDK for building enclave applications in C and C++. An enclave application partitions itself into two components:

  1. An untrusted component (called the host) and
  2. A trusted component (called the enclave).

  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.

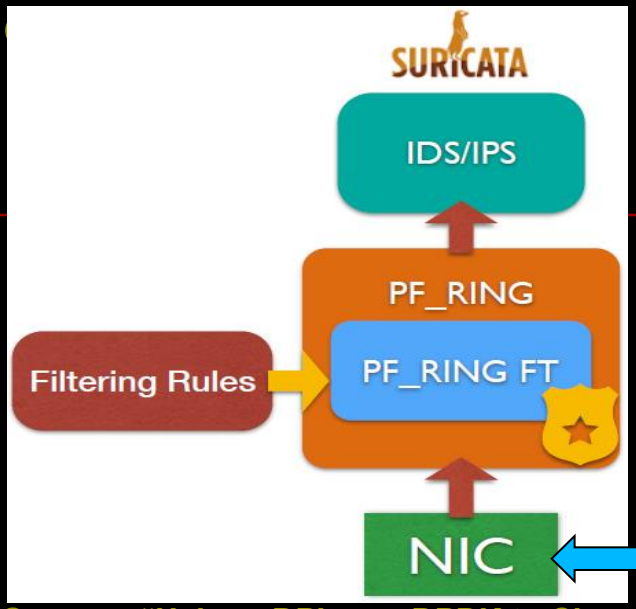
3) eBPF is playing a more and more important role in Linux system security

- 

  Source: "Back to the future with eBPF", Beatriz Martínez Rubio, KubeCon Europe 2019

- …

eBPF-based IDS



or a SmartNIC

Source: "Using nDPI over DPDK to Classify and Block Unwanted Network Traffic", Luca Deri, DPDK Summit NA 2018

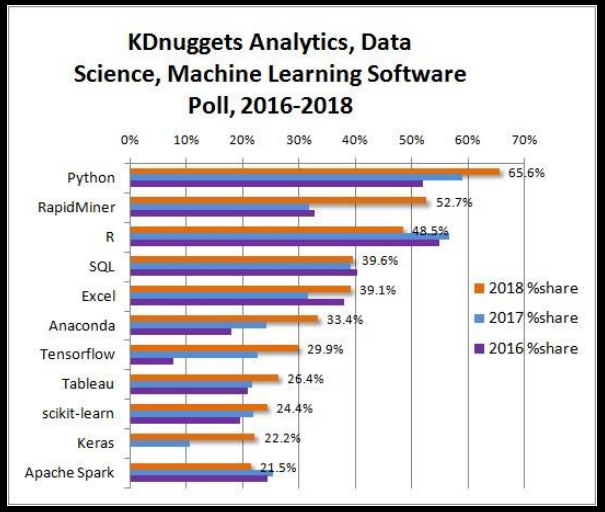
- https://github.com/ntop/nDPI
  //Open Source Deep Packet Inspection Software Toolkit
- https://suricata-ids.org/
- a network intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM)
- https://lwn.net/Articles/737771 Using eBPF and XDP in Suricata
- http://suricata.readthedocs.io/en/latest/capture-hardware/ebpf-xdp.html?highlight=XDP

XII. Miscs

1) New Python Runtime

Why is it

- https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html



- Important Python Projects for HCI

  Build: Meson, SCons… DevOps: Ansible, SaltStack…

  AI: Keras, Scikit-Learn… Data Analyst: PyData, PySpark…

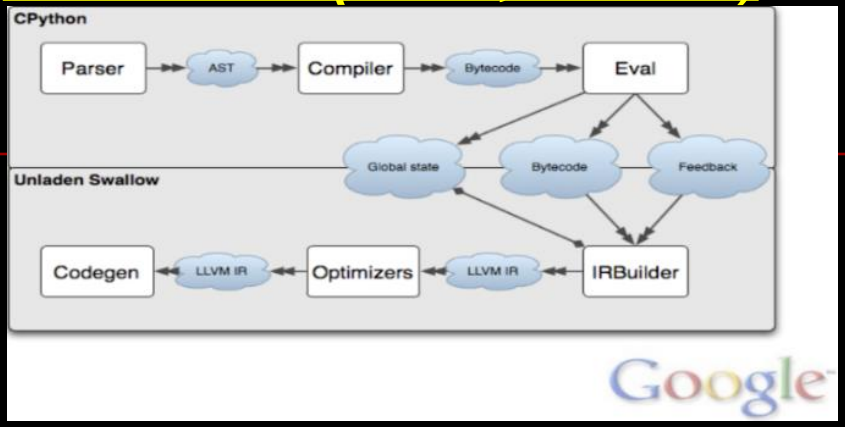
  Cloud/DataCenter: OpenStack, Airship(partially), Apache Airflow…

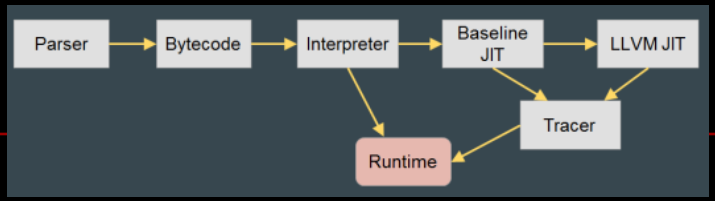
  Kubernetes Operator Pythonic Framework (Kopf)

  Security: a swiss knife for hackers…

1.1 Previous Methods

LLVM-based (VMKit, MCJIT…)

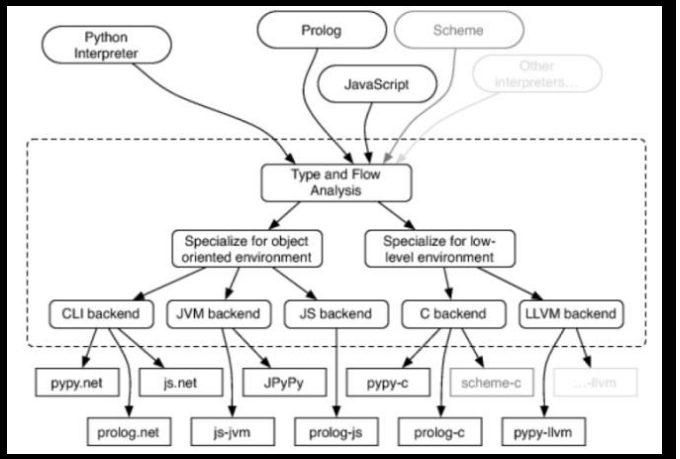


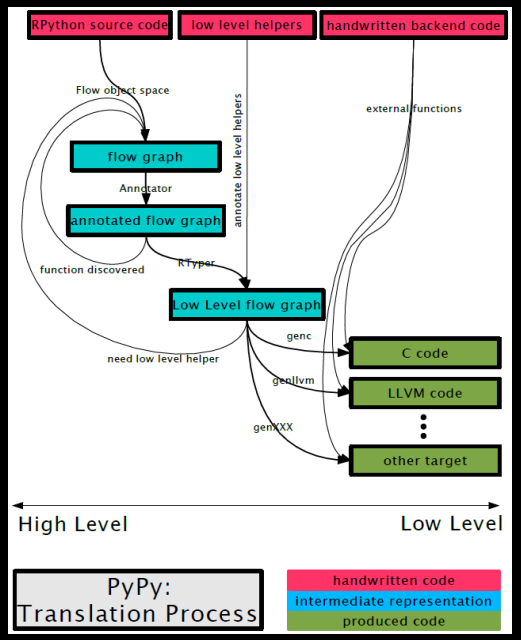


PySton Dropbox

Src2Src

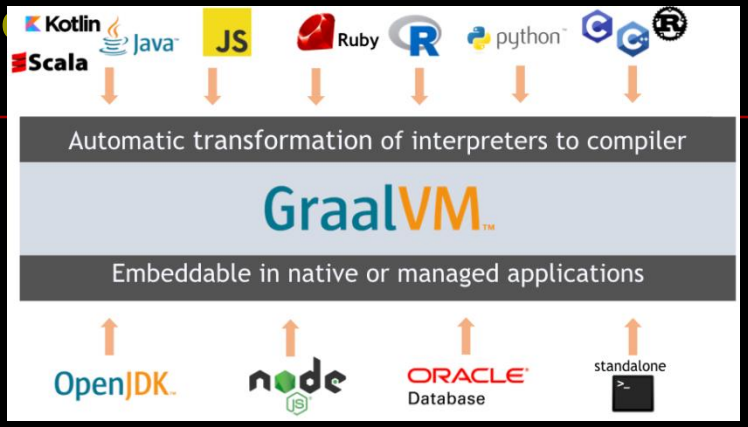
RPython + Meta-tracing





1.2 New Ideas

GraalVM

- 

RustPython

- https://github.com/RustPython 

JITs

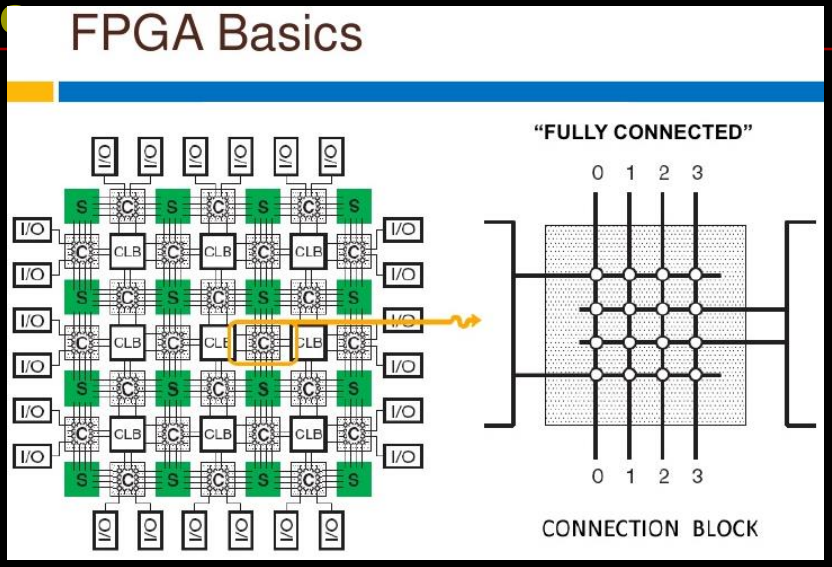
- https://github.com/thautwarm/restrain-jit
- https://github.com/Microsoft/Pyjion
- …

BYOD

- The initial demo of my own solution DPython will come in next year…

2) FPGA

- https://en.wikipedia.org/wiki/Field-programmable_gate_array
- https://en.wikipedia.org/wiki/FPGA_prototyping
-



- ASIC vs GPU vs FPGA as AI accelerator

2.1 Next Steps in our new development cluster

Networking

- FPGA-based SmartNIC prototyping
- FPGA-based programmable smart switch
- …

AI

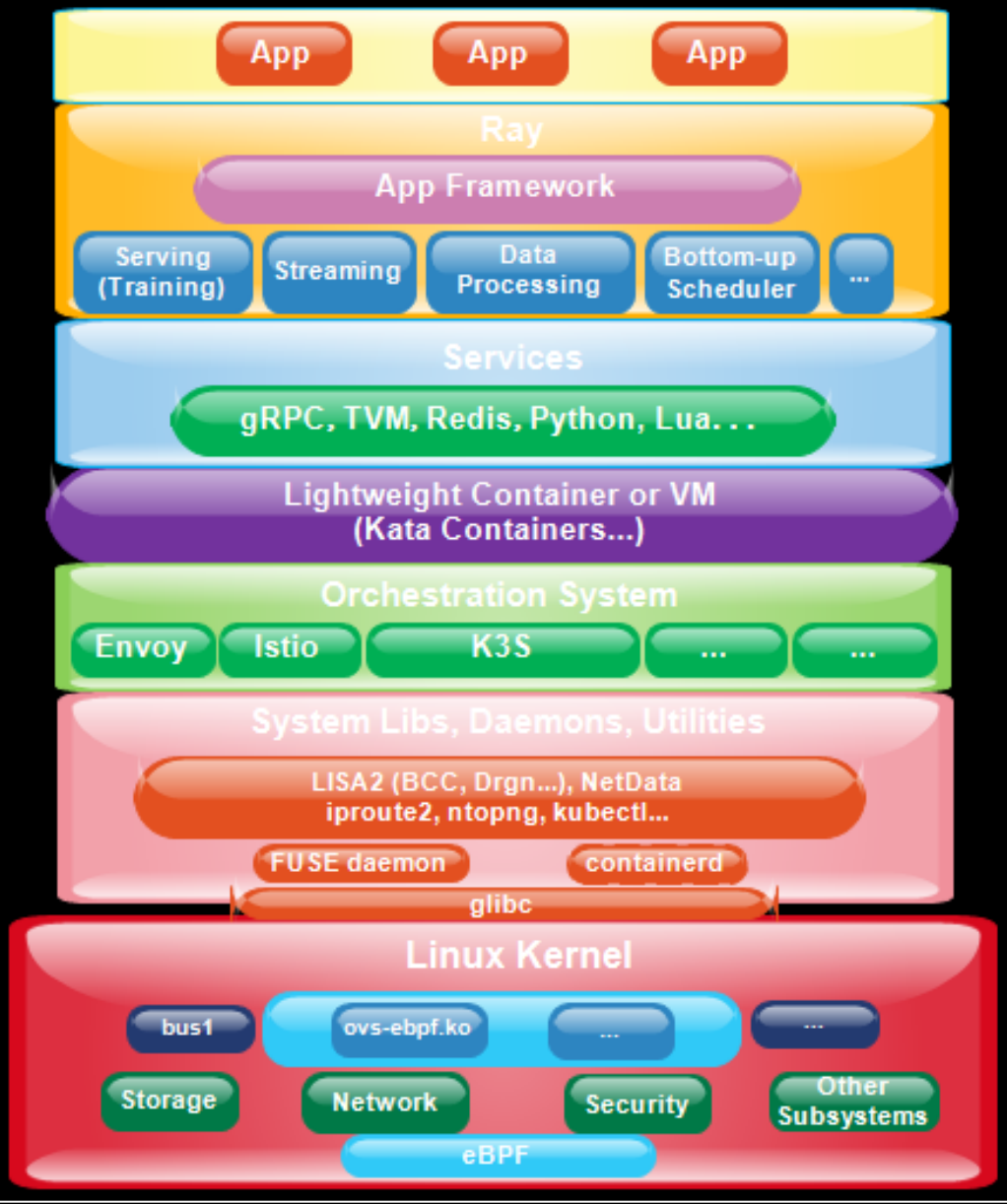
- FPGA-based AI accelerator
- …

XIII. eBPF-centric New Design

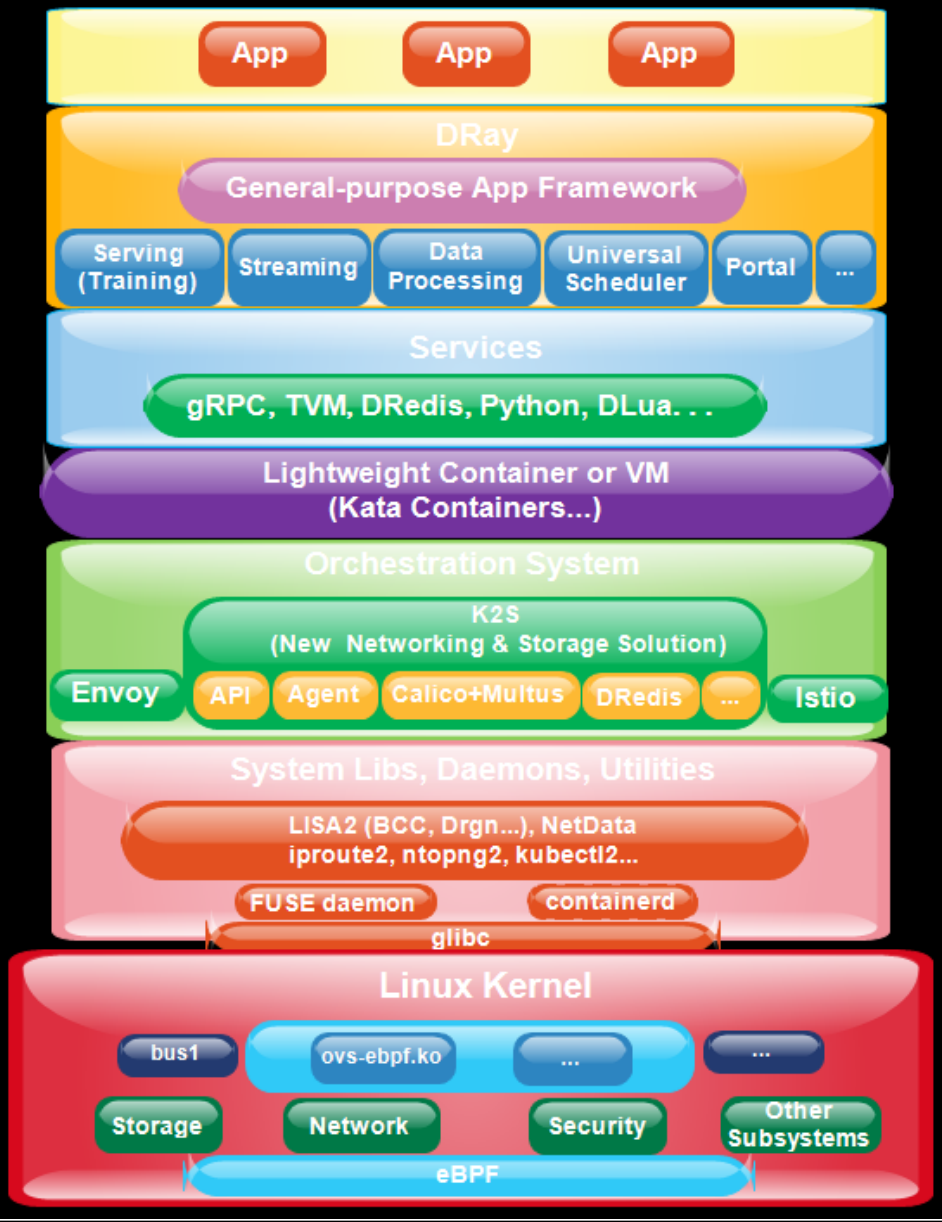
1) System Architecture

- an eBPF-centric new lightweight solution to meet our design goals & principles that described in section II, which is going to be divided into four stages

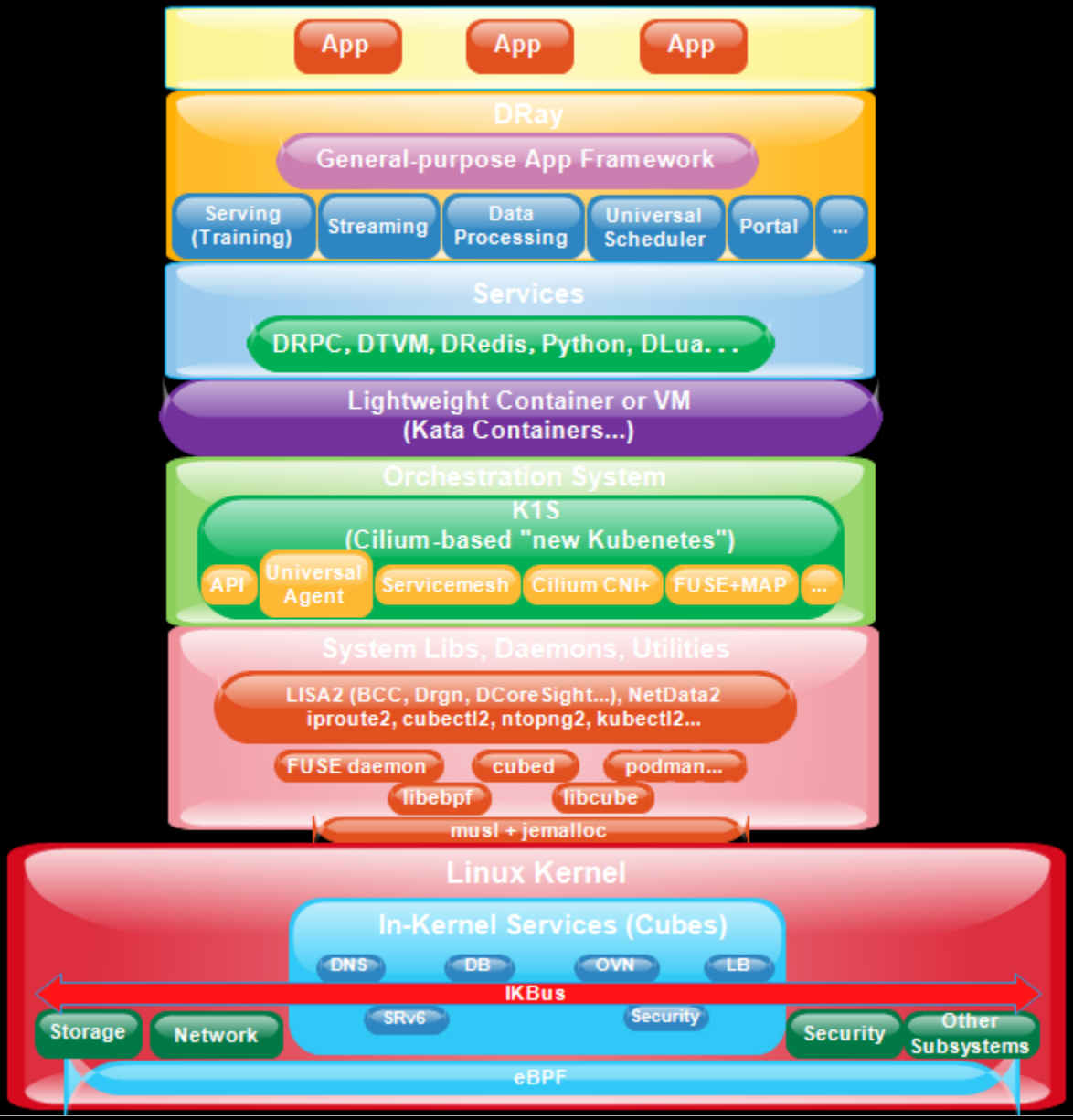
Stage 1


App
App
App
Ray
App Framework
Serving (Training)
Streaming
Data Processing
Bottom-up Scheduler
...
Services
gRPC, TVM, Redis, Python, Lua. . .
Lightweight Container or VM
(Kata Containers...)
Orchestration System
Envoy
Istio
K3S
...
...
System Libs, Daemons, Utilities
LISA2 (BCC, Drgn...), NetData
iproute2, ntopng, kubectl...
FUSE daemon
containerd
glibc
Linux Kernel
bus1
ovs-ebpf.ko
...
...
Storage
Network
Security
Other Subsystems
eBPF

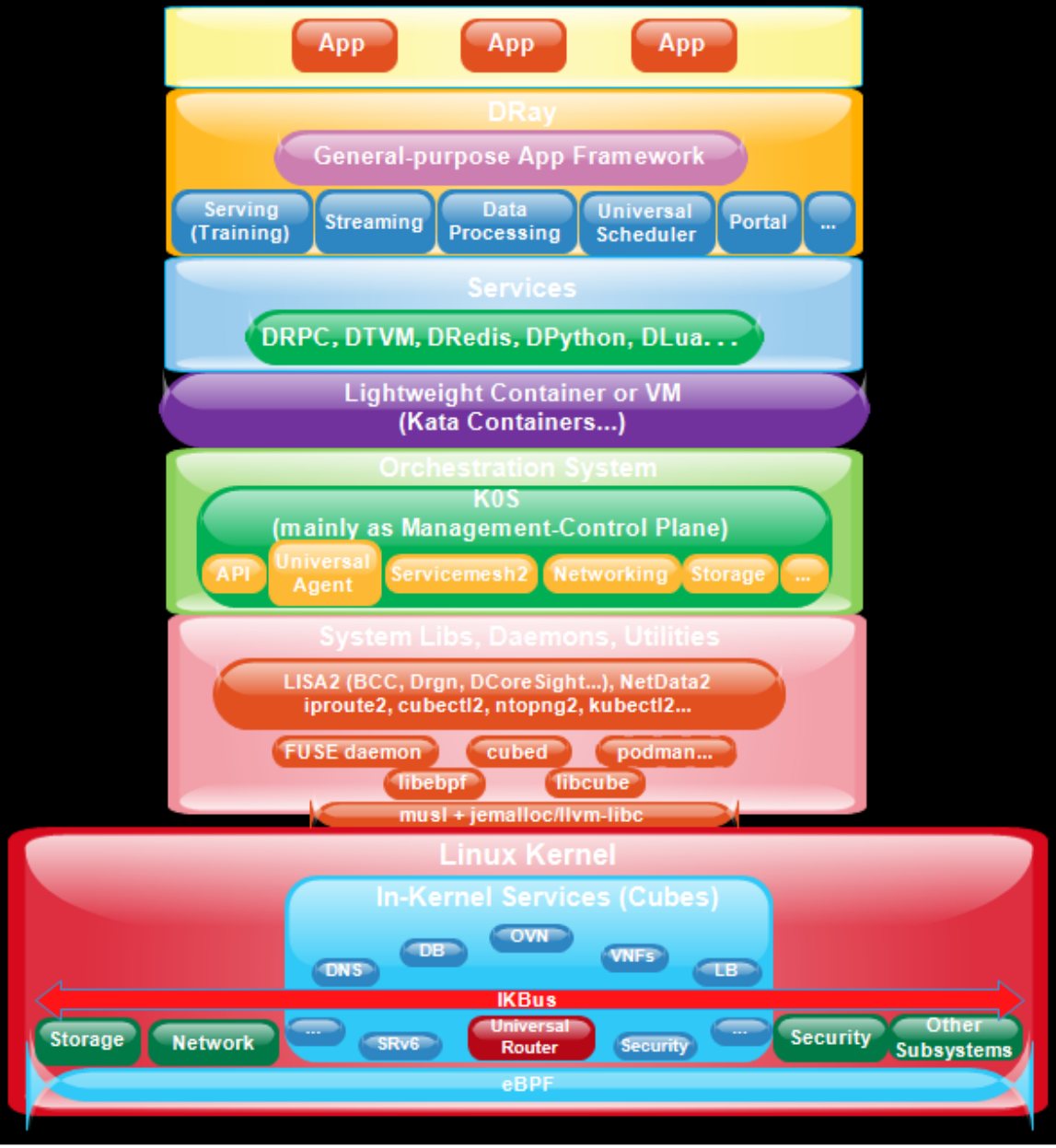
Stage 2


App
App
App
DRay
General-purpose App Framework
Serving (Training)
Streaming
Data Processing
Universal Scheduler
Portal
...
Services
gRPC, TVM, DRedis, Python, DLua. . .
Lightweight Container or VM
(Kata Containers...)
Orchestration System
K2S
(New Networking & Storage Solution)
Envoy
API
Agent
Calico+Multus
DRedis
...
Istio
System Libs, Daemons, Utilities
LISA2 (BCC, Drgn...), NetData
iproute2, ntopng2, kubectl2...
FUSE daemon
containerd
glibc
Linux Kernel
bus1
ovs-ebpf.ko
...
...
Storage
Network
Security
Other Subsystems
eBPF

Stage 3


App
App
App
DRay
General-purpose App Framework
Serving (Training)
Streaming
Data Processing
Universal Scheduler
Portal
...
Services
DRPC, DTVM, DRedis, Python, DLua...
Lightweight Container or VM
(Kata Containers...)
Orchestration System
K1S
(Cilium-based "new Kubenetes")
API
Universal Agent
Servicemesh
Cilium CNI+
FUSE+MAP
...
System Libs, Daemons, Utilities
LISA2 (BCC, Drgn, DCoreSight...), NetData2
iproute2, cubectl2, ntopng2, kubectl2...
FUSE daemon
cubed
podman...
libebpf
libcube
musl + jemalloc
Linux Kernel
In-Kernel Services (Cubes)
DNS
DB
OVN
LB
IKBus
Storage
Network
SRv6
Security
Security
Other Subsystems
eBPF

Stage 4


App
App
App
DRay
General-purpose App Framework
Serving (Training)
Streaming
Data Processing
Universal Scheduler
Portal
...
Services
DRPC, DTVM, DRedis, DPython, DLua...
Lightweight Container or VM
(Kata Containers...)
Orchestration System
K0S
(mainly as Management-Control Plane)
API
Universal Agent
Servicemesh2
Networking
Storage
...
System Libs, Daemons, Utilities
LISA2 (BCC, Drgn, DCoreSight...), NetData2
iproute2, cubectl2, ntopng2, kubectl2...
FUSE daemon
cubed
podman...
libebpf
libcube
musl + jemalloc/llvm-libc
Linux Kernel
In-Kernel Services (Cubes)
DNS
DB
OVN
VNFs
LB
IKBus
Storage
Network
...
SRv6
Universal Router
Security
...
Security
Other Subsystems
eBPF

1.1 TuobaOS

- https://github.com/Yuwenfeng2019/TuobaOS
- developer friendly Linux distribution images that target at on Open Hardware development boards to achieve the following goals gradually:

Stage 1

- base on an existed distribution like Fedora, Manjaro, and so on
- upstreaming Linux Kernel with all the necessary HCI related features (e.g., eBPF, Virtualization, Debugging, Networking, Storage…)enabled according to our design
- all the necessary development required packages (such as that for programming language runtimes, developer toolkits, virtualization softwares, devops utilities, HPC/AI frameworks, security tools, and so on) are preinstalled
- …

Stage 2

- get rid of the old GNU softwares and corresponding dependencies as much as possible, such like:

  1) remove GCC and use LLVM as the default toolchain to build everything like Linux Kernel

  2) replace glibc with musl libc (http://www.musl-libc.org/)
- upstreaming Linux Kernel with our own customization, especially for eBPF related code, and an In-Kernel messaging bus
- …

Stage 3

- replace musl libc with llvm-libc (https://llvm.org/docs/Proposals/LLVMLibC.html) if the later is mature enough
- upstreaming Linux Kernel with all the In-Kernel services work as expected according to our design
- …

The first release of TuobaOS images will come by the end of this year…

XIV. Wrap-up

- nearly every subsystem of Linux is or will be effected by eBPF, and eBPF is getting better, which is also hacker friendly.
- smart and modular hardware design is the general situation.
- it's time to release the power of cluster computing on high cost–performance ratio SBC or SOM/COM.
- emerging hardware techniques like Persistent Memory (MRAM, PCRAM, ReRAM…) will dramatically change the underlying system design both in hardware and software.
- our design and implementation for HCI on Edge will meet the trend of Hardware-Software & Kernel Space/User Space unified design of a modern computer system, which aims at efficiency and flexibility.
- PoC and protyping development work will be gradually published at conferences or github -- I will continuously post the latest progress and promise this will be a fully open source project!
- Currently, there is no uniform solution for Edge Computing, since it relies primarily on actual requirements from the enterprise side, while such requirement may different from each other, so it leaves too much room for your imagination!

Q & A

Thanks!

Reference

Slides/materials from many and varied sources:

- http://en.wikipedia.org/wiki/
- http://www.slideshare.net/
- https://developer.arm.com/architectures/instruction-sets
- https://www.python.org
- http://llvm.org
- http://www.brendangregg.com/
- https://en.wikipedia.org/wiki/Just-in-time_compilation
- https://www.infoq.com/news/2019/05/kubernetes-future/
- https://www.infoq.cn/article/apache-arrow
- …

…