Cilantro

Performance-Aware Resource Allocation for General Objectives via Online Feedback

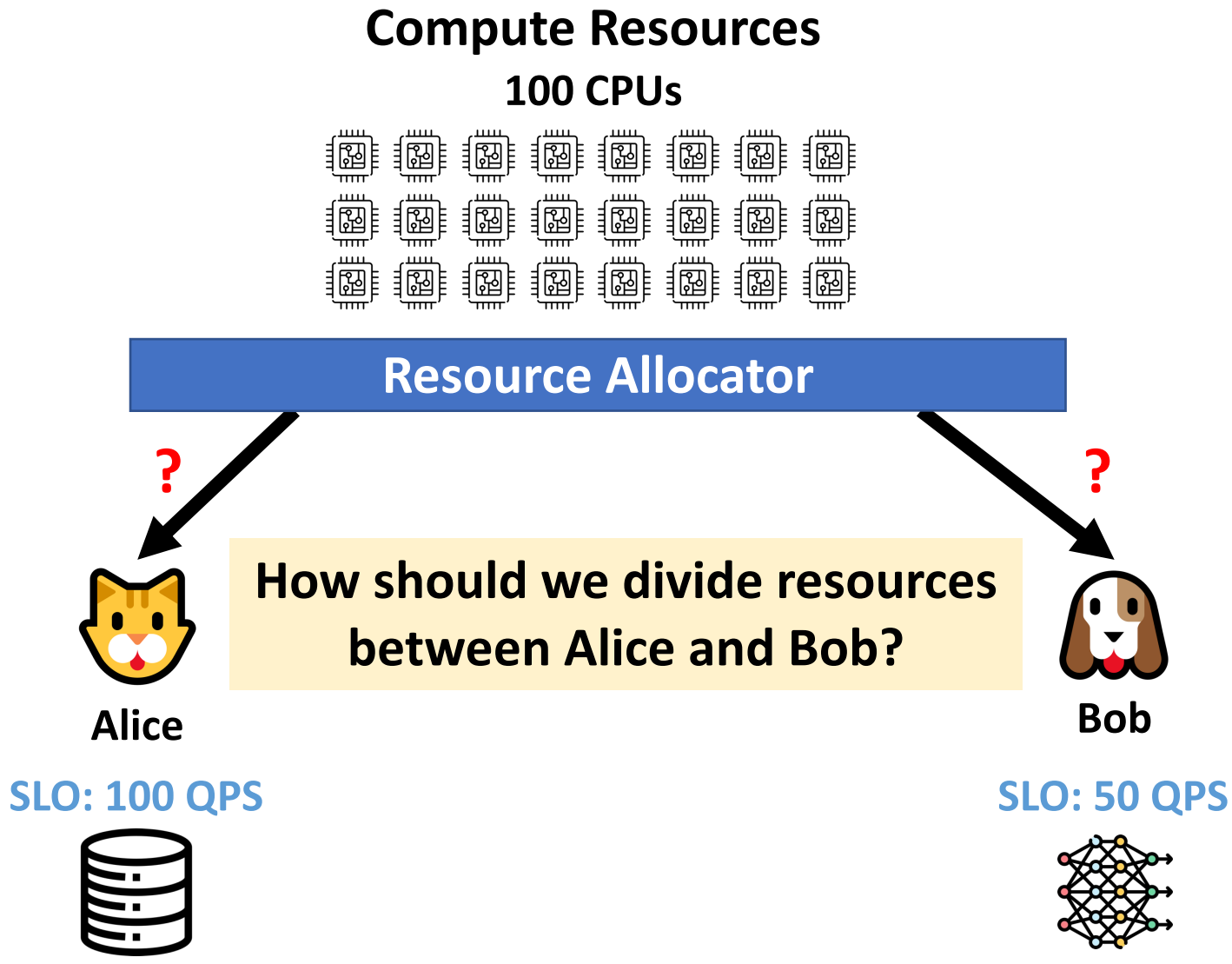
Romil Bhardwaj, Kirthevasan Kandasamy, Asim Biswal, Wenshuo Guo,

Benjamin Hindman, Joseph Gonzalez, Michael Jordan, Ion Stoica

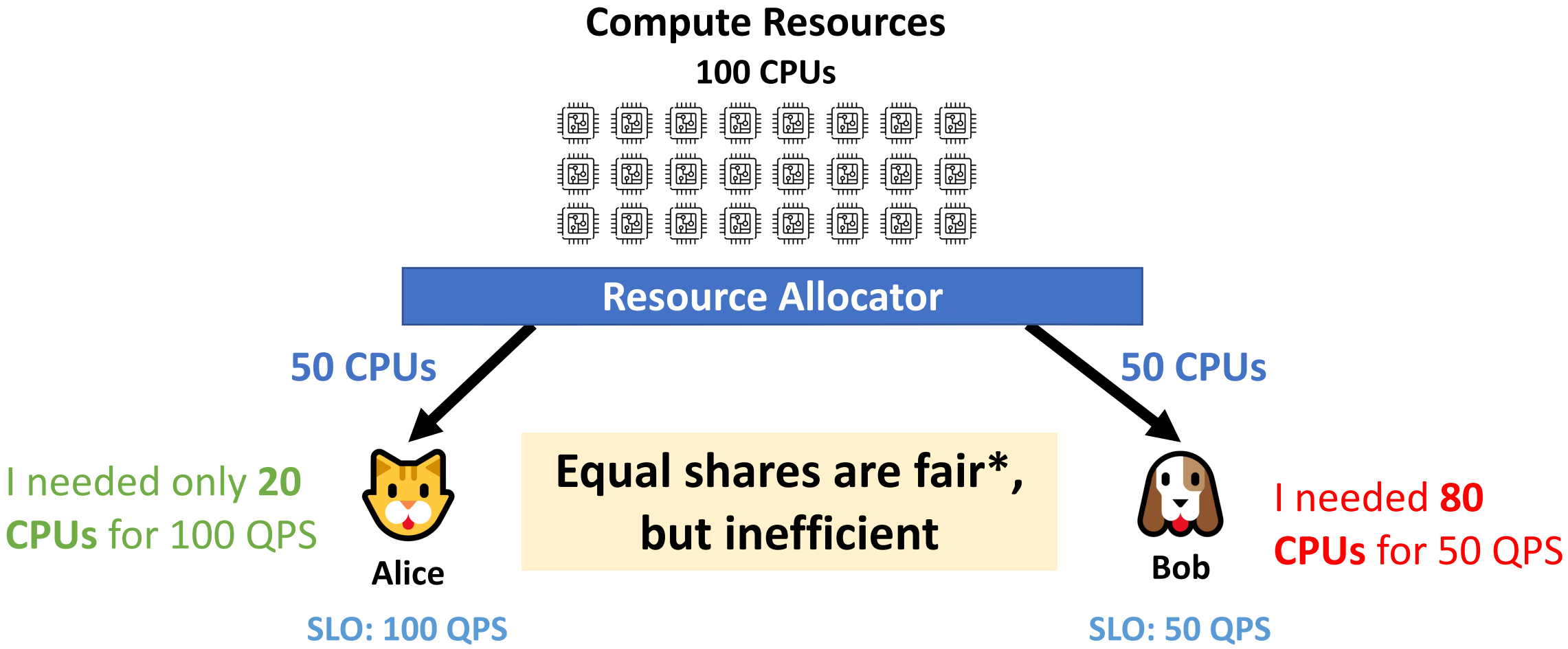
Berkeley
UNIVERSITY OF CALIFORNIA

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Resource allocation in multi-tenant clusters

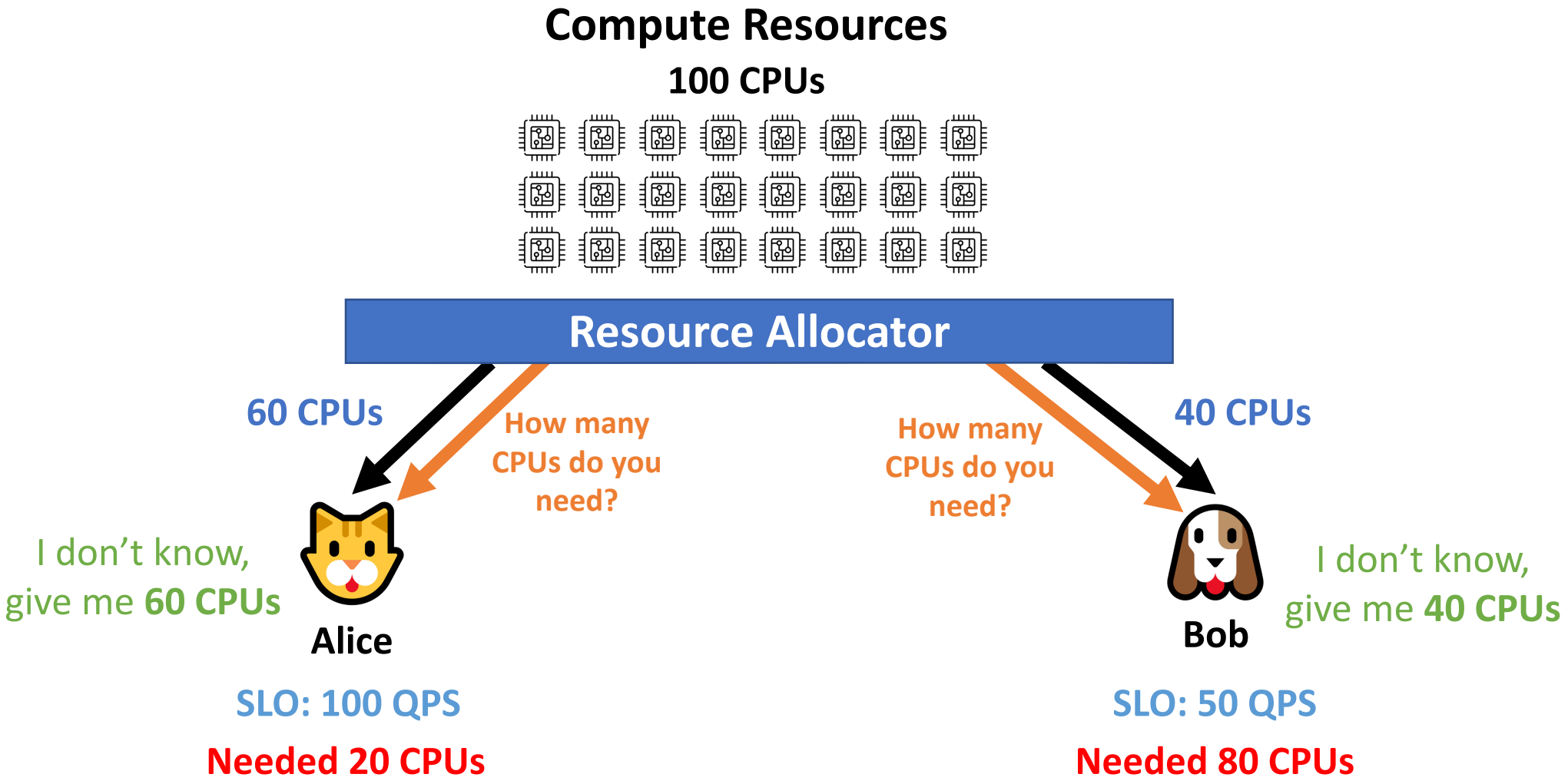


Equal Allocation



*fair = proportional shares

Asking for resource demands

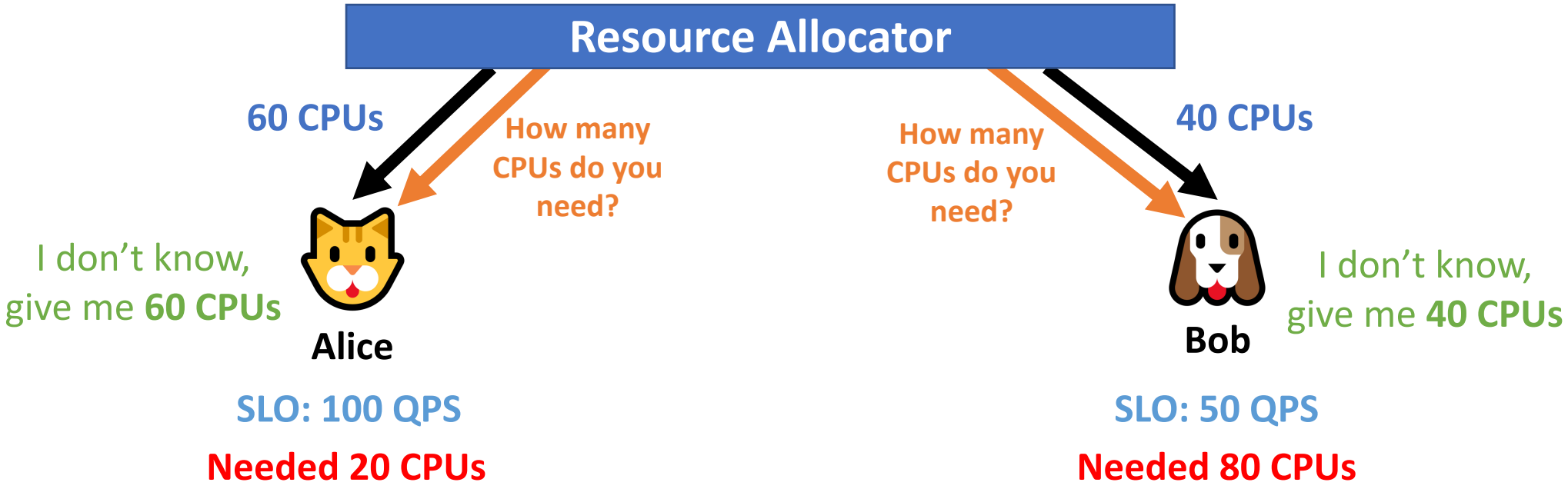


Observed in: Delimitrou & Kozyrakis '14 (PARTIES), Abdu Jyothi et al '16 (Morpheus), Rzadca et al '20 (Google Autopilot)

Asking for resource demands

Compute Resources
100 CPUs

Inaccurate resource demands result in inefficient allocations



Observed in: Delimitrou & Kozyrakis '14 (PARTIES), Abdu Jyothi et al '16 (Morpheus), Rzadca et al '20 (Google Autopilot)

Asking for resource demands

Users have performance objectives, not resource demands

How can we do performance-aware scheduling?

Alice

SLO: 100 QPS

Bob

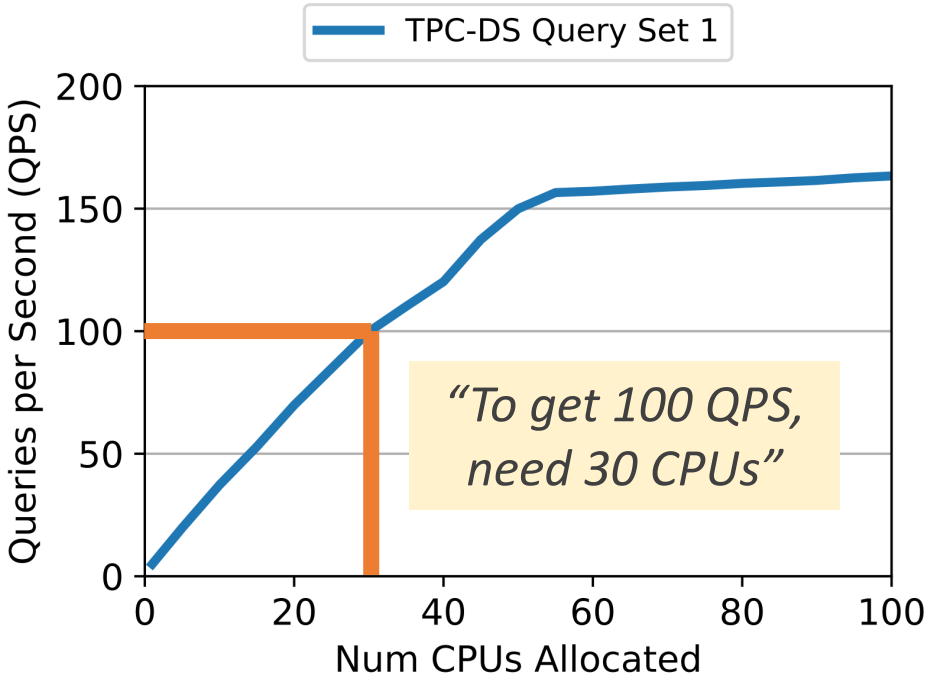
SLO: 50 QPS

Observed in: Delimitrou & Kozyrakis '14 (PARTIES), Abdu Jyothi et al '16 (Morpheus), Rzadca et al '20 (Google Autopilot)

Towards performance-aware scheduling

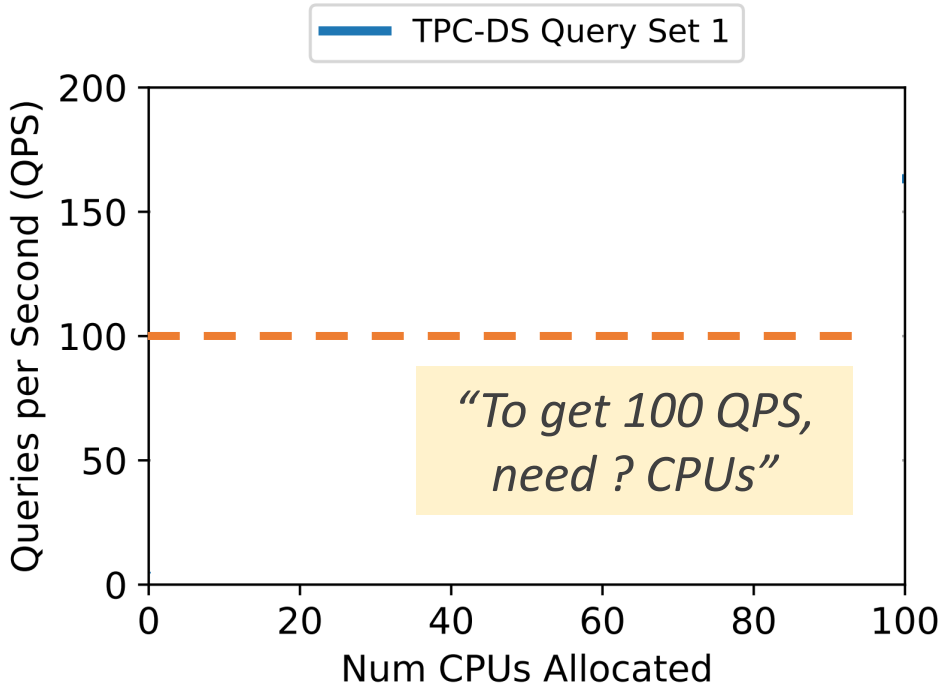
Alice
SLO: 100 QPS

- Users know their **performance objectives (SLOs),** but not the resources required to achieve SLOs
- A **resource-performance mapping** can map SLOs to resource demands

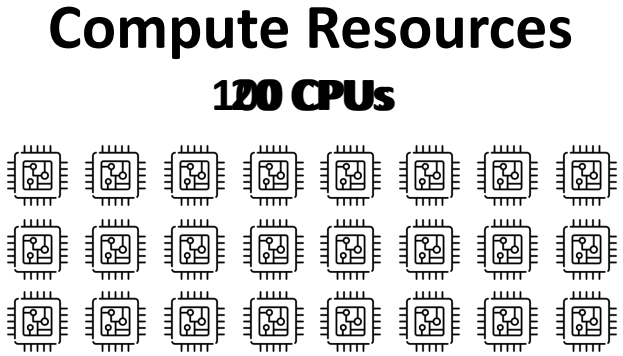


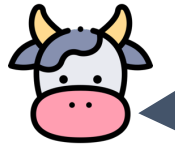
Towards performance-aware scheduling

Challenge 1 - Resource-performance curves are not known



Resource allocation objectives are diverse





Admin

Objective

Maximize the average QPS over all users

Challenge 2 – Scheduling objectives are diverse and require a general resource allocator to satisfy objectives

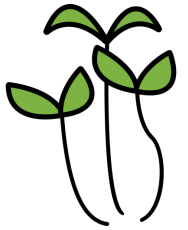
I need **20 CPUs** for 100 QPS

Alice

Bob

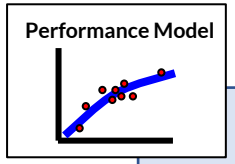
I need **80 CPUs** for 50 QPS

Cilantro



- Cilantro is a **general framework** to enable **performance-aware scheduling** in clusters

Challenge 1 – Resource-performance curves are unknown



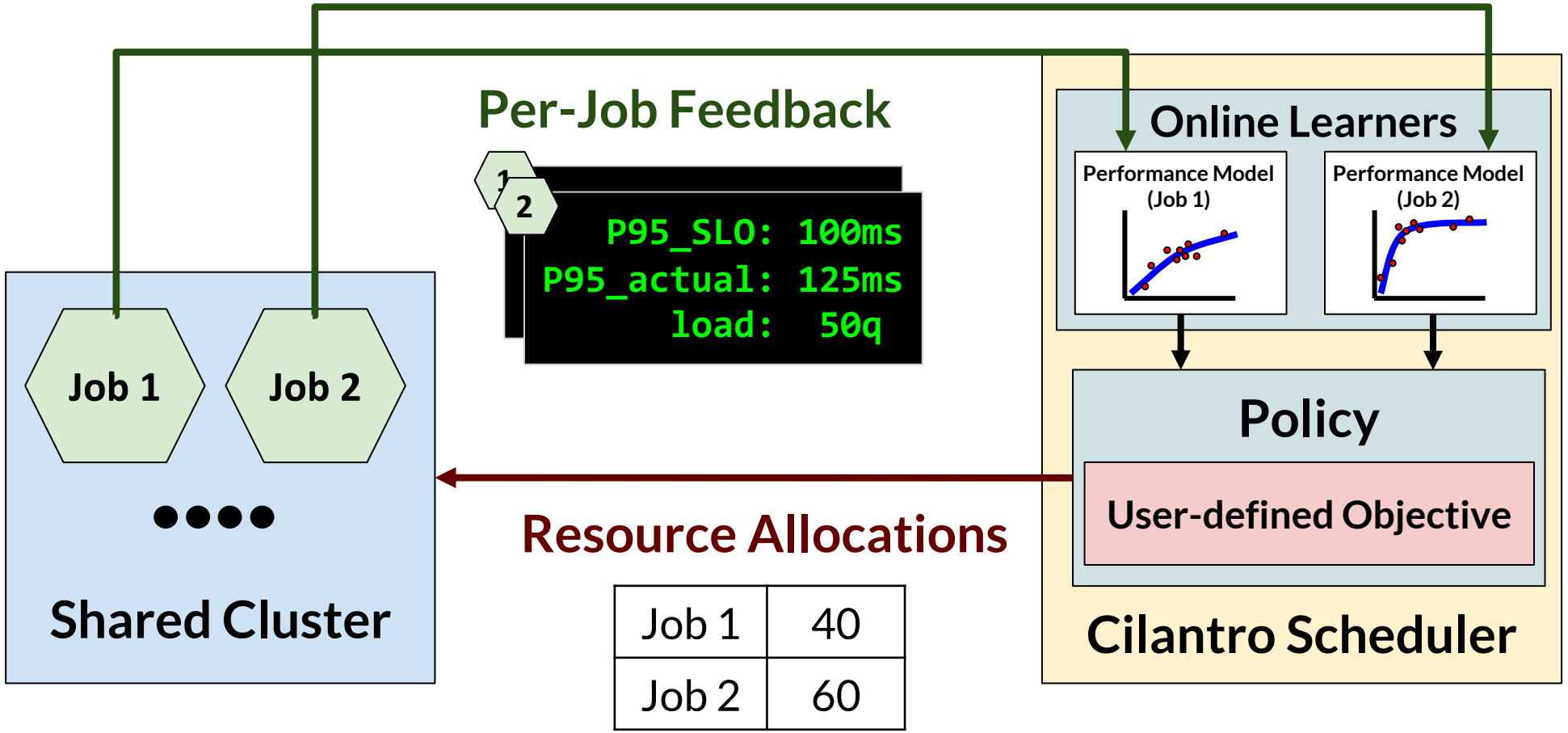
Use **online learning to generate progressively accurate** resource-performance models

Challenge 2 - Support a diverse set of objectives

Decouple allocation policy from learning mechanisms by using resource-performance models

Cilantro workflow

Creating a closed scheduling loop of resource allocations and job feedback



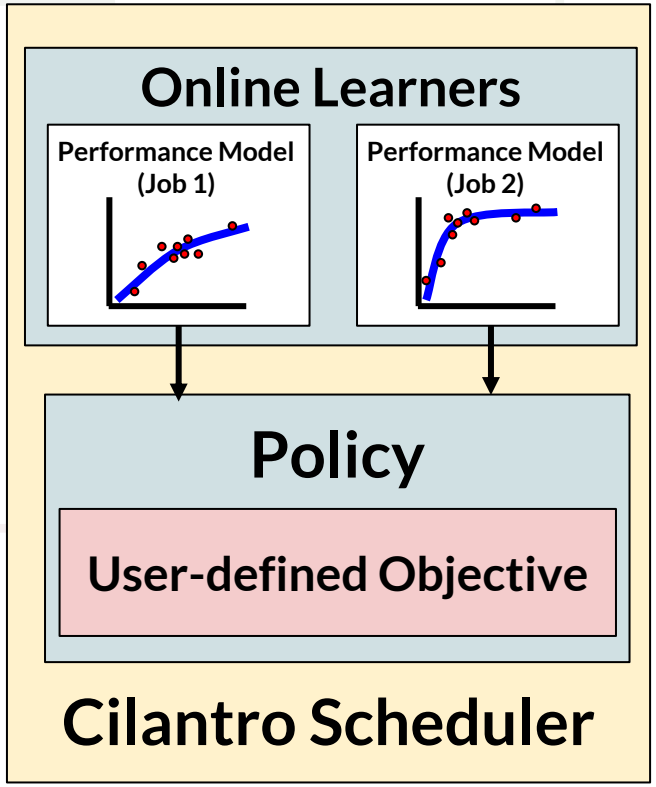
Challenges in Online Learning

Large and expensive search space

Reduce search space by defining a constrained utility model

Model estimates have uncertainty

Make policies uncertainty-aware



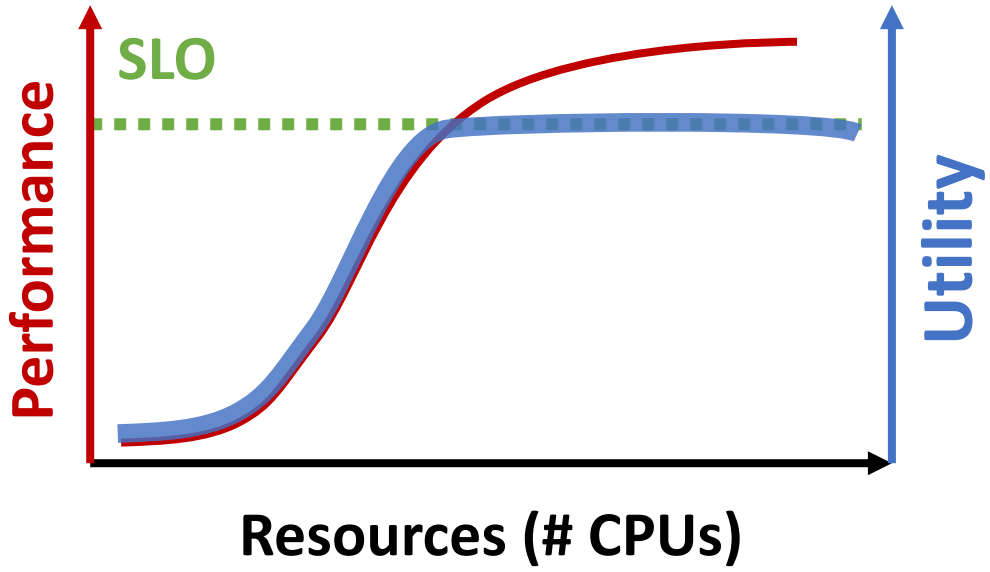
Cilantro utility model

Each job has an unknown resource-performance curve

SLO (Service level objective) is the performance goal specified by the user

Utility is a function of performance. It must be:

1. **Monotonically non-decreasing**
2. **Clipped at SLO**



These properties are desirable because

1. **Makes exploration tractable**
2. **Reflective of real-world "utility"**

Utility u, num jobs n, allocation a, job load l

Common Scheduling Objectives

General Performance objectives

Social Welfare
"Maximize mean utility across all jobs"

$$a = \underset{a_j}{\operatorname{argmax}} \frac{1}{n} \sum_{j=1}^{n} u_j(a_j, l_j)$$

Egalitarian Welfare
"Maximize minimum utility across all jobs"

$$a = \underset{a_j}{\operatorname{argmax}} \left(\min(u_j(a_j, l_j)) \right)$$

Application Performance objectives

Minimize end-to-end latency
of a collection of microservices

$$a^r = \underset{a \in A^r}{\operatorname{argmax}}(u(a, l))$$

Demand-based objectives

No Justified Complaints (NJC) Fairness
"Guarantee at least fair-share"

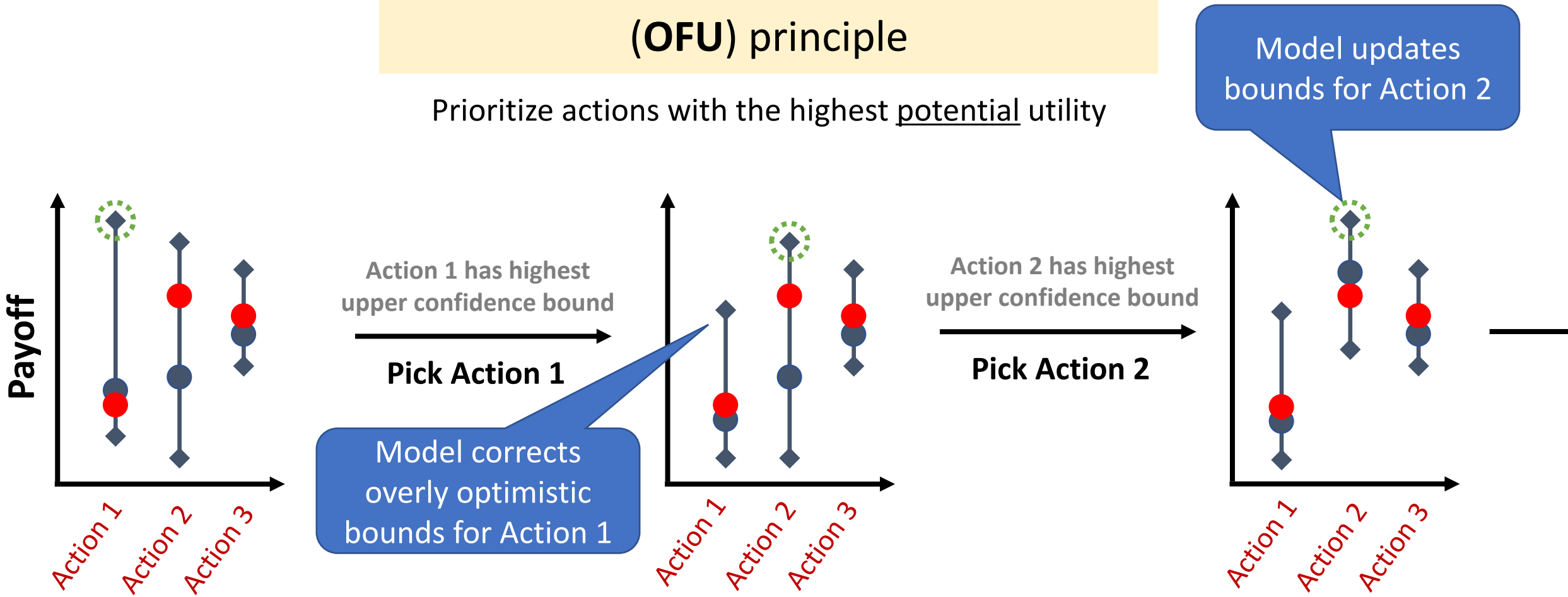
$$a = \underset{a_j}{\operatorname{argmax}} \left(\min_{j \in \{1,\ldots n\}} \left(\frac{u_j(a_j, l_j)}{u_j(R/n, l_j)} \right) \right)$$

Making policies uncertainty-aware

Real payoff
Expected payoff
Confidence bounds

"Optimism in the Face of Uncertainty"[1]
(**OFU**) principle

Prioritize actions with the highest potential utility

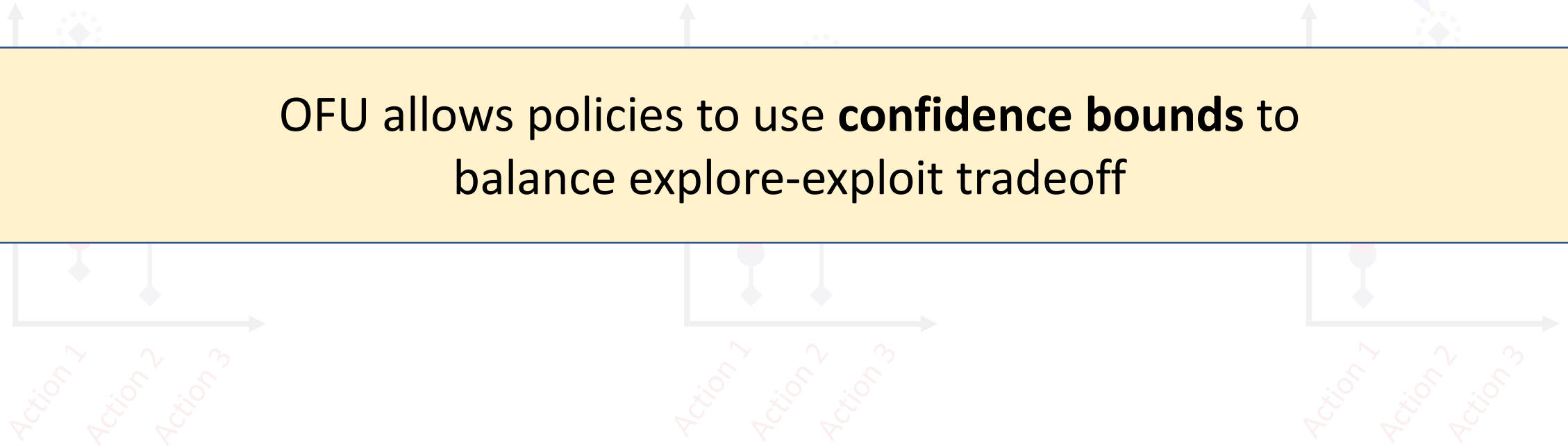


[1] "Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems", Bubek et. al 2012

Making policies uncertainty-aware

"Optimism in the Face of Uncertainty"[1]
(**OFU**) principle

Prioritize actions with the highest potential utility



OFU allows policies to use **confidence bounds** to balance explore-exploit tradeoff

[1] "Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems", Bubek et. al 2012

Utility u, num jobs n, allocation a, job load l

OFU-Adapted Objectives

Performance-based objectives

Social Welfare
"Maximize mean utility across all jobs"

$$a = \underset{a_j}{\operatorname{argmax}} \frac{1}{n} \sum_{j=1}^{n} \widehat{u_j}(a_j, \widehat{l_j})$$

Egalitarian Welfare
"Maximize minimum utility across all jobs"

$$a = \underset{a_j}{\operatorname{argmax}} \left(\min(\widehat{u_j}(a_j, \widehat{l_j})) \right)$$

Application-specific objectives

Minimize end-to-end latency
of a collection of microservices

$$a^r = \underset{a \in A^r}{\operatorname{argmax}} \left(\hat{u}(a, \hat{l}) \right)$$

Demand-based objectives

NJC Fairness
"Guarantee at least fair-share"

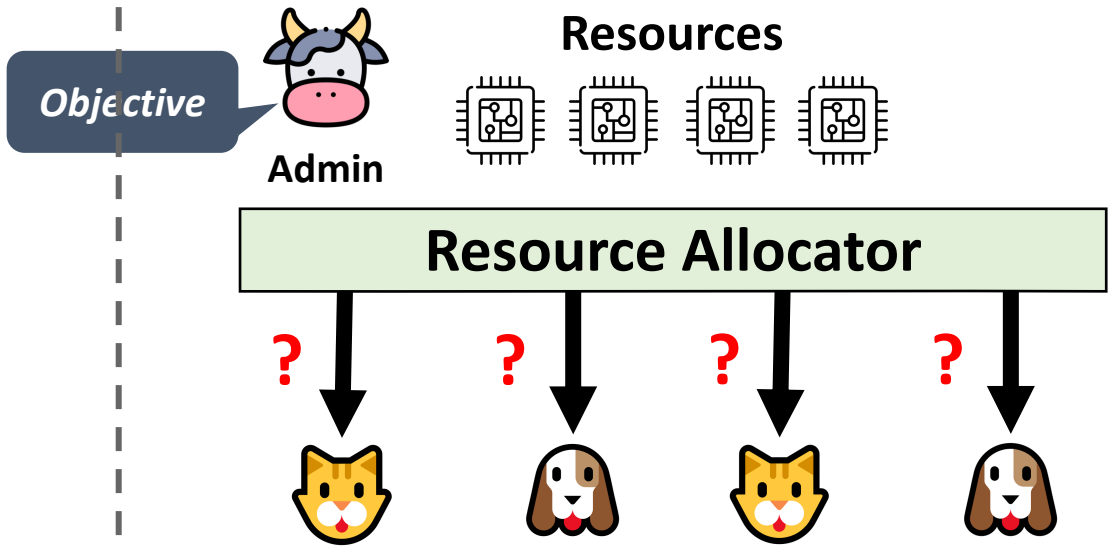
Details in the paper!

Applications of Cilantro

This Talk

Multi-tenant Cluster Sharing

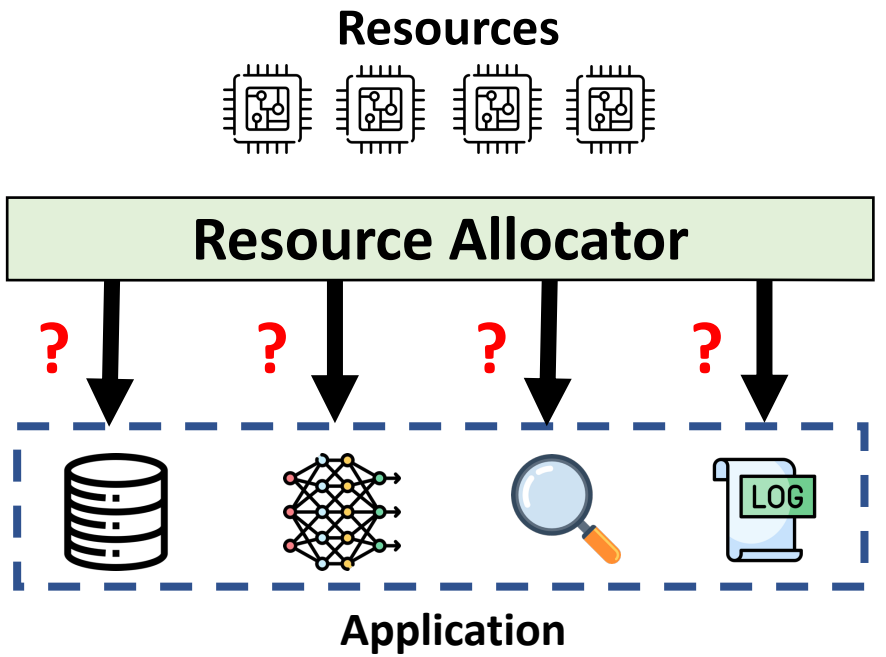
Allocating resources to independent jobs to maximize stated objective



In the paper

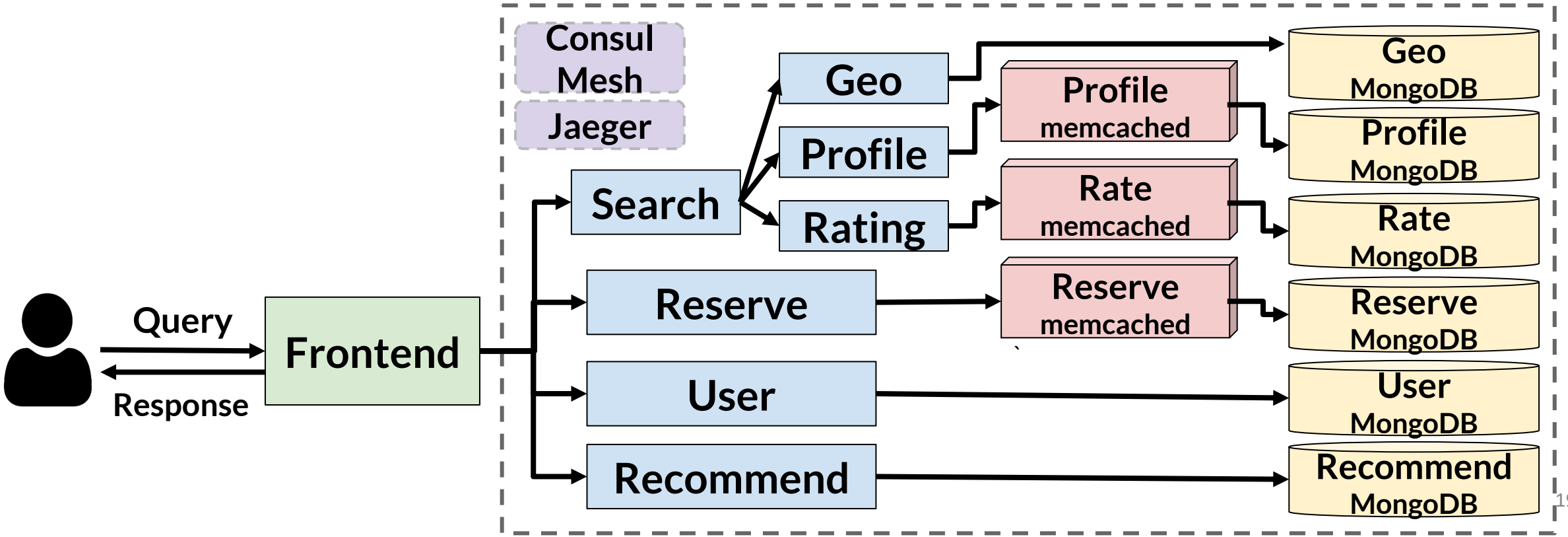
Microservices Resource Allocation

Allocating resources to microservices to minimize end-to-end application latency



Evaluation – Microservices

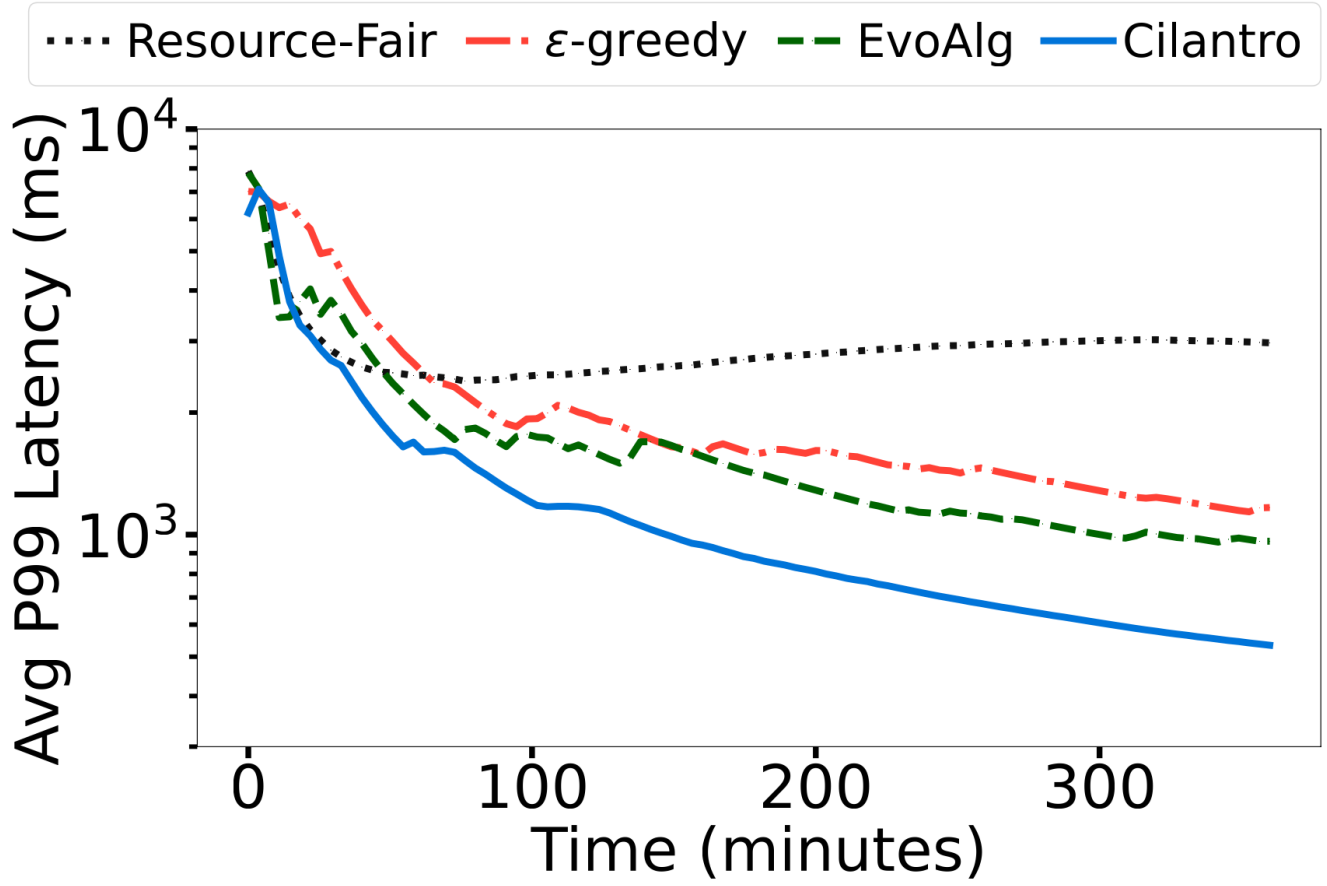
- Hotel Reservation benchmark from DeathStarBench
- Contains 19 microservices serving one endpoint



Evaluation – Microservices

- Hotel Reservation benchmark from DeathStarBench
- Contains 19 microservices serving one endpoint

Evaluation - Microservices



Cilantro reduces P99 latency to 0.57x of the best baseline
without any visibility into the performance or structure of component microservices

Cilantro Summary

github.com/romilbhardwaj/cilantro

romilb@eecs.berkeley.edu

- Cilantro is a **general, performance-aware** cluster scheduling framework
- Users do not state their resource demands - they provide **SLOs and feedback**.
- Online learning estimates **resource-performance curves** and **policies adapt to uncertainty in estimates**
- Improves utilities up to 1.2 – 3.7× in cluster sharing and reduces latencies by 43% for microservices