K-Bench: A Framework to Prescriptively Benchmark Kubernetes

Yong Li and Karthik Ganesan

VMware

Motivation

- When it comes to performance, multiple aspects to look at:
  - Control plane aspects when deploying and manipulating K8s objects:
    - Responsiveness
    - Scalability
    - Resiliency
  - Data plane or application performance:
    - Core Infrastructure capabilities (Compute, I/O, network)
    - Resource efficiency
    - Performance isolation characteristics
- Lack of a configurable and an accurate tool to benchmark these different aspects

K-Bench Overview

- A configurable framework to prescriptively deploy and manipulate K8s objects
  - Benchmark both control and data plane aspects with ease
    - E.g., Deploy 1000 nginx pods concurrently and observe at pod startup latencies
    - E.g., Deploy a persistent volume and record asynchronous rd/wr bandwidths to it from a pod
- Extensible design, rich configuration space
  - Bring your own workloads and orchestrate your target workflows with ease
  - A simple and prescriptive config file represents a use-case
- Provides an intuitive set of performance metrics and detailed diagnostic data to help resolve performance issues

K-Bench Overview

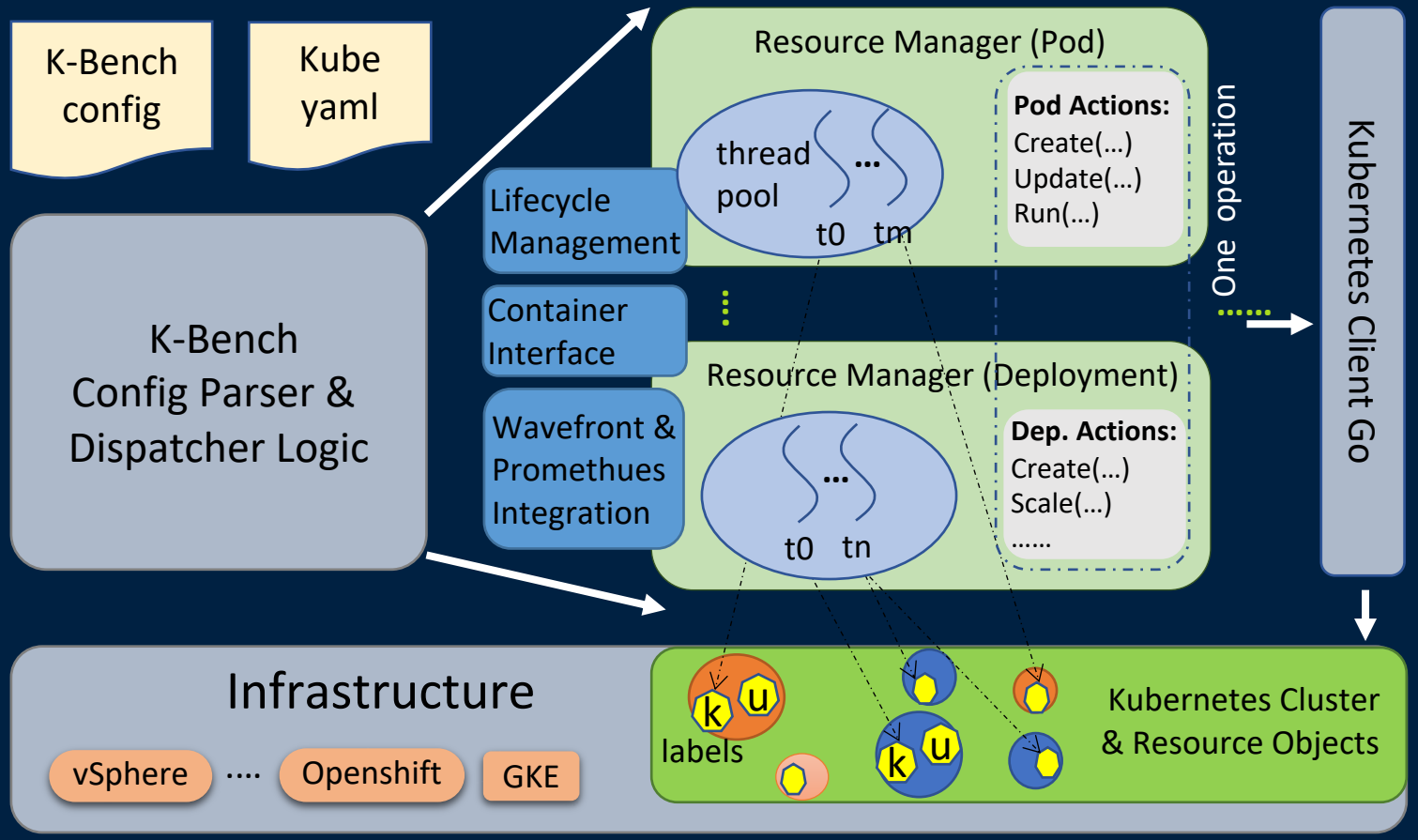
- Control Plane:
  - Accurate and fine-grained critical path latencies
    - E.g., pod scheduling, initialization, startup latencies
  - Client-server hybrid & event driven approaches
    - Tackle eventual consistency in K8s
- Data Plane:
  - Evaluate application performance with containerized benchmarks in K8s pods
  - Built-in workloads to stress specific infrastructure resources
    - E.g., Redis memtier (compute/memory) , FIO (I/O), Iperf3/qperf (network)
  - Scale-up, scale-out & mix resource usage to study infrastructure performance
  - Built-in blueprints of workflows to evaluate different aspects

K-Bench Control Plane Basics & Terminology

- Control plane action
  - CREATE, DELETE, LIST, SCALE, etc.
  - Can be resource type specific (e.g., CREAT for Pod, SCALE for Deployment)
  - Run with action specific options (yaml spec)
  - Multiple actions run one after another on the same resource object can form a chain
- Operation
  - Contains a collection of action chains, each executed for a particular resource type
  - Action chains for different resource types run in parallel
- Predicate
  - A condition under which an operation is triggered
- Labels & filters
  - K-Bench labels (k-label) and use specified labels (u-labels)

K-Bench Control Plane Framework

- Config file specifies:
  - Resource types
  - Actions & operations
  - Concurrency
  - Filters & labels
  - Predicates
  - Execution plan
- Data plane:
  - Container interface
- Telemetry & monitoring:
  - Wavefront & Prometheus integration
- Run on many k8s platforms



Example Resources, Actions & Config Options

- Pod supports CREATE, LIST, GET, RUN, COPY, UPDATE, and DELETE actions.
  - All actions have some common options, e.g.: *Count* (concurrency) and *SleepTime*
  - CREATE has *ImagePullPolicy*, *Image*, *YamlSpec*, etc., options.
  - RUN supports *Command* option.
  - COPY supports *LocalPath*, *ContainerPath,* etc., options.
  - CREATE, LIST, RUN, COPY provide *LabelKey* and *LabelValue* options.
- Deployment:
  - Supports all pod actions, and SCALE.
  - CREATE has specific options such as *NumReplicas*
- ReplicationController and StatefulSet: similar to Deployments.
- Other resource types:
  - Namespace, Service, ConfigMap, Endpoints, Event, ComponentStatus, Node,
  - LimitRange, PersistentVolumeClaim, PersistentVolume, PodTemplate, ResourceQuota,
  - Secret, ServiceAccount, Role, RoleBinding, ClusterRole, ClusterRoleBinding, etc.,

Example Configurations

Predicate:

- Operation runs when condition is met

Typed actions inside one operation:

- *Create* deployments with dep.yaml, with *4* go routines
- *Create* pods with *pause* image and then *Run* start_work.sh in pods with given labels, with *2* go routines

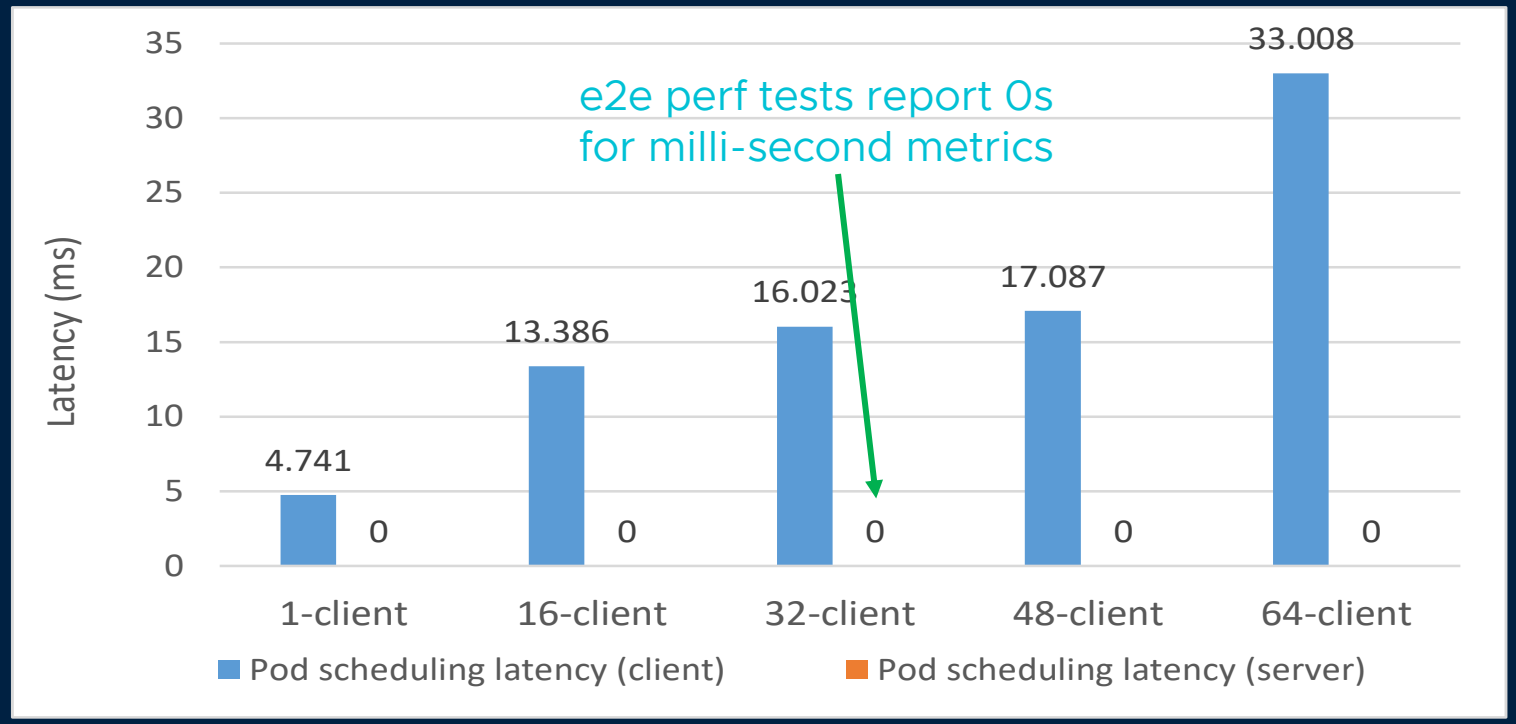
More operations can be configured

- Operations run sequentially

```
{
 "Timeout": 60000,
 "Operations": [
  { # first operation
   "Predicate": { "Resource": "mypod/initializer" }
   "Deployments": {
    "Actions": [ {
       "Act": "CREATE",
       "Spec": { "YamlSpec": "./dep.yaml" } }
    ],
    "Count": 4
   },
   "Pods": {
    "Actions": [
     { "Act": "CREATE" },
     { "Act": "RUN",
       "Spec": {
        "Command": "/root/start_work.sh",
        "LabelKey": "podName",
        "LabelValue": "myPod" } }
    ],
    "Count": 2
   }
  },
  { # second operation...}
 ]
}
```

Benchmarking Control Plane

- SIG compliant API metrics
- Fine-grained critical path latencies
  - E.g., scheduling, initialization, image pulling, startup latencies
- Improved accuracy
  - Client-server hybrid timing



Benchmarking Data Plane – Orchestrate Any Workflow

- Leverage K-Bench container interface to orchestrate real world workloads
  - CREATE -> deploy labelled K8s resources with your target containers
    - These labels will enable user to filter and select specific objects on which following operations act on
  - COPY -> copy workload artifacts into containers
  - RUN -> run commands inside pods to trigger workflows
    - Use condition-based predicates to trigger workflows
    - Predicates can be K8s system based or evaluated in-container
    - E.g., wait until a server process is up in server pod before a client pod generates load
  - COPY -> copy results out of the pods to the client
  - DELETE -> delete the created artifacts and trigger next workflow

Benchmarking Data Plane – Leverage Pre-integrated Blueprints

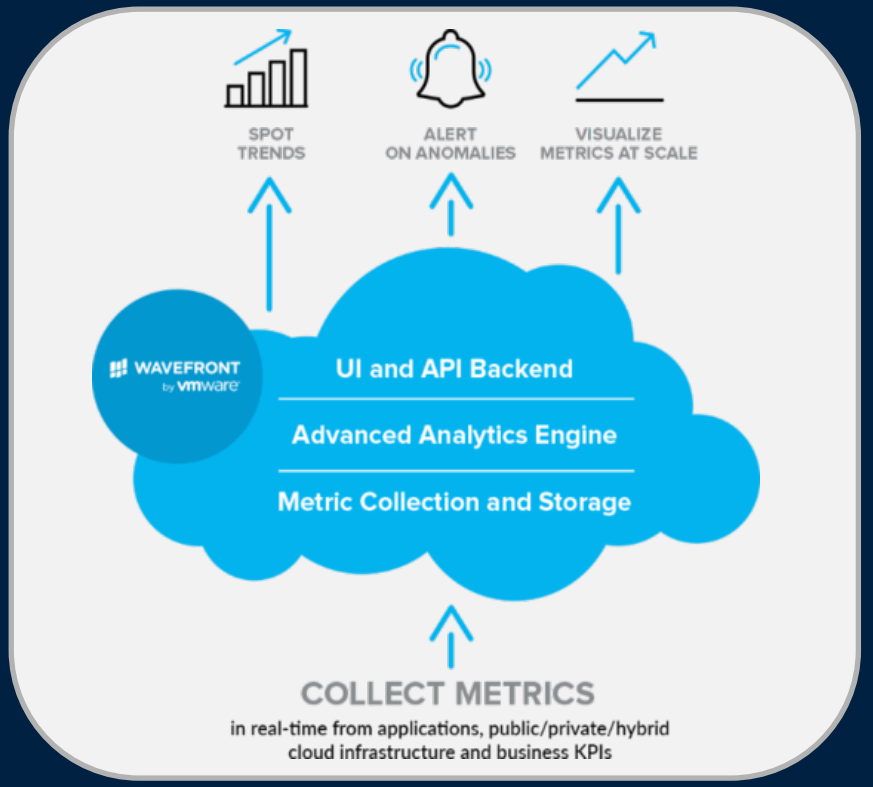
- Containerized workloads to stress different infrastructure resource dimensions
  - CPU, Memory, I/O, Network, hardware accelerators [Future]
  - Integrated workloads: Redis Memtier, FIO, IOping, Iperf3, Qperf, etc.
  - Pre-Integrated blueprints: Redis Memtier pod density for aggregate performance

Metrics	Resource category	Benchmark	Notes
Txn throughput	CPU/Memory	Redis Memtier	Aggregate transaction throughput
Txn latency	CPU/Memory	Redis Memtier	Transaction latency
I/O bandwidth (IOPS)	I/O	FIO	Rd/Wr bandwidth for various rd-wr ratios, block sizes on ephemeral and persistent volumes
I/O Latency (ms)	I/O	Ioping	I/O latency on Ephemeral and Persistent volumes
Network b/w	Network	Iperf3	Inter-pod TCP, UDP bandwidth. Blueprints with varying pod placements on nodes, zones, regions
Network latency	Network	Qperf	Inter-pod network latency for TCP and UDP packets. Blueprints with varying pod placements

Diagnostic Data and Dashboarding

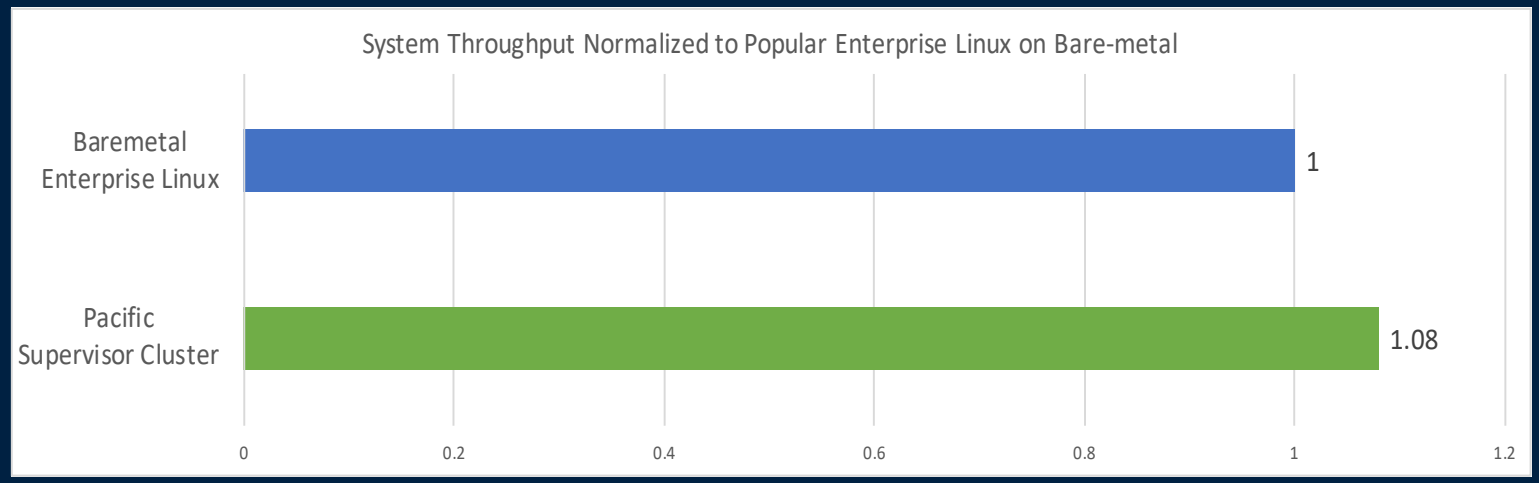
- End results only paint the final picture
  - Analysis and improvements need infrastructure diagnostics

- Diagnostic Telemetry:
  - Support to inject performance and diagnostic data to dashboarding services like Wavefront/Grafana
  - Distributed Telegraf data collectors with generic output plugins
  - 1000s of hand-crafted performance metrics can be monitored for Linux and ESX K8s nodes



Dataplane Use-cases – Example 1

- Cluster-level aggregate transaction throughput
  - Deploy a standard Java benchmark inside multiple K8s pods
  - Find maximum cluster level aggregate transaction throughput
  - Compare performance of two K8s clusters with same hardware resources
    - Further explain the results using insights from the infrastructure diagnostics



VMware CEO Pat Gelsinger and PE Joe Beda announce project Pacific at VMworld 2019 opening keynote using results generated by K-Bench

https://blogs.vmware.com/performance/2019/10/how-does-project-pacific-deliver-8-better-performance-than-bare-metal.html

Dataplane Use-cases – Example 2

- Example Blueprint: "dp_network_internode"
  - Automatically deploys two pods on two nodes using anti-affinity rules
  - Iperf3 run across the pods
    - Provides inter-pod TCP/UDP bandwidth
  - Qperf run across the pods
    - Provides inter-pod TCP/UDP latency

- These blueprints can be run as a suite to get all these key metrics in a nutshell

Summary

- K-Bench is a highly configurable and easy-to-use benchmark framework to evaluate Kubernetes performance
- It can be valuable for competitive benchmarking of K8s platforms, identify and improve performance issues
- K-Bench is open sourced: https://github.com/vmware-tanzu/k-bench
  - If you are interested, please consider using the tool, providing us feedback and contributing to the project

Thank You!
Any Questions?