Predictive Auto-scaling in the Kubernetes Cluster Manager

THANK YOU

- Professor Albrecht, Professor, Williams College
- Brendan Burns, Lead Engineer for Kubernetes, Google

1.
Goals

Why do we care?

General Goal

Contribute to distributed system's ability to reliably and resourcefully do large, varying amounts of computation.

Specific Goal

To maximize the sum of two Kubernete's metrics: Efficient Resource Utilization and Quality of Service

Unpacking this Goal

Kubernetes

An open-source cluster manager from Google.

Efficient Resource Utilization (ERU)

Is the application efficiently using the resources it is given?

Quality of Service (QOS)

Is the application accomplishing its stated purpose?

The goal is balance.

Increasing ERU/QOS while decreasing the other is easy, we seek to increase the summation.

2. Accomplishing General Goals

How do cluster managers like Kubernetes reliably and resourcefully perform large, varying amounts of work?

What is a cluster manager?

- A cluster is a collection of commodity computers linked by a local-area network.
- A cluster manager admits/runs/monitors user submitted jobs on the cluster.

Benefits of Cluster Managers

Cluster managers allow us to perform computational work that could never be performed on a single computer.

What are some cluster managers?

Borg

Decades old cluster manager from Google. The closed-source precursor to Kubernetes.

Mesos

A low-level cluster manager. If Borg is Ubuntu, Mesos is the Linux Kernel.

Apache YARN

A cluster manager originally for Apache Hadoop.

More about Kubernetes

Kubernetes specific terms

Pods

A stateless, replicable wrapper around an containerized application.

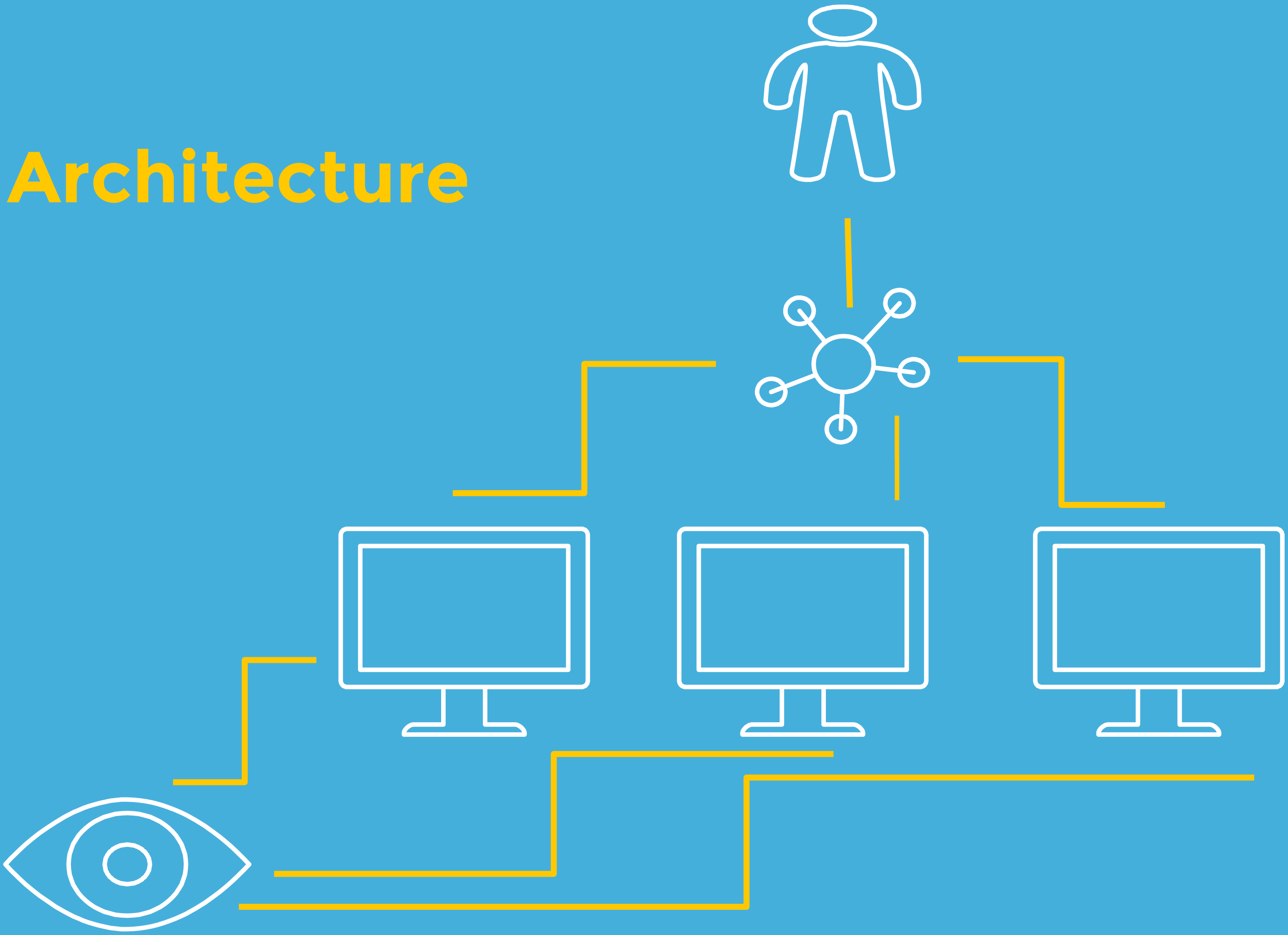
Replication Controllers

A controller for ensuring a given number of replica pods exist.

Services

A single point of load-balancing access for requests to replica pods.

Architecture

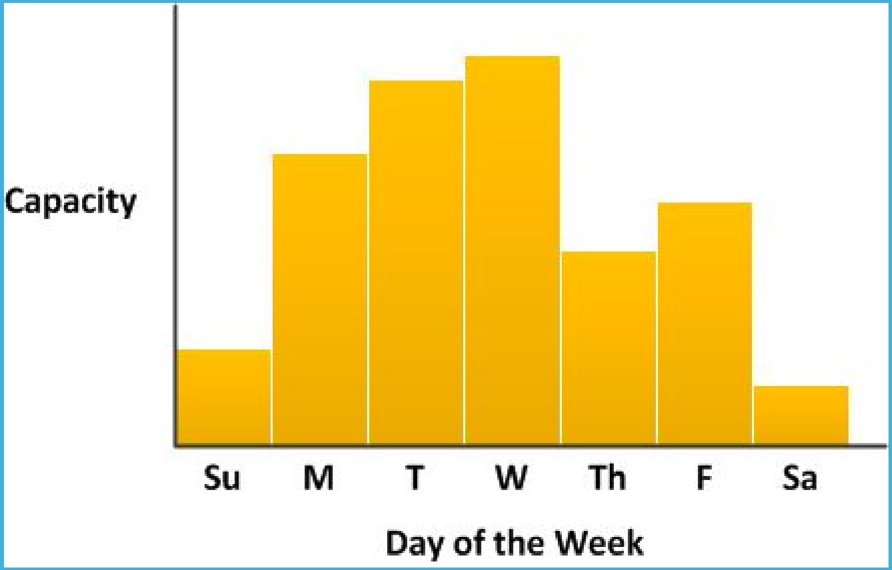


3.
Accomplishing Specific Goals

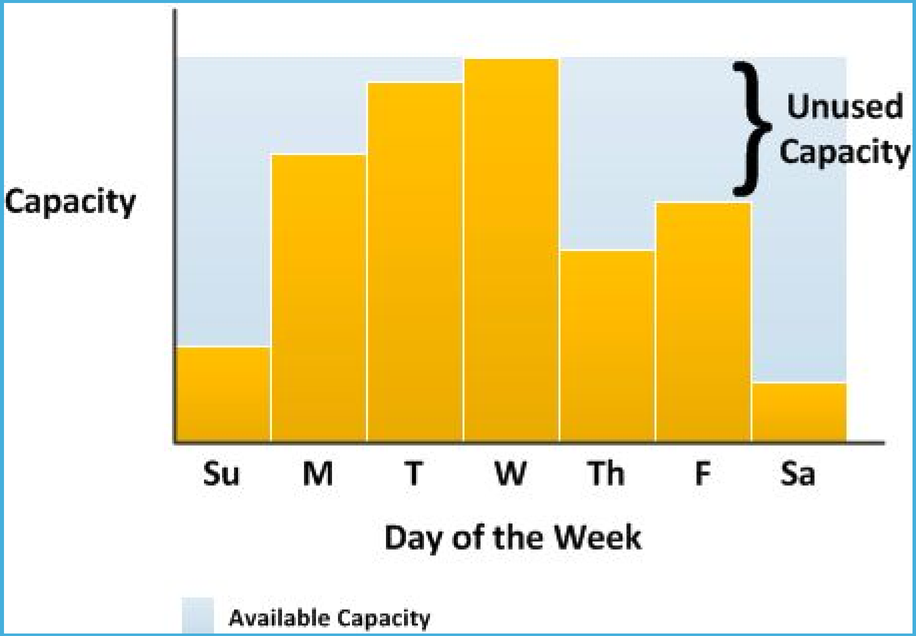
How does (predictive) auto-scaling in Kubernetes improve the summation of ERU and QOS?

Benefits of auto-scaling

Imagine the following capacity for an application running on a cluster manager…



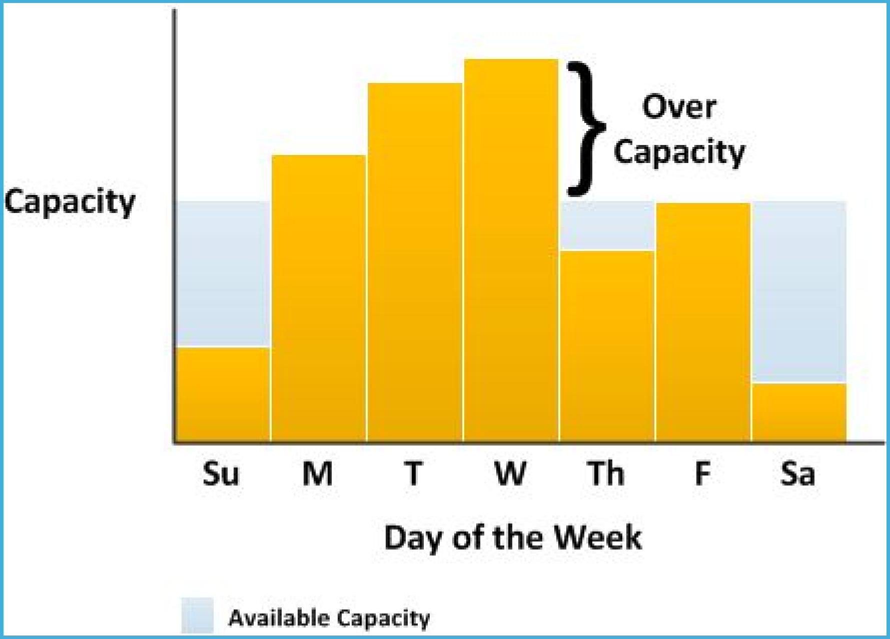
If we do not have auto-scaling



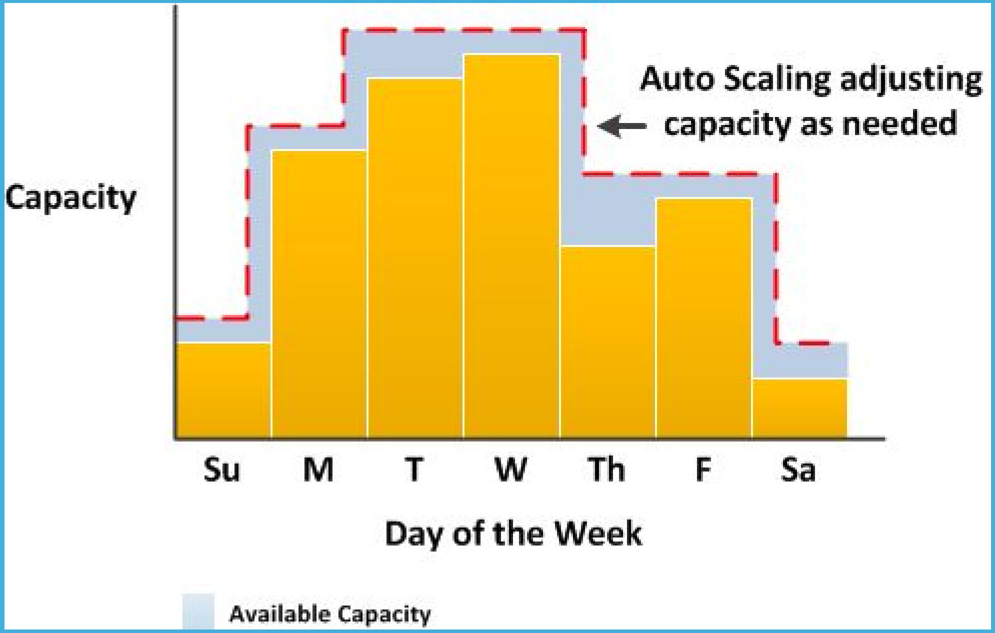
We can assign our application the most resources it will ever need… but poor ERU.

If we do not have auto-scaling

We can assign the average amount of capacity needed, improving ERU, but decreasing QOS.



If we do have auto-scaling



We can assign the application the exact resources it needs, when it needs them... improving the summation of ERU and QOS.

What are the different characteristics of auto-scaling?

Horizontal vs Vertical

How is an application given the extra resources that it needs?

Reactive vs Predictive

Does auto-scaling occur based on the current or future state of the cluster?

What are the major types of auto-scaling?

Threshold-based Rule Policies

Scale if the current resource usage is not in accordance with a set of predefined rules.

Time-series Analysis

Auto-scale based on repeating pattern in the application load.

Control-theory (Feedback Control)

Scale such that the resource usage is in accordance with predefined guidelines.

Current State of Auto-scaling in Kubernetes

Kubernetes currently implements reactive, horizontal feedback control based auto-scaling of pods.

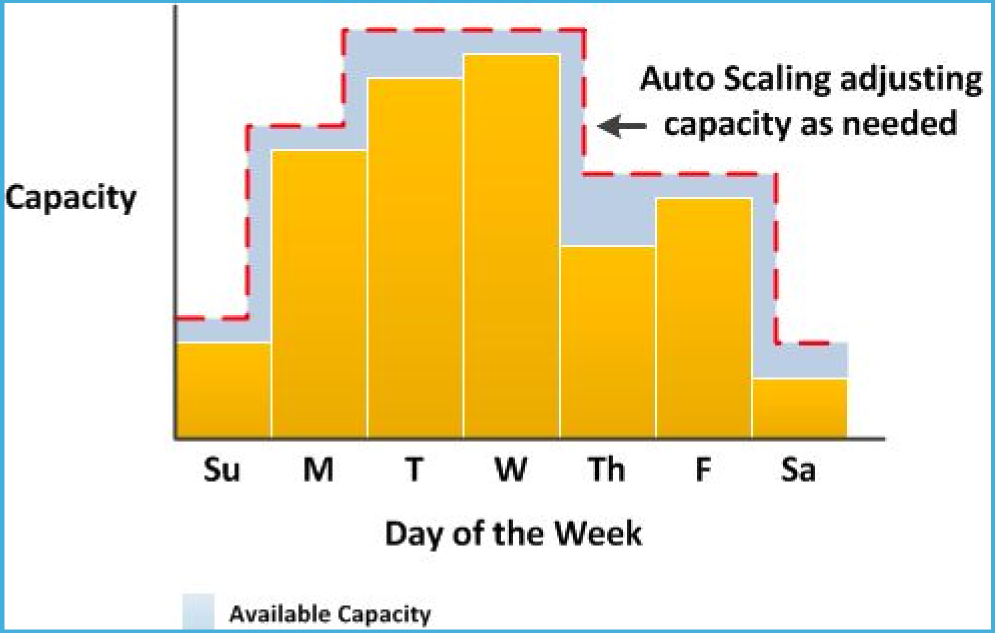
4.

Concerns with Auto-scaling in Kubernetes

Are there ways to improve the summation of ERU and QOS?

Delayed Pod Initialization Time

What if it takes a long time for a pod to be ready to share in the computational work?



5.
Improvements to Auto-scaling in Kubernetes

What if we add prediction?

Benefits of adding prediction

- Predictive, horizontal feedback control based auto-scaling of pods
- Improves QOS without decreasing ERU

A case study

- Imagine at 5:50pm, we need 100 pods, and it 6pm, we need 200 pods.
- Imagine pods take 10 minutes to create.

Reactive

1.
At 6:00pm, reactive auto-scaling says create 100 pods.

2.
From 6:00 to 6:10, wait for pods to initialize.

3.
At 6:10, all the needed pods will be working.

For 10 minutes, the application operates with only half the resources it needs, while we wait for the replica pods to initialize.

Predictive

1.
At 5:50pm, predictive auto-scaling says create 100 pods.

2.
From 5:50 to 6:00, wait for pods to initialize.

3.
At 6:00, all the needed pods will be working.

The application always has the resources that it needs.

Implementation questions?

How long does it take for a pod to be ready to share in the work?

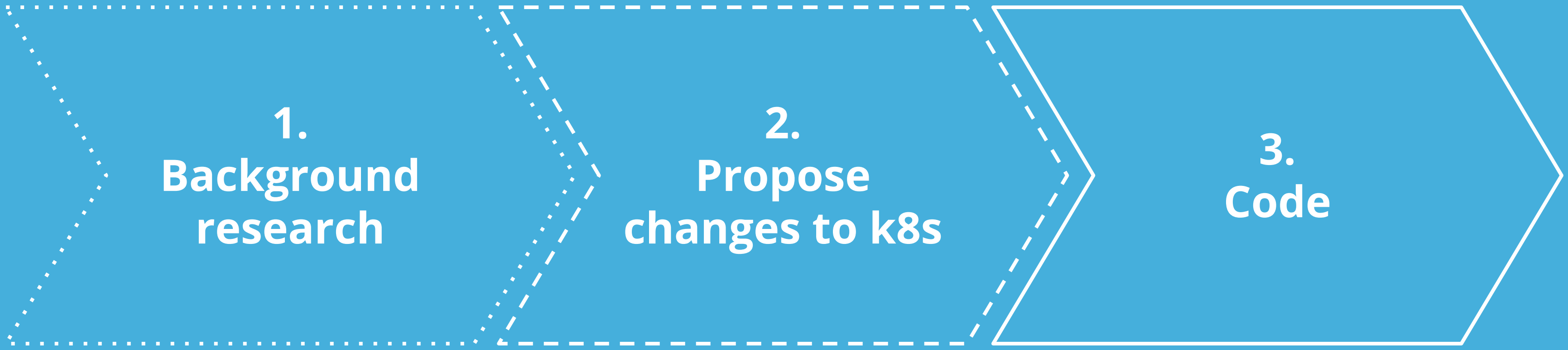
How can we predict the future resource utilization of an application?

Should this behavior be enabled by default?

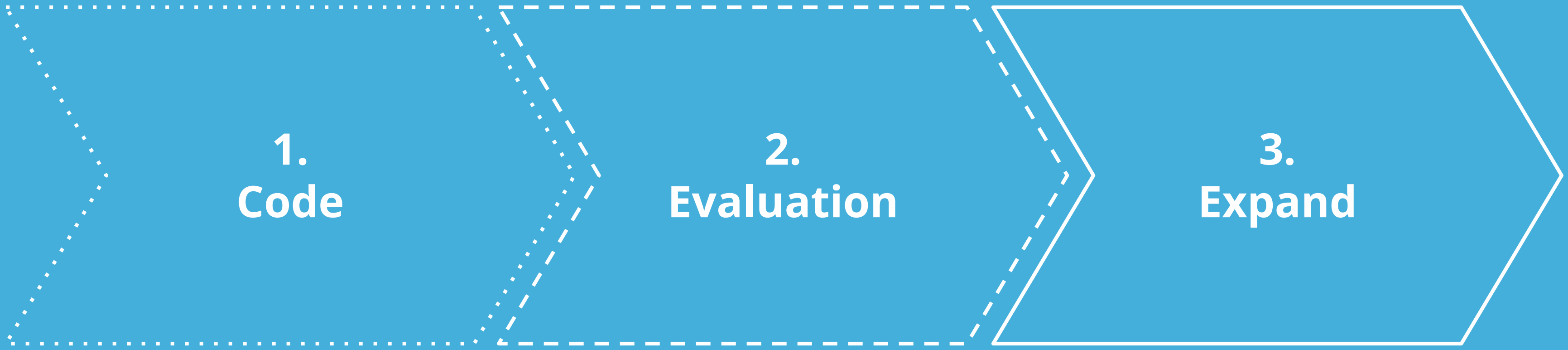
6.
Status of Work

What has been done and what is left to do?

Current State



Future Work



7. Evaluation

How will we know if we're successful?

Is predictive auto-scaling **beneficial?**

- Does predictive auto-scaling improve the summation of ERU and QOS?
  - How to combine ERU and QOS?
  - What applications will we try to auto-scale?
  - What will be their external environment?

THANKS!

Any questions?

CREDITS

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash