Trends in Cloud-Native

Performance & Efficiency

@melaniecebula

Disclaimers:

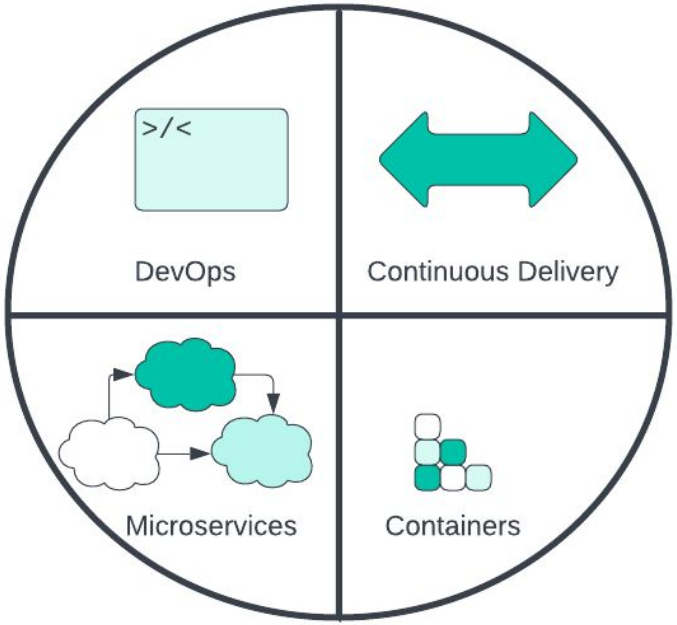
This talk contains my personal views, and is not about my employer or my employer's views.

This talk contains predictions and speculation, which could be wrong.

Agenda

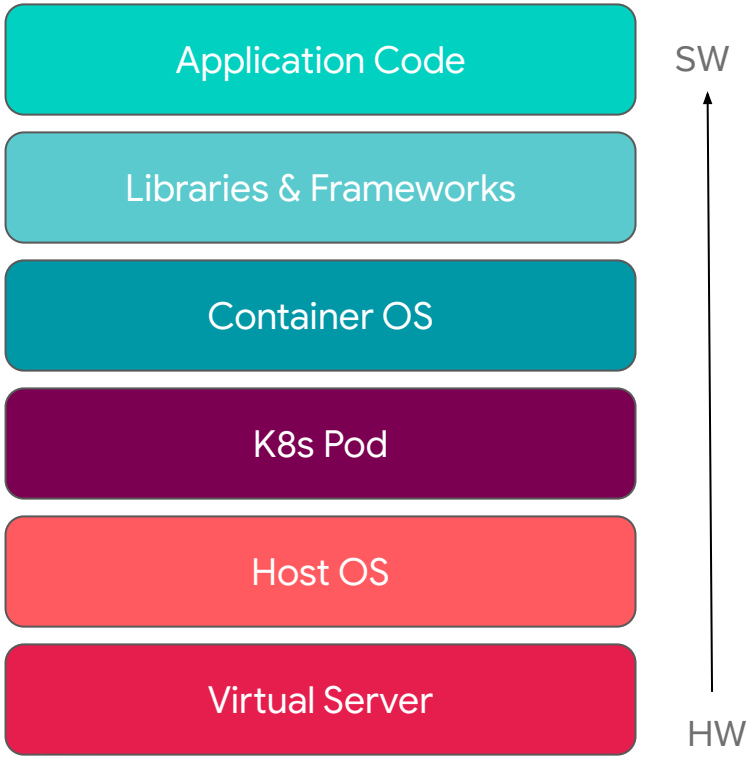
- Overview
- Hardware in the Cloud
  - Processors, Memory, Disk
  - Pricing
  - Instance Sizing
- OS, Kernel, & JVM
- Schedulers & Containers
- Perf Tools
- Networking in the Cloud
  - Service Mesh, eBPF

What is Cloud-Native?



LeadDev

What is Performance & Efficiency?



Performant Application Code

Profiling & Tooling

K8s HPA Tuning

JVM & Kernel Tuning

Modern Frameworks & SW

Modern OS, Kernel, and HW

Scheduling, Cluster Autoscaling & Binpacking

HW Capabilities

LeadDev

Cloud-native changes everything

Running compute as a distributed system with containerized multi-tenant workloads has a huge impact on how you approach performance & efficiency work.

Hardware in the Cloud

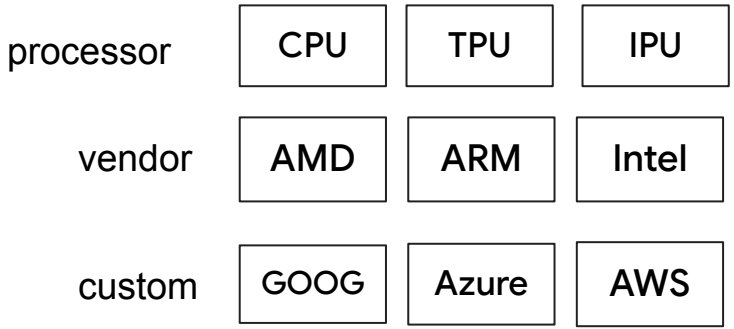
Processors, Memory, Disk

Trend: New Processors in the Cloud

- Intel Xeon:
  - 2022: Ice Lake
  - Late 2022/2023: Sapphire Rapids
    - Google announced system-on-chip (SoC) with custom Intel infrastructure processing unit (IPU)
- AMD:
  - 2022: 3rd-gen AMD EPYC “Milan”
  - 2023: 4th-gen AMD EPYC “Genoa”
- Ampere ARM-based processors:
  - 2022: Altra @ Google, Microsoft Azure
  - AmpereOne

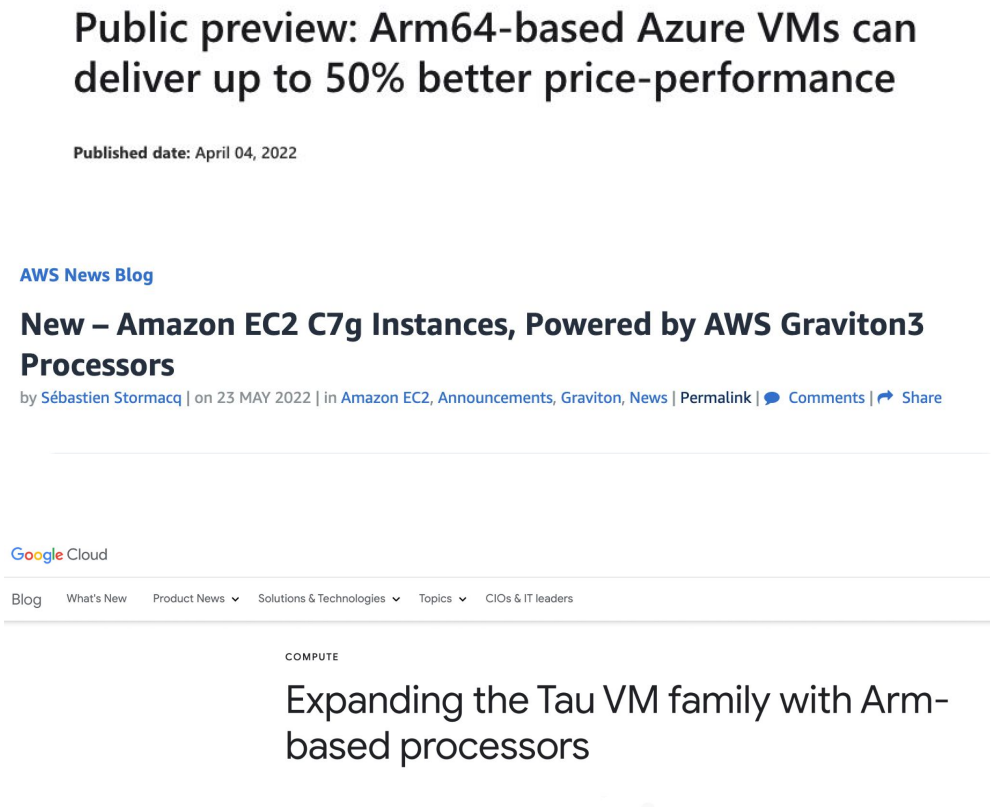
Trend: Custom processors in the Cloud

- "Systems on Chip" (SoC) design
  - Build vs Buy:
    - CPU, TPU, IPU
- Amazon ARM (Graviton 3)
- Ampere Altra Cloud Native Processors (ARM)
- Google Intel Xeon Sapphire Rapids with custom IPU



Trend: Rise of ARM

The rise of arm64 and SoC



Public preview: Arm64-based Azure VMs can deliver up to 50% better price-performance

Published date: April 04, 2022

AWS News Blog

New – Amazon EC2 C7g Instances, Powered by AWS Graviton3 Processors

by Sébastien Stormacq | on 23 MAY 2022 | in Amazon EC2, Announcements, Graviton, News | Permalink | Comments | Share

Google Cloud

Blog What's New Product News Solutions & Technologies Topics CIOs & IT leaders

COMPUTE

Expanding the Tau VM family with Arm-based processors

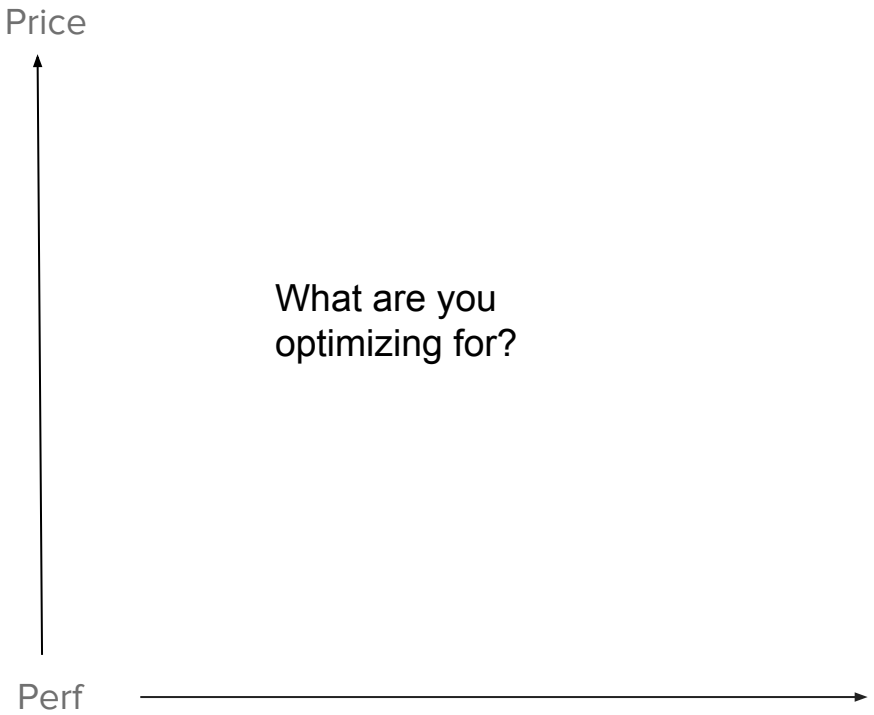
LeadDev

Trend: Cloud processors target Efficiency

- Cloud-provider specific optimizations (e.g. “secret sauce”)
- Custom Chips for Cloud -> Custom Chips for Cloud Native
- Optimized based on Microservices, Schedulers, Containers, etc
  - Microservices spend a lot of time in I/O
- Efficiency > performance
  - ARM
  - Intel p cores and e cores

Trend: More CPU Choices

- Price
- Performance
- Efficiency
- Also:
  - Architecture



What are you optimizing for?

Trend: Microservices & Memory

- Many workloads are actually memory I/O bound
- DDR5 (Double Data Rate 5) is a big deal
  - Advertised 50% better bandwidth
- Which processors have it?
  - AWS / Graviton3 (GA) first cloud processor to have DDR5
  - AMD Genoa
  - AmpereOne

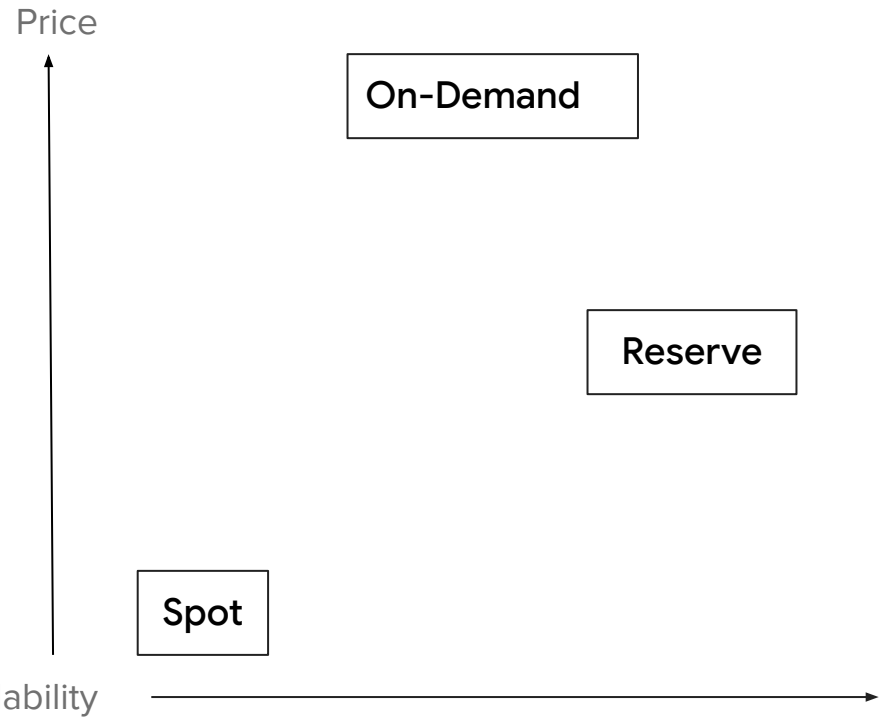
Trend: IOPS-heavy Services & Storage

- NVMe 2.0 (Non-Volative Memory Express)
  - Improved throughput, IOPS
- PCIe 5.0 (Peripheral Component Interconnect Express)
  - Improved Speed & Bandwidth
- Coming Soon!
- Also: Block Storage has improved

Pricing

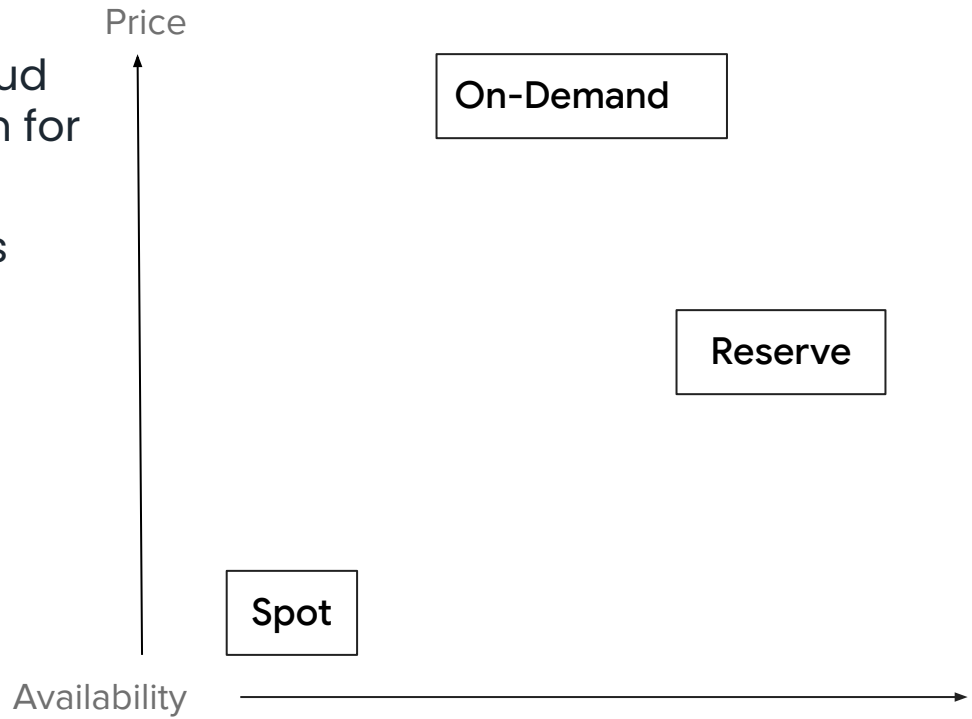
Pricing & Availability Guarantees

- Different tiers of pricing
  - On-Demand: Pay as You Go
  - Reserve: Pay up Front & Save
  - Spot: Save on Extra Capacity, But No Guarantees



Trend: Solving for excess (or lack of) capacity

- The overall trend is the all cloud providers now have a solution for excess capacity
- Market Dynamics (e.g. Azure's evict at your set price)



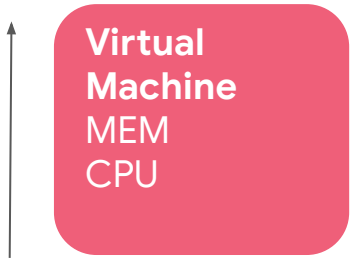
LeadDev

Trend: Performance, Price, and Availability

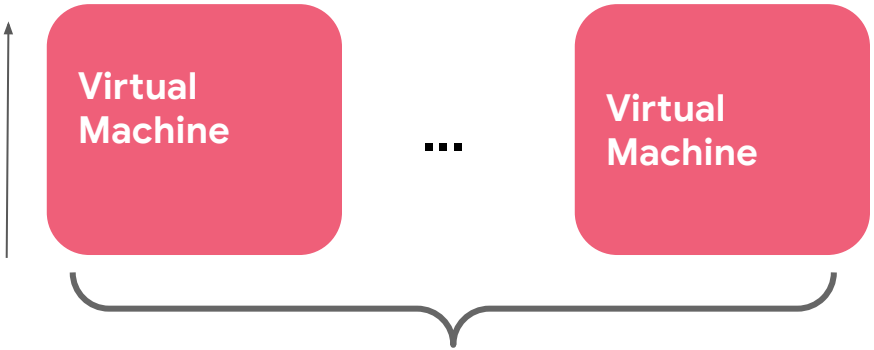
- What are you optimizing for?
  - Absolute _?
- More likely:
  - Pay for performance & availability where you need it
  - Save on costs with price/performance (e.g efficiency) where you don't

Instance Sizing

Instance Sizing: Vertical vs Horizontal Scaling

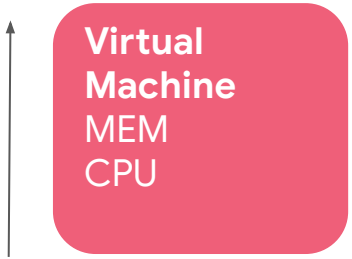


Vertical scaling:
increasing resources of an existing instance(s)



Horizontal scaling:
increasing number of instances

Horizontally scaling in the cloud



Vertical scaling: increasing resources of an existing instance(s)

- Limits on Vertical scaling (capped VM sizes)
- Larger instances have multi-socket costs (NUMA)
- Why pay it?
  - Use 2 single-socket instances instead of 1 two-socket instance
- Horizontally scale instead!

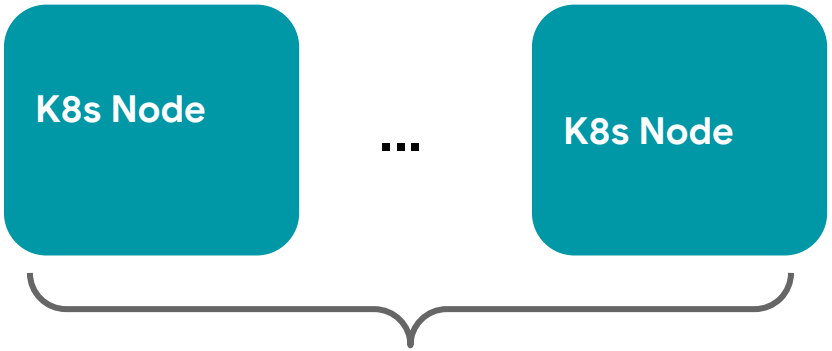
But in K8s, there's also Overhead

K8s Node
Daemons
System Pods
Agents

Scheduling Overhead: Resources used per instance by scheduling logic.

- If instance overhead is high enough, you may want to vertically scale as much as you can
- Minimize overhead, vertically scale, then horizontally scale

And Cluster Overhead



Horizontal scaling: increasing number of clusters

Scheduling Overhead: Resources used per cluster by scheduling logic.

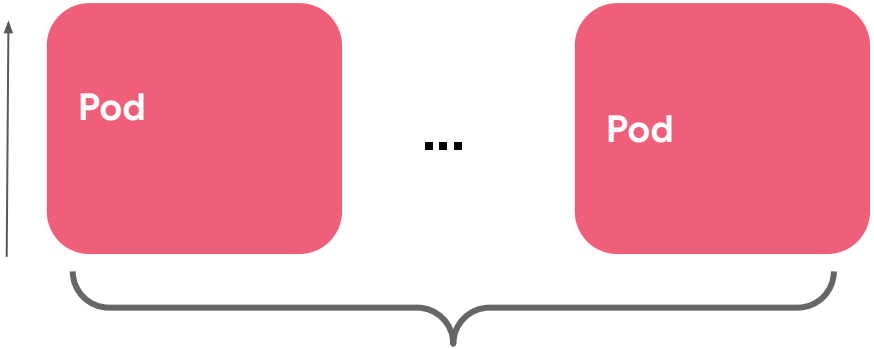
- Clusters themselves also have per-cluster overhead
  - Not to mention operational overhead
- "Vertically scale" each cluster until you hit the max number of instances
- "Horizontally scale" by adding another cluster

Hardware & Multi-Tenancy

- Before: Optimize the instance for the workload
- Now: Multiple tenants per instance
- New problems:
  - Resource contention
  - Scheduling challenges
  - Binpacking

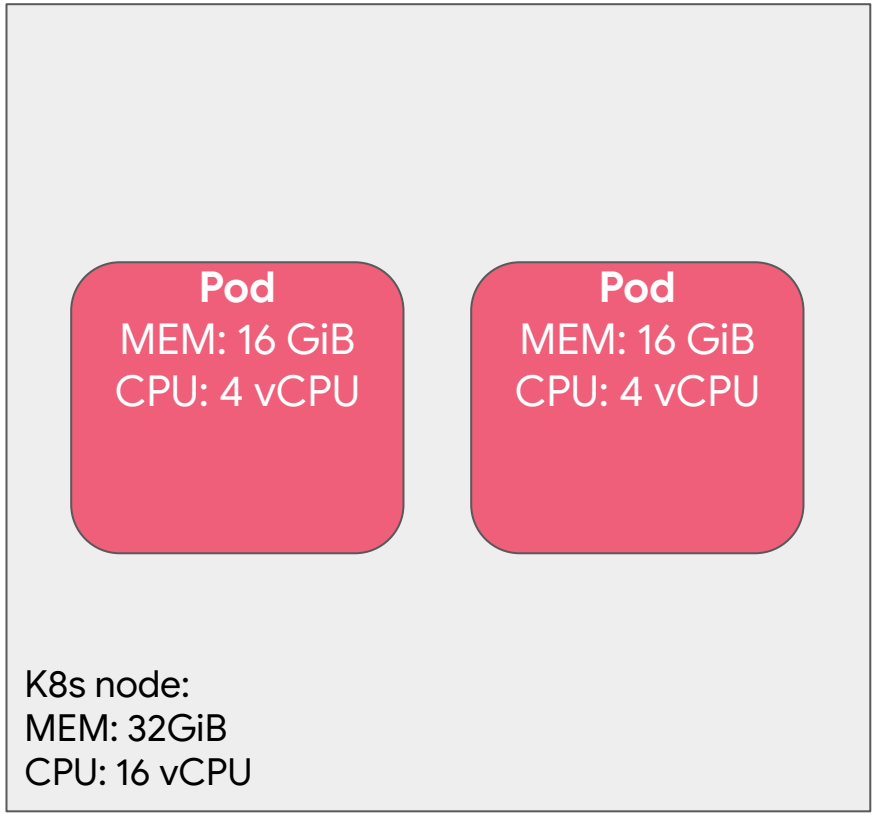
K8s

Vertical scaling:
increasing resources
of an existing pod(s)



Horizontal scaling:
increasing number
of pods

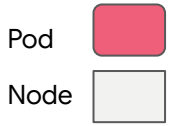
Example: Inefficient Binpacking

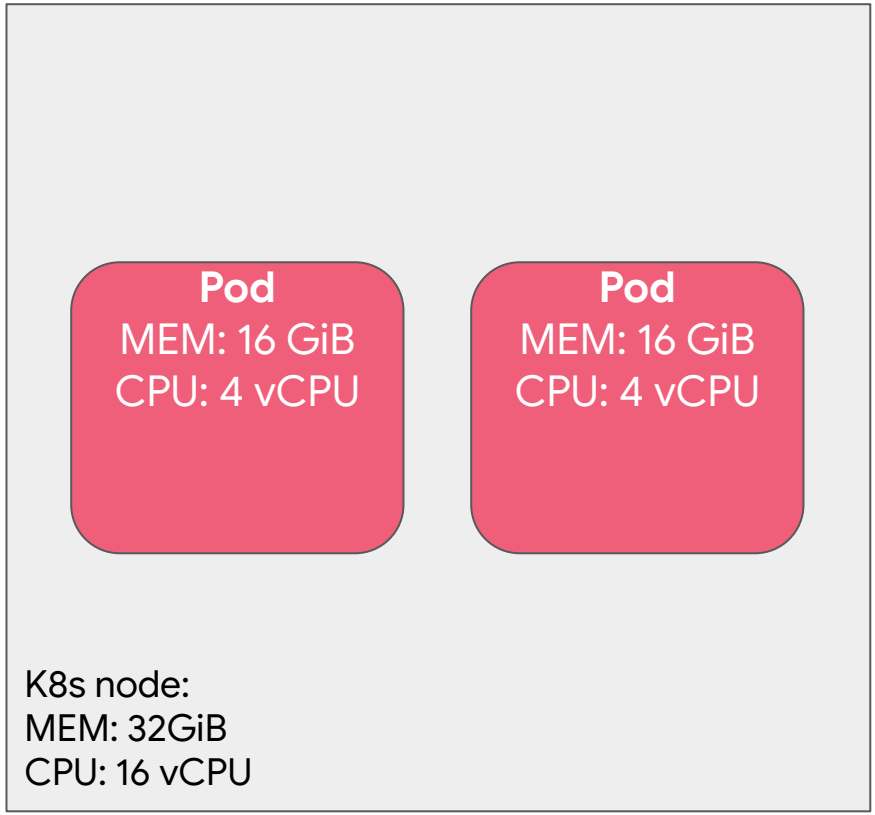


Pod
Node

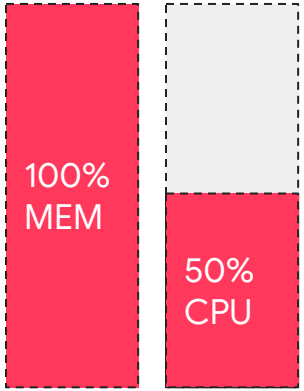
LeadDev

Example: Inefficient Binpacking



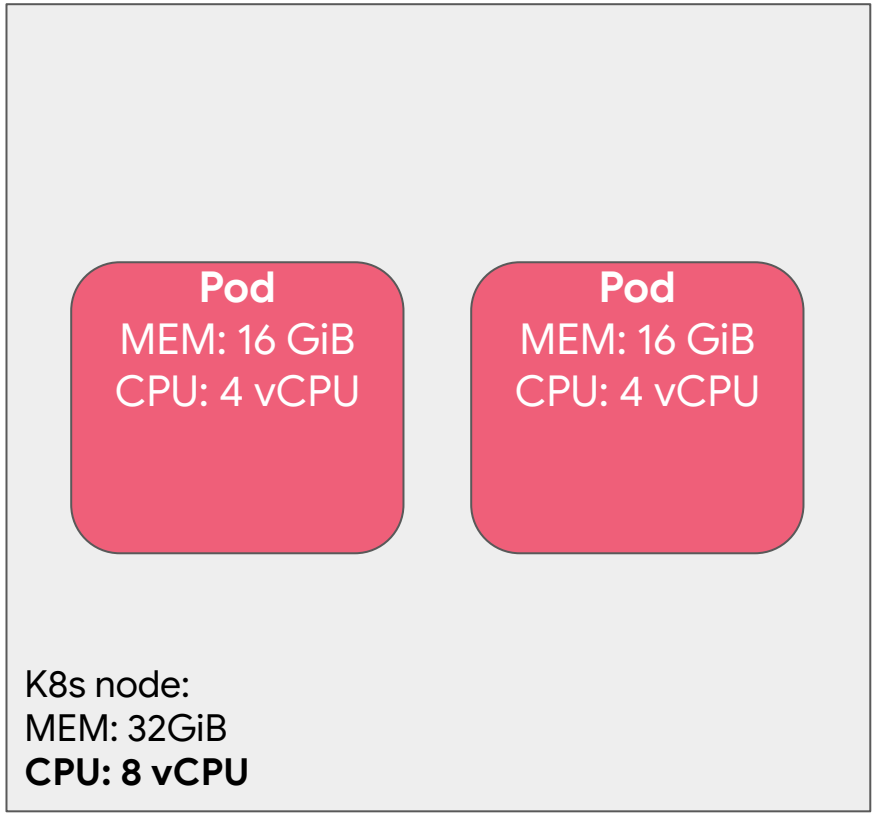


Node Utilization



LeadDev

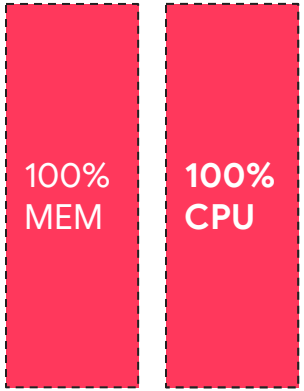
Example: Efficient Binpacking



Pod

Node

Node Utilization



LeadDev

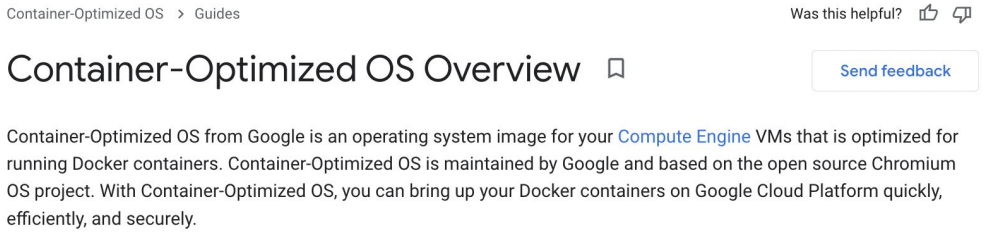
Which HW should you use?

- Requirements:
  - Resources (CPU, MEM, GPU, Storage)
  - Architecture
- Optimizations:
  - Price
  - Performance
  - Availability
  - Sizing

OS, Kernel, & JVM

Trend: custom cloud OS

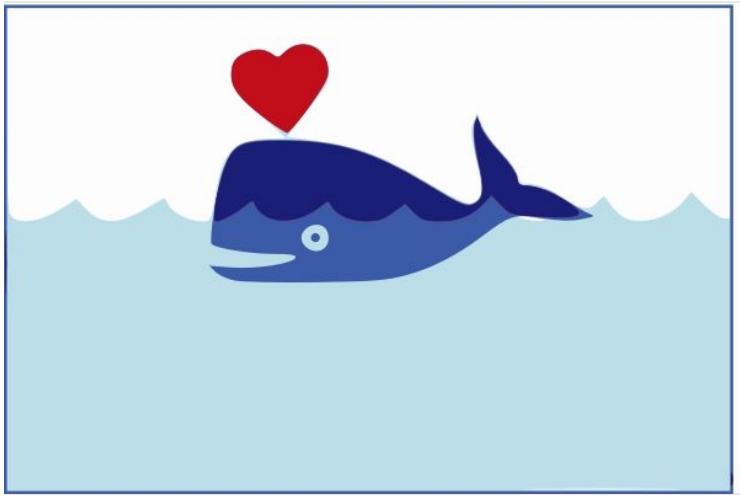
- Cloud-specific OS' based off open-source OS's
  - e.g. AL2, AL2022, Container-Optimized OS (COS)
- Clouds offer open-source OS's too
  - e.g. Ubuntu, RHEL, Fedora, Debian



reference: https://aws.amazon.com/linux/amazon-linux-2022
reference: https://cloud.google.com/container-optimized-os/docs/concepts/features-and-benefits

Trend: container OS optimizations

- Container-optimized
- Slimmer
- Distroless
  - Application & runtime deps only
  - No package managers
  - No shell access



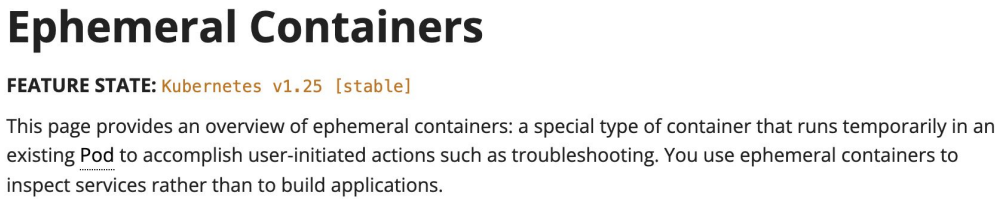
LeadDev

reference: https://freesvg.org/whale-animation

Trend: Secure Profiling & Debugging

In a Container World

- Distroless (no shell access), container hardening (non-root)
- Some perf tools are not “container aware”
- Debugging must evolve
  - e.g. with ephemeral containers for profiling



Ephemeral Containers

FEATURE STATE: `Kubernetes v1.25 [stable]`

This page provides an overview of ephemeral containers: a special type of container that runs temporarily in an existing Pod to accomplish user-initiated actions such as troubleshooting. You use ephemeral containers to inspect services rather than to build applications.

Kernel Improvements

5.10

- New features & perf improvements
  - Optimizations for new processors
  - BPF improvements
  - Improved write perf, throughput, better compatibility with storage devices
  - MultiPath TCP
  - PSI
  - io_uring (for I/O- bound workloads)

Kernel Improvements

5.15+

- 5.15
  - New standard for latest Cloud OS's
  - Continued HW improvements (Intel Alder Lake, AMD)
  - I/O improvements
- 5.19
  - Many Intel & AMD improvements
  - Networking improvements (io_uring)
- 6.0
  - Improved NUMA balancing
  - Improved behavior in both reduced-capacity (load balancing) and spare capacity (idle CPU behavior)
  - Improved core scheduling

JVM Improvements

- Java 11
  - ZGC
  - **Parallel full G1GC**
  - Low-overhard heap profiling
  - GC adaptive thread scaling
  - JMH JDK microbenchmarks
- Java 15
  - **ZGC production-ready**
- Java 17
  - Vector API
  - Parallel GC improvements
  - **Perf improvements incl. G1 and ZGC**
- Java 18/19
  - Light on perf / preview mode

LeadDev

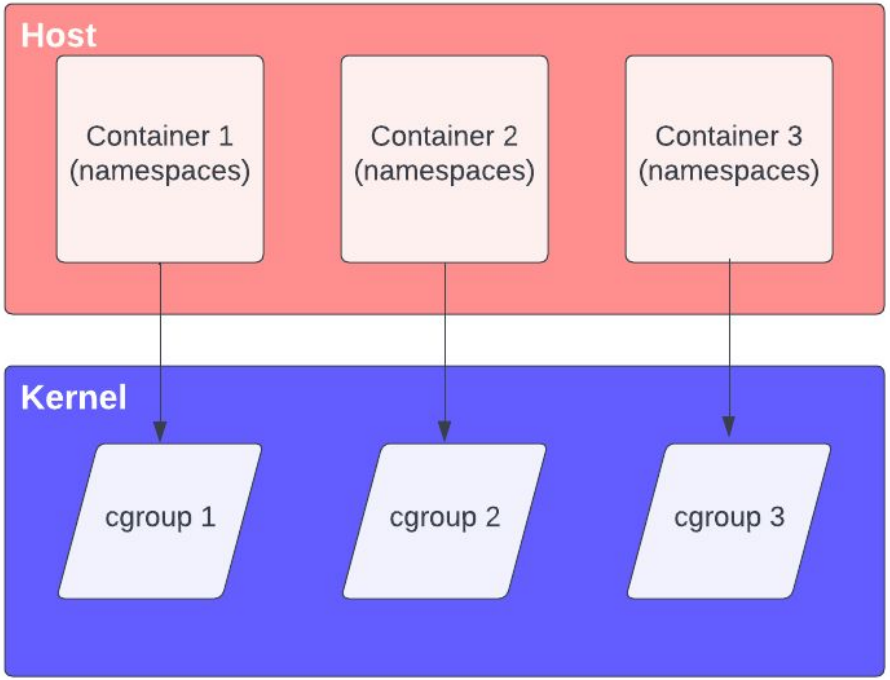
reference: https://freesvg.org/java

Schedulers & Containers

Trend: better container resource handling

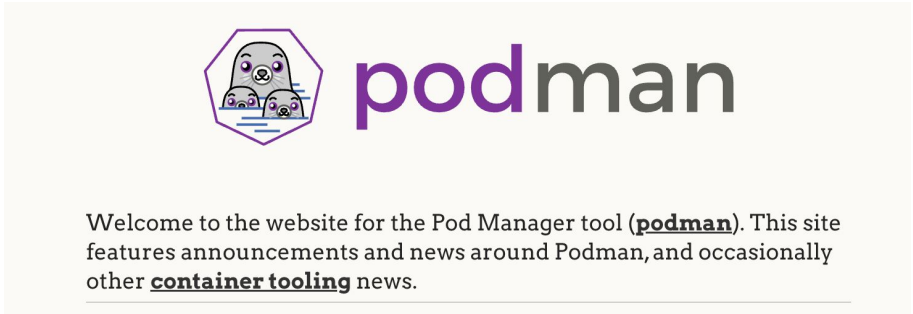
Cgroup v2

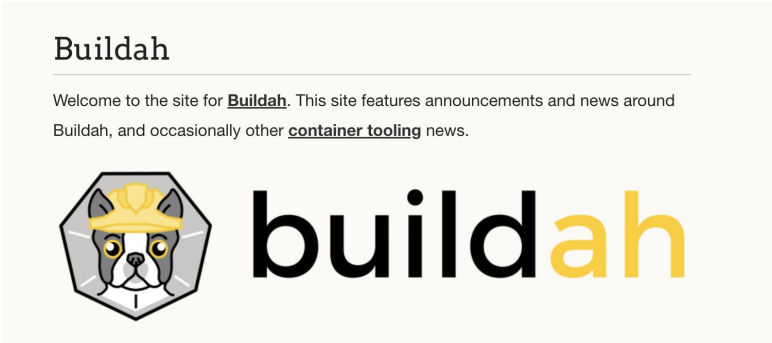
- Containers are implemented with cgroups
- Pressure Stall Information (PSI)
  - Identifies and quantifies disruptions caused by *all* resources crunches
  - Avoid OOM kills!
- Better resource allocation and isolation



Trend: “Dockerless”

- Specifications (OCI, CRI) for containers and container runtimes
- Docker being replaced by other tools
  - K8s replace Docker Daemon with CRI-O
  - New tools (Podman, Buildah) for building & running OCI containers



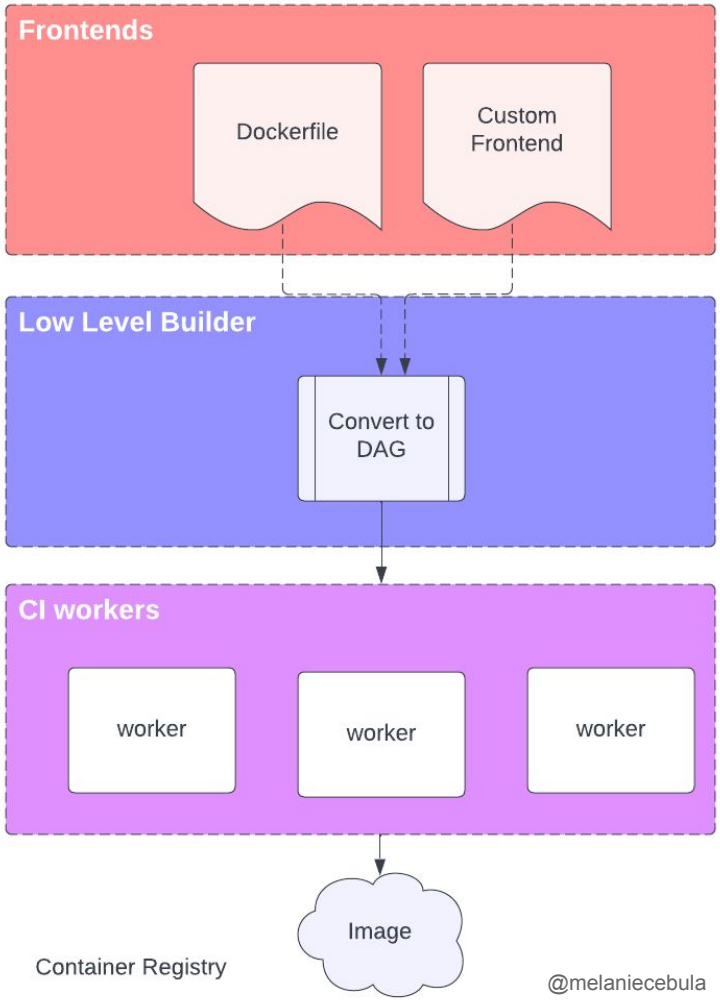


LeadDev

reference: https://podman.io/
reference: https://buildah.io/

Trend: “Dockerless”

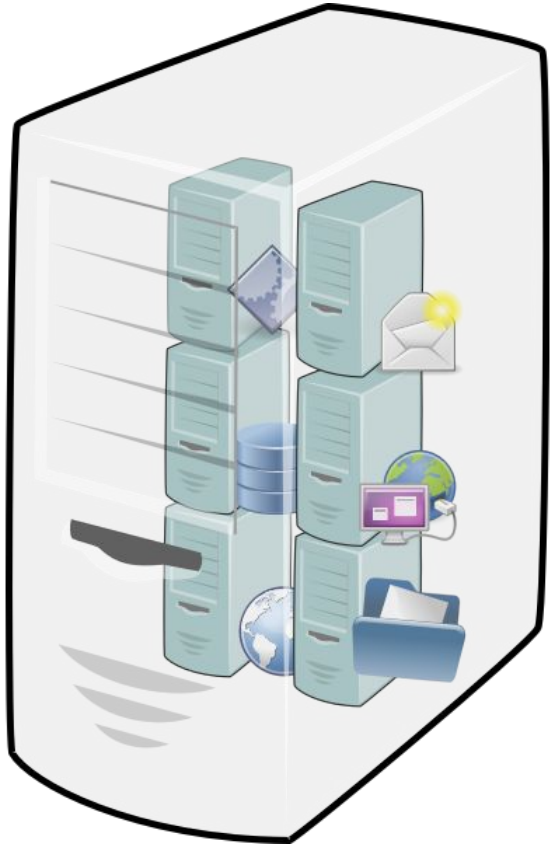
- Docker backend -> replaced by Buildkit
- Docker frontend (Dockerfile) -> replaced by custom image definition frontend?
  - More control over abstractions (OS, arch, packages)



Trend: (Lightweight) VMs are cool again?

E.g. AWS Firecracker

- Advent of microVMs
- Fast start time (similar to containers)
- Fast performance (similar to HW)
- Dedicated kernels, better security, better resource isolation



LeadDev

reference: https://freesvg.org/virtual-machine-host

Trend: “VM support” in schedulers

- Can you use VMs if you’re using K8s?
- Actually, yes!
- Sysbox, container runtime that supports running VM workloads in VM-approximate containers

Nestybox

Sysbox Container Runtime

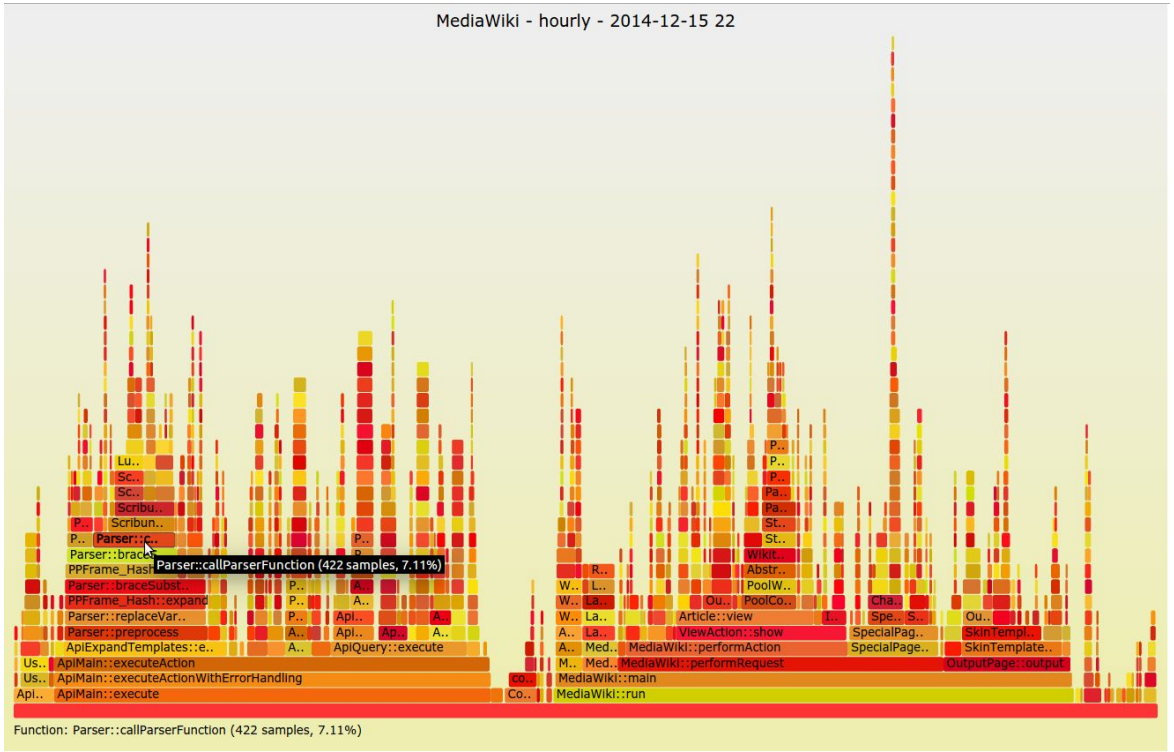
(Community Edition)

license Apache-2.0 build passing chat on slack

Perf Tooling

Perf Tooling

Flamegraphs



reference: https://commons.wikimedia.org/wiki/File:MediaWiki_flame_graph_screenshot_2014-12-15_22.png

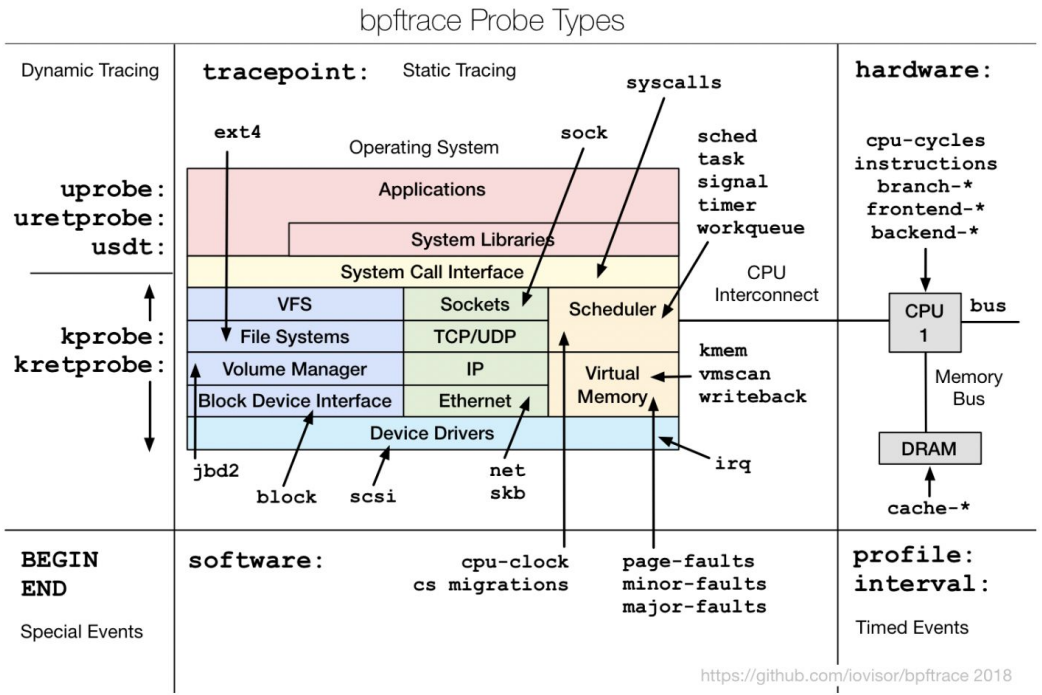
Trend: continuous -> automated analysis?

Continuous Profiling and Analysis

reference: https://github.com/Netflix/flamescope

Next Gen Perf Tooling

bpftrace



reference: https://github.com/iovisor/bpftrace

Prediction: all-in-one cloud debugging tool?

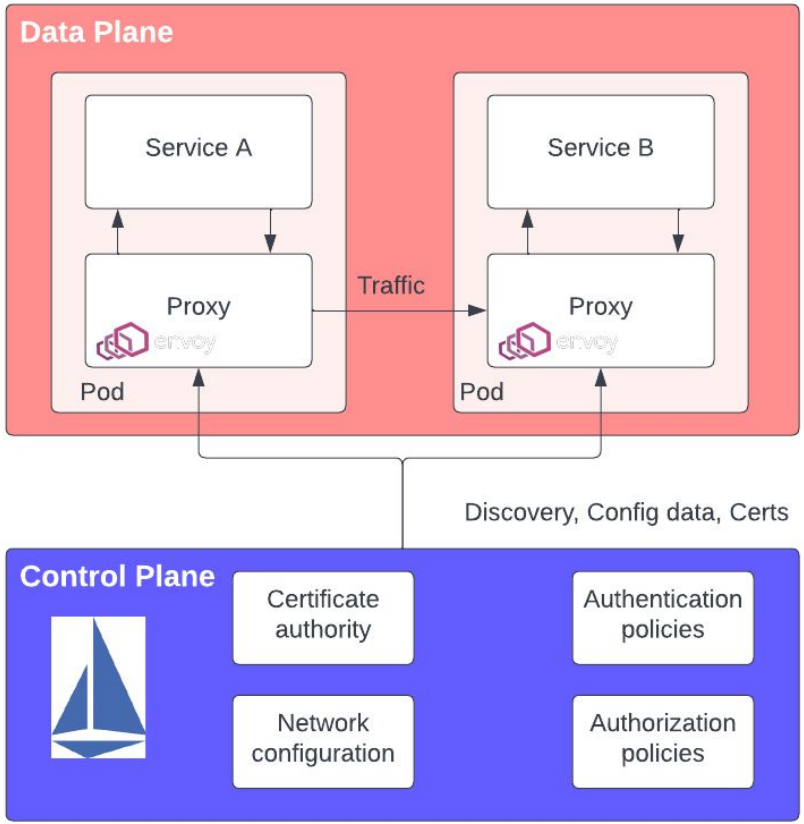
- The biggest issue with all these tools is none of them “zoom out” and “drill down” across layers
- Layers: Application, Container, Scheduler, OS, Kernel, HW, etc.
- More complexity than *ever* in the cloud
- Debugging is time-consuming and expensive

reference: https://github.com/iovisor/bpftrace

Networking in the Cloud

Service Mesh

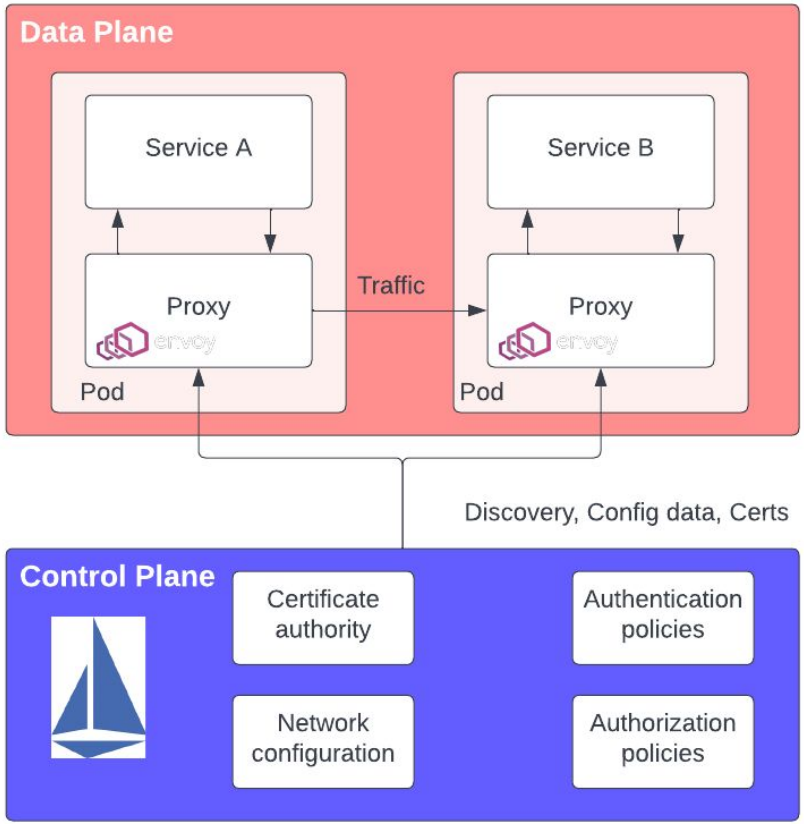
- Service-to-Service communication moved outside of applications
- Avoids language-specific libraries!



LeadDev

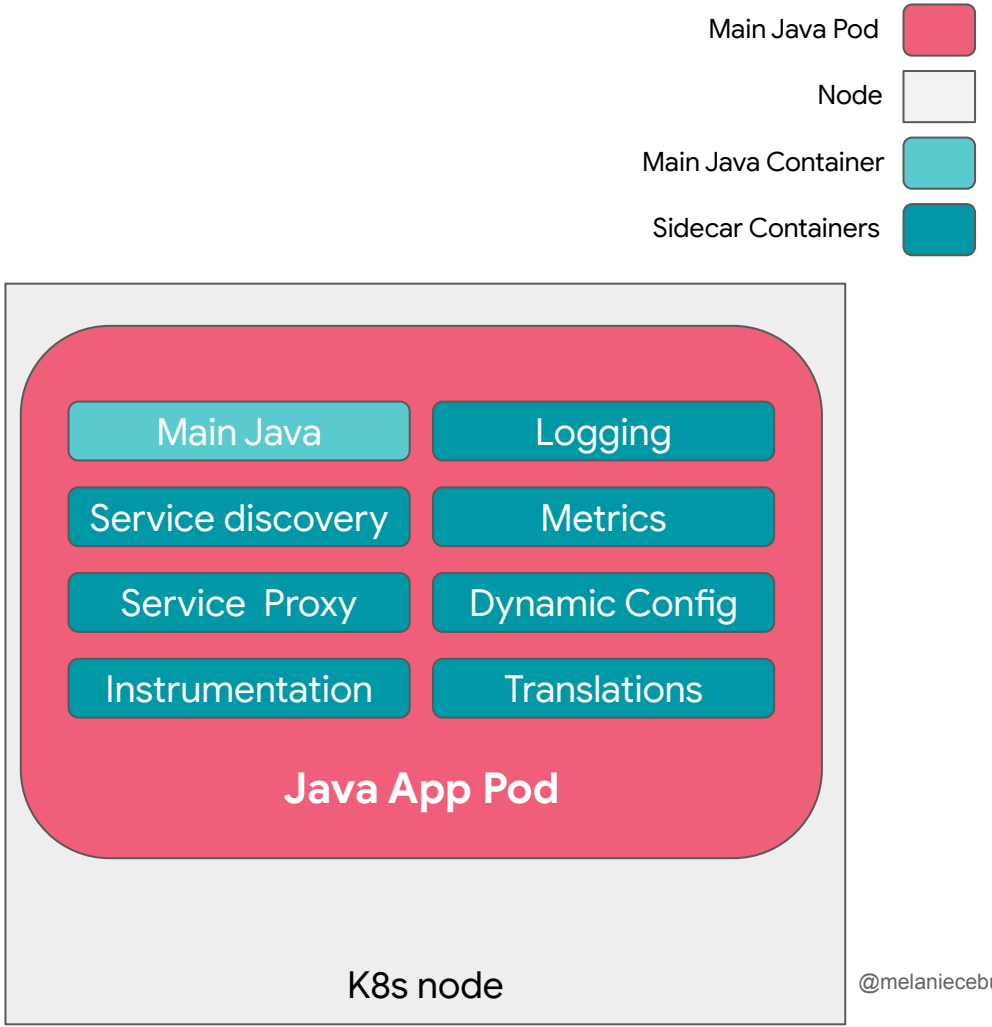
Service Mesh

- Performance:
  - Rightsizing proxies for CPU/MEM
  - Depends on size of configuration state (e.g. number of listeners, clusters, and routes)



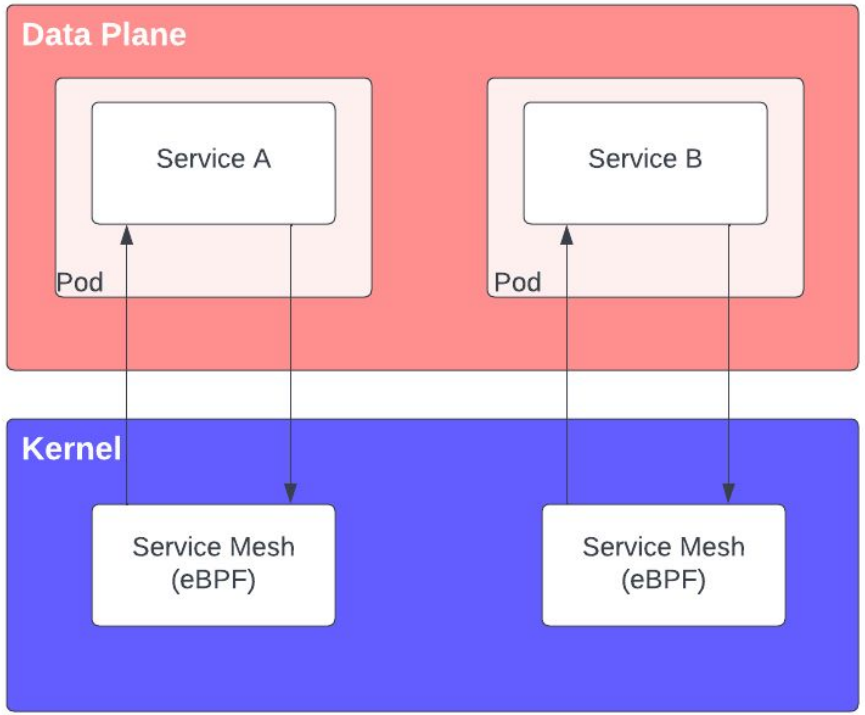
Sidecars

- General problem:
  - As shared concerns are moved into sidecar containers
  - Sidecar resource footprint is significant



Trend: Use eBPF for networking… or another model

- eBPF:
  - Allows applications to do certain types of work in the kernel
  - Potential for network, security, observability usecases
  - Way better performance
  - But, harder to write & reason about
- eBPF in the CNI too!



Summary

1. Custom Cloud Hardware
   a. SoC, Rise of ARM, targeting efficiency
2. Era of CPU Choice
   a. Price, Performance, Availability, Sizing, Architecture
3. Microservices change resource requirements & scaling considerations
4. Multi-tenancy & clusters introduce challenges
   a. resource contention and binpacking efficiency
5. Custom Cloud OS, Container-optimized OS, and Distroless
6. Cgroups v2 & Kernel
   a. better resource contention & isolation (PSI)
7. “Dockerless” trend
   a. container runtime, builds, interface
8. Lightweight VM offerings & integration into schedulers
9. Continuous Profiling and analysis towards more automation and all-in-one debugging
10. eBPF or another model will change networking in the cloud

Thank you.

@melaniecebula