INTRODUCTION TO CLOUD SYSTEMS

Lecture 5 – DPDK, VPP, ServiceMesh

Previous lecture

CNI

eBPF

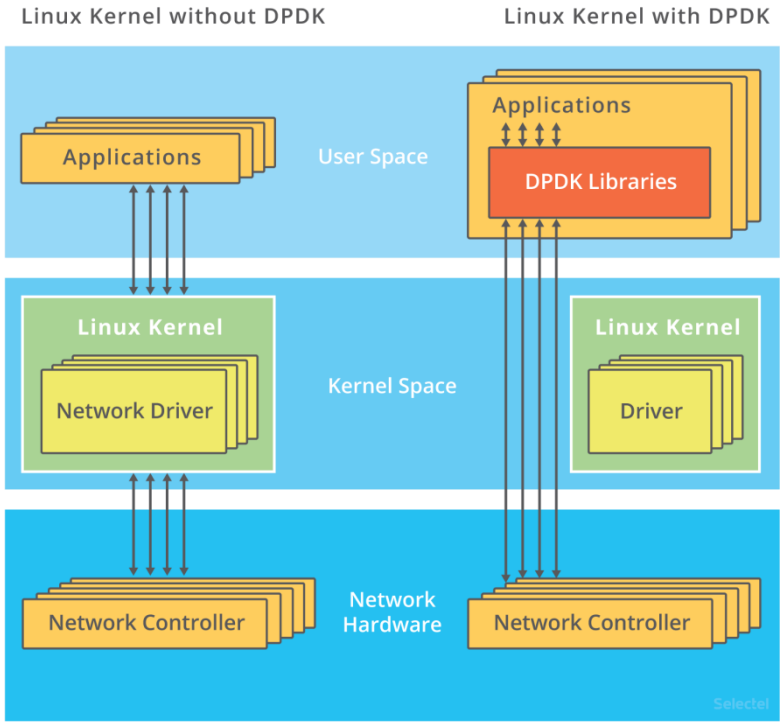
DPDK - Data Plane Development Kit

DPDK is the Data Plane Development Kit that consists of libraries to accelerate packet processing workloads running on a wide variety of CPU architectures.

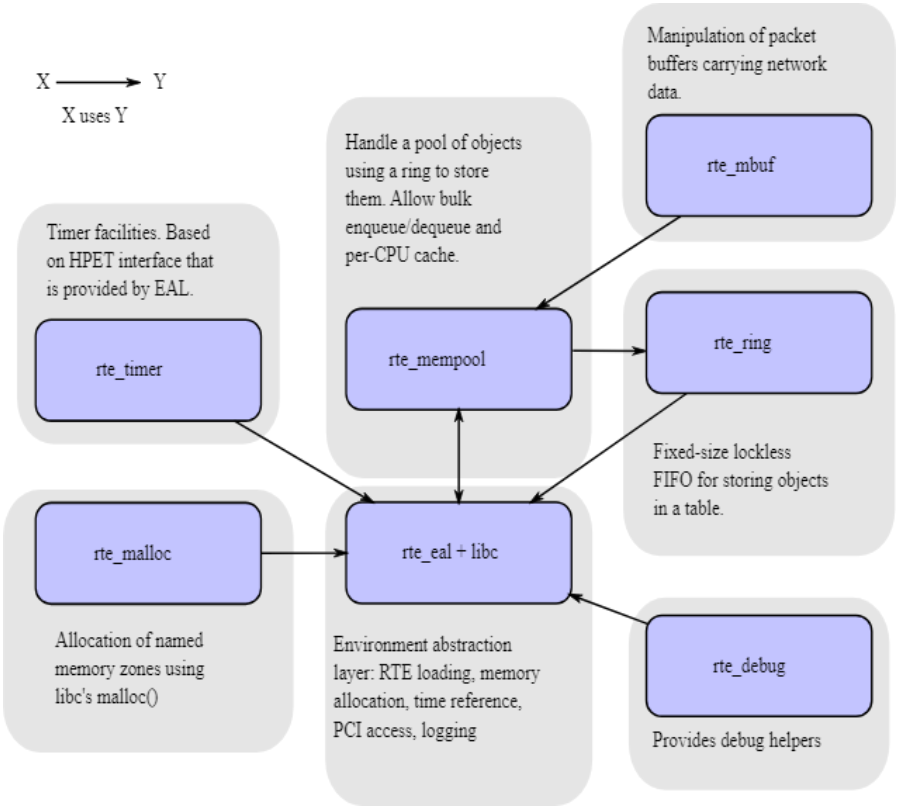
DPDK

DATA PLANE DEVELOPMENT KIT

DPDK - Data Plane Development Kit



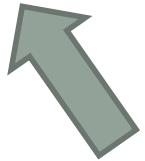
DPDK - Data Plane Development Kit



DPDK - Data Plane Development Kit

The DPDK has five core components that are responsible for sending packets from point A to B in the framework:

- **The EAL (Environment Abstraction Layer)**
- MBUF
- The DPDK MEMPOOL
- The RING
- The TIMER library

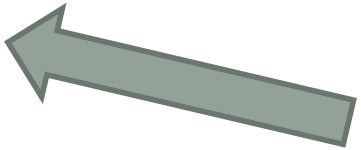


“is responsible for gaining access to low-level resources such as hardware and memory space.”

DPDK - Data Plane Development Kit

The DPDK has five core components that are responsible for sending packets from point A to B in the framework:

- The EAL (Environment Abstraction Layer)
- **MBUF**
- The DPDK MEMPOOL
- The RING
- The TIMER library

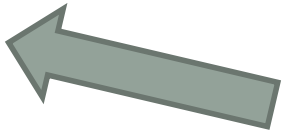


is a specific data structure that carries network packets as messages. This happens in the background without the cloud database user's knowledge.

DPDK - Data Plane Development Kit

The DPDK has five core components that are responsible for sending packets from point A to B in the framework:

- The EAL (Environment Abstraction Layer)
- MBUF
- **The DPDK MEMPOOL**
- The RING
- The TIMER library



is the library for creating allocated memory packets. They are of fixed size. The DPDK uses a MEMPOOL handler for storing free objects.

DPDK - Data Plane Development Kit

The DPDK has five core components that are responsible for sending packets from point A to B in the framework:

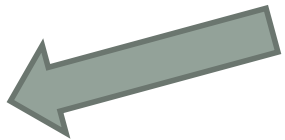
- The EAL (Environment Abstraction Layer)
- MBUF
- The DPDK MEMPOOL
- **The RING**
- The TIMER library

library manages log-less queues/messages sent between threads, cores, or other parallel entities. In relation to the database as a service model, the RING library helps with data packet instructions.

DPDK - Data Plane Development Kit

The DPDK has five core components that are responsible for sending packets from point A to B in the framework:

- The EAL (Environment Abstraction Layer)
- MBUF
- The DPDK MEMPOOL
- The RING
- **The TIMER library**

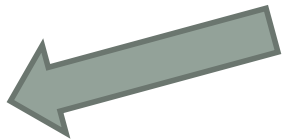


library is a DPDK service that offers asynchronistic callback functions. There is an interface to add, delete, or restart a timer

DPDK - Data Plane Development Kit

The DPDK has five core components that are responsible for sending packets from point A to B in the framework:

- The EAL (Environment Abstraction Layer)
- MBUF
- The DPDK MEMPOOL
- The RING
- **The TIMER library**



library is a DPDK service that offers asynchronistic callback functions. There is an interface to add, delete, or restart a timer

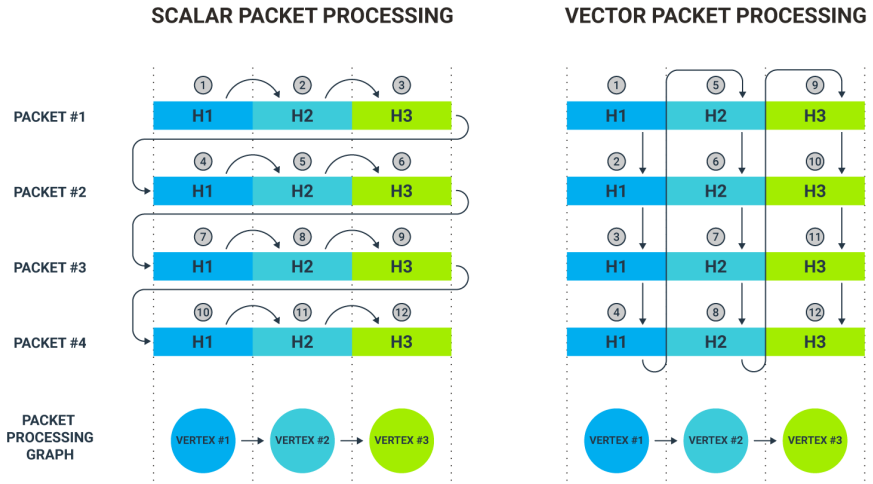
VPP - Vector Packet Processing

Vector Packet Processing (VPP) platform is an extensible, open-source framework, which offers the functionality of network switches or routers. Vector processing is the process of processing multiple packets at a time, with low latency. Single packet processing and high latency are present in the scalar processing approach, which VPP aims to make obsolete.

VPP uses the Data Plane Development Kit device drivers and libraries for many of its layer 1 functions - however, this functionality is separated into an optional plugin-in for VPP.

VPP - Vector Packet Processing

Regardless of the specific hardware or software implementation at the dataplane level, packet processing can be generally described as a sequence of operations that are performed on packets. This can be represented as a graph in which each vertex corresponds to an operation.



VPP - Vector Packet Processing

VPP & FD.io

The open source VPP platform is based on Cisco's proven VPP technology and is part of the Fast Data Project (FD.io), a Linux Foundation Networking (LFN) projects. The key characteristics of the VPP framework are the following:

- runs as a standard Linux user-space process
- can be deployed on bare metal, VM or container
- supports multiple processor architectures (x86/64, ARM-AArch64)
- is a modular platform built on a packet processing graph, an abstraction of how a VPP processing pipeline is organized:
  - vertices in the graph are small and loosely coupled, making it relatively easy to add new vertices or rewire existing ones
  - external plugins are supported (shared libraries loaded at runtime) that can introduce new graph vertices or rearrange the packet processing graph
  - a single vector of packets processed through the graph typically contains up to 256 packets

VPP - Vector Packet Processing

VPP & FD.io

The open source VPP platform is based on Cisco's proven VPP technology and is part of the Fast Data Project (FD.io), a Linux Foundation Networking (LFN) projects. The key characteristics of the VPP framework are the following:

- provides a variety of interfaces including those optimized for different scenarios such as container-to-container connectivity (“memif” interface) or VM-to-VM connectivity (“vhost-user” interface), etc.
- can use DPDK technology to connect to physical NICs
- offers built-in tracing capabilities for various types of interfaces
- has its own test suite (functional and performance testing), which is implemented through an associated project called Continuous System Integration and Testing (CSIT)
- can be configured via CLI-based utilities offered out-of-the-box (e.g. vppctl, vat)
- offers multi-language API bindings

VPP - Vector Packet Processing

Where is VPP used?

Ligato

Ligato is an open-source Golang framework for developing Cloud Native Network Functions (CNFs).

Container Network Interface

A specification and libraries for writing plugins to configure network interfaces in Linux containers.

Network Service Mesh

Network Service Mesh (NSM) is a kind of system that allows you to handle complex L2 / L3 use cases in Kubernetes.

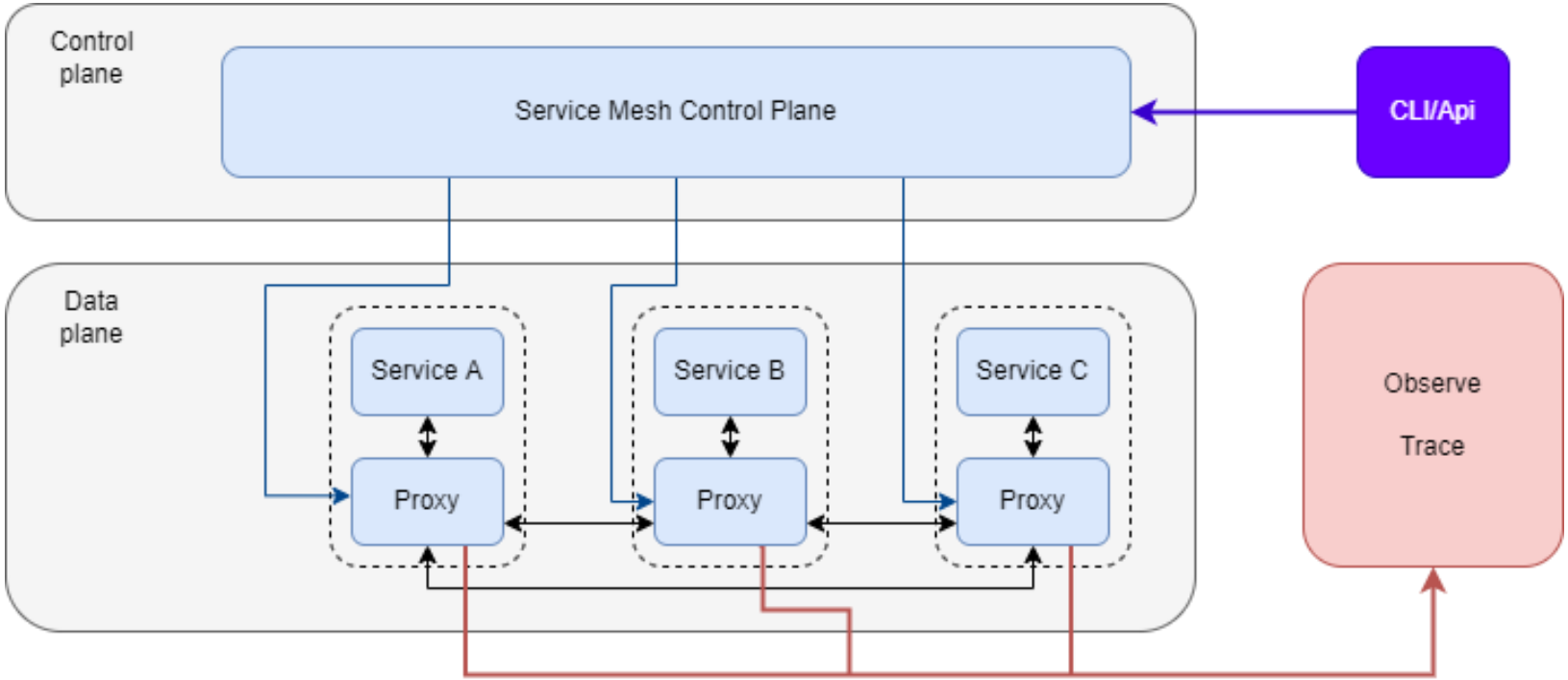
TNSR

TNSR is a “high-performance software router based on Vector Packet Processing

ServiceMesh

Service Mesh is a network of interconnected traffic intermediaries (proxies) that are globally configured. It consists of two basic elements: Control Plane and Data Plane. The Data Plane is responsible for forwarding traffic between services, and Control Plane is responsible for configuring the rules for how this traffic should be routed.

ServiceMesh



ServiceMesh

Data plane:

- traffic management and filtering
- Routing
- communication with services
- Service Discovery
- mTLS encryption
- collecting records
- enriching HTTP requests with additional headers required for authentication or enabling distributed tracing

Control plane:

Single component responsible for the configuration and control of the data plane layer. The component can communicate with each proxy sidecar within the infrastructure, which allows you to send a new configuration to each of the proxies and the other way around - collecting metrics and monitoring the service status. Main features:

- can report the state of the cluster in the form of visualizations and graphs
- visualization of metrics

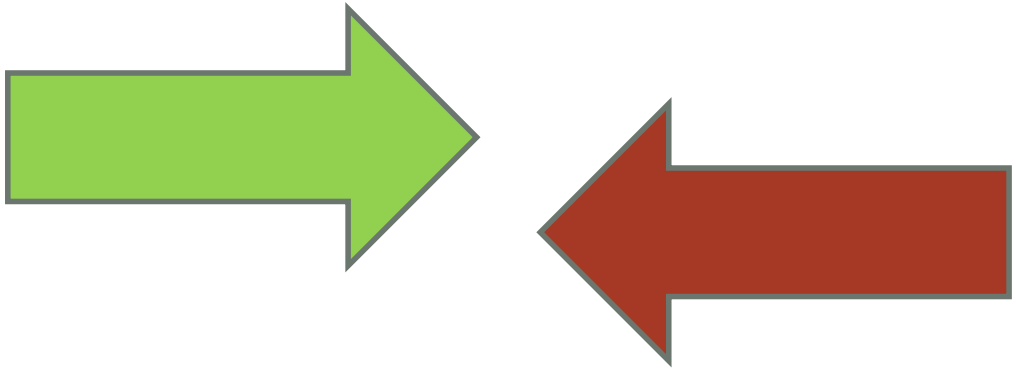
ServiceMesh

Pros:

- Traffic management
- Observability
- Security capabilities

Cons:

- Costs
- Competencies
- Greater load



ServiceMesh – Intel Xeon Optimization

Intel® Advanced Vector Extensions 512

Intel Advanced Vector Extensions 512 (Intel AVX-512) is a set of instructions that can accelerate performance for vector processing–intensive workloads. Vector processing performs an arithmetic operation on a large array of integers or floating point numbers in parallel. Examples of applications in which vector processing can be highly intensive include scientific simulations and 3D modeling.

With ultra-wide 512-bit vector-operations capabilities, Intel AVX-512 can handle your most demanding computational tasks. 3rd Gen Intel Xeon Scalable processors are specifically built with the flexibility to run computationally intensive workloads on the same hardware as your existing workloads, so that you do not need to invest in additional hardware to run your demanding computational workloads.

ServiceMesh – Intel Xeon Optimization

Accelerate data-center performance with Intel AVX-512

3rd Gen Intel Xeon Scalable processors with Intel AVX-512 have a built-in performance advantage for computationally intensive workloads, whereas AMD EPYC processors are currently only available with AVX2, which can lead to significant performance advantages for Intel Xeon Scalable processors in cases such as image-classification, HPC, AI, and web encryption. A key difference between Intel AVX-512 and AVX2 in AMD processors is the auto-vectorizing compilers that Intel offers. The Intel compilers automatically enable applications to use vector instructions for Intel AVX-512. When coupled with profiling tools that help find high-impact vectorization opportunities to safely speed up applications, Intel compilers are fundamental to making efficient use of the wider vectors in Intel AVX-512.

ServiceMesh – Intel Xeon Optimization

The performance enhancements from Intel AVX-512 - including faster workload speeds and more efficient data processing - are available immediately to your applications. Business applications do not need to be modified to take advantage of the performance improvements made possible by Intel AVX-512. In fact, your applications running on servers powered by 3rd Gen Intel Xeon Scalable processors might already be benefiting from the performance gains provided by Intel AVX-512.

And for applications running in the cloud, Intel AVX-512 is also available in Intel-based public-cloud instances to suport hybrid environments.

ServiceMesh – Intel Xeon Optimization

Some application use cases that benefit the most from Intel AVX-512 include:

- HPC: scientific simulations, DNA sequencing, 3D advanced modeling, and financial analytics
- **Cryptography and data compression, working with Intel Crypto Acceleration and Intel QuickAssist**

Technology (Intel QAT)

- Image and audio/video processing
- AI and DL, working with Intel Deep Learning Boost (Intel DL Boost)