Only Slightly Bent:
Uber's Kubernetes Migration Journey
Microservices

Yunpeng Liu @Uber

Uber

Agenda

- Uber Compute Infrastructure Overview
- Motivation to Migrate to Kubernetes
- Evaluation of Kubernetes for Uber
- Broader Integration with Uber Ecosystem
- Key Takeaways
- Q&A

Uber

15+ billion trips

15M+ trips per day

6 continents, 65 countries and 700+ cities

100M monthly active users

3.9M+ active drivers

27,000+ employees worldwide

3000+ engineers worldwide

Compute Infrastructure Scale

1,000s of Microservices

5,000+ of Builds per day

1,000+ Microservices deployed per day

10,000+ instances deployed per day

100K+ Service containers per cluster

~1M Batch containers per day

35+ clusters

Stateless Services

- 1,000s of microservices
- Mostly containerized and running on Mesos
- A long journey full of migrations
  - Bare metal
  - Containerized with static placement
  - Mesos + Aurora
  - Mesos + Peloton
- Multi-Region & Multi-Zone
- Tripod strategy

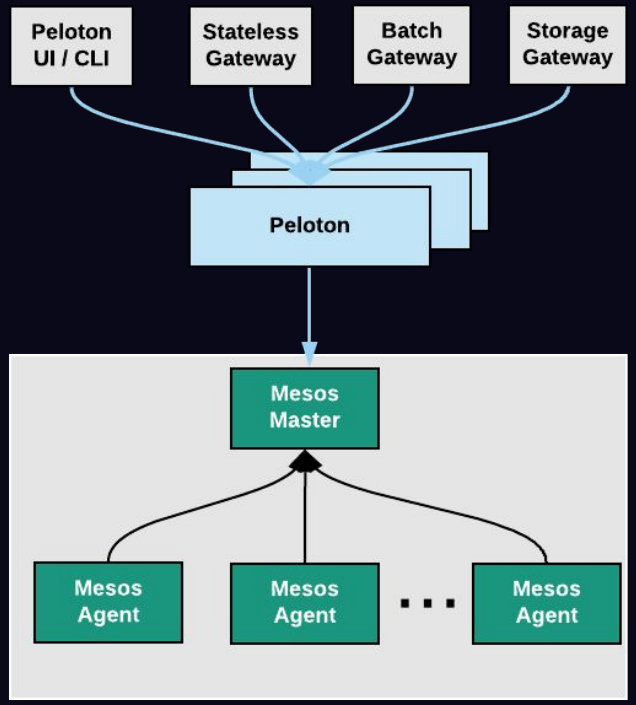
Other Compute Use Cases

- Large volume of batch workloads on Compute clusters
- Co-locating mixed workloads (stateless and batch) on shared clusters
- Workloads requiring hardware customization running on dedicated hardwares
- Distributed deep learning on GPUs

Customized Workload Scheduler

Peloton

- Unified resource scheduler for co-locating mixed workload on compute clusters
- Integrates with Spark, TensorFlow, uDeploy
- Can be run on-premise or in the Cloud
- Elastic resource sharing among teams and organizations



Motivation to Migrate to Kubernetes

Business requirements - AuthN and AuthZ for inter-service communication (secure service mesh)

- Existing solution is not extensible
- Native support for pod in Kubernetes
- Sidecar, Security plugins available out of the box
- Init container support
- Thriving community support/convergence for Kubernetes
- Natively supported by new Cloud native applications

Evaluation of Kubernetes for Uber

Concerns

- **Scalability**
  - Clusters scaling ability beyond 5k nodes
  - Volume of batch workloads (1m/1k containers launched per day/second)
- **Portability**
  - Customized stateless and batch service orchestration system
  - Unique service deployment characteristics
  - Customized cluster networking setup

Scalability

Benchmarking etcd

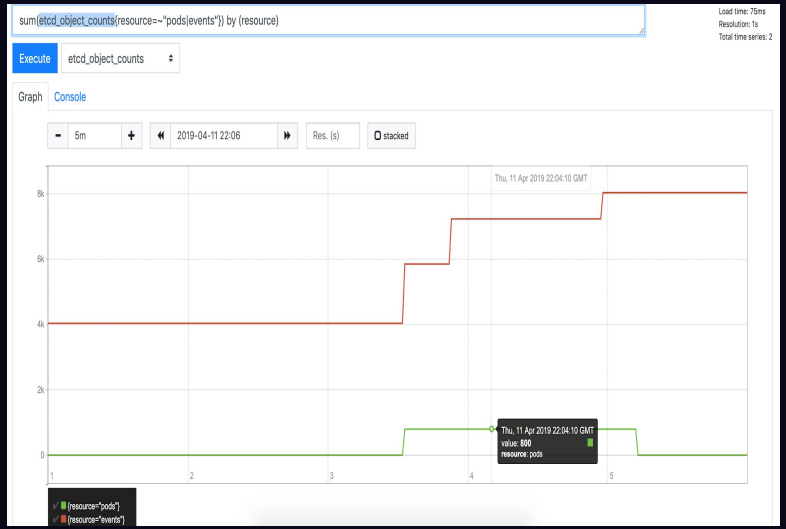
- Write throughput - Key size and value size
  - 50k writes / sec, 150k reads / sec
  - Large write value size (> 256 bytes)
  - Value size affects write throughput
- Write throughput - number of watches
  - Drops significantly after hundreds of watches
- Memory consumption - number of watches
  - Millions of watches with a few GB memory

```
benchmark --endpoints="http://${H1}:2379"
--target-leader --conns=100 --clients=1000
put --key-size=8 --sequential-keys
--total=100000 --val-size=256
===> RPS = 35k

benchmark --endpoints="http://${H1}:2379"
--target-leader --conns=100 --clients=1000
put --key-size=8 --sequential-keys
--total=100000 --val-size=2560
===> RPS = 19.5k

benchmark --endpoints="http://${H1}:2379"
--target-leader --conns=100 --clients=1000
put --key-size=8 --sequential-keys
--total=100000 --val-size=5000
===> RPS = 16k
```

Scalability

Benchmarking etcd and kube-apiserver

- Random pod to node binding
- Throughput - Number of pods and nodes
  - 40k pods, 8k nodes, **over minutes**
- Number of pod events **~10x** number of pods



Scalability

Drop pod events

- Consumes network and storage resource of api server and etcd
- Persistent state vs. logs
- Throughput improvement
  - 40k pods, 8k nodes, **30 seconds**
- Other evaluations around pod deletion, resource consumption etc.

Scalability

Conclusion

- etcd performance is heavily impacted by number of watches (number of nodes), but overall is good enough for Uber’s footprint
- etcd + kube-apiserver performance on pod creation/launch is good enough for Uber’s footprint after dropping pod events

Portability

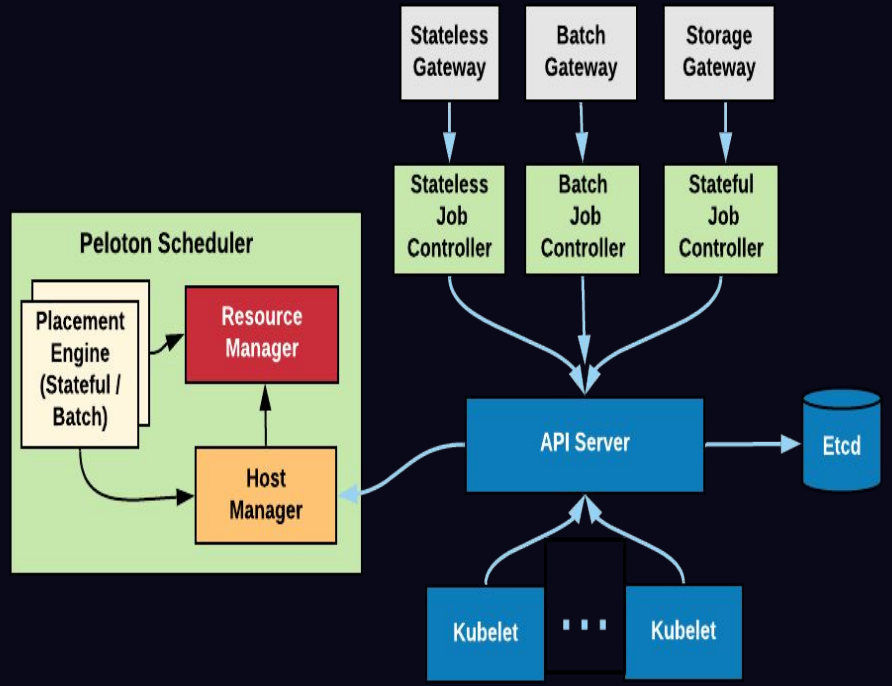
Peloton as a Kubernetes plugin

Pros

- Idiomatic for Kubernetes ecosystem
- Reuse Kubernetes functionalities
- Eventually we want to get here

Cons

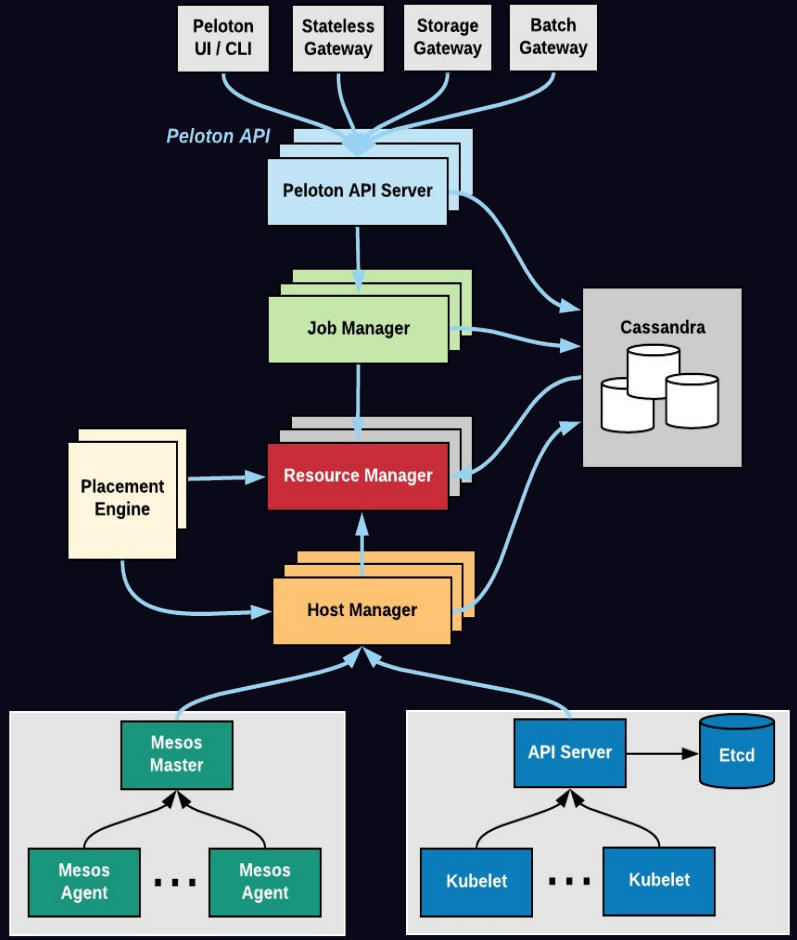
- Large scope, especially as first step
- Migration is costly
- Many translation layers needed



Portability

Peloton talks to Kubernetes & Mesos

- Kubernetes and Mesos cluster use same Peloton control plane
- Peloton Host Manager hides away both Kubernetes and Mesos
- Same Peloton Job Spec can be launched on either Mesos or Kubernetes host



Portability

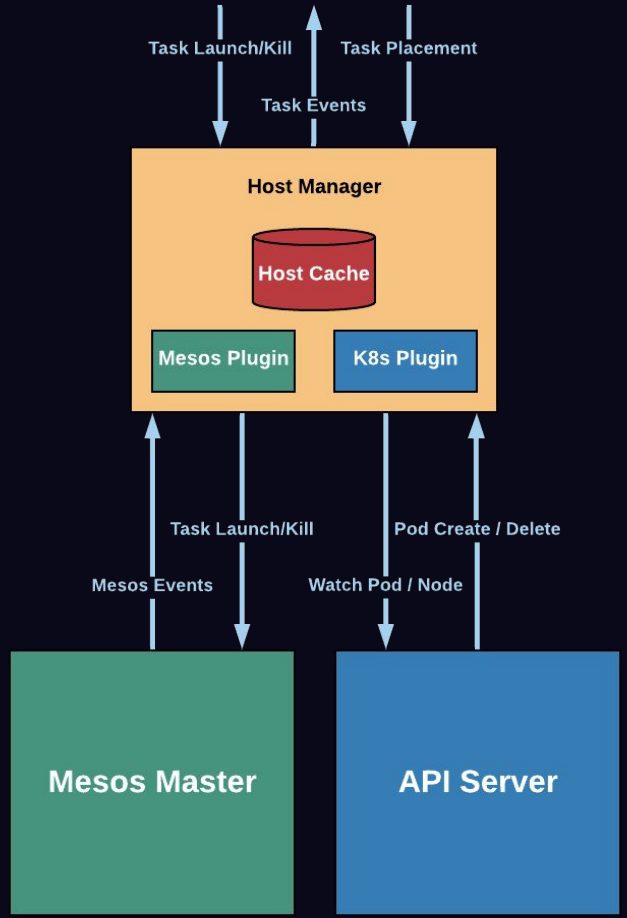
Peloton talks to Kubernetes & Mesos

Pros

- Shared control plane (Mesos + Kubernetes)
- Smaller scope of changes
- Invisible to clients, transparent migrations

Cons

- Not idiomatic
- Fundamental differences between Mesos & Kubernetes
- Consistency challenges



Portability

Conclusion

Peloton talks to Kubernetes & Mesos

- Customized stateless and batch service orchestration system
- Unique service deployment characteristics
- Different networking setup which is customized for Uber use case

Broader Integration with Uber Ecosystem

- Logging infrastructure
  - Kafka and ElasticSearch
- Security infrastructure
  - SPIRE
- Service discovery and routing
  - Customized software networking system

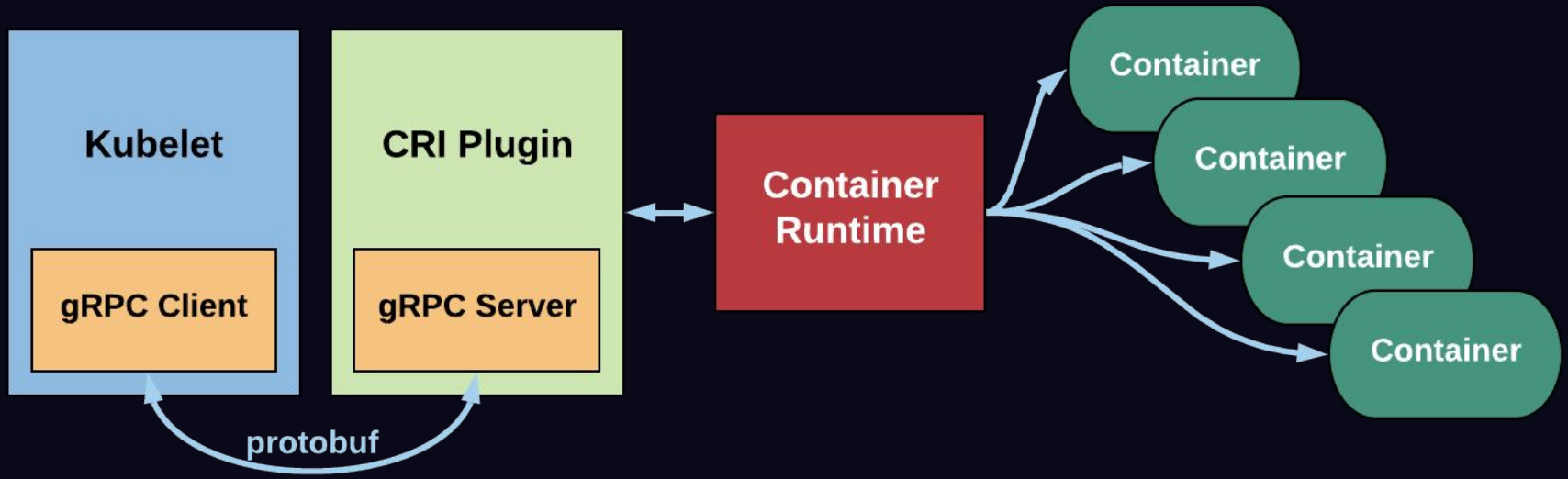
Case Study 1 - Logging

Requirements

- Support container logging to stdout/stderr and on-disk files
- Local log persistence after killing the pod
- Custom log file location

Case Study 1 - Logging

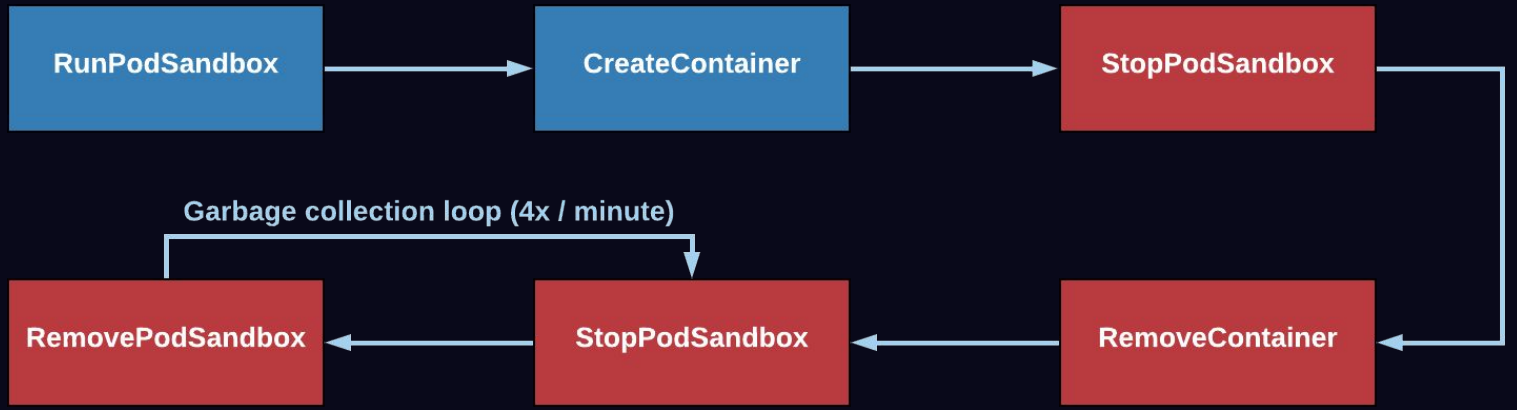
Solution - CRI customization



Kubernetes CRI Plugin

Case Study 1 - Logging

Solution - CRI customization



Flowchart of key RPCs in CRI

Case Study 1 - Logging

Solution - Custom Docker logging driver

- Responsible for logging to stdout/stderr
- Custom log file location
- Handle lossless log rotation and log compression

Case Study 2 - Networking

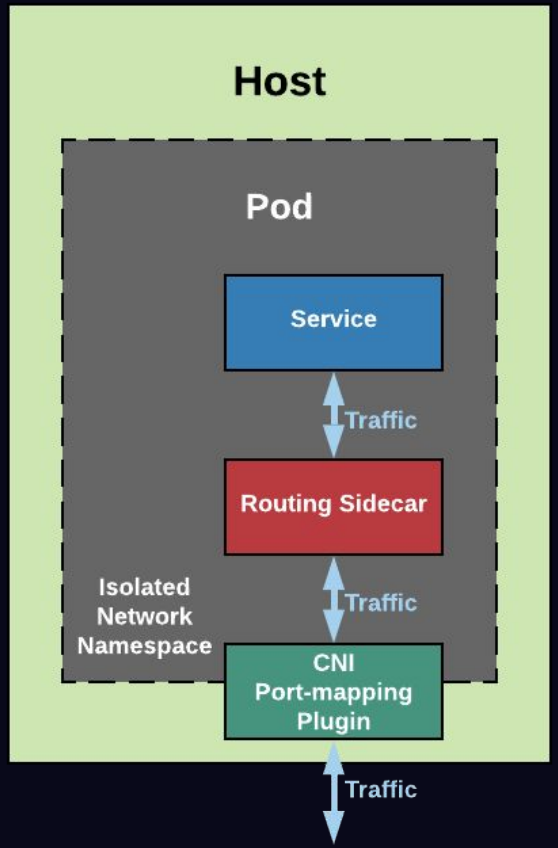
Current State of the World

- Customized discovery and routing system relying on host IP and dynamic port allocation
- No IPtables management
- Service dependency on static ports on host network interface

Case Study 2 - Networking

Solution - CNI plugin + CRI customization

- Manage IPtables via CNI port-mapping plugin
- Port mapping via CRI customization by
  - Modify iptables rules for the pod network namespace
  - Clean up iptables rules before stopping pod sandbox



Key Takeaways

- Active community engagement is important
  - Drop pod events
- Hands on evaluation helped in decision making
  - Scalability concern
- Portability is the key for massive migration
  - Peloton / kube-apiserver integration, Uber ecosystem integration etc.

Q & A