



Peloton: A Unified Scheduler for Web-scale Workloads on Apache Mesos & Kubernetes

Min Cai, Uber

Nitin Bahadur, Uber

Igniting opportunity by setting the world in motion

Uber



10+ billion trips

15M+ trips per day

6 continents, 65 countries and 600+ cities

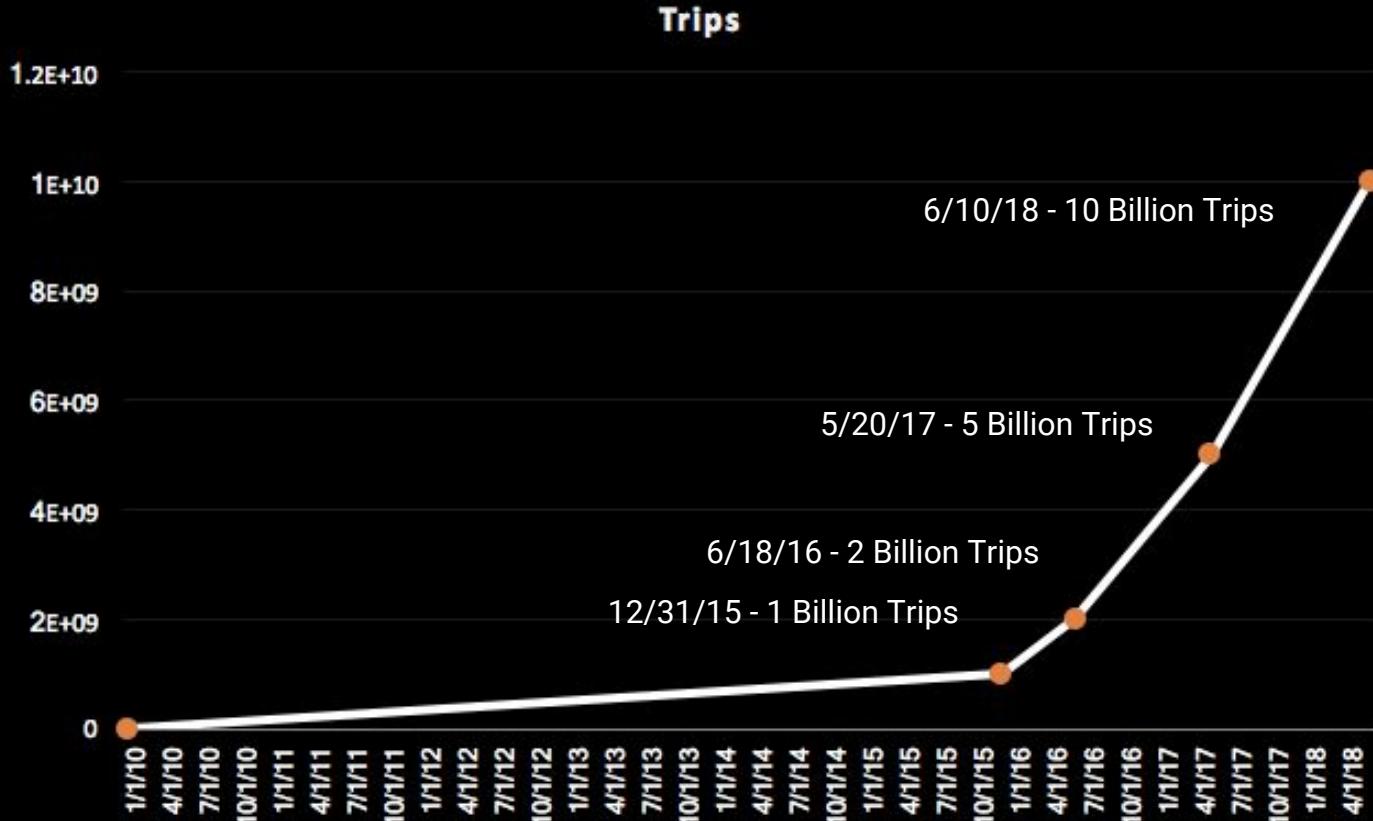
75M active monthly users

3M+ active drivers

16,000+ employees worldwide

3000+ developers worldwide

Business



Compute Infrastructure Scale

1000s of Microservices

1000s of Builds per day

10,000+ instances deployed per day

100K+ Service containers per cluster

~1M Batch containers per day

~1000s GPUs per cluster

25+ clusters

Uber stateless services run on
Mesos Today

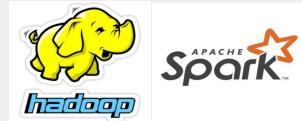
New Compute Cluster Use Cases

- Large scale batch jobs for autonomous vehicle use-cases
 - 100K tasks per job and millions tasks per day
- Elastic resource sharing among organizations and teams
- Co-locating mixed workloads on shared clusters
- Distributed deep learning on GPUs

Uber Cluster Workloads



Stateless Jobs



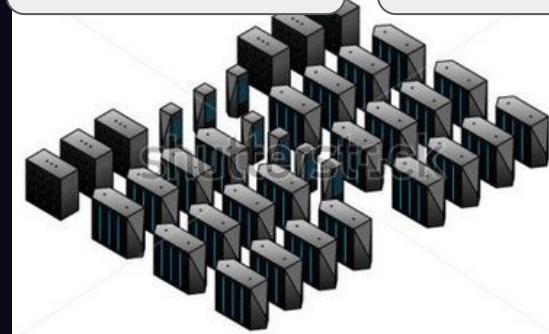
Batch Jobs



Stateful Jobs



Daemon Jobs



* Apache Hadoop, Cassandra, Spark, and Kafka logos are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks. TensorFlow and the TensorFlow logo are trademarks of Google Inc. Redis is a trademark of Redis Labs Ltd. Any rights therein are reserved to Redis Labs Ltd. Any use by Uber Technologies is for referential purposes only and does not indicate any sponsorship, endorsement or affiliation between Redis and Uber Technologies.

Co-locate Cluster Workloads

Why

- Improve cluster utilization
- Reduce the need to buy extra capacity for big spikes like NYE
- Use DR capacity for batch jobs in All-Active setup
- Batch jobs are ideal for resource overcommit

Issues

- Expensive to preempt online jobs that are latency sensitive

Existing Cluster Management Solutions



Comparison of Cluster Manager Architectures

	Borg	YARN	Mesos	Kubernetes
Application Workflow (e.g Map-Reduce)	Controller Job	Application Master		Custom Resources
Job/Task Lifecycle				kube-controller
Task Placement	Borgmaster		Frameworks (Cassandra, Spark, etc)	
Task Preemption		Resource Manager		kube-scheduler
Resource Allocation			Mesos Master	
Task Execution	Borglet	Node Manager	Mesos Agent	kubelet

Why Not Use Other Existing Schedulers

- **Borg** is not an open source solution
- **YARN** is a batch scheduler for Hadoop with no or very limited support for stateless, stateful, and daemon jobs.
- **Kubernetes**
 - It hasn't been able to scale to the large clusters that Uber requires, i.e. 10,000 plus. Federation is still in infancy.
 - Elastic resource sharing is not supported.
 - Not ideal for batch workloads, due to the high-churn nature of batch jobs.

Introducing Peloton

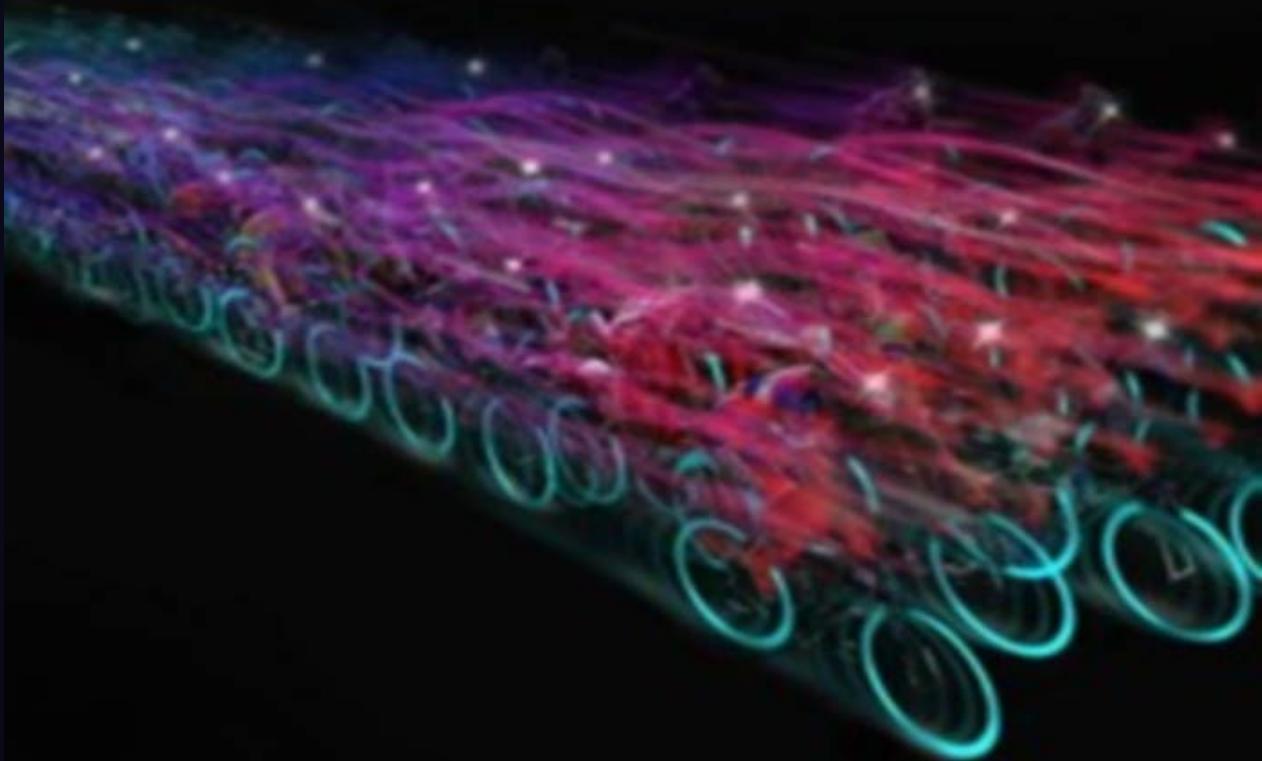


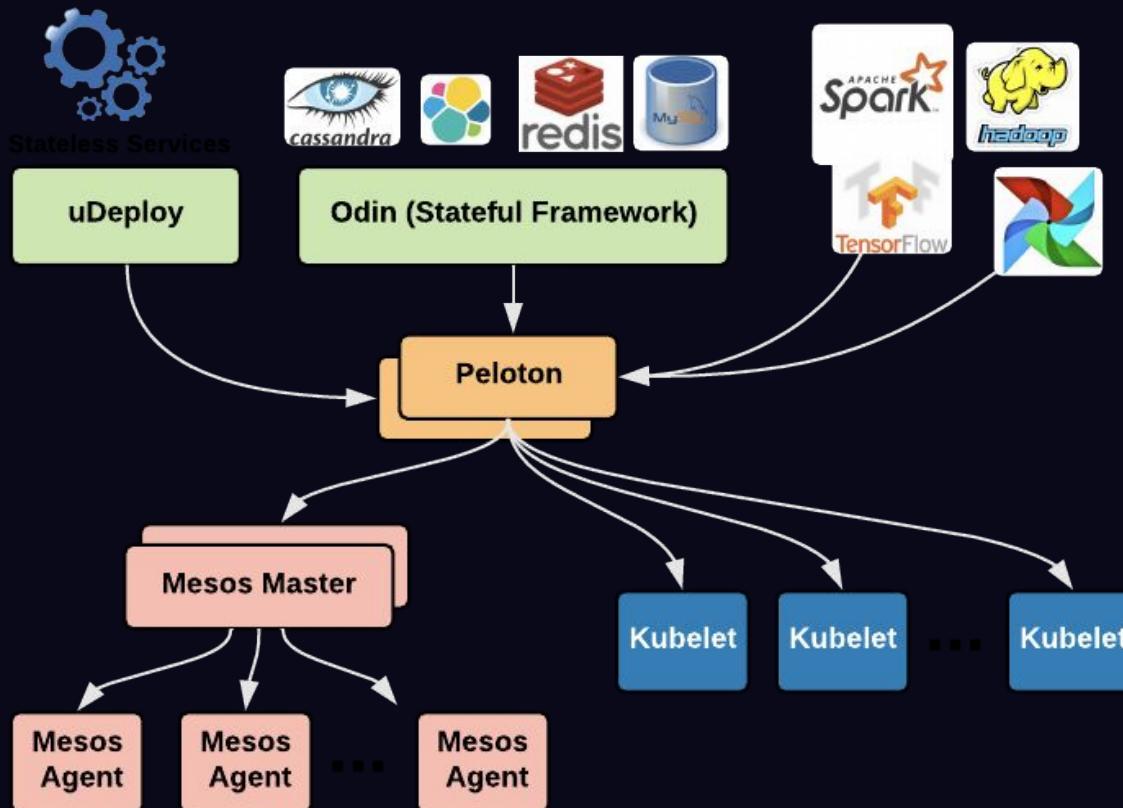
Image Source: <https://framemissing.bandcamp.com/album/ghost-peloton-soundtrack>

What is Peloton?

- *Unified Resource Scheduler* for co-locating mixed workload on compute clusters @ Uber
- Integrates with Spark, TensorFlow, YARN, uDeploy, etc.
- Can be run on-premise or in the Cloud



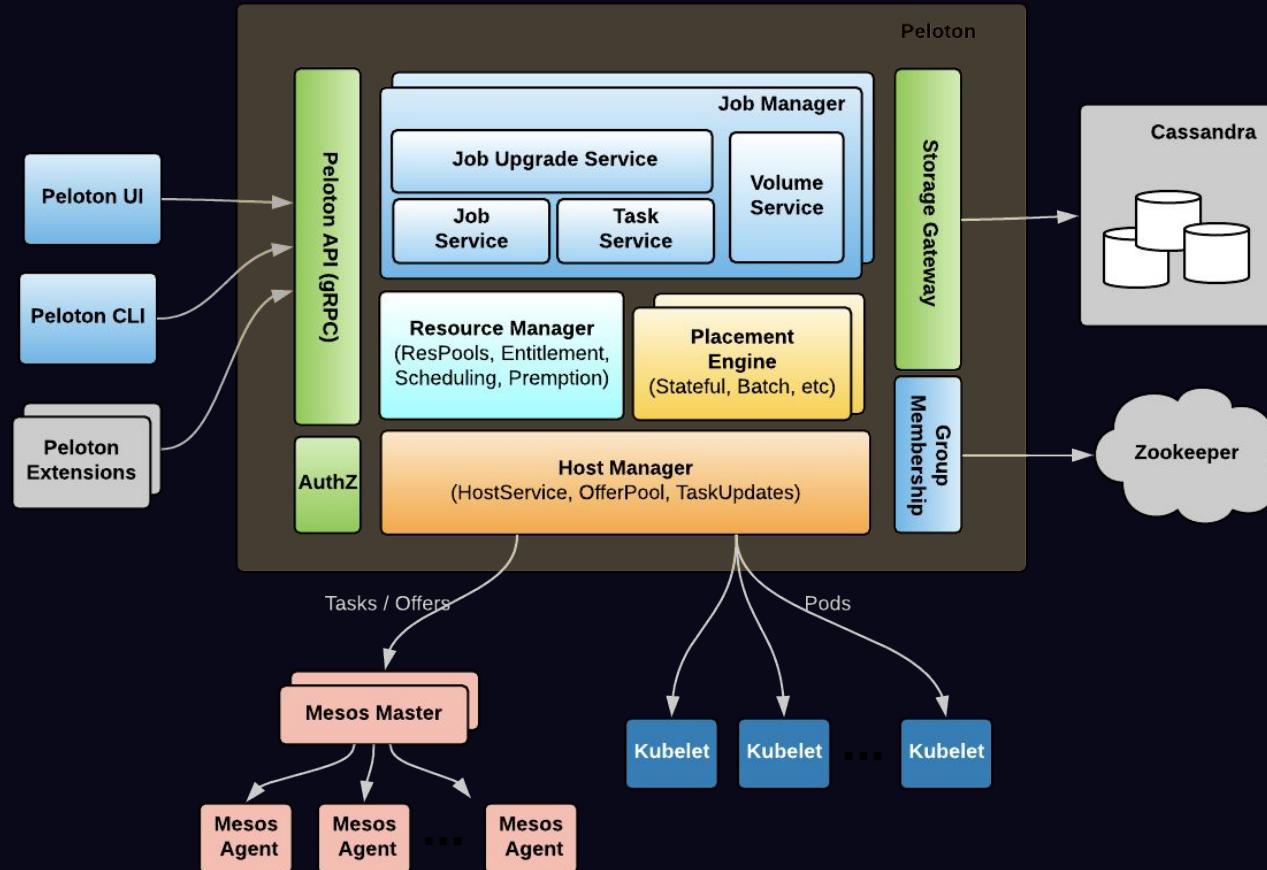
Peloton Overview



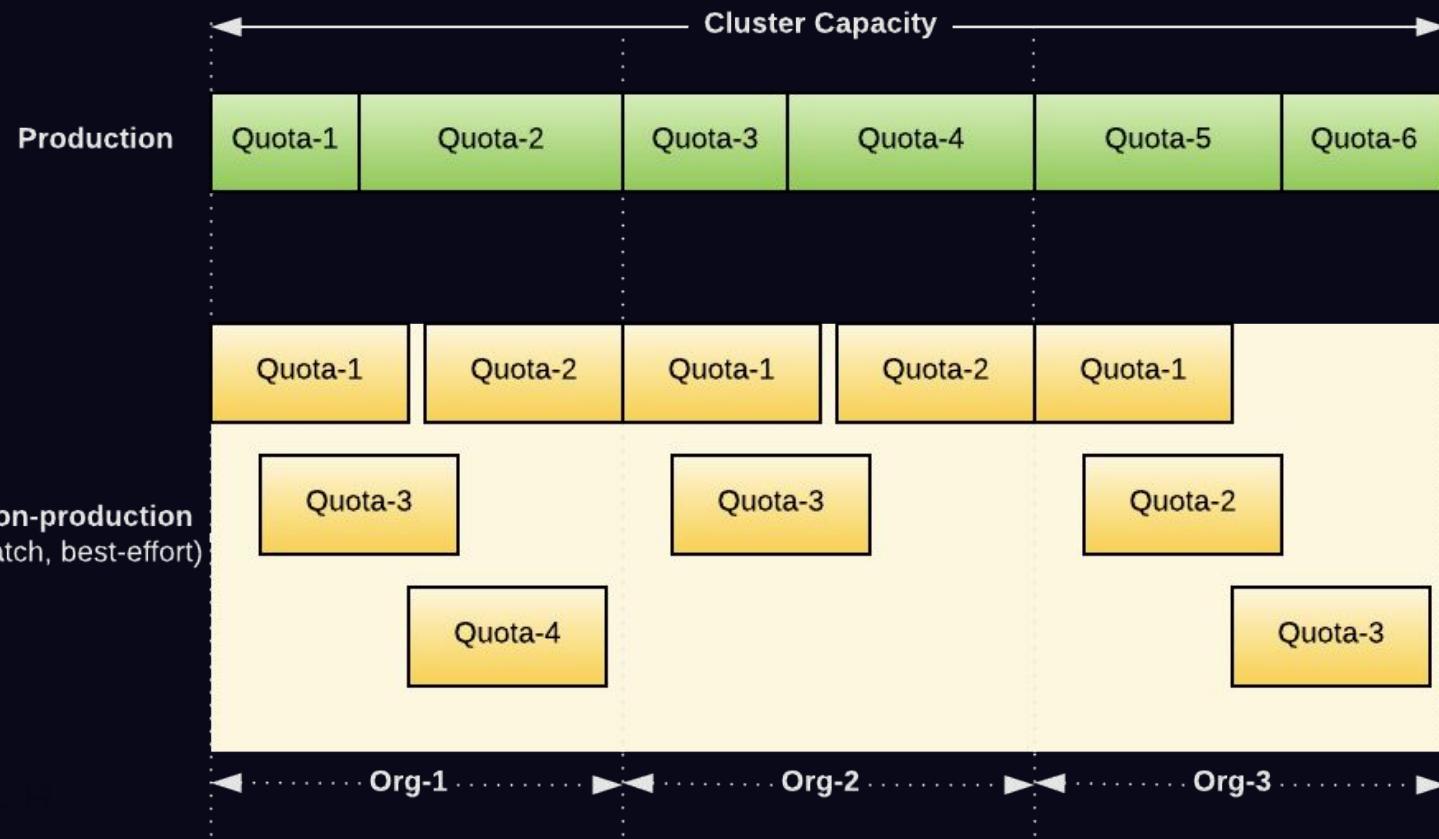
Comparison of Cluster Scheduler Architectures

	Borg	YARN	Mesos	Kubernetes	Peloton
Application Workflow (e.g Map-Reduce)	Controller Job	Application Master		Custom Resources	Controller Job
Job/Task Lifecycle	Borgmaster	Resource Manager	Frameworks (Cassandra, Spark, etc)	kube-controller	Peloton
Task Placement				kube-scheduler	
Task Preemption					
Resource Allocation			Mesos Master		Mesos Master
Task Execution	Borglet	Node Manager	Mesos Agent	kubelet	Mesos Agent

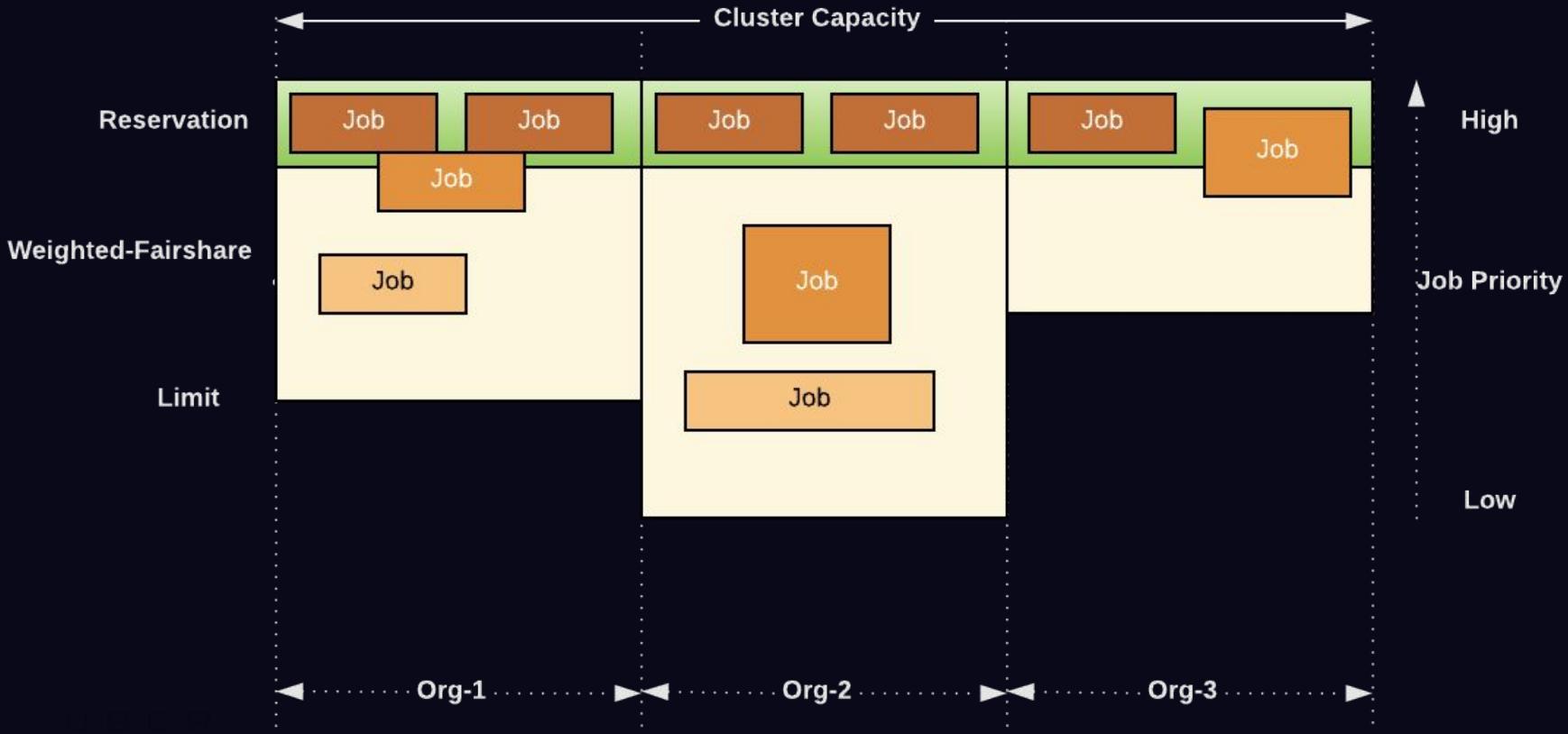
Peloton Architecture



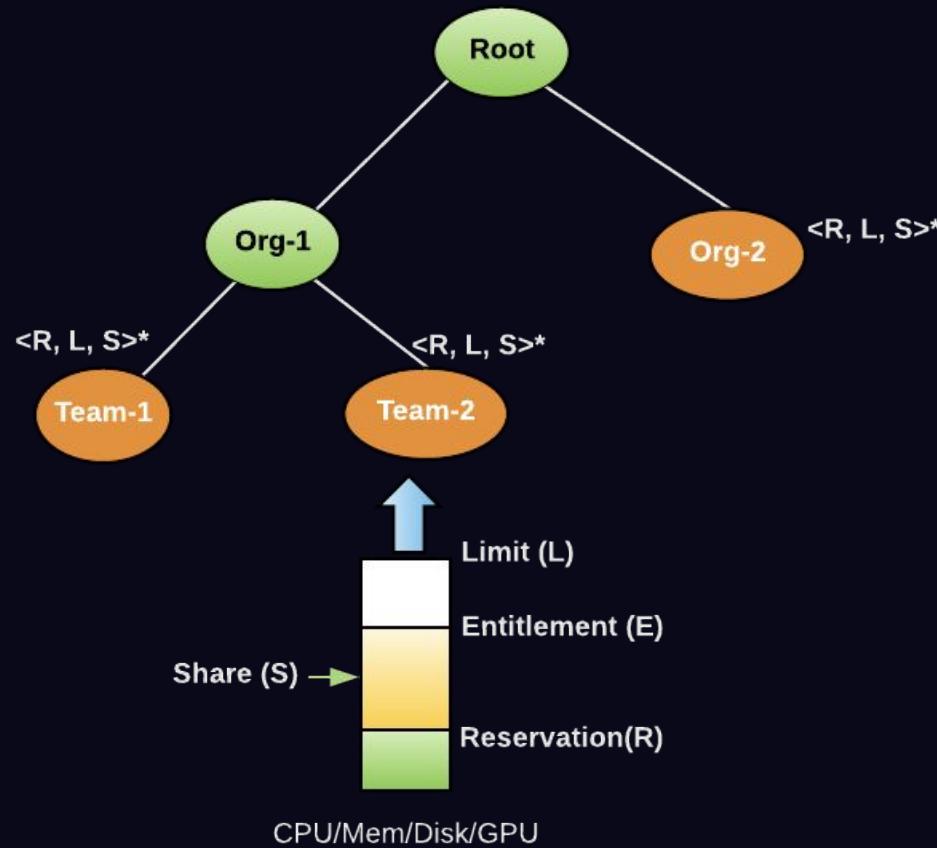
Priority-based Quota (Borg Model)



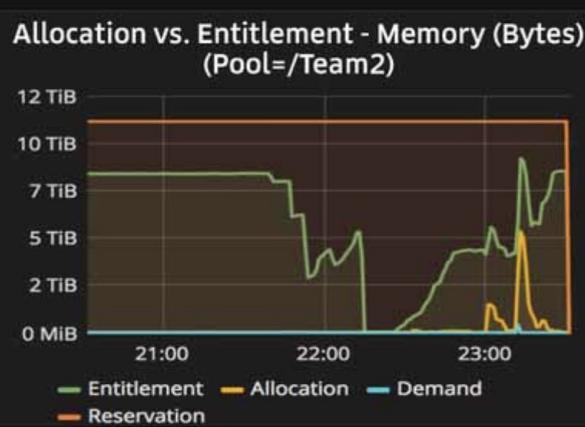
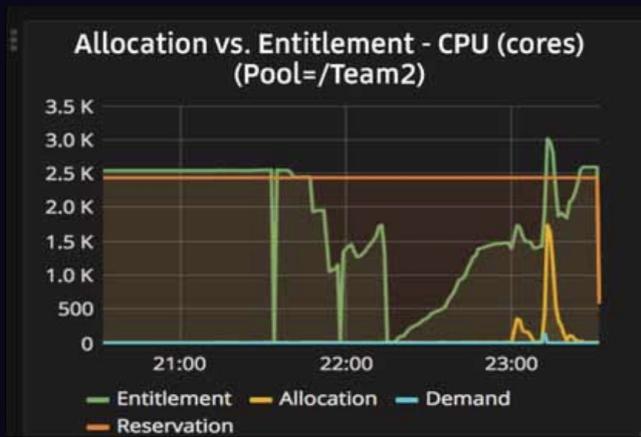
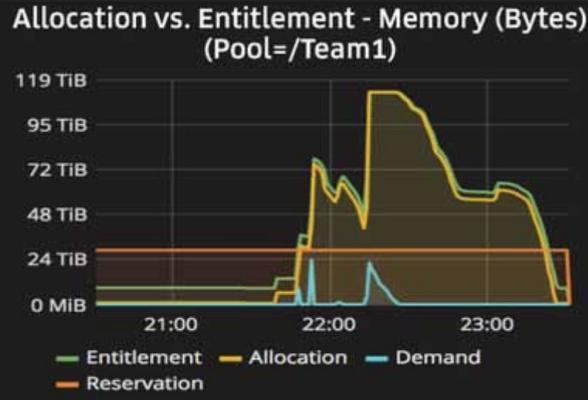
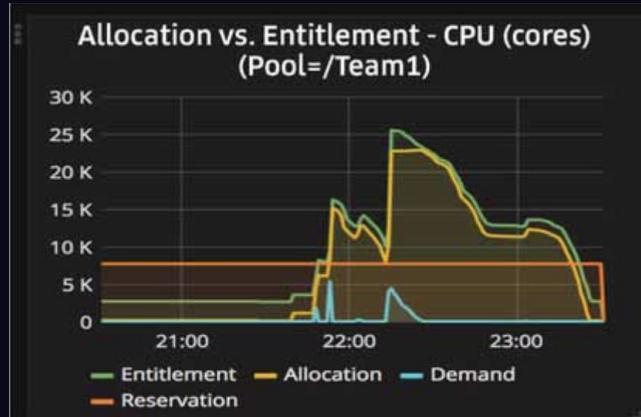
Hierarchical Max-Min Fairness



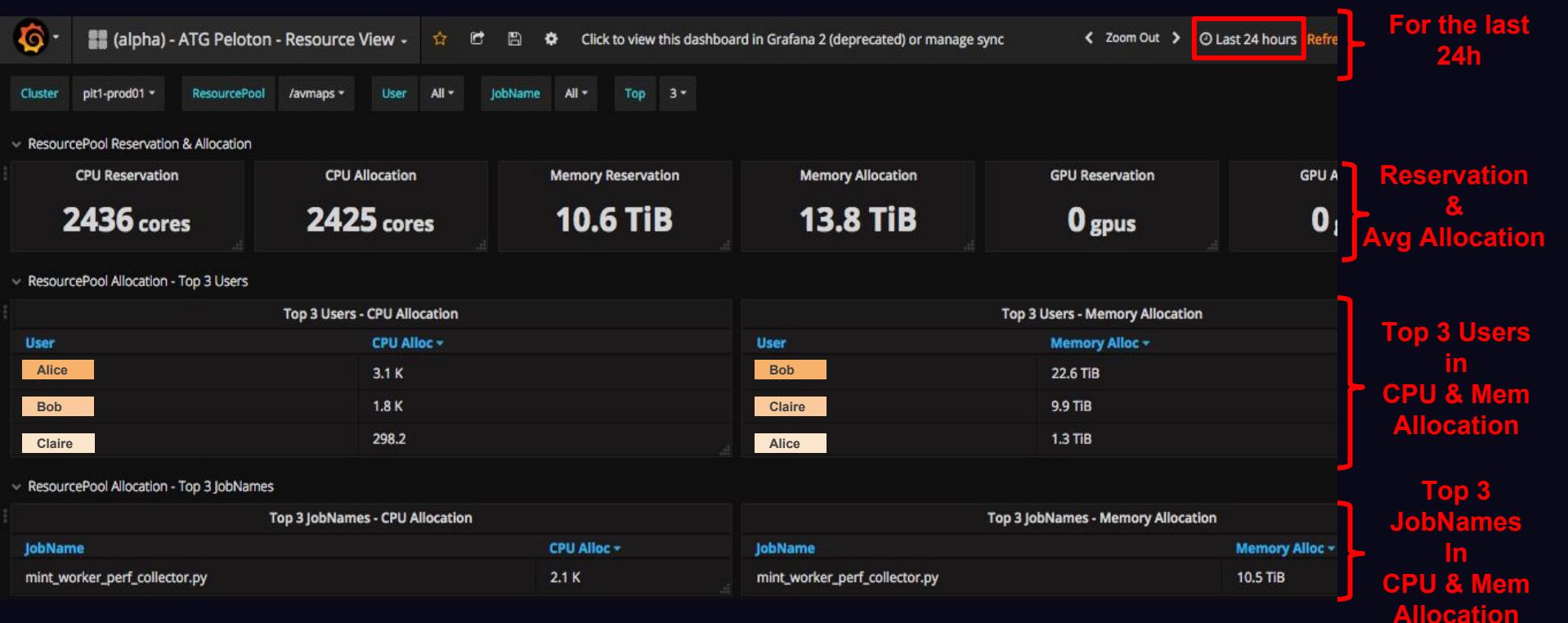
Hierarchical Resource Pools



Resource Pool Elasticity



Resource Mgmt Features

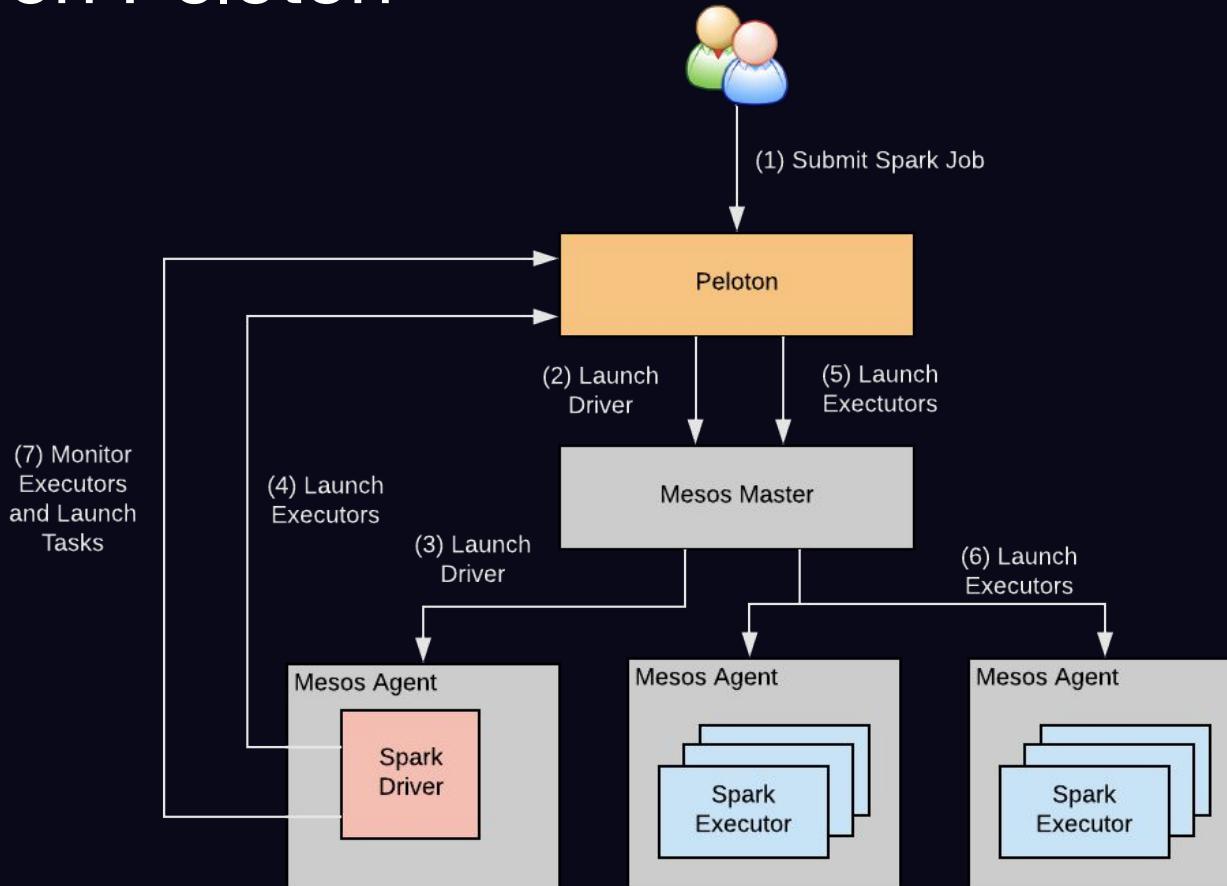


Peloton Workloads @Uber

Batch Jobs on Peloton

- Running Batch Jobs in multiple datacenters
 - Spark
 - Distributed TensorFlow
 - Large Scale Maps workloads
 - Large scale Autonomous batch workloads
 - Feature parity with YARN for Uber workloads
- Scale
 - **8K+** Hosts, **~2.5K+** GPUs
 - **3M+** jobs monthly, **36M+** containers monthly

Spark on Peloton



GPUs & Deep Learning

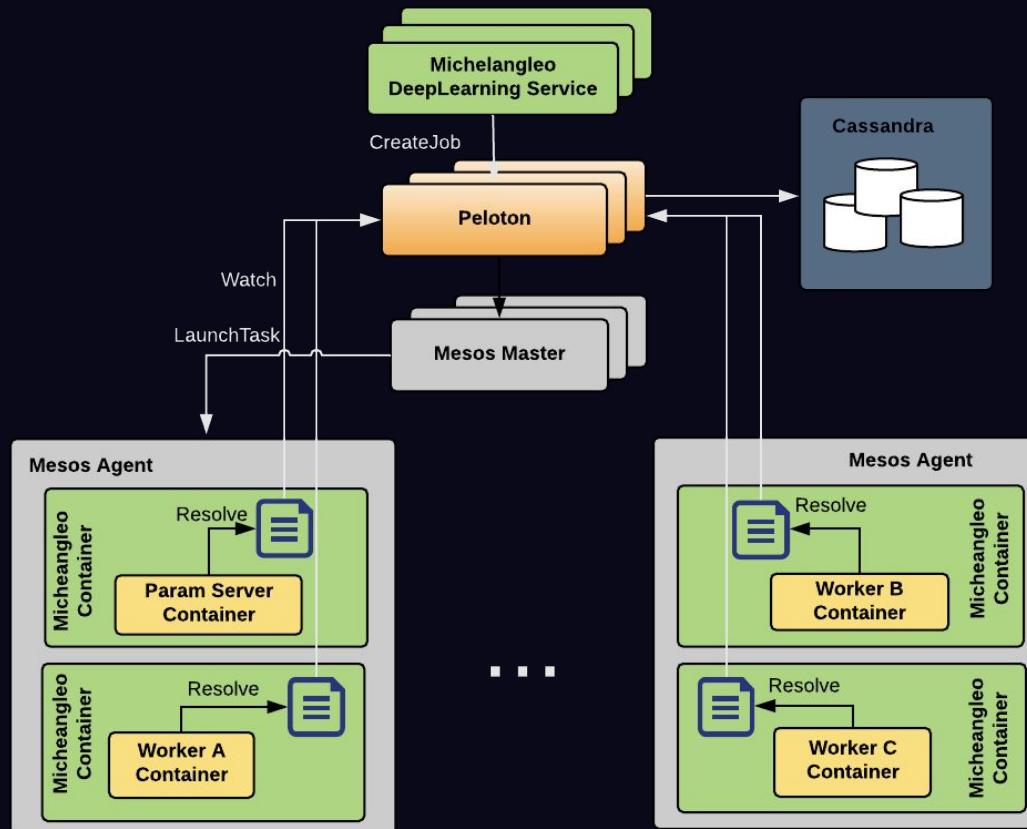
- Self-Driving Vehicles
- Trip Forecasting
- Fraud Detection
- More ...



Distributed TensorFlow Challenges

- Elastic GPU Resource Management
- Locality and Network-aware Placement
- Gang Scheduling
- Task Discovery
- Failure Handling

Distributed Tensorflow on Peloton



Stateless Services

1000s of microservices, growing day by day

- Over-allocated & under-utilized
- Resource over-commit & pre-emption in-use

Team is actively working on migrating services off Apache Aurora to Peloton.



Stateful services



Large scale Storage & Data applications

- Currently running on dedicated bare metal clusters



Peloton target => 2020 H1



Kubernetes & Peloton

Best of both worlds?

Kubernetes

- Widely adopted container orchestration system

Peloton

- Intelligent scheduler built for web-scale workloads

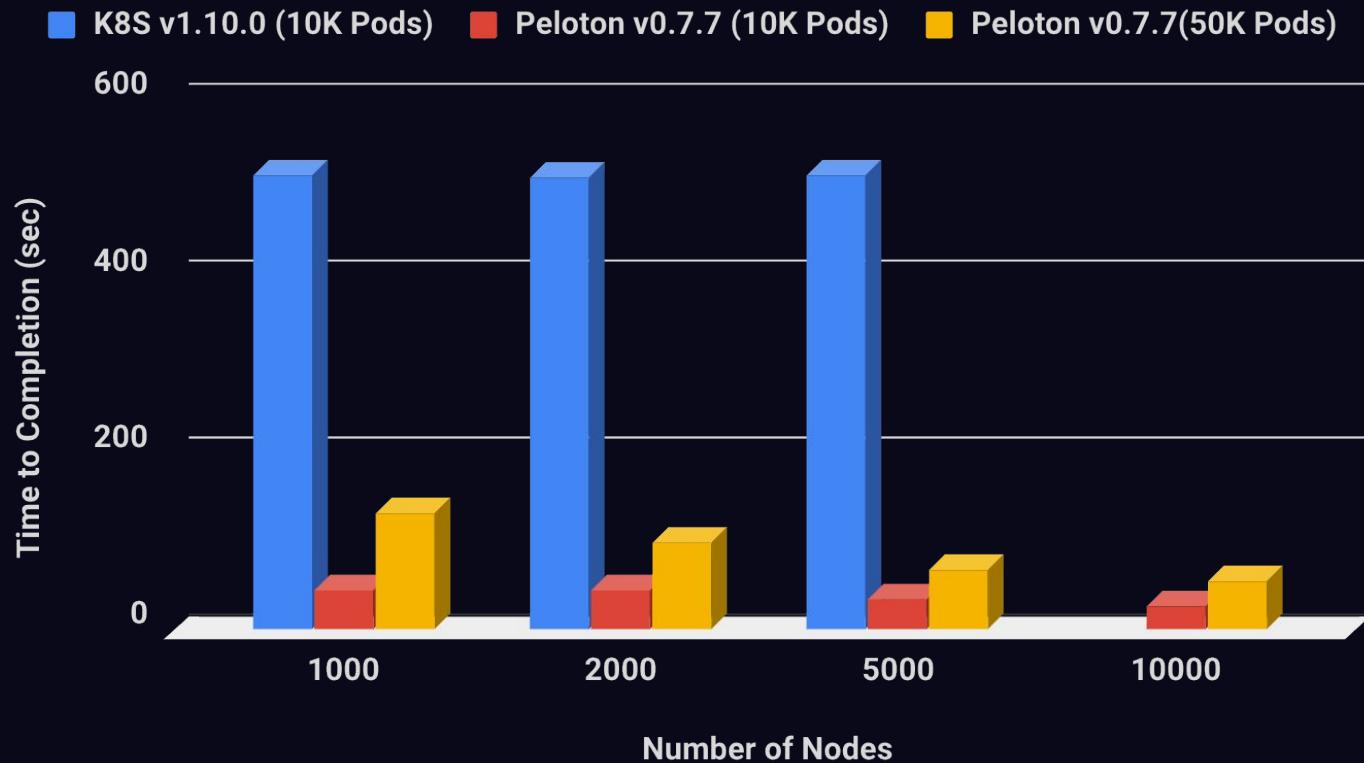
Why consider Kubernetes?

- Lots of features and extensions for mixed workloads
 - Pod, Deployment, StatefulSet, Job, DaemonSet, etc
- Growing community and ecosystem support
- More adoption and native integration from many open source projects
 - E.g. Spark, Flink, Kafka, Tensorflow etc
- Cloud native support in AWS, GCP, and Azure as managed clusters
- Fill the gap for features unavailable in today's Uber Compute offerings
 - StatefulSet
 - Auto-scaling
- Feasible extension model that allows other Uber teams such as Software Networking, Storage, Data, and Security teams to build extensions.

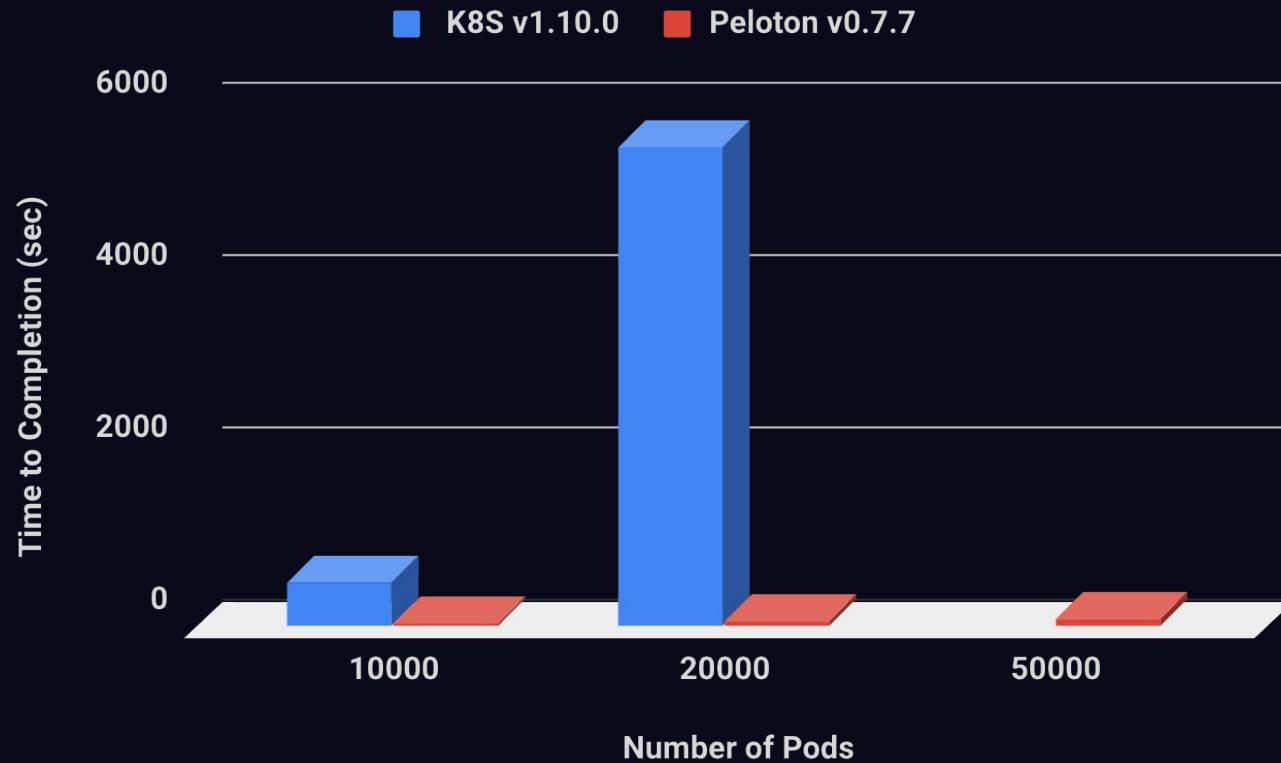
Benchmarking Peloton and Kubernetes

- Running Kubernetes and Peloton as *virtual clusters* on top of Peloton
- Scale the cluster sizes from 1K to 10K nodes
- Scale the batch and stateless jobs from 10K to 100K containers
- Measure the performance for the following scenarios:
 - Total time to completion for a batch job
 - Total time to rolling upgrade a stateless job

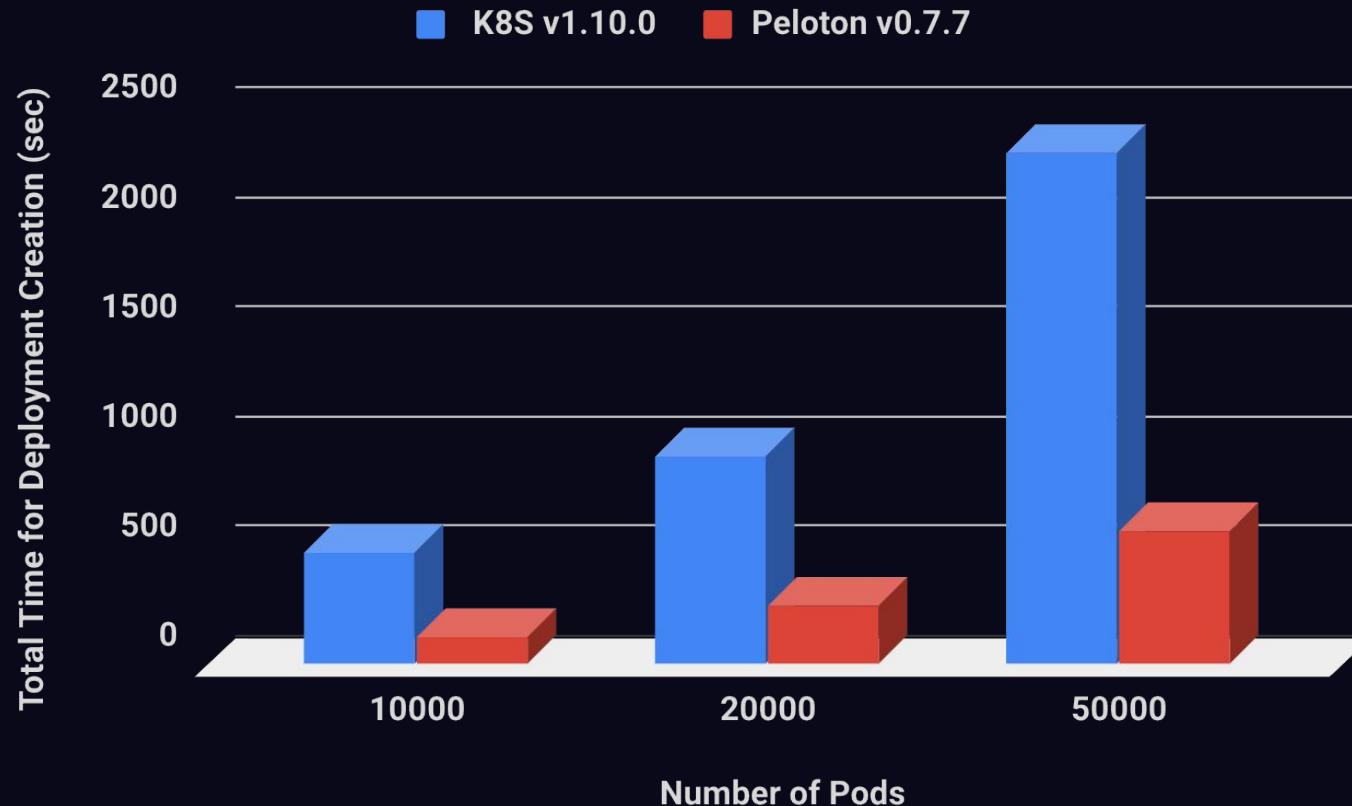
Time to Completion for Batch Jobs



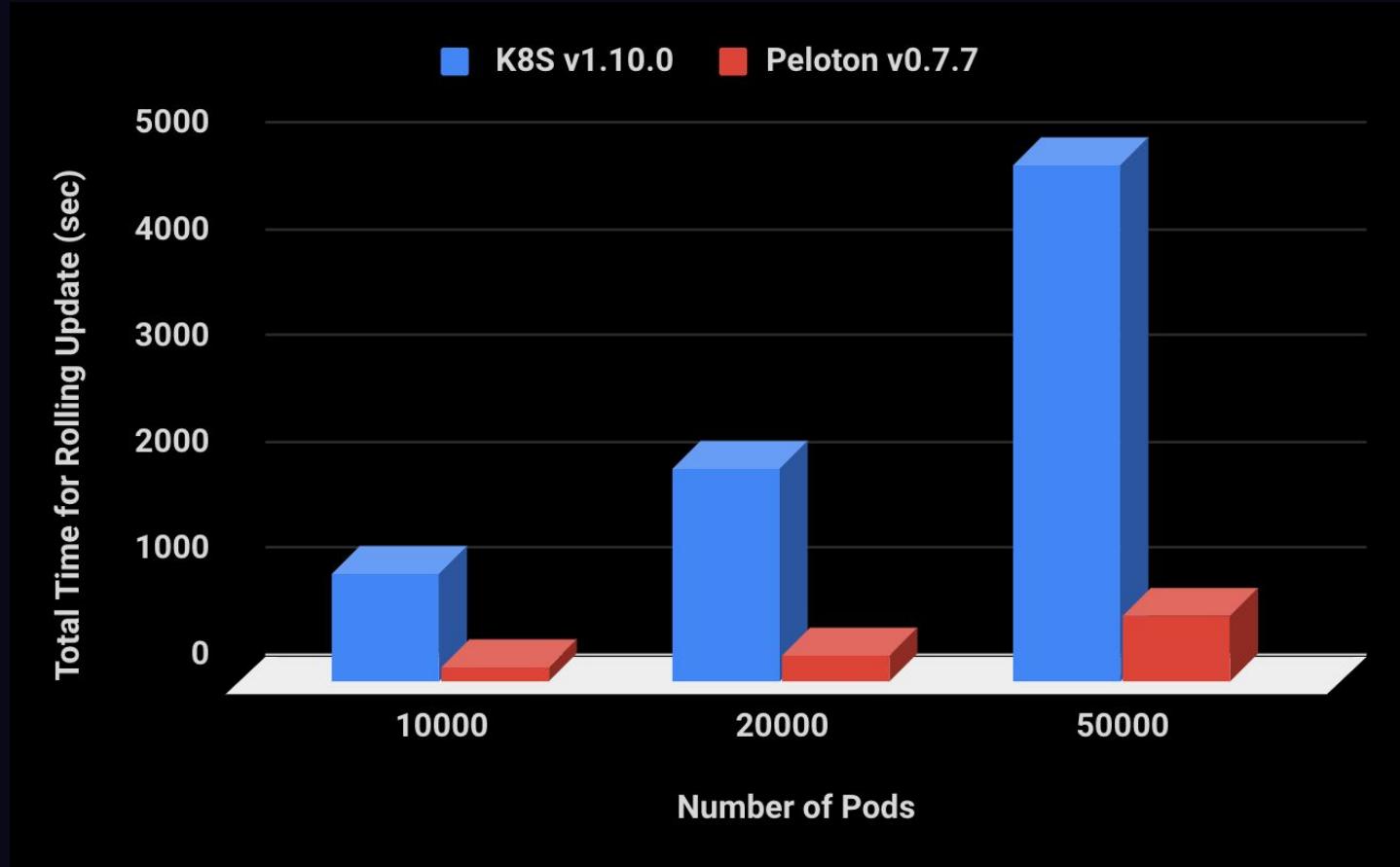
Time to Completion for Batch Jobs (2K Nodes)



Time for Deployment Creation (2K Nodes)



Time for Deployment Rolling Update (2K Nodes)



Peloton & Kubernetes Integration

Why K8s?

- Enables Uber to stay with current technology trends and leverage open-source

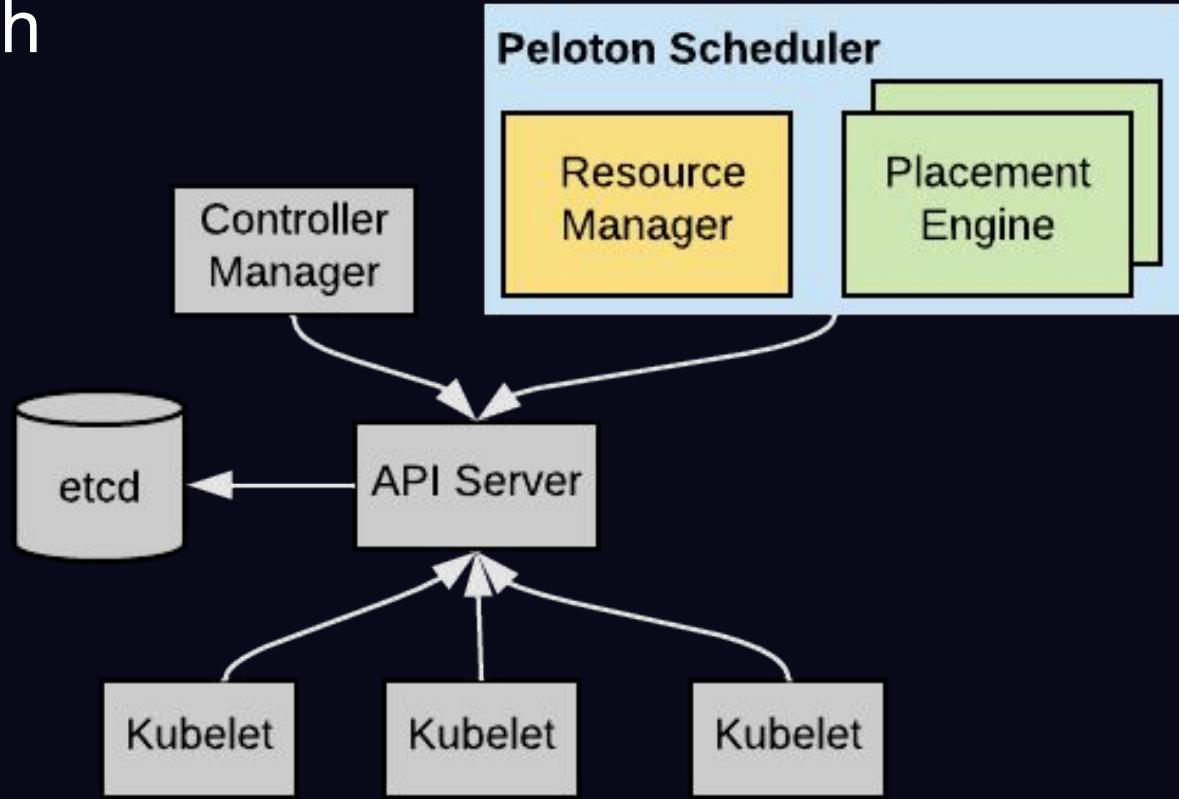
Why Peloton?

- Meets Uber specific scale & customization needs
- Provides a migration path from Mesos to Kubernetes without impacting Uber workloads

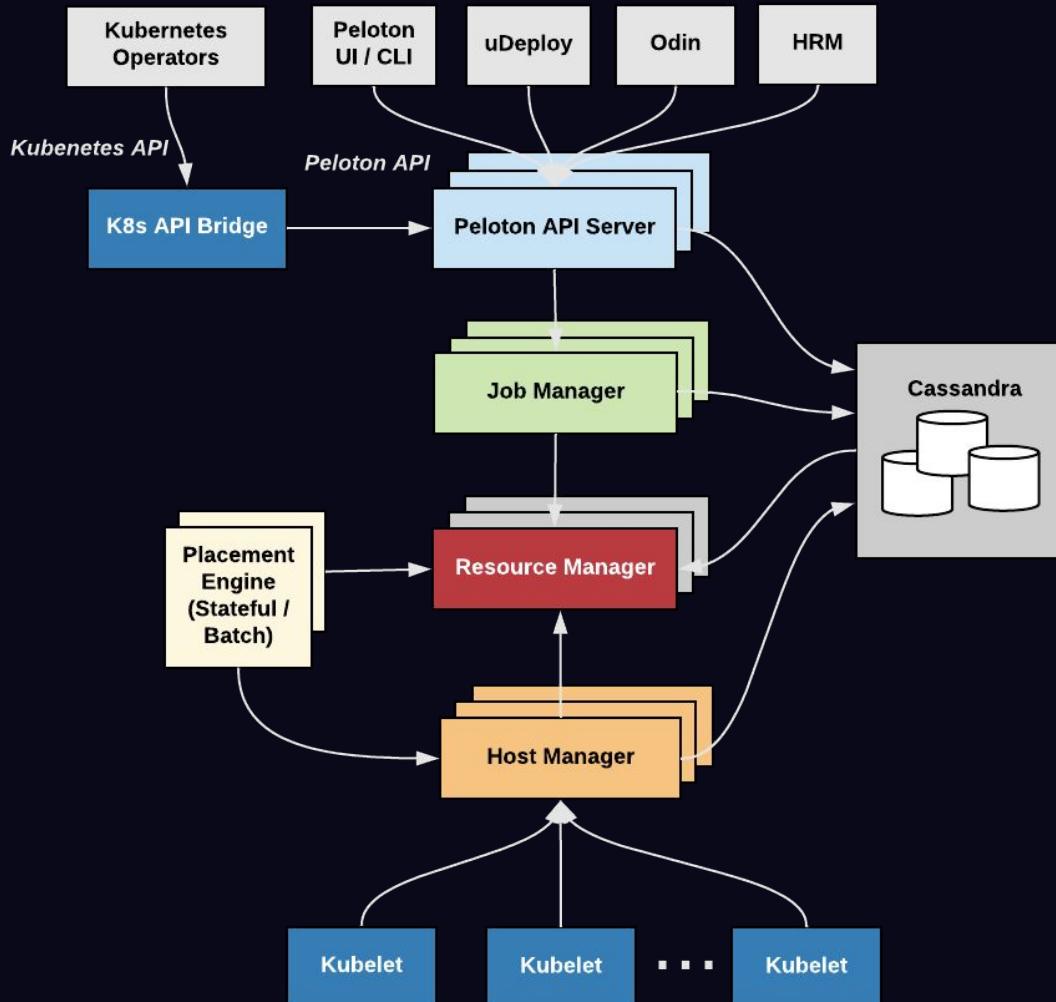
Icing on the cake

- Enables other large Mesos-based companies a way to transparently migrate from Mesos to K8s

Kubernetes with Custom Scheduler



Kubernetesize Peloton



Summary

- Peloton has been deployed in production at Uber for over a year
- It's designed from day-1 to run alongside any container orchestration system
- Engineering blog - eng.uber.com/peloton



We are hiring!

www.uber.com/careers/

Uber

eng.uber.com/peloton

Proprietary and confidential © 2018 Uber Technologies, Inc. All rights reserved. No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval systems, without permission in writing from Uber. This document is intended only for the use of the individual or entity to whom it is addressed and contains information that is privileged, confidential or otherwise exempt from disclosure under applicable law. All recipients of this document are notified that the information contained herein includes proprietary and confidential information of Uber, and recipient may not make use of, disseminate, or in any way disclose this document or any of the enclosed information to any person other than employees of addressee to the extent necessary for consultations with authorized personnel of Uber.