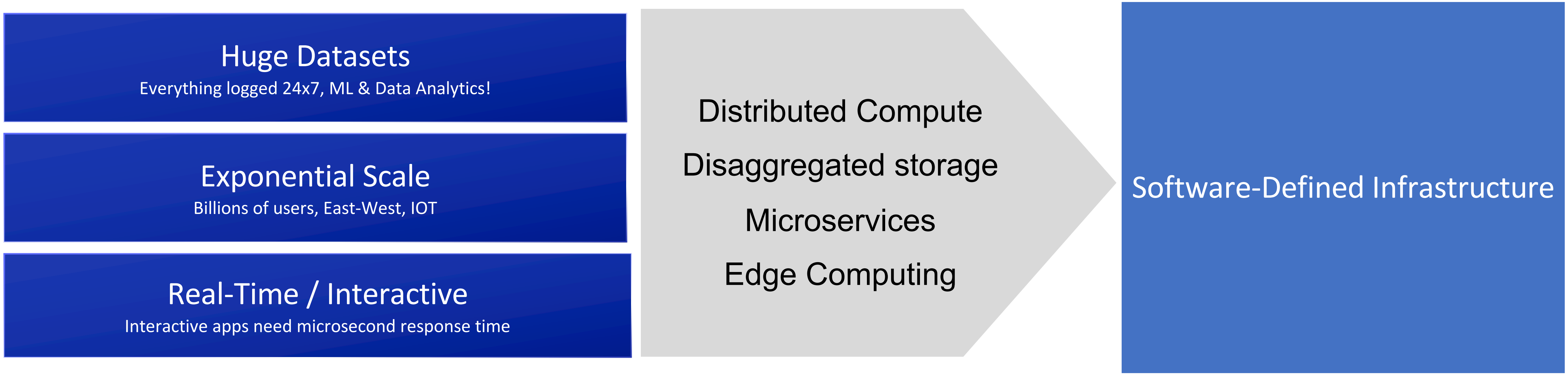
# Introducing IPDK

Deb Chatterjee, Sr Dir Eng @Intel, presenter
Dan Daly, Sr PE @Intel

# Infrastructure Trends



Huge Datasets

Everything logged 24x7, ML & Data Analytics!

Exponential Scale

Billions of users, East-West, IOT

Real-Time / Interactive

Interactive apps need microsecond response time

Distributed Compute

Disaggregated storage

Microservices

Edge Computing

Software-Defined Infrastructure

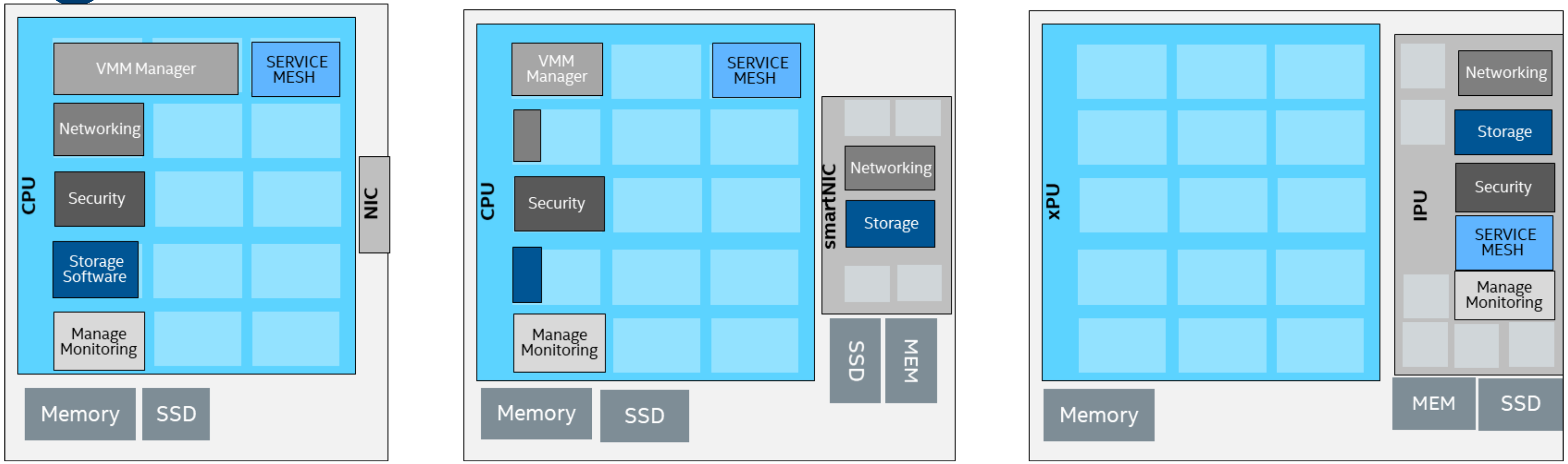
A pervasive theme across industries

- Same modular software runs everywhere, from the data center to IoT device
- Open, modular software is driving the pace of this software revolution

# Architectural Compartmentalization and Domain-Specific Hardware

- Mismatch of software to hardware abstractions and trust boundaries
- Hypervisors are unable to effectively abstract domain-specific hardware
- Desire to use entire host CPU for application workloads

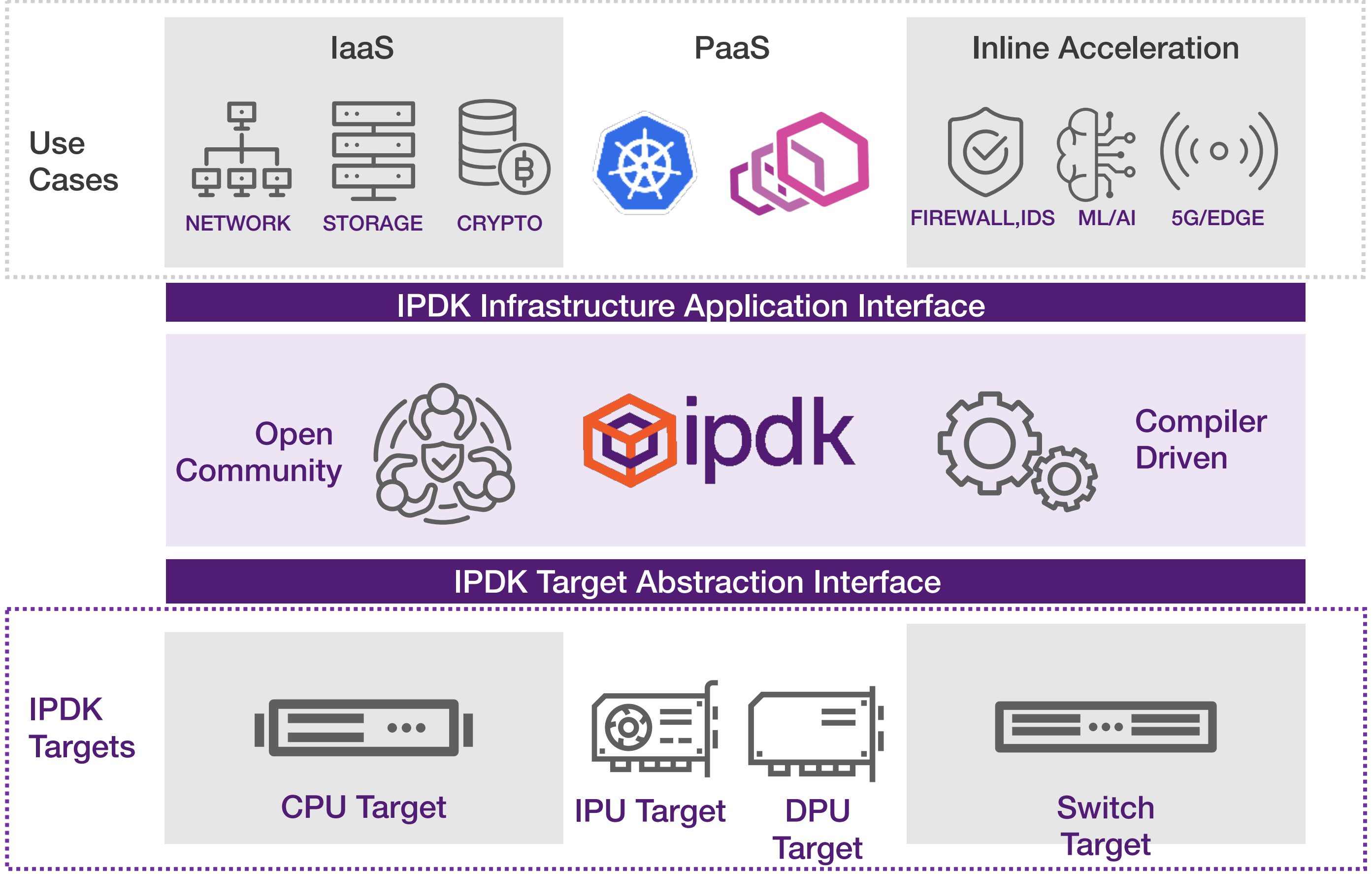
# Emergence of the IPU



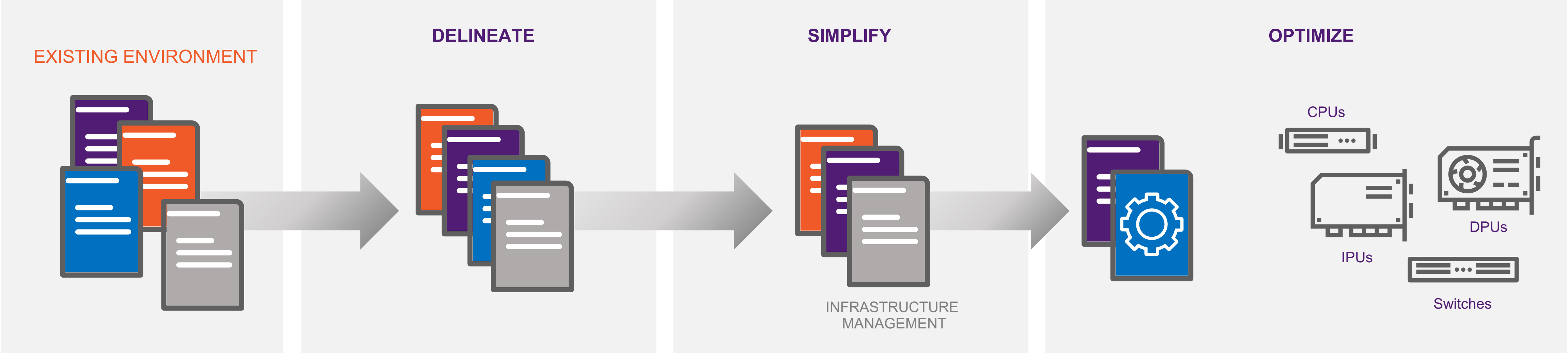
1. Efficient  high-performance software programmable multi-core CPUs
2. Flexible and programmable acceleration engines
3. SW-defined device functions and rich programmability

# IPDK Overview

- IPDK is a development framework
- **community-driven**
- **target agnostic**
- runs on CPU, IPU, DPU, or switch.



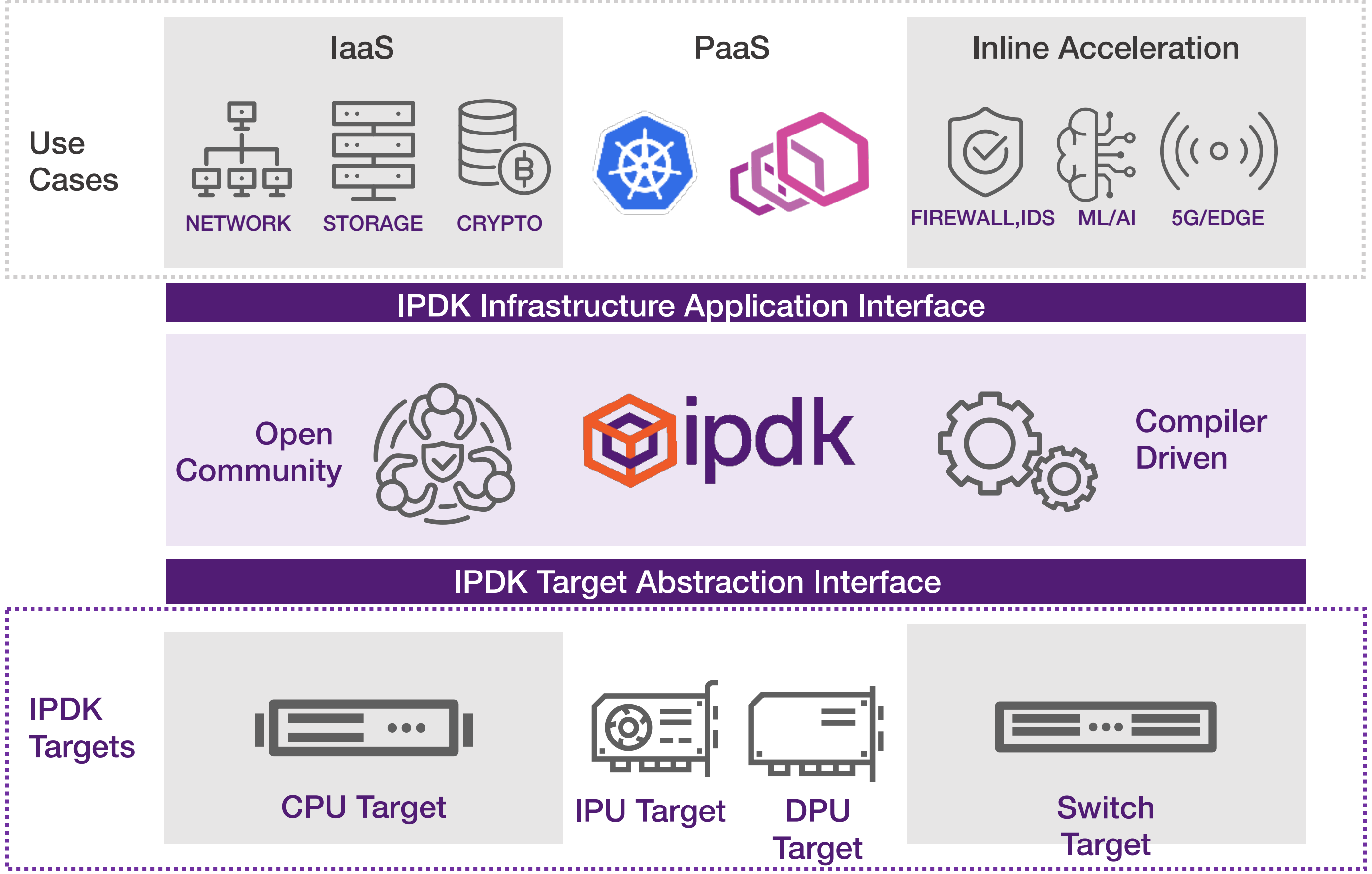
# IPDK Approach



1. **Delineate** Business Logic vs. Infrastructure
2. **Simplify** Infrastructure Management
3. **Optimize** Infrastructure using a Compiler-Driven Target Abstraction

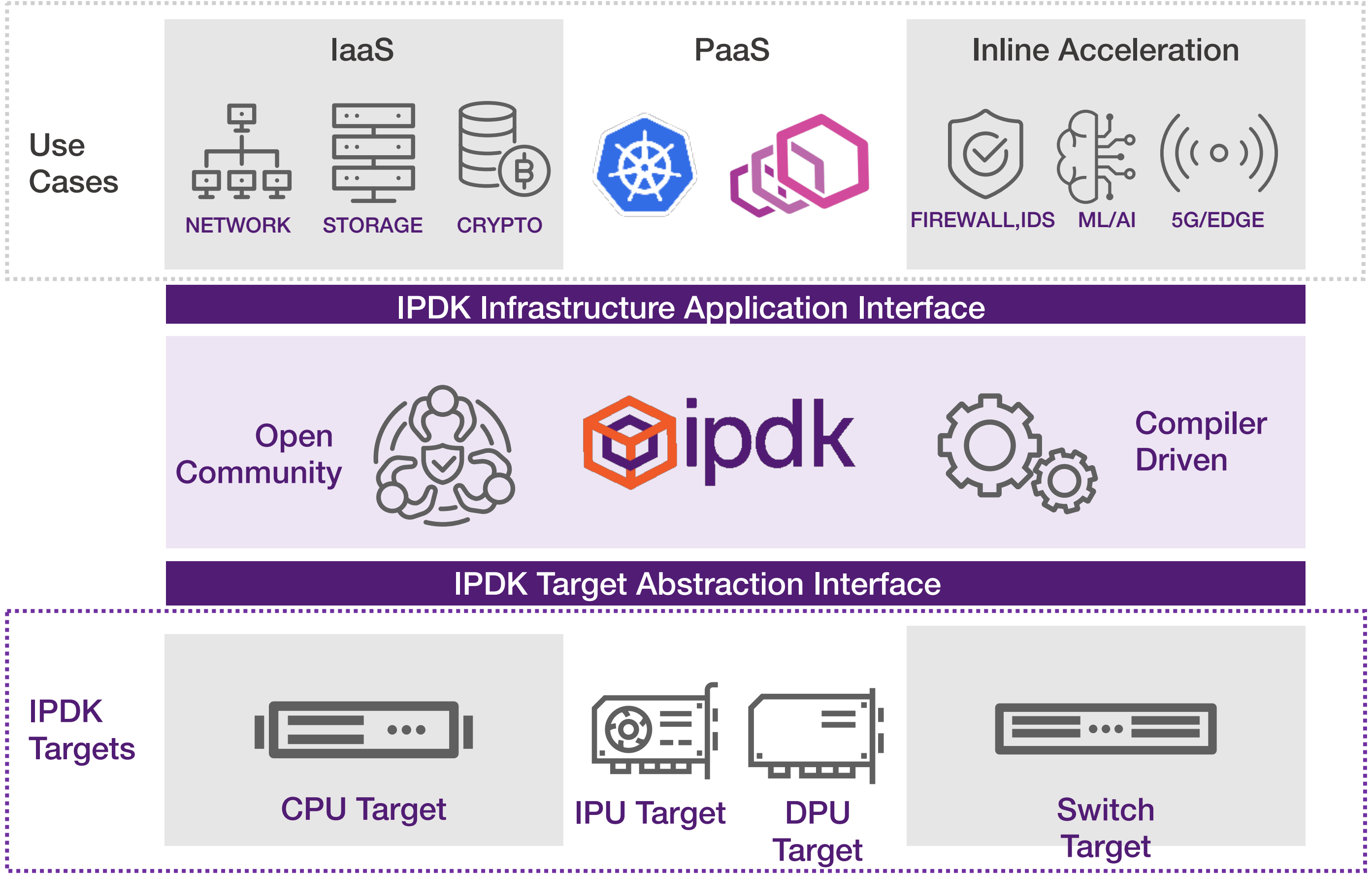
# IAI, TAI, TDI – the IPDK Standard Interfaces

- **Infrastructure Application Interface (IAI)**
- **Target Abstraction Interface (TAI)**
- **Table Driven Interface (TDI)**



# IPDK journey is use-case driven

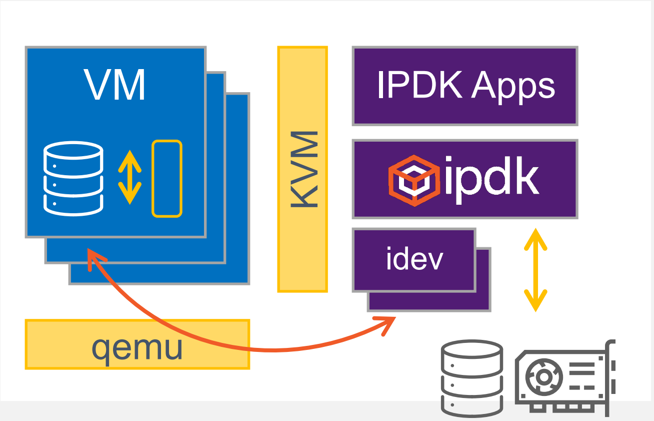
1. **Infrastructure-as-a-Service**
2. **Platform-as-a-Service**
3. **Inline Acceleration**



# Example Use Case:  IaaS

- **Common Control**
- **Common Interfaces**
- **Target Abstraction**

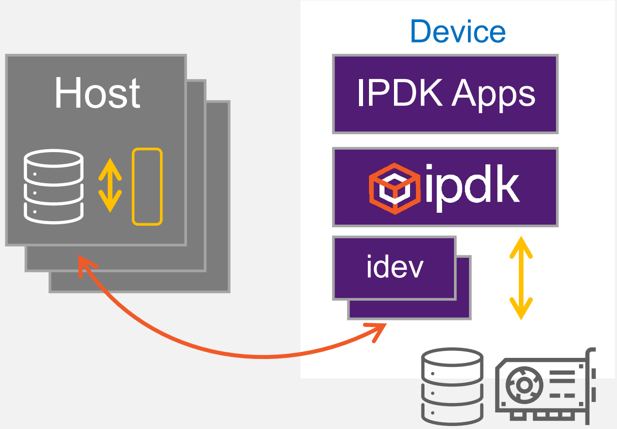
## VM HOSTING



- Insert/Delete devices into **VMs**
- **Direct attached** devices (drives, NICs, Accel)

**Software Hypervisor I/O**
Virtual Switch, Block Storage & Crypto

## BARE METAL HOSTING

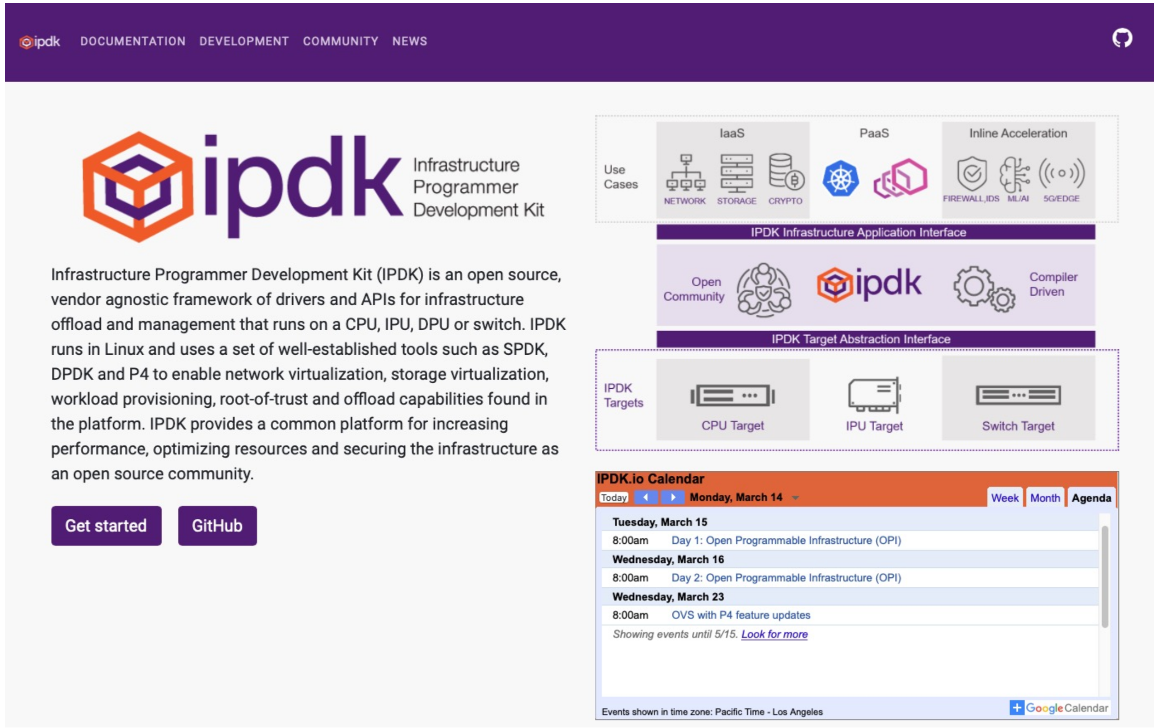


- Insert/Delete devices into **hosts** (bare metal, VMs inside bare metal)
- **Direct and virtual** devices (drives, NICs, Accel)

**Hardware Hypervisor I/O**
Virtual Switch, Block Storage & Crypto

# Open-Source Development

- **Recipes and ingredients**
- **Open-Source Development & Governance**
- **Development has started, join us!**
  Collaborate on Slack , Github & IPDK.io



Infrastructure Programmer Development Kit (IPDK) is an open source, vendor agnostic framework of drivers and APIs for infrastructure offload and management that runs on a CPU, IPU, DPU or switch. IPDK runs in Linux and uses a set of well-established tools such as SPDK, DPDK and P4 to enable network virtualization, storage virtualization, workload provisioning, root-of-trust and offload capabilities found in the platform. IPDK provides a common platform for increasing performance, optimizing resources and securing the infrastructure as an open source community.

IPDK.io Calendar

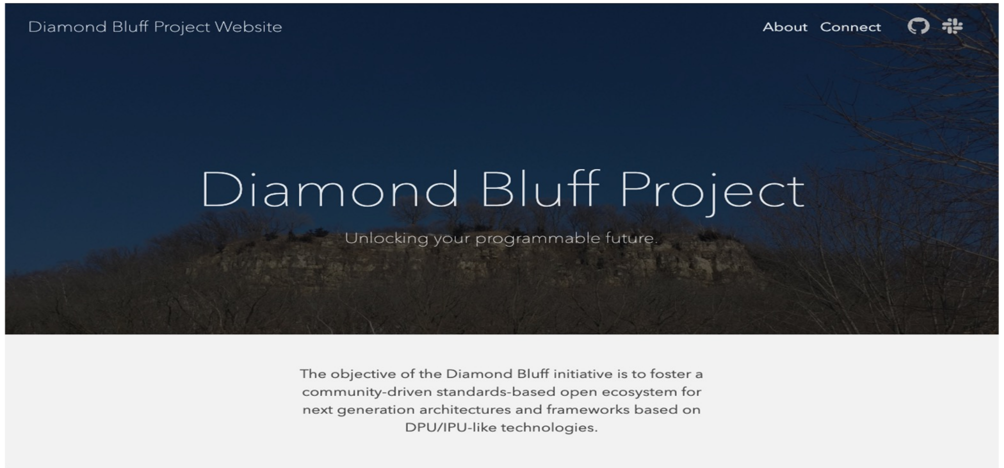
Today ◀ ▶ Monday, March 14 | Week | Month | Agenda

**Tuesday, March 15**
8:00am Day 1: Open Programmable Infrastructure (OPI)
**Wednesday, March 16**
8:00am Day 2: Open Programmable Infrastructure (OPI)
**Wednesday, March 23**
8:00am OVS with P4 feature updates

*Showing events until 5/15. Look for more*

Events shown in time zone: Pacific Time - Los Angeles

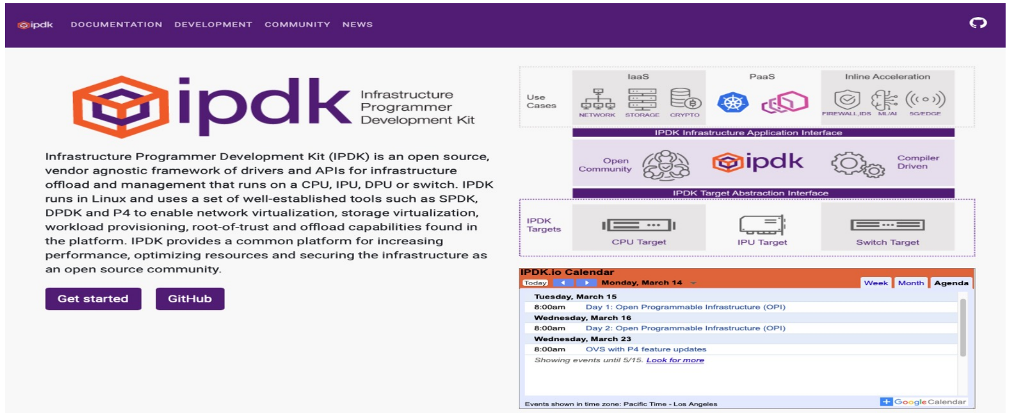
# IPDK, Diamond Bluff, OPI



Diamond Bluff Project

Unlocking your programmable future.

The objective of the Diamond Bluff initiative is to foster a community-driven standards-based open ecosystem for next generation architectures and frameworks based on DPU/IPU-like technologies.



Infrastructure Programmer Development Kit (IPDK) is an open source, vendor agnostic framework of drivers and APIs for infrastructure offload and management that runs on a CPU, IPU, DPU or switch. IPDK runs in Linux and uses a set of well-established tools such as SPDK, DPDK and P4 to enable network virtualization, storage virtualization, workload provisioning, root-of-trust and offload capabilities found in the platform. IPDK provides a common platform for increasing performance, optimizing resources and securing the infrastructure as an open source community.

IPDK.io Calendar

Today ◀ ▶ Monday, March 14 Week Month Agenda

Tuesday, March 15

8:00am Day 1: Open Programmable Infrastructure (OPI)

Wednesday, March 16

8:00am Day 2: Open Programmable Infrastructure (OPI)

Wednesday, March 23

8:00am OVS with P4 feature updates

*Showing events until 5/15. Look for more*

Events shown in time zone: Pacific Time - Los Angeles

+ Google Calendar

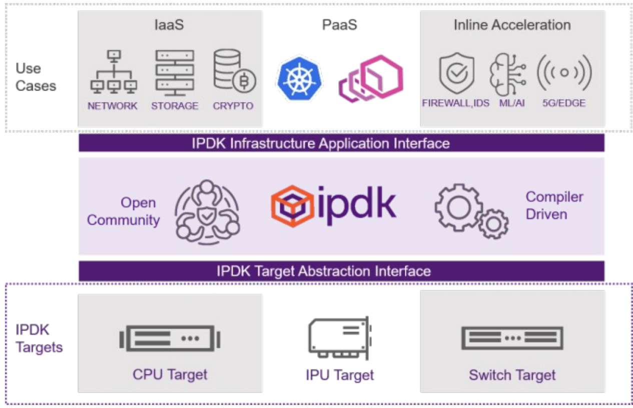
# From Dell - Need for an Open API for D/IPU

- Define standard mechanisms for Service Deployment
- Support of a Multi-Vendor Open D/IPU API definition and adoption for
  - Storage Services
  - Network Services
  - Security Services
  - AI/ML
  - Telemetry
  - System and Lifecycle Management
- Reuse Existing or define new common APIs for Configuration, Management and Consumption

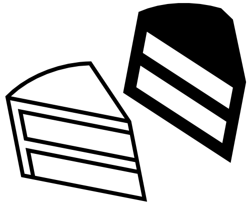
# From Lightbits - An IPDK Shopping List

1. “here's our cluster's discovery endpoint, here's the UUID of the volume we want, now surface it as a local NVMe device on the host, connected to this PF or VF”
2. A joint API that is common to most if not all SmartNICs and IPUs
   a. For configuring remote storage
   b. For deployment and provisioning of local services
   c. For VXLANs and network virtualization
   d. For network transport security, e.g., IPsec
   e. For storage data-at-rest encryption/decryption
   f. For end-to-end data integrity configuration (e.g., DIF)
   g. For resource metering and limiting (bandwidth and/or IOPs QoS, rate limiting)
   h. For billing?
3. Support for controlling IPUs both locally from the host and remotely from some centralized management layer
   a. potentially different mgmt access transports, security considerations, "ownership", etc.
4. Simplicity - keep the APIs and abstractions as simple as possible but no simpler. Clear and concise error reporting.
5. Robustness - the APIs should be race-free, safe in the face of retries/crashes/outages/concurrency. For block storage, "it *usually* works" is not considered acceptable.
6. Ultimately: “do one thing and do it well”

# From Ericsson - Could IPDK support Telcos requirements



- IPDK should
  - – Allow multi-vendor IPU/DPU for Telco Operator
  - – Allow mix of IPU/DPU and CPU based networking stack for Telco Operator
  - – Enable IPU/DPU SW stacks with large portability for Telco Vendors
    - Enable CPU usage for deployment without IPU/DPU
    - Enable functional portability over CPU, IPU/DPU and Programmable switches where deployed
  - – Enable SW application portability with low need for re-verification
    - Over different IPU/DPU, CPU and Cloud providers
  - – Fit the Cloud Native paradigm and be seamlessly exposed through K8s Infrastructure
    - Must support the Telco functional extensions e.g. for secondary networking



Work together to grow the shared cake instead of chasing growth of each small slice or crumble

# From Dell - Separating Business Apps from Infrastructure

- **Business Apps** run on the Node

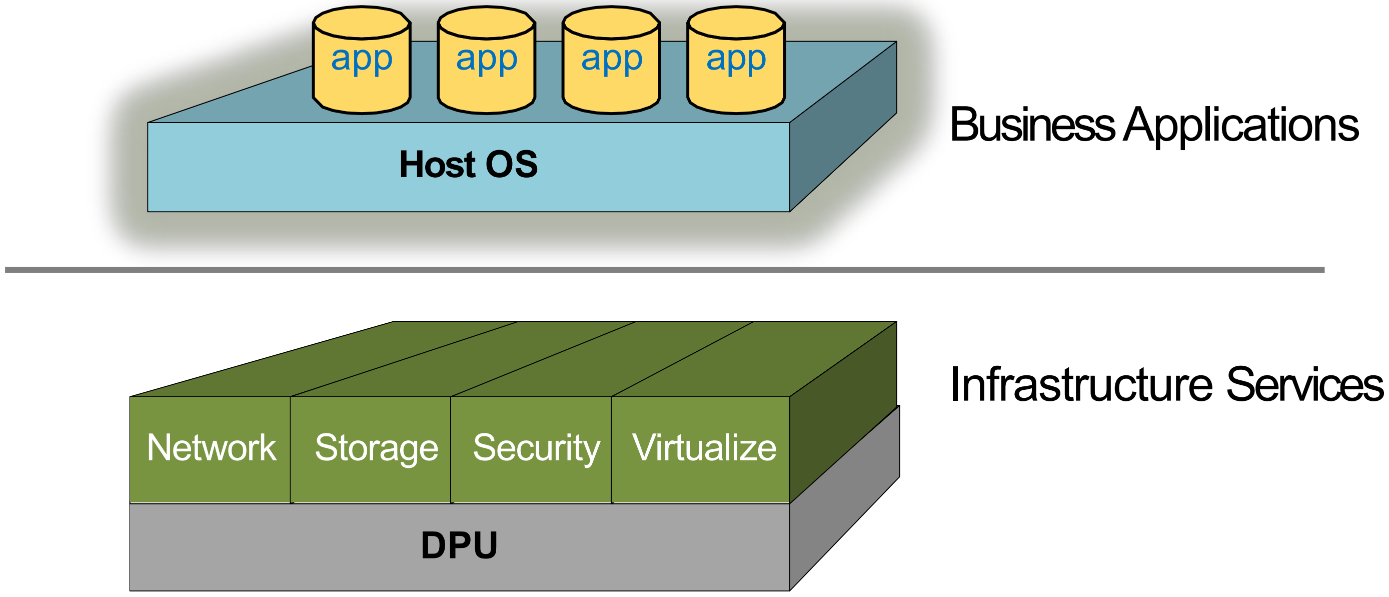
- **Infrastructure Apps** are Services running on the DPU
  D Network
  D Storage
  D Security
  D Virtualization

- Why move Infrastructure off the node?

  - *“30% of CPU cores are being used for datacenter infrastructure needs”*

  -*“It would take 125 cores to run all the Security, Network, and Storage offloads at 125Gbps”*

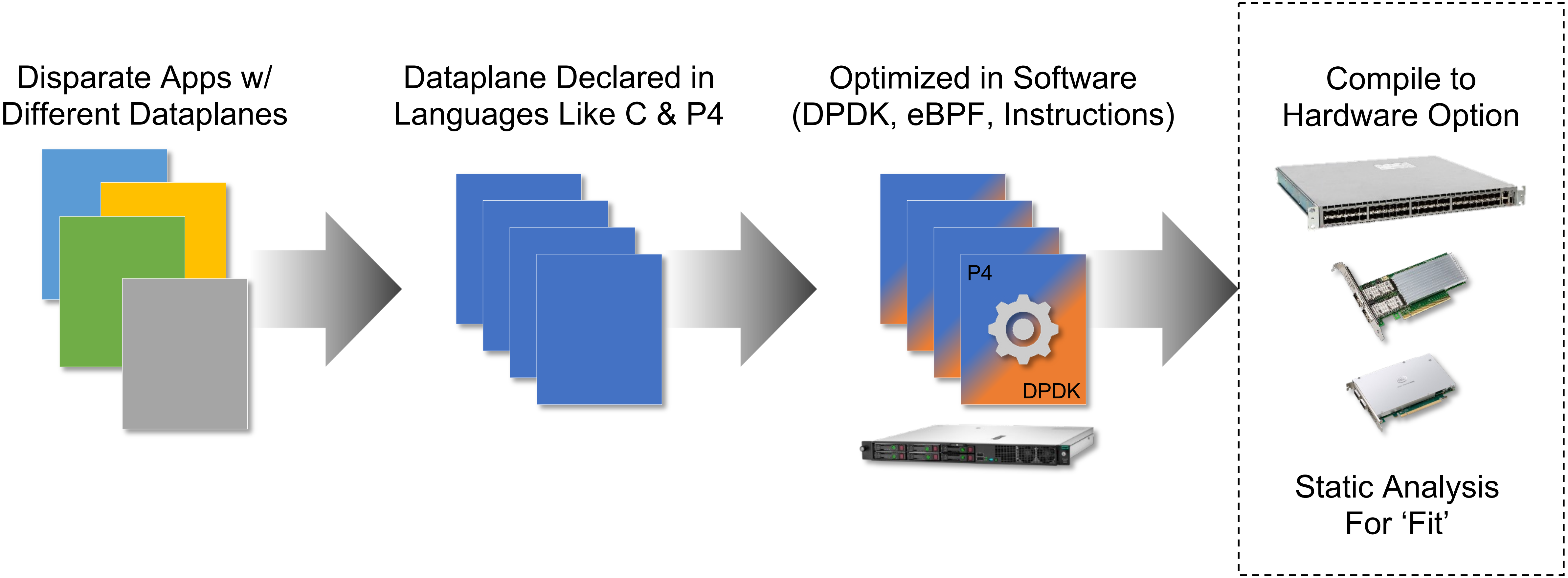
  *Jensen Huang, NVIDIA CEO, @ 2020 GTC Keynote*



# From Marvell - OCTEON IPDK PoC conclusion

- IPDK working smoothly on OCTEON DPU
  - ARM support was missing - added and upstreamed by Marvell
- p4 DPDK target
  - Performance limitations - CPU Scalability
- PCI Interface support missing
  - Virtio only
  - Required for external interface
  - Required for DPU->Host interface

# End-to-End Infrastructure Programming



Disparate Apps w/ Different Dataplanes

Dataplane Declared in Languages Like C & P4

Optimized in Software (DPDK, eBPF, Instructions)

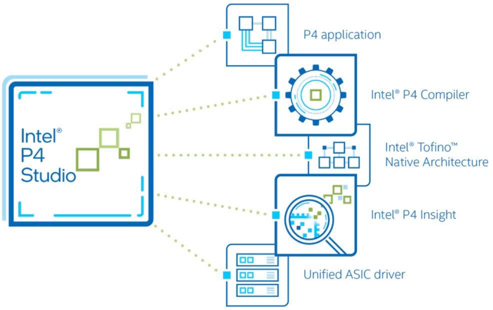
Compile to Hardware Option

Static Analysis For ‘Fit’

# Programmable Infrastructure Ecosystem Using P4

Growing Ecosystem

SDE & Compiler



Network Analytics

+

P4 targets –
- SW
- IPU ASICs
- IPU in FPGAs
- Switches

=

New Functionality

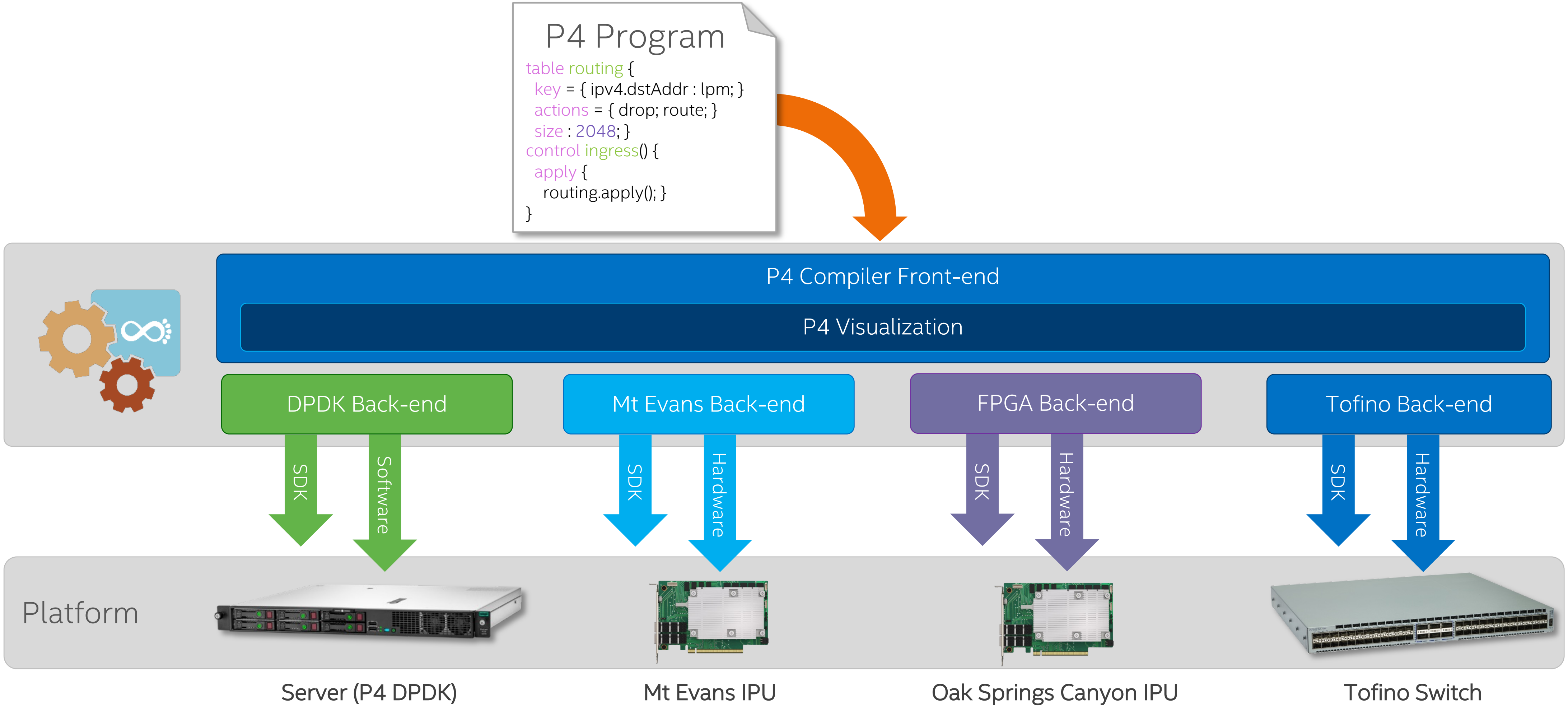
Differentiation

Rapid Innovation

Workload Acceleration

Network Analytics

# A single Programming Model Across Servers, IPUs, FPGAs & Switches



P4 Program

```
table routing {
 key = { ipv4.dstAddr : lpm; }
 actions = { drop; route; }
 size : 2048; }
control ingress() {
 apply {
   routing.apply(); }
}
```

P4 Compiler Front-end

P4 Visualization

DPDK Back-end

Mt Evans Back-end

FPGA Back-end

Tofino Back-end

SDK

Software

SDK

Hardware

SDK

Hardware

SDK

Hardware

Platform

Server (P4 DPDK)

Mt Evans IPU

Oak Springs Canyon IPU

Tofino Switch

# P4 demo on SW target

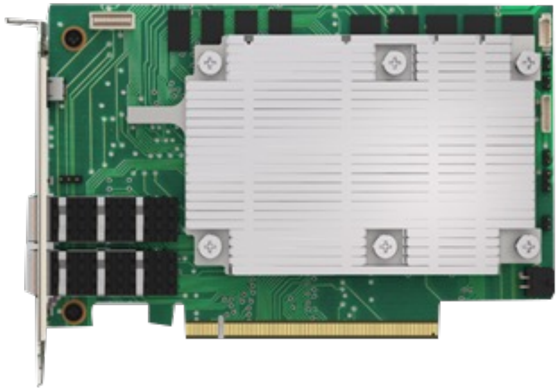
- Presented by Sandeep Nagapattinam from EPG SW
- Works on the P4 DPDK backend
  - A special P4 DPDK compiler backend was written
  - A special P4 DPDK packet processing library was developed
- Uses P4-OVS as control plane
- Please view in the tutorial

# P4 demo on Tofino Target

- Presented by Sayan Bandyopadhy from XFG

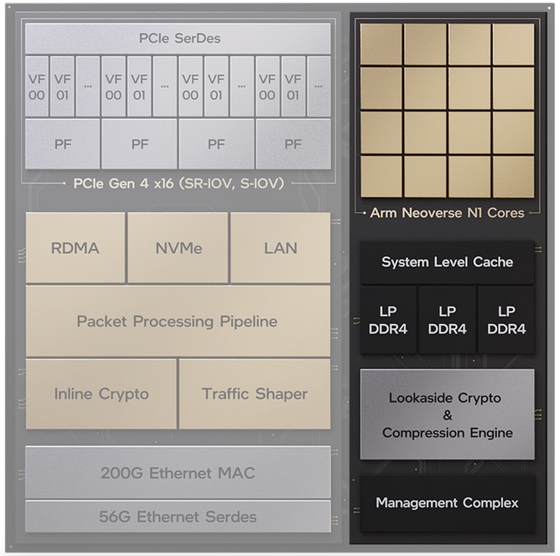
# P4 demo on Big Spring Canyon FPGA IPU Target

- Presented by Anbuvelu Venkataraman from EPG SW



# P4 demo on Mt Evans Target

- Presented by Nupur Uttarwar from EPG SW



# Linux_networking.p4 – starting point of P4-OVS

```
 table ipv4_tunnel_term_table {
        key = {
            local_metadata.tunnel.tun_type : exact @name("tunnel_type");
            hdr.outer_ipv4.src_addr : exact @name("ipv4_src");
            hdr.outer_ipv4.dst_addr : exact @name("ipv4_dst");
        }
        actions = {
            @tableonly decap_outer_ipv4;
            @defaultonly NoAction;
            //      @defaultonly set_exception;
        }
        default_action = NoAction;
}
    action set_tunnel(ModDataPtr_t tunnel_id, ipv4_addr_t dst_addr) {
        vendormeta_mod_action_ref = vendormeta_mod_action_ref | (16w1 << VXLAN_ENCAP);
        vendormeta_mod_data_ptr = tunnel_id; /* ptr can be tunnel_id */
        local_metadata.ipv4_dst_match = dst_addr;
        local_metadata.is_tunnel = 1;
    }
```

```
control linux_networking_control(inout headers_t hdr,
        inout local_metadata_t local_metadata,
        in pna_main_input_metadata_t istd,
        inout pna_main_output_metadata_t ostd)
{
    ActionRef_t vendormeta_mod_action_ref = (16w1 << NO_MODIFY);
    ModDataPtr_t vendormeta_mod_data_ptr = 0xFFFF;
    ModDataPtr_t vendormeta_neighbor_mod_data_ptr = 0xFFFF;
    action do_recirculate() {
        //    recirculate();
    }
    action set_exception(PortId_t vport) {
        send_to_port(vport);
        local_metadata.exception_packet = 1;
    }
```

# Connection_tracking.p4

```
 control MainControlImpl(
    inout headers_t  hdr,
    inout metadata_t meta,
    in    pna_main_input_metadata_t  istd,
    inout pna_main_output_metadata_t ostd)
{
    action drop () {
       drop_packet();
    }

    // Inputs from previous tables (or actions, or in general other P4
    // code) that can modify the behavior of actions of ct_tcp_table.
    bool do_add_on_miss;
    bool update_aging_info;
    bool update_expire_time;
    ExpireTimeProfileId_t new_expire_time_profile_id;

    // Outputs from actions of ct_tcp_table
    bool add_succeeded;

    action tcp_syn_packet () {
       do_add_on_miss = true;
       update_aging_info = true;
       update_expire_time = true;
       new_expire_time_profile_id = EXPIRE_TIME_PROFILE_TCP_NEW;
    }
    action tcp_fin_or_rst_packet () {
       update_aging_info = true;
       update_expire_time = true;
       new_expire_time_profile_id = EXPIRE_TIME_PROFILE_TCP_NOW;
    }
```

```
 table set_ct_options {
    key = {
       hdr.tcp.flags: ternary;
    }
    actions = {
       tcp_syn_packet;
       tcp_fin_or_rst_packet;
       tcp_other_packets;
    }
    const entries = {
       TCP_SYN_MASK &&& TCP_SYN_MASK: tcp_syn_packet;
       TCP_FIN_MASK &&& TCP_FIN_MASK: tcp_fin_or_rst_packet;
       TCP_RST_MASK &&& TCP_RST_MASK: tcp_fin_or_rst_packet;
    }
    const default_action = tcp_other_packets;
  }
  action ct_tcp_table_hit () {
 if (update_aging_info) {
       if (update_expire_time) {
          set_entry_expire_time(new_expire_time_profile_id);
          // This is implicit and automatic part of the behavior
          // of set_entry_expire_time() call:
          //restart_expire_timer();
       } else {
          restart_expire_timer();
       }
       // a target might also support additional statements here
    } else {
```

# Container_load_balancing.p4

```
 control MainControlImpl(
   inout headers_t  hdr,
   inout main_metadata_t meta,
   in    pna_main_input_metadata_t  istd,
   inout pna_main_output_metadata_t ostd)
{
   //vendormeta_t vendormeta;
   bool do_clb_pinned_flows_add_on_miss = false;
   bool add_succeeded = false;
   FlowId_t my_flow_id = (FlowId_t)0;

   action update_src_ip_mac(bit<48> new_smac, bit<32> new_ip) {
       hdr.ethernet.srcAddr = new_smac; //TODO: how to use meta in main_metadata_t
       hdr.ipv4.srcAddr = new_ip;
   }

   table write_source_ip_table {
       key = { meta.mod_blob_ptr : exact; }
       actions = { update_src_ip_mac; }
       size = 2048;
   }

   action set_source_ip (bit<24> ptr) {
       meta.mod_action = (ActionRef_t)WRITE_SRC_IP; // from mod_hints.p4
       meta.mod_blob_ptr = (ModDataPtr_t)ptr;
   }
```

```
 action pinned_flows_hit (FlowId_t flow_id, PortId_t p,
                 ModDataPtr_t ptr) {
    // This action should only be executed for Tx packets.
    meta.dst_port = p;
    send_to_port(p);
    meta.mod_action = WRITE_DEST_IP;
    meta.mod_blob_ptr = ptr;
  }

  // Note: This action does nothing at all if
  // do_clb_pinned_flows_add_on_miss is false.
  action pinned_flows_miss() {
    if (do_clb_pinned_flows_add_on_miss) {
      //my_flow_id = allocate_flow_id();//DPDK doesn't yet support
allocate_flow_id()
      my_flow_id = (FlowId_t)0;
      add_succeeded =
        add_entry(action_name = "pinned_flows_hit", // action name
             action_params = (clb_pinned_flows_hit_params_t)
                     {flow_id = my_flow_id,
                      p = meta.dst_port,
                      ptr = meta.mod_blob_ptr});
    }
  }
```

# Summary

- IPDK is a target and platform-agnostic Infrastructure Programming Kit
- IPDK is entirely in open source and in active development. Please come and join us!
- IPDK is a part of OPI and will shortly move under Linux foundation
- First major IPDK release is 22.07, in July of this year. Next release is 23.01, in January 2023. Two releases will be made every year
- P4 is a cornerstone of IPDK. We hope to create newer use cases for P4 through IPDK, such as the ones shown. We are also extending P4 support into Linux kernel
- That’s all! If you have questions, please write to me or Dan
- Deb.Chatterjee@intel.com
- Dan.daly@intel.com

# Thank You

More IPDK information on
www.ipdk.io