Evolution of Compute Stack for the Cloud

Deep Samal,

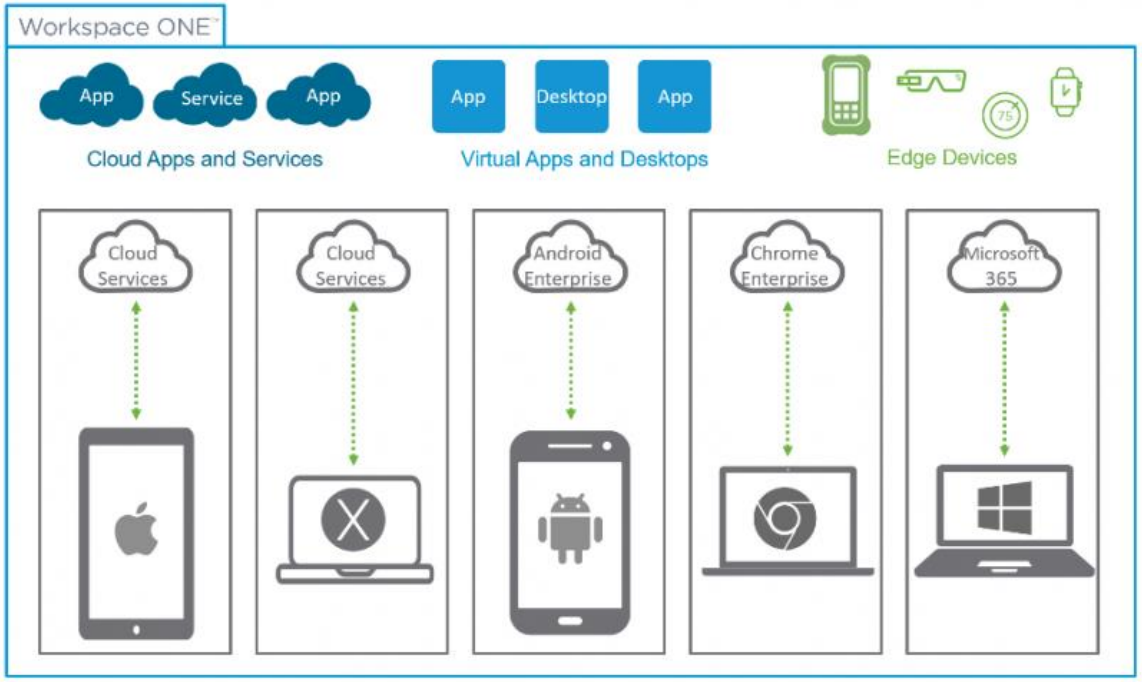
Staff Engineer, Performance Engineering

VMware/Broadcom/Omnissa

Introduction

- Staff Engineer
  - EUC Vmware, Broadcom, Omnissa
  - Performance Engineering Team, UEM

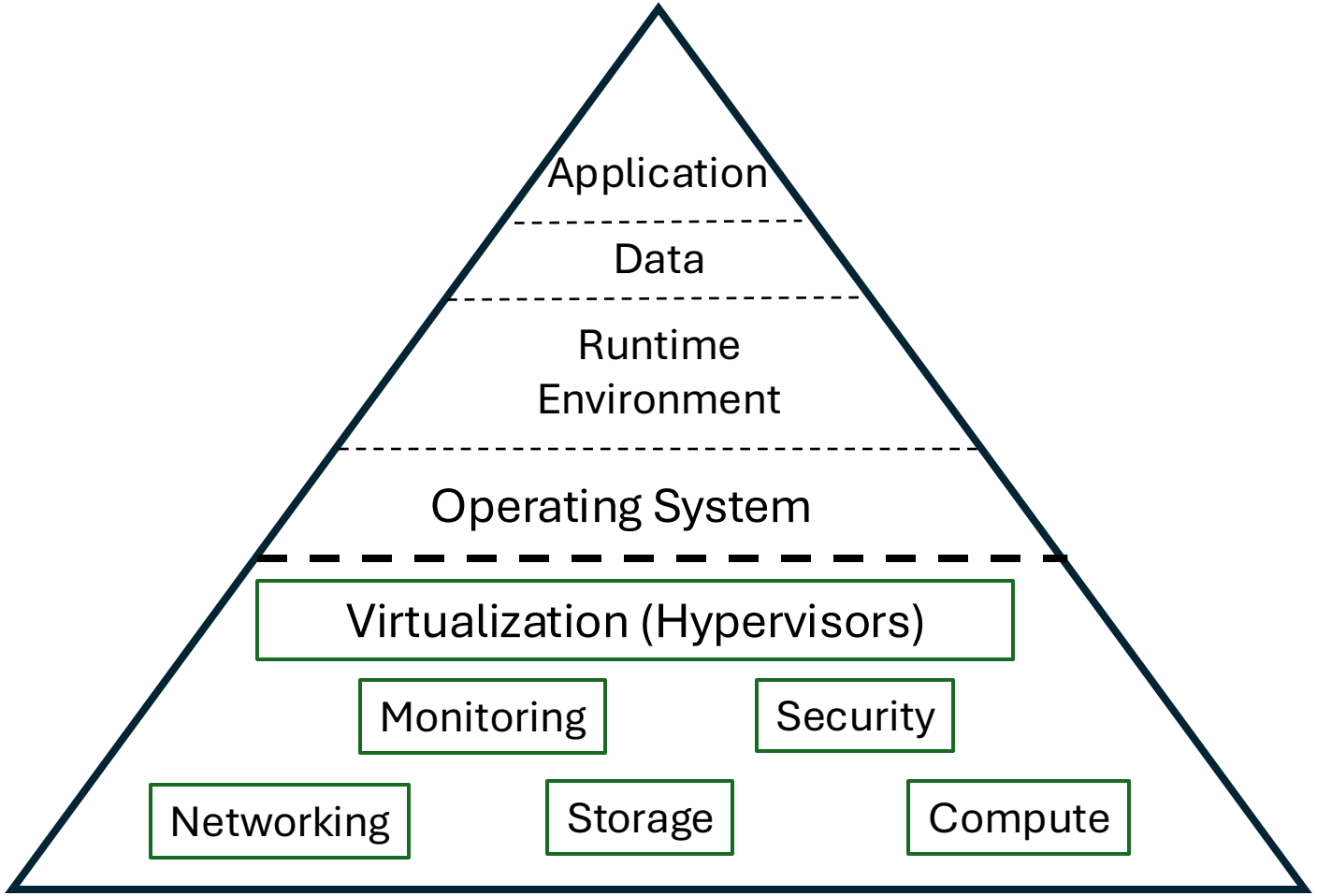
- PhD, ECE, Georgia Tech:
  - Advisors: Marilyn Wolf and Saibal Mukhopadhyay
  - Closed Loop Perception for Resource Efficient Autonomous System
    - Autonomous Systems need to adapt to the dynamic environments they operate in.
    - Control the complexity of autonomous perception systems (DNNs) dynamically according to risk/safety.



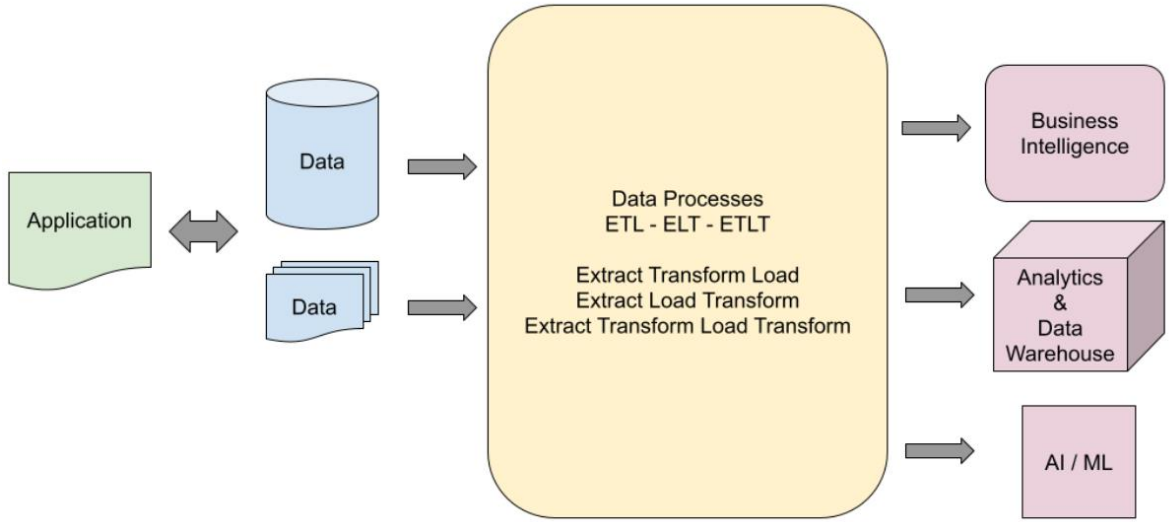
https://blogs.vmware.com/euc/2018/08/workspaceone-uem-vmworld.html

Infrastructure as a Service (IaaS)

- Components are loosely coupled
- Runtime Scalability
- Heterogenous Configurations
  - Compute Optimized
  - Memory Optimized
  - …
- Farm of specialized Hardware
- Bookkeeping



Data Pipeline – Assembly Line for Information



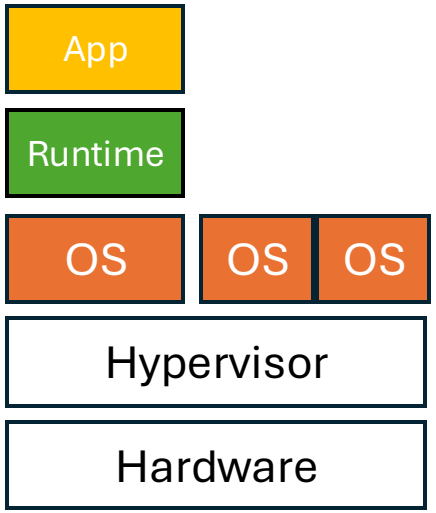
- Data pipeline is the process of getting the data from various sources, processing it and finally delivering desired output to consumers
  - Source: IoT devices, API calls, changes in databases etc.
  - Process: data normalization, filtering, formatting
  - Output: Analytics Applications, Monitoring, Data lake, another pipeline
- Different kinds of pipelines such as ETL, ELT etc.
- Architecture and components vary depending on desired use-cases such as batch-processing vs real-time etc.

https://cloud.google.com/blog/topics/developers-practitioners/what-data-pipeline-architecture-should-i-use/

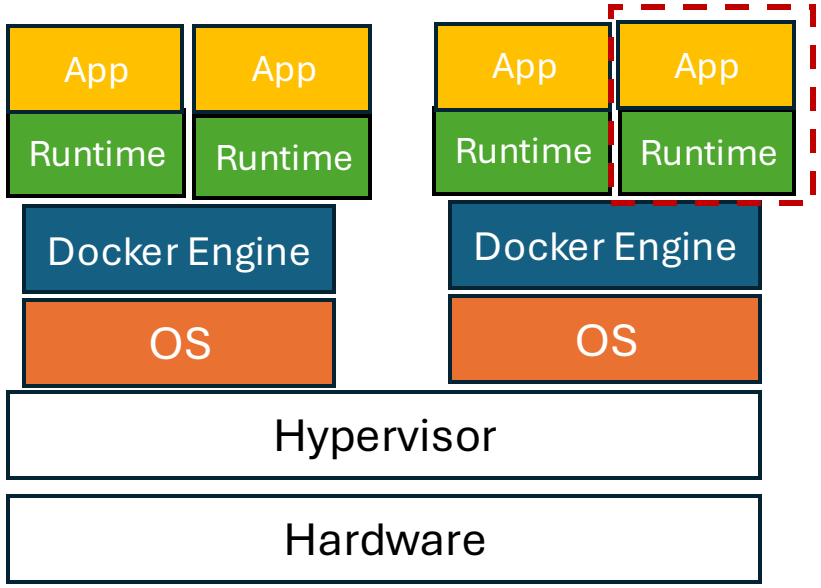
Virtualization

We can have multiple applications on same VM but there will be clash such as runtime library version etc.

Virtual Machines



Containers

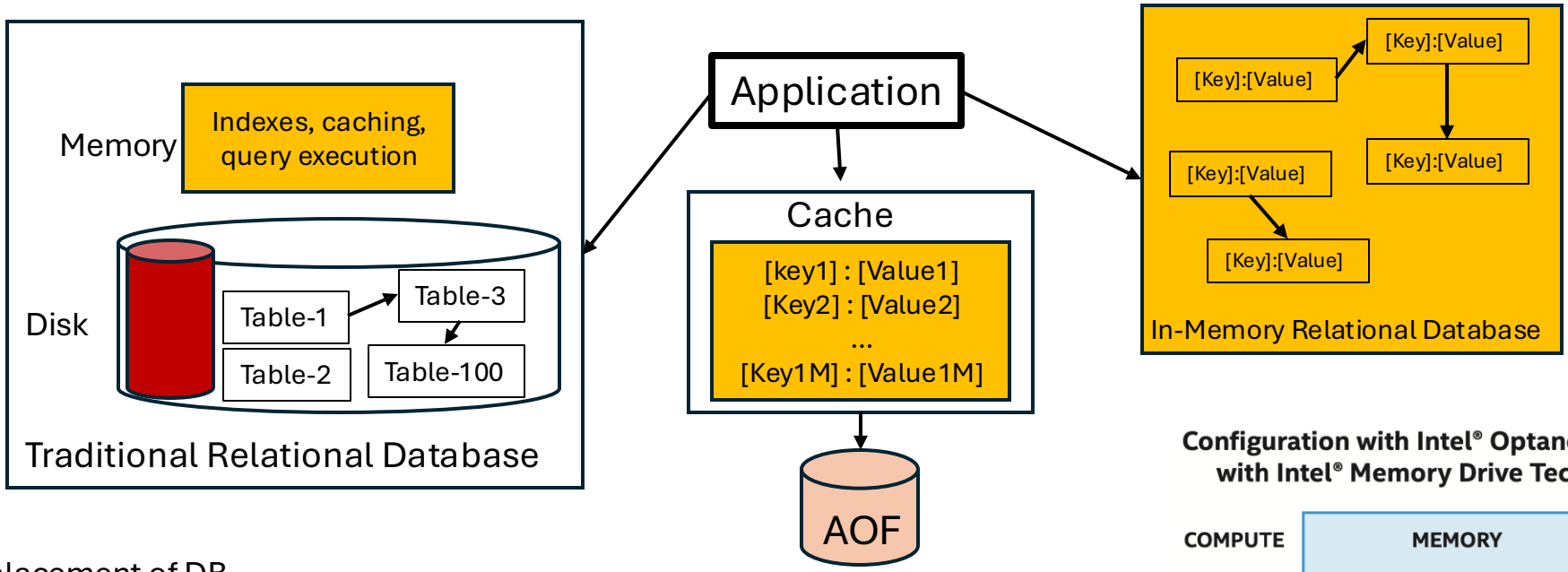


Application developers build containers and do not worry about how to deploy

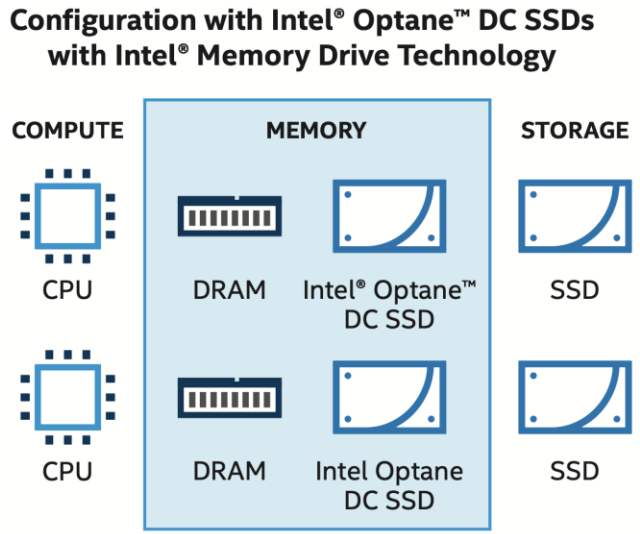
Different Apps have different compute profile

- Docker engine is responsible for isolation and management of containers.
- Application Processes inside containers run as native processes on the OS.
- Processes within a container are isolated from rest of the processes via Linux control groups
- How does the footprint on the hardware change due to containerized applications?
  - Context switching
  - Cache and Memory Access pattern

Data Store (Database, Cache)



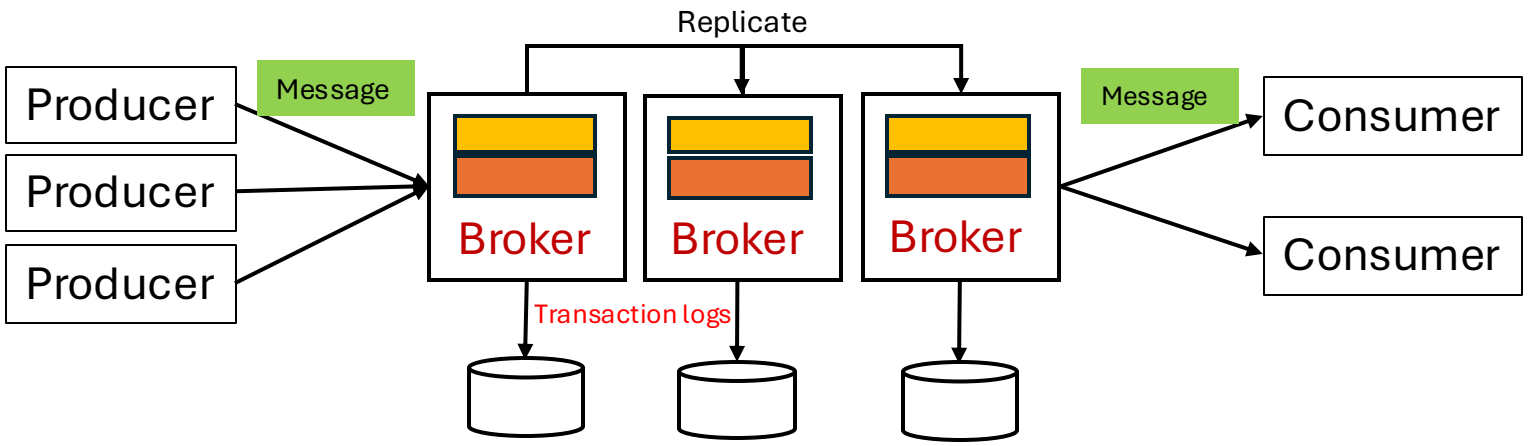
- Caching is not a replacement of DB
  - No persistence
  - Relational Nature is not captured
  - Unlike H/W Memory Management consistency is not automatically enforced
  - Value field can be hash-tables, list, bit-maps etc.
- In-memory DB try to bridge the difference between DB and cache by creating the relationship on data stored in Key Value format

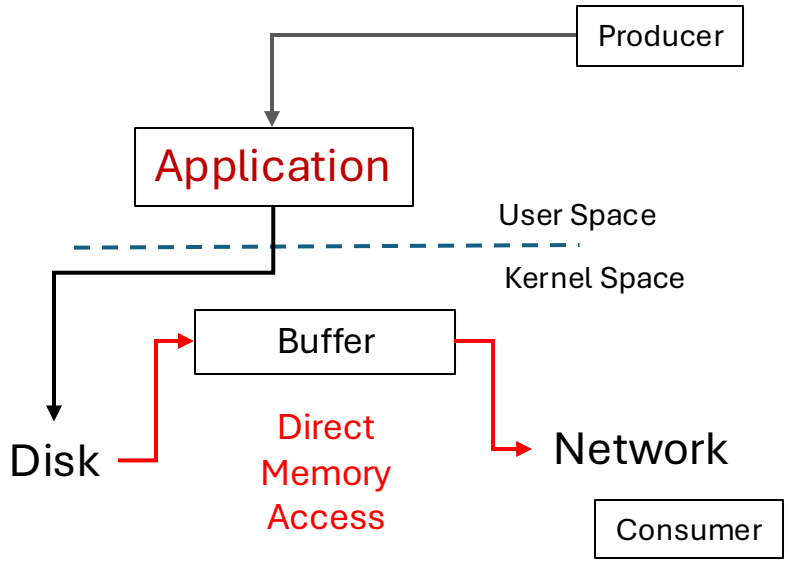


Source: Intel Solution Brief: Optimizing Microsoft SQL Server Databases to Accelerate Response Time and Throughput

Credit: Dennis Matovu, Staff Engineer, Vmware/Broadcom/Omnissa

Information Exchange





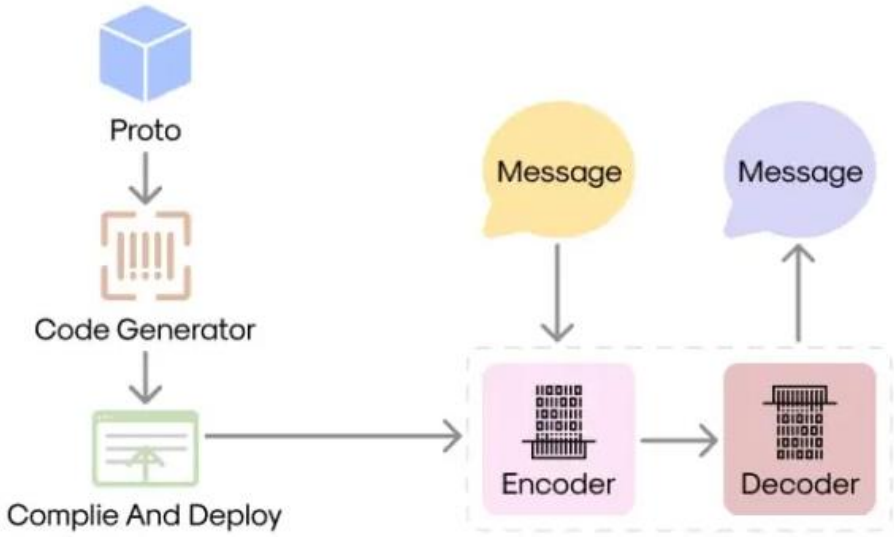
- Multiple type of messages are handled via topics.
- High throughput is achieved via Direct Memory Access

Data Serialization Technique

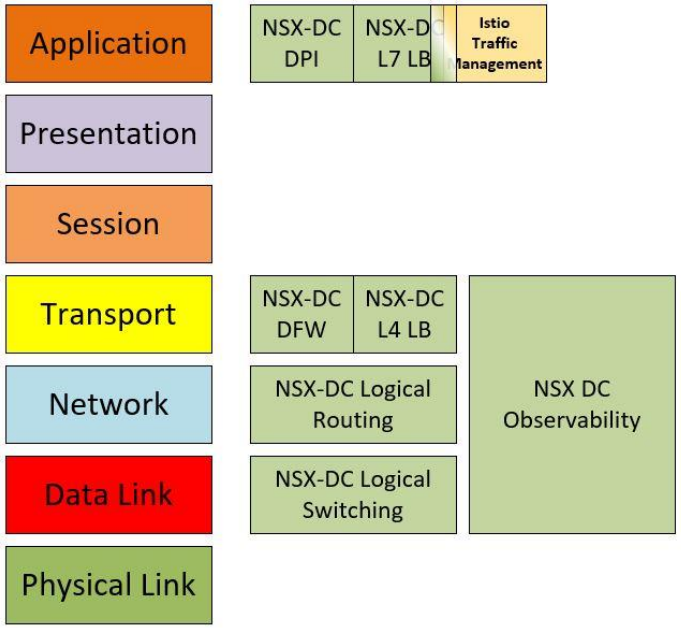
- JSON – Javascript Object Notification
  - Human readable serialization standard
  - Key-Value format
  - Demo- http://mpsoc-forum.org

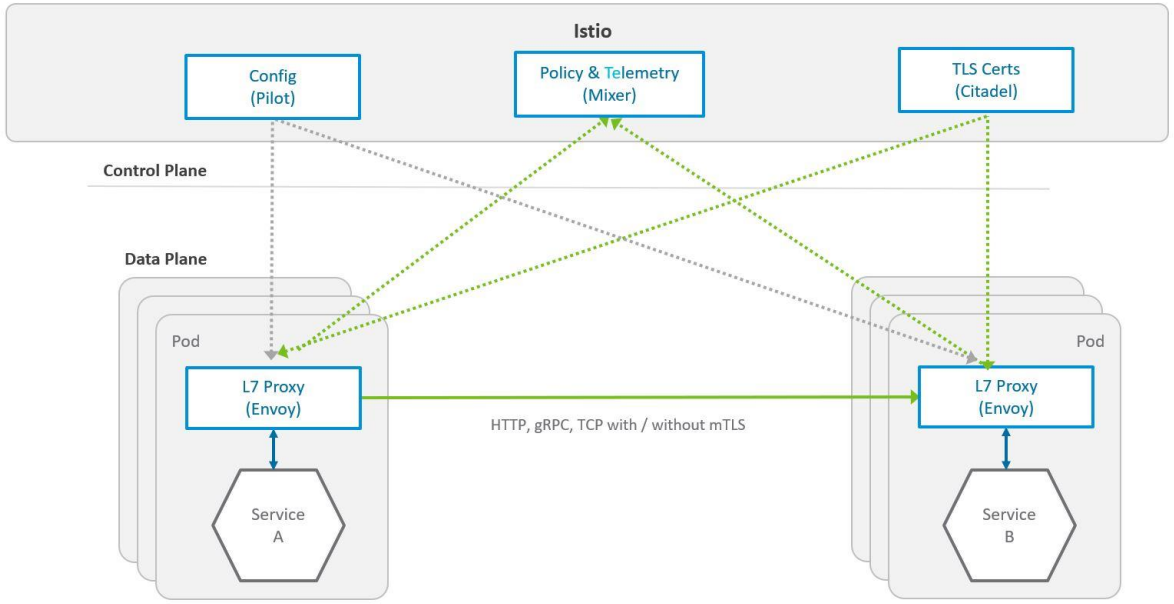
- Protobuf – Protocol Buffer
  - Binary format, not human readable
  - Fast serialization
  - Language agnostic but has to be compiled
  - Think structures in C

- Data format has to be converted into code!



Network Stack – Traffic Management

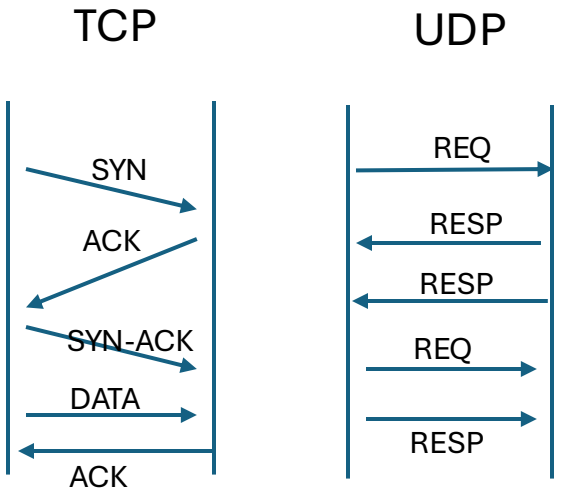




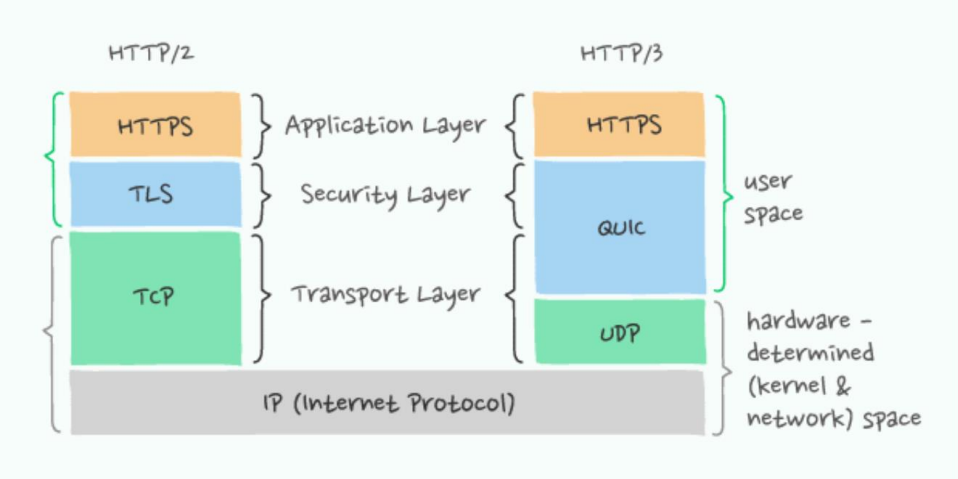
- Virtualization of Network OSI Layers
  - Low level structures are virtualized to give better control and observability
  - Low level control for high- availability and infrastructure management
  - High level N/W layer virtualization for application level control.
  - Objective is to achieve high-availability

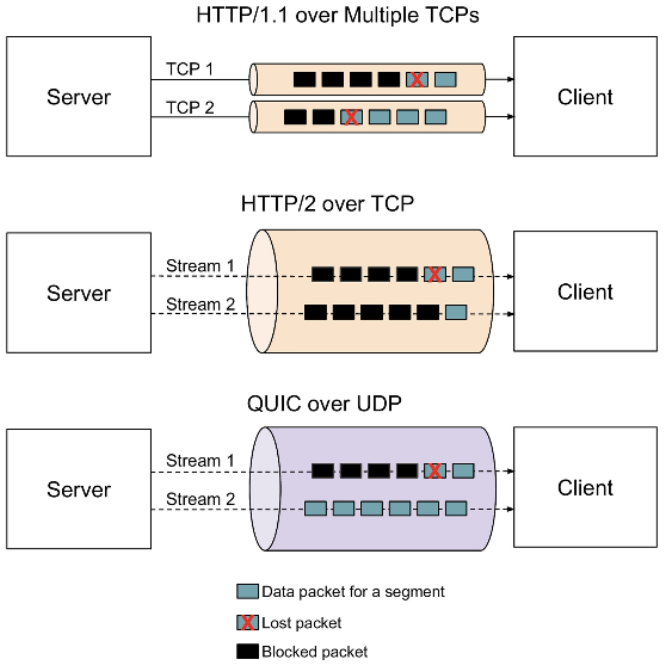
https://blogs.vmware.com/networkvirtualization/2019/04/how-istio-nsx-service-mesh-and-nsx-data-center-fit-together.html/

Network Stack - Protocol



Connection-less





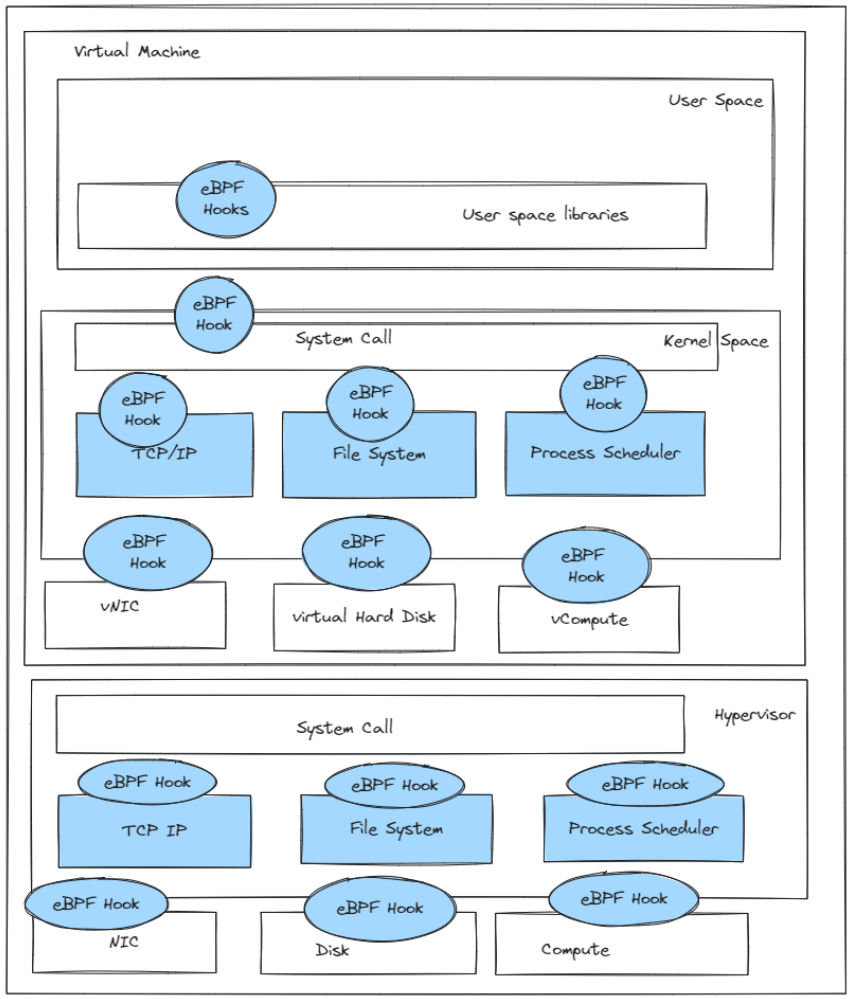
- Transport layer is a bottleneck
- Security, reliability and speed are not mutually exclusive
- Control network at application layer

https://ably.com/topic/http-2-vs-http-3
http://austincorso.com/2019/12/29/http3-quic-tls13.html

eBPF – Bypassing the stack

- Can be placed on hardware to kernel to System calls to user-space library in non-invasive manner

- Works on Containers
- Data from eBPF hooks can be feed to AI based observability systems to know why error happened
  - Difference between monitoring and observability is important here
- Observability data is usually huge this a trade-off should be analyzed when deciding the amount of observability data customer/user can afford

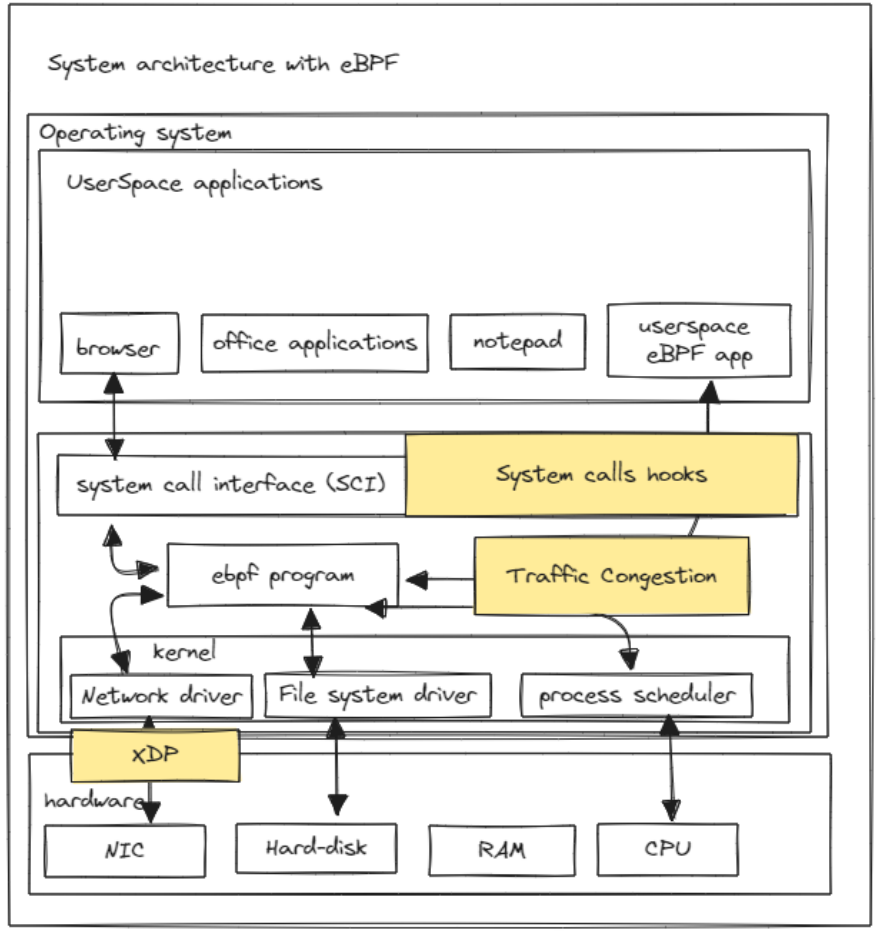


Credit: Puran Chand, Staff Engineer, Vmware/Broadcom/Omnissa

eBPF – Bypassing the stack

- eBPF allows user level programs to run at kernel level
- Uses system call hooks
- Strict compilation rules and code isolation

- More efficient implementing user-level control for low-level operations such as file-system, network etc.
- Efficient when compared to agents or sidecars



Credit: Puran Chand, Staff Engineer, Vmware/Broadcom/Omnissa

Conclusion

- Evolution is in progress
- Workloads became compute heavy and low-level components of the computing stack became a limitation in scalability and high-availability
- This led to virtualization of hardware, network stack, drivers etc.
  - Control and monitoring was at infrastructure level
- Next generation workloads were highly dynamic
  - Infrastructure level control of low-level stack was not enough
  - Application-level control -> stack duplication
  - Observability -> Intelligent control
  - Degradation of performance and resource utilization
- Application-level control executed directly at low-level stack