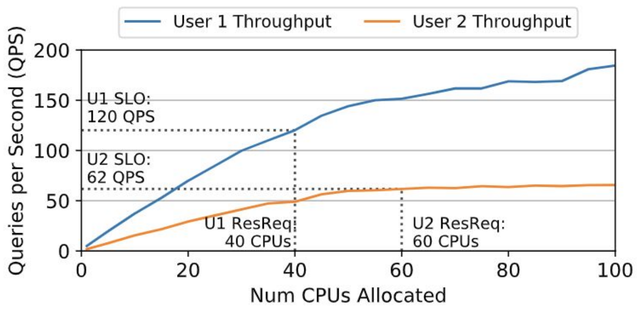
Online Learning for Performance-Aware Resource Allocation

Project 11: Wenshuo Guo (in collaboration with Romil Bhardwaj et al)

Problem

- Traditional systems for resource allocation in **multi-tenant environments** have either aimed at providing **fairness** (in data centers), relied on users to specify their resource requirements, or estimating the resource requirements via **surrogate metrics** (e.g. CPU utilization).
- These approaches fall short on what a user cares most: ***how well their job performs in the real world.***
- In this project, we argue that resource allocation systems should directly account for **real world performance** considerations of the users and build *Cilantro*, a framework for **performance-aware resource allocation in data centers and the cloud.**

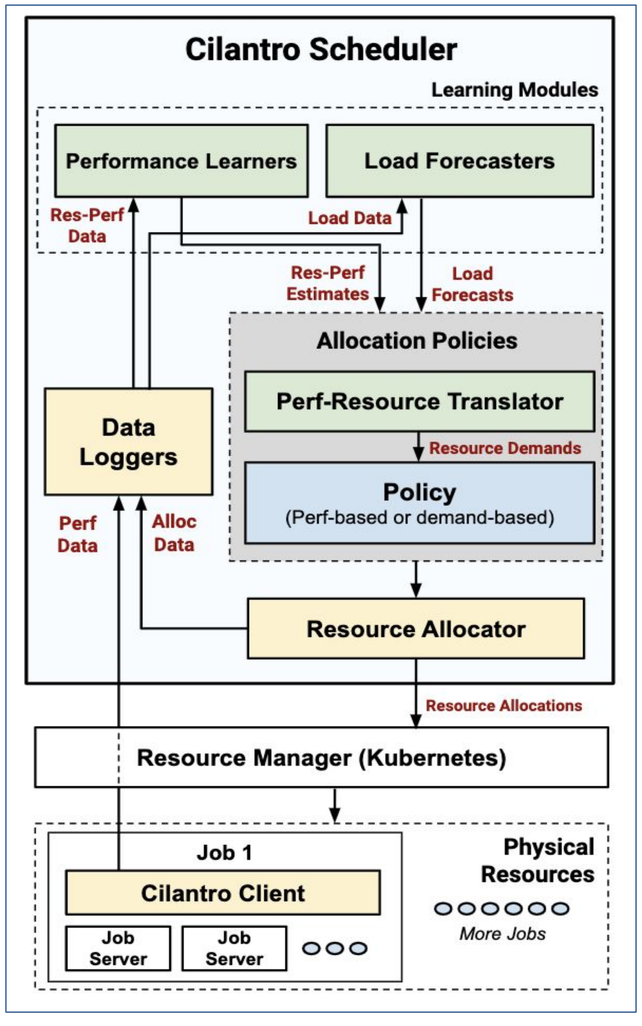
Background



- When allocating resources, a cluster manager should ensure that **the allocations fulfill the user(s)'s overall goals.**
- Consider an example where two users are sharing a cluster of 100 CPUs. The first user's SLO is 120 query-per-second (QPS) which requires 40 CPUs, and the second user's SLO is 62 QPS which requires 60 CPUs.
- A performance-oblivious resource-based fair allocation policy will simply allocate 50 CPUs to each user: only the first user will achieve her SLO while the second one will not;
- If we allocate 40 CPUs to user 1 and 60 to user 2, they both achieve their SLOs. While this might seem unfair from a resource point of view, user 1 may not have complaints since her SLO is satisfied.

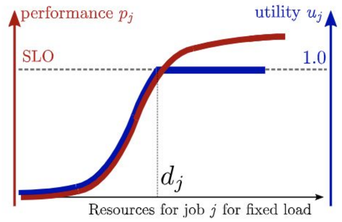
Solution: A Performance-Aware Scheduling Framework

- In this project, we introduce *Cilantro*, a **performance-aware scheduling** framework.
- At the core of *Cilantro* is **an online learning mechanism** which forms feedback loops with jobs to allocate resources and get performance feedback from jobs.
- **A pool of independent learners** analyze this feedback and learn increasingly accurate models of resource-performance curves for each application.
- These models serve as a translation layer for scheduling policies which **converts the performance goals to resource requirements** and vice-versa.



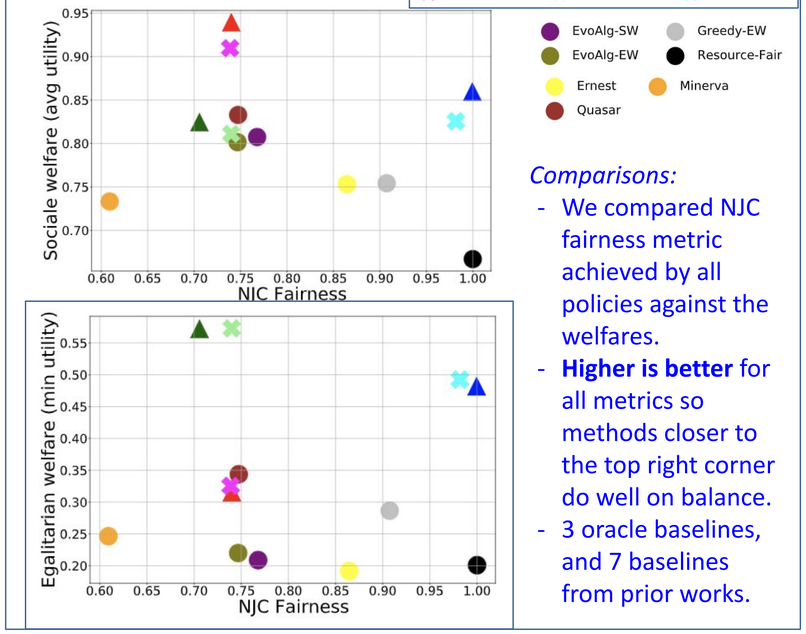
Results

1. **From raw performance metrics to utilities:** the learning algorithm learns an SLO-aware utility (blue curve) derived from the raw performance metric (red curve).



2. **Allocation policies in fixed clusters:**
   - **Social welfare:** $W_{\mathrm{S}} = \frac{1}{n}\sum_{j=1}^{n} u_j(a_j, \ell_j)$.
   - **Max-min fairness:** $W_{\mathrm{E}} = \min_{j\in\{1,\ldots,n\}} u_j(a_j, \ell_j)$.
   - **No justified complaints (NJC) fair division:** (R: total resource) $F_{\mathrm{NJC}} = \min_{j\in\{1,\ldots,n\}} \frac{u_j(a_j, \ell_j)}{u_j(R/n, \ell_j)}$

3. Fixed cluster experiments
   - We use a cluster of 1000 CPUs composed of 250 AWS m5.xlarge instances which have 4 CPUs each.
   - The Cilantro scheduler runs on its own m5.xlarge instance.
   - We use the above 4 workloads to create 20 jobs: 3 DB-0 users, 7 DB-1 users, 3 prediction serving users, 7 machine learning training users



Comparisons:
- We compared NJC fairness metric achieved by all policies against the welfares.
- **Higher is better** for all metrics so methods closer to the top right corner do well on balance.
- 3 oracle baselines, and 7 baselines from prior works.