Cilantro:Performance-Aware Resource Allocation for General Objectives via Online Feedback

汇报人丨许树旺　时间丨2023年12月

目录

CONTENTS

01

Abstract & Introduction

YANXIAOZHAO

Abstract

PLEASE ENTER THE TOPIC

Traditional systems for allocating finite cluster resources among competing jobs have either aimed at providing fairness,relied on users to specify their resource requirements,or have estimated these requirements via surrogate metrics.

These approaches **do not account fora job’ s real world performance.**

Existing performance-aware systems use offline profiled data and/or are designed for specific allocation objectives.

At the core of Cilantro is **an online learning mechanism** which forms feedback loops with the jobs to estimate the resource to performance mappings and load shifts.

This relieves users from the onerous task of job profiling and **collects reliable real-time feedback.**

This is then used to **achieve a variety of user-specified scheduling objectives.**

Cilantro handles the uncertainty in the learned models by adapting the underlying policy to work with confidence bounds.

Introduction

PLEASE ENTER THE TOPIC

The goal of cluster resource managers is to allocate a finite amount of scarce resources to competing jobs.

When doing so,we should ensure that the allocations **fulfill the users' and the organization' s overall goals.**

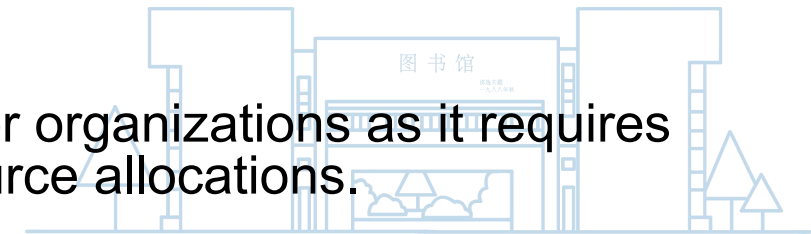
Despite extensive theoretical work,**performance-aware scheduling has remained challenging** since the resource-to-performance mappings are usually unavailable in practice.

Such profiling has three limitations:

1、First,**offline profiled resource-to-performance mappings may not reliably reflect a job' s performance in a production environment**,as it may not capture the interference from other jobs and the server' s performance variability.

2、Second,**jobs' resource requirements change with time due to varying loadand** profiling typically cannot account for these changes.

3、Third,**such profiling is burdensome for users** and expensive for organizations as it requires a large pool of resources to exhaustively profile a wide range of resource allocations.



Introduction

PLEASE ENTER THE TOPIC

This informs the first requirement for this work:

obtain the resource-to-performance mappings in the production environment where the job will be run.

Even if the resource-to-performance mappings are known,the choice of scheduling policy depends on the objective of the end-users.

while end users may find it relatively easy to state their objective,it is harder to design a policy to achieve it.

This inform sour second requirement:

support a diverse set of user-defined scheduling objectives.

Introduction

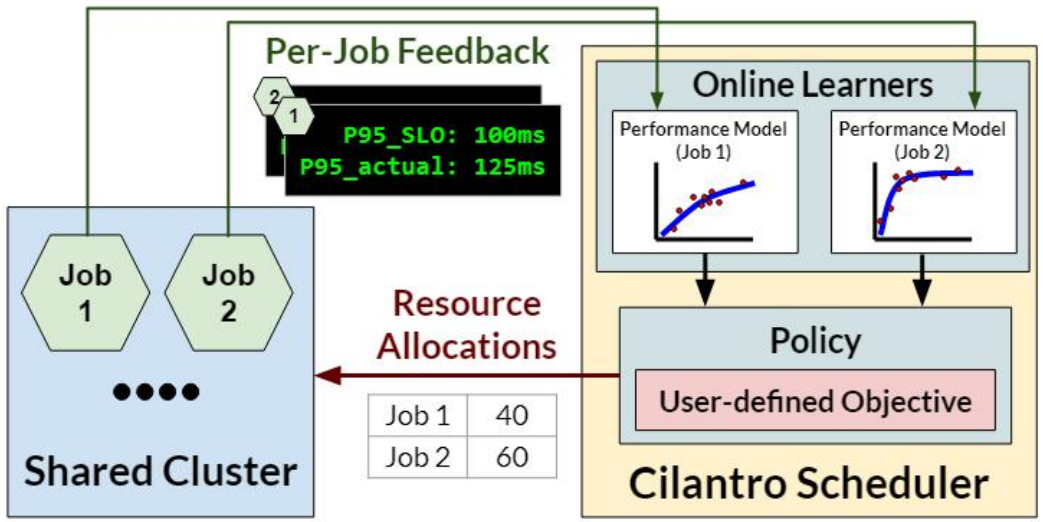
PLEASE ENTER THE TOPIC

To address these requirements,we introduce Cilantro,a frame-work for performance-aware allocation of a single fungible resource type among competing jobs.

In Cilantro,end **users first declare their desired scheduling objective.**

To satisfy the first requirement,a pool of performance learners and load forecasters analyzes live feedback from jobs and learns models to estimate resource-performance curves and load shifts for each job.

To satisfy the second requirement,Cilantro’ s scheduling policies,which are automatically derived based on the users’ objectives,leverage these estimated models to compute allocations for each job.



Introduction

PLEASE ENTER THE TOPIC

Our proposed solution solves two key challenges：

1、First：

estimating resource-to-performance mappings online can be notoriously difficult due to highly stochastic nature of real-time production environments,unexpected load shifts,**especially in the early stages when there is insufficient data.**

To operate without accurate estimates,**Cilantro informs scheduling policies with confidence intervals of its estimates.**

Policies are designed to account for this uncertainty when making allocation decisions until the estimates become more accurate.

Accounting for this uncertainty helps Cilantro conservatively explore the space of allocations making it robust to environment stochasticity and also to the idiosyncrasies specific to the performance models used.

Introduction

PLEASE ENTER THE TOPIC

Our proposed solution solves two key challenges：

2、Second：

supporting a diversity of objectives in the same frame-work is challenging. The monolithic design of end-to-end feedback-driven approaches restricts them only the objective they were originally designed for.

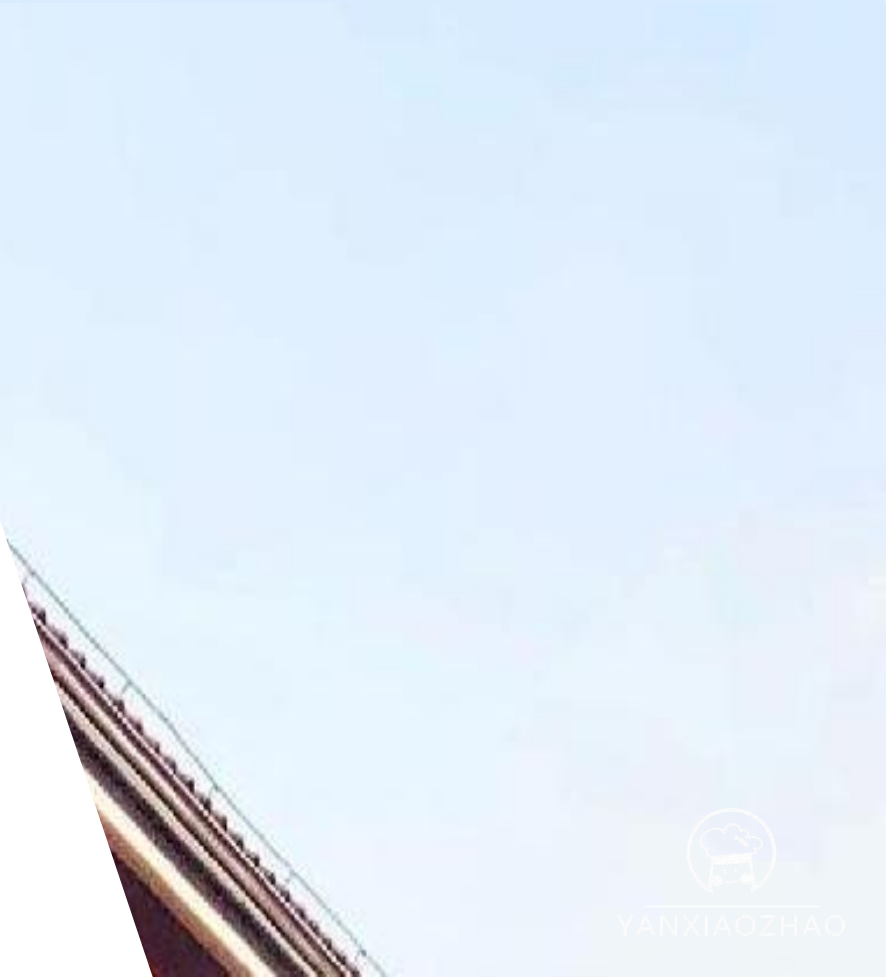
Instead,**Cilantro achieves generality in supporting custom objectives by decoupling the learning mechanisms from the allocation policy.**

This decoupling is necessary as **it allows us to account for the effect of each job’ s performance and load shifts on the objective individually.**

Moreover,this decoupling has other in-tangible benefits:it leads to a more transparent design which is easy to debug than monolithic systems which directly optimize for end-to-end performance,and if online job feedback cannot be obtained for a particular job,it is easy to swap the learners with profiled information or other sensible defaults.

02

Architecture & Policies



Cilantro Architecture

PLEASE ENTER THE TOPIC

Cilantro is a performance-aware scheduling framework that can optimize for various scheduling objectives without requiring any a priori knowledge of the resource-performance mapping of the workloads.

The design of Cilantro is informed by the following two key insights：

1、Performance-aware policies rely on accurate estimates of resource-to-performance mappings and load shifts.

Offline profiling of these resource-performance mappings can be inaccurate due to unpredictability in server and application performance and changing traffic patterns.

Adapting to these changes necessitates continuously learning and predicting these unknowns in an online manner.

2、Decoupling learning mechanisms and policies enables diverse scheduling objectives.

As different scheduling policies optimize different criteria,**it may be challenging for a scheduling framework to generally support different policy types.**

Cilantro Architecture

PLEASE ENTER THE TOPIC

Cilantro is composed of two key components:

1、the centralized Cilantro scheduler
2、the Cilantro clients

the centralized Cilantro scheduler:

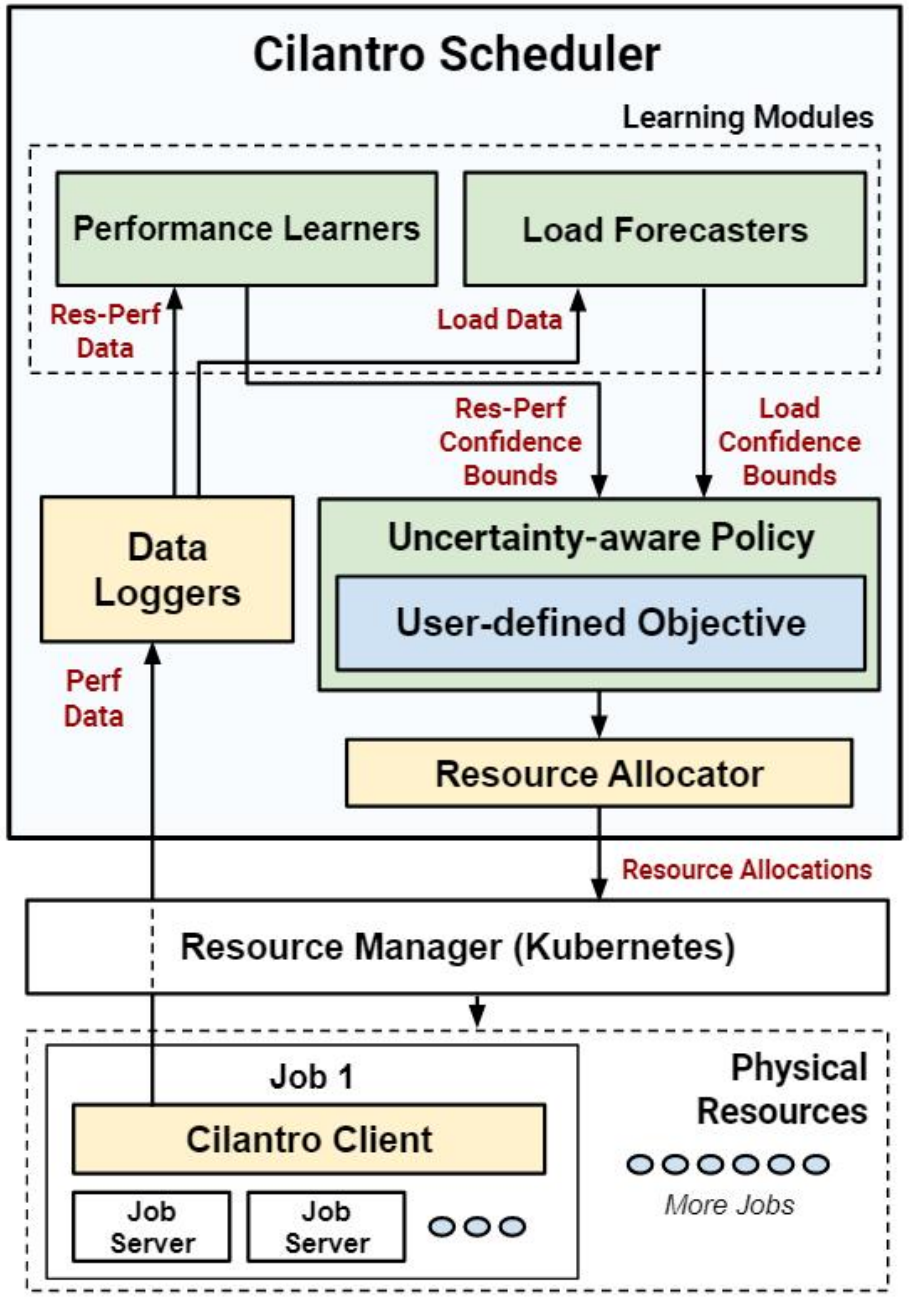
1.Data loggers.

2.Performance learner.

3.Load forecasters.

4.Uncertainty-aware Policy.

5.Resource allocator.



Policies

PLEASE ENTER THE TOPIC

We now describe our policies for performance-aware resource allocation in two settings:

multi-tenant resource allocation in a fixed cluster.
allocating finite resources to constituent micro services of an application.

1、Demand-based policies:

These policies apply when job shave a well-defined SLO and it is possible to define its demand dj.Such policies will compute allocations based on the demands of all jobs.

This requires knowledge of the demand,which in turn depends on the performance mapping.

2、An NJC policy:

This policy proceeds it eratively.

While this policy maynot maximize any welfare,it achieves Pareto-efficient user utilities.

Another advantage of this policy is that it is strategy-proof,i.e a user does not gain additional utility by falsely stating their demand.

Policies

PLEASE ENTER THE TOPIC

Online learning policies in Cilantro：

Our policies will operate on lower and upper confidence bounds obtained from the load forecasters and performance learners instead of the direct estimates;

doing so accounts for the uncertainty in the learned models and encourages a policy to conservatively explore the space of allocations until the estimates become accurate.

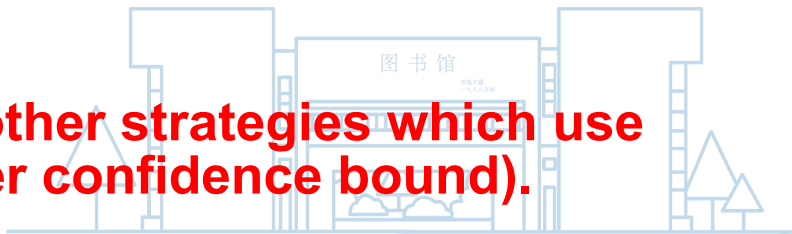
Cilantro′ s policies will proceed sequentially in allocation rounds.On round r,Cilantro chooses an allocation a(r)=(a(r)1,...,a(r)n)based on the feedback from all jobs up to now and the specific scheduling objective.

Welfare-based online policies:

For welfare-based policies,**Cilantro adopts the optimism in the face of uncertainty principle.**

OFU stipulates that,to maximize an uncertain function,we should choose actions which maximize an upper confidence bound(UCB)on the function.

Both the oretically and empirically,**OFU is known to outperform other strategies which use direct estimates or those which are pessimistic(i.e.maximize lower confidence bound).**



Policies

PLEASE ENTER THE TOPIC

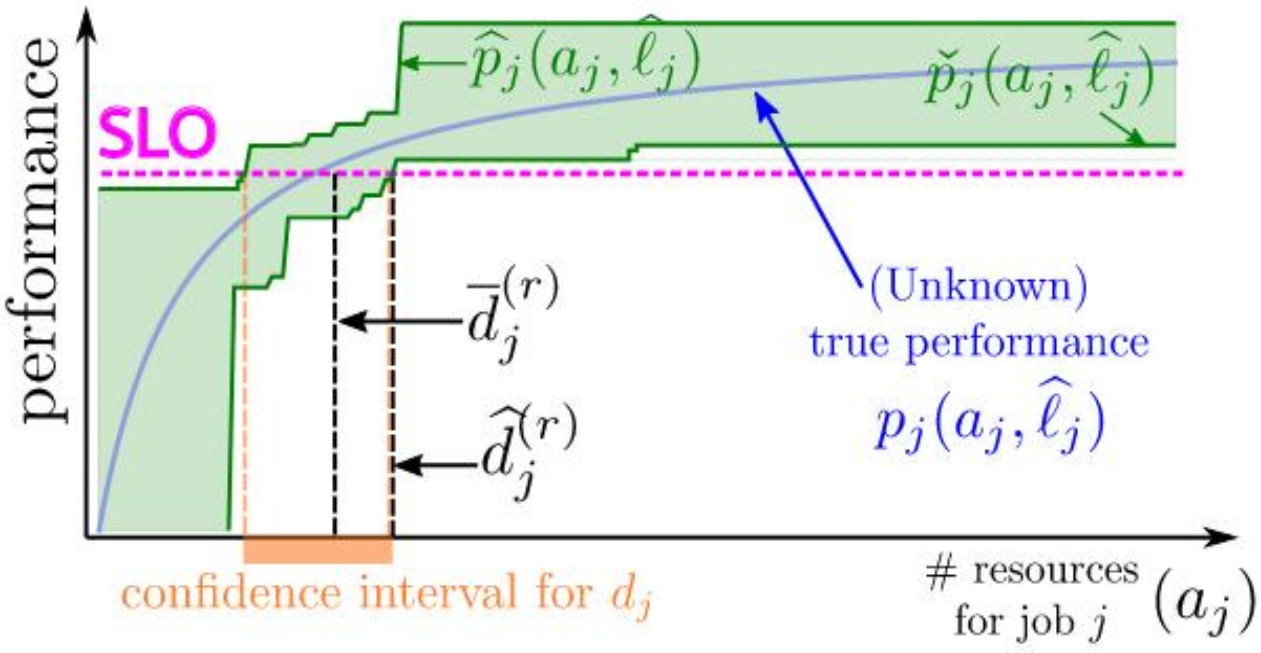
Illustration of Cilantro′ s uncertainty-aware demand-basedpolicies：

We first **obtain a UCB ℓjfrom the load forecaster**,**whichensures that we have a conservative estimate on the job′ s load.**

We **show the SLO(pink)**,**the slice of the unknownperformance curve(blue) when the load is ℓj,and the confidencereg obtained from past data(green).**

The LCB pjand UCB p jon p j(a,ℓ j)are given by the lower and upper boundaries of theconfidence region(solid green lines).

A confidence interval for thedemand(orange) can be obtained by the region where p j, pjintersectthe SLO line.



To obtain a recommendation,we compute a UCB d(r)jon the demand(where SLO intersects pj)and d(r)jvia equation(5).

03

Discussion & Implementation

Discussion

PLEASE ENTER THE TOPIC

We now present a discussion on **Cilantro′ s operation under various adversarial conditions that may occur in deployment.**

1.Cilantro provides three fallback options when online feedback is not available.

1、**First**,**Cilantro allows a user to use a profiled model i**nstead of online feedback.

2、**Second**,it allows using proxy metrics from the Kubernetes API instead of real-world performance.In such cases,a user should specify how these proxies are tied to their utility and/or demand.

3、**Third**,if neither of these is possible,we allow the user to directly **submit an estimate for their resource demand which will then be fed to the policy** when determining allocations.In such cases,we assume that utility increases linearly up to the demand when computing allocations.

2.Learning in unpredictable environments.

Some situations,such as unexpected load spikes for web services or interference between jobs,are fundamentally hard to predict.

Cilantro′ s uncertainty-aware design provides a degree of resilience against these unpredictable changes， as we show init′ s robustness to noise in load and resource demand estimates

Implementation

PLEASE ENTER THE TOPIC

The Cilantro scheduler is implemented in 7600 lines of Python code,as a standalone scheduler for Kubernetes.Resource reallocation events are triggered by a timer-based event,which is raised every 2 minutes in our experiments.

To execute updated resource allocations received from policies,we horizontally scale the workloads by adding more replicas to their Kubernetes deployment.

The frequency of performance feedback depends on the application and the environment.

Specifying utilities and objectives.

Utilities of jobs are calculated based on the performance metrics collected by the Cilantro clients in the last resource allocation round.

Learning models and load forecasters.

04

Evaluation & Conclusion

Evaluation

PLEASE ENTER THE TOPIC

We evaluate Cilantro in two settings described:

1、In the multi-tenant setting

Cilantro′ s online learning policies,which do not start with any prior data,are competitive with oracular policies which have access to jobs′ resource to performance mappings obtained after several hours of profiling.

2、In the micro services setting

Cilantro is able to support the completely different objective of minimizing end-to-end latency.It outperforms three other baselines and reduces the P99 latency to×0.57 that achieved by the next best performance-aware baseline.

In our micro benchmarks，we show that Cilantro′ s allocation policies are inexpensive,evaluate its fallback options when performance metrics are unavailable,and demonstrate its robustness to errors in feedback and choices for performance learner and forecaster models.

Evaluation

PLEASE ENTER THE TOPIC

Multi-tenant cluster sharing

We first evaluate Cilantro′ s multi-tenant policies(§4.1.2)on a 1000 CPU cluster shared by 20 users.

Evaluation on performance-aware fairness metrics.We first compare all 15 baselines on the **social welfare(1),egalitarian welfare(2),and the NJC fairness criteria(3).**

While the oracular methods perform best on their respective metrics,we find that the online learning policies in Cilantro come close to matching them.Resource-Fair achieves a perfect NJC score by definition,but performs poorly on social and egalitarian welfare as it is performance oblivious.

We found that **Greedy-EW,Parties,and MIAD were sensitive to the amount by which we changed the allocations based on feedback**;when tuning them,we found that they were either too slow or too aggressive when responding to load shifts.Next,the learning models used by Quasar and Ernest were not able to accurately estimate the demands in our experiment.

Finally,**the evolutionary baselines were inefficient**,taking along time to discover the optimal solution.They,however,were effective within Cilantro′ s welfare policies when you need to optimize a cheap analytically computable function as they can be run for several iterations.

Evaluation

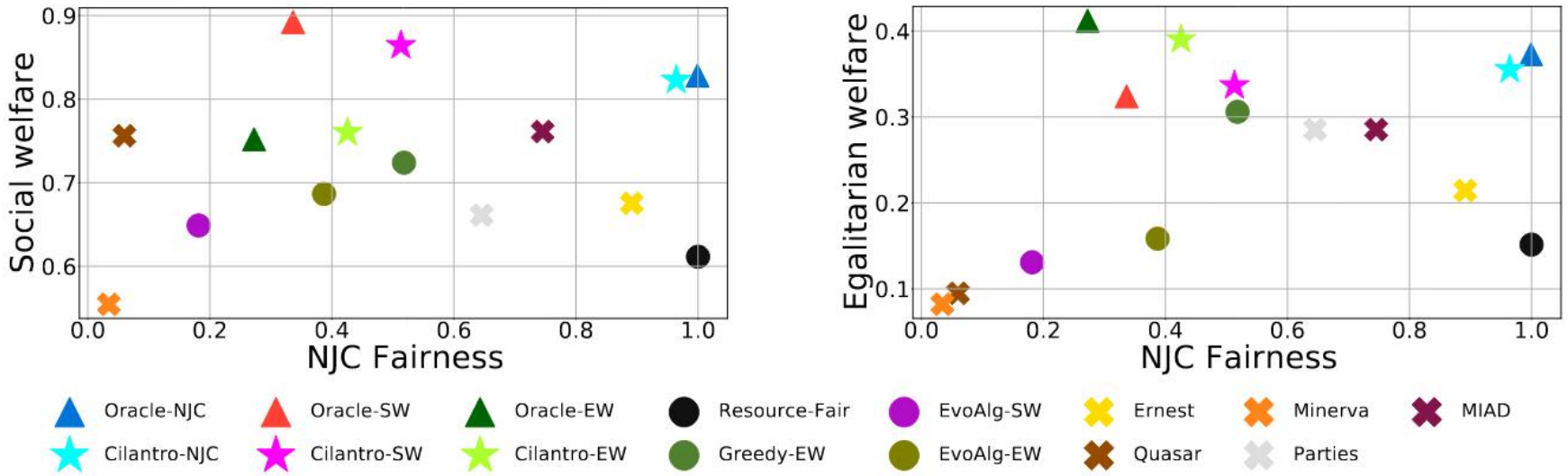
PLEASE ENTER THE TOPIC

NJC fairness vs the social and egalitarian welfare for all policies：

We report the average value over the 6 hour period.

Higher is better for all metrics,so closer to the top right corner is desirable.The Oracle-SW,Oracle-EW policies optimize for the social andegalitarian welfare when the performance mappings are known and Oracle-NJC achieves maximum fairness while improving cluster usage.

The corresponding **Cilantro policies are designed to do the same without a priori knowledge of the performance mappings.**

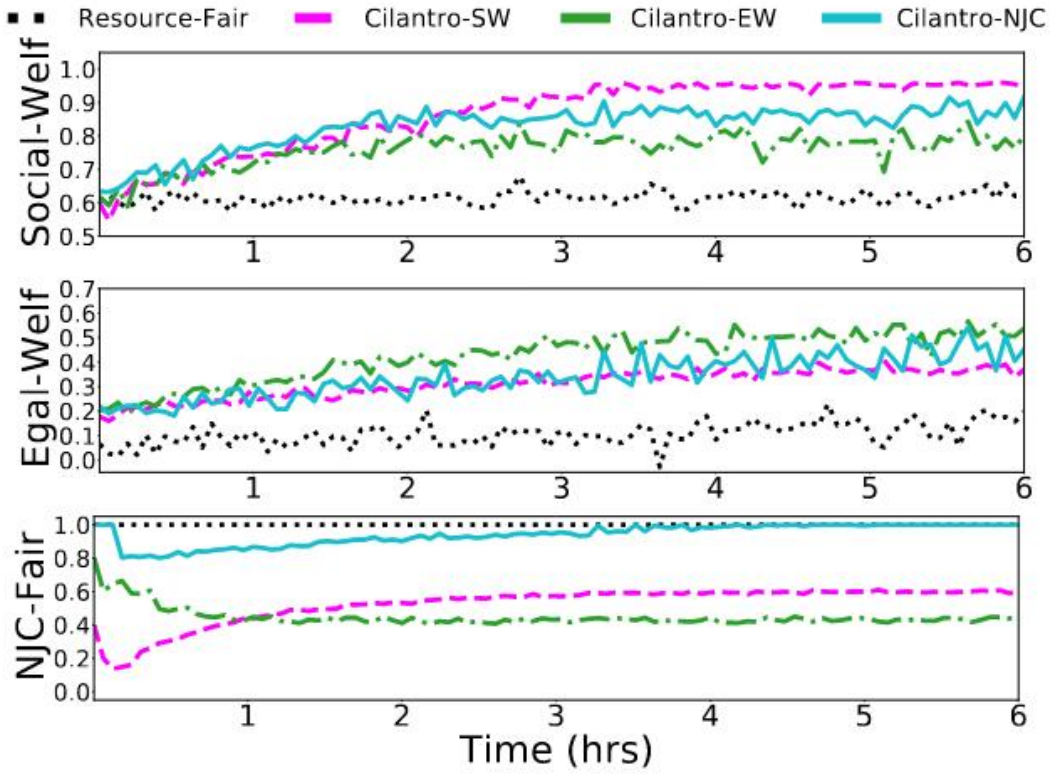


Evaluation

PLEASE ENTER THE TOPIC

Convergence over time of social,egalitarian welfares andNJC fairness for the three Cilantro policies：

maximized social welfare by allocating more resources tojobs that can quickly achieve high utility.



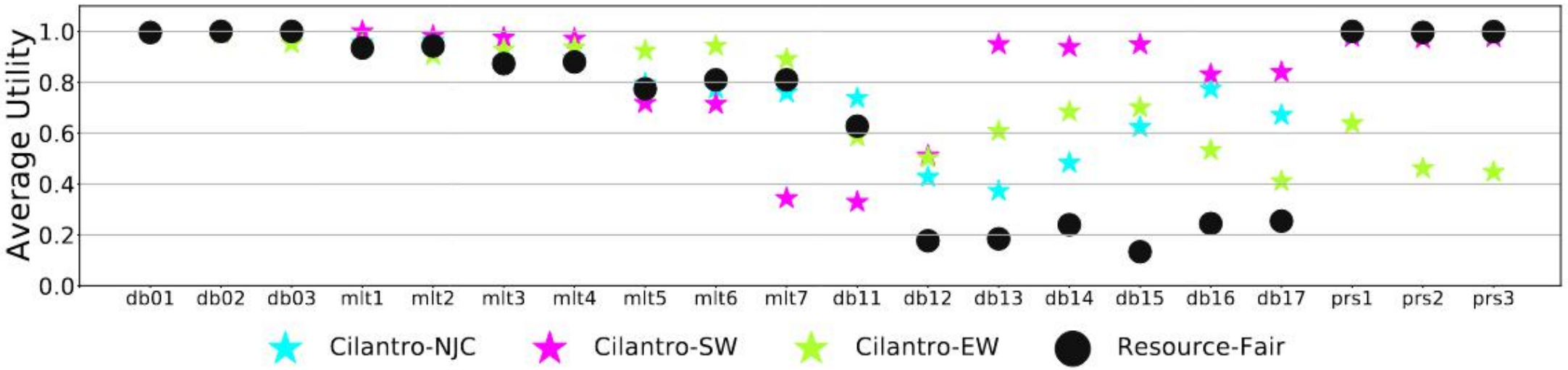
Evaluation

PLEASE ENTER THE TOPIC

Resource allocation for Microservices

The average utility achieved by the 20 jobs for the three online learning methods in Cilantro and Resource-Fair.

Here,db0x,mltx,db1x,and prsx refers to jobs using the DB-0,ML training,DB-1,and prediction serving workloads.



Evaluation

PLEASE ENTER THE TOPIC

The social welfare(1),egalitarian welfare(2),NJC fairness metric(3),and the effective resource usage(8)for all 13 methods.

Higheris better for all four metrics,and the maximum and minimum possible values for all metrics are 1 and 0.

The values shown in bold have achievethe highest value for the specific metric,besides the oracular policies.Resource-Fair has NJC fairness FNJC=1 by definition.

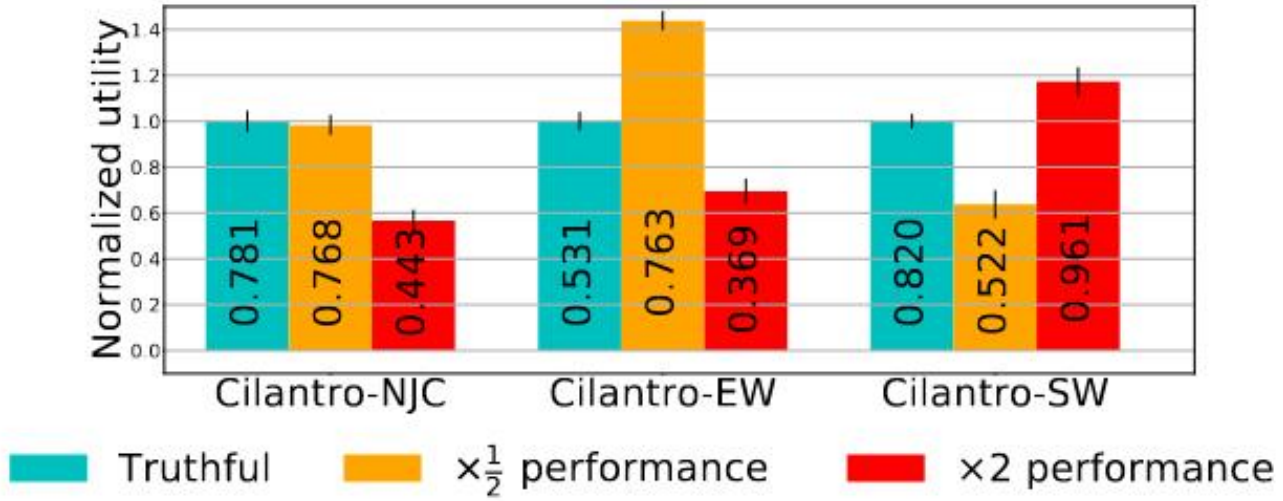
Policy	Social Welfare, (W_S)	Egalitarian Welfare (W_E)	NJC Fairness (F_{NJC})	Useful resource usage
Oracle-SW	0.892 ± 0.004	0.324 ± 0.008	0.336 ± 0.004	0.964 ± 0.002
Oracle-EW	0.752 ± 0.003	0.412 ± 0.007	0.272 ± 0.002	0.997 ± 0.000
Oracle-NJC	0.828 ± 0.002	0.373 ± 0.008	0.999 ± 0.000	0.991 ± 0.000
Cilantro-SW	$\mathbf{0.864 \pm 0.006}$	0.337 ± 0.013	0.513 ± 0.020	0.818 ± 0.012
Cilantro-EW	0.760 ± 0.007	$\mathbf{0.390 \pm 0.020}$	0.426 ± 0.037	$\mathbf{0.954 \pm 0.012}$
Cilantro-NJC	0.823 ± 0.002	0.355 ± 0.005	$\mathbf{0.964 \pm 0.006}$	0.931 ± 0.003
EvoAlg-SW	0.649 ± 0.017	0.131 ± 0.016	0.182 ± 0.048	0.671 ± 0.021
EvoAlg-EW	0.687 ± 0.011	0.158 ± 0.012	0.387 ± 0.040	0.700 ± 0.009
Resource-Fair	0.611 ± 0.002	0.151 ± 0.006	$\mathbf{1.000 \pm 0.000}$	0.766 ± 0.001
Greedy-EW	0.724 ± 0.005	0.306 ± 0.006	0.518 ± 0.009	0.882 ± 0.004
Ernest	0.675 ± 0.002	0.214 ± 0.005	0.891 ± 0.013	0.774 ± 0.002
Quasar	0.756 ± 0.002	0.095 ± 0.003	0.060 ± 0.003	0.706 ± 0.002
Minerva	0.555 ± 0.017	0.082 ± 0.006	0.034 ± 0.005	0.407 ± 0.023
Parties	0.661 ± 0.002	0.285 ± 0.006	0.645 ± 0.000	0.766 ± 0.001
MIAD	0.761 ± 0.002	0.285 ± 0.005	0.745 ± 0.000	0.766 ± 0.001

Evaluation

PLEASE ENTER THE TOPIC

The utility of db16 under the three online learningpolicies,when they report truthfully,when they under-report,andwhen they over-report.

The plot normalizes with respect to truthfulreporting,but the bars are annotated with the absolute value.or uses the allocation with the smallest observed P99 latencywith probability 2/3.



Both Cilantro and EvoAlg explore early on(Fig.12-Center),but as they find better values,exploration shrinks as they focus on testing more promising allocations.

However,Cilantro′ sOFU-based online learning policy is able to do this more effectively than EvoAlg.

ε-greedy explores aggressively even in later stages and is unable to adequately exploit good candidates it may have discovered in the early stages.

Conclusion

PLEASE ENTER THE TOPIC

We described Cilantro,a performance-aware framework for the allocation of a finite amount of resources among competing jobs.

Our motivations were:

(i)resource allocation policies should be performance-aware and based on real-time feedback in production environments,

(ii)schedule rsshould accommodate diverse allocation objectives.

We designed Cilantro to address these challenges by decoupling the performance learning from the policies and informing the policies of uncertainties in performance estimates,thus enabling the realization of several performance-aware policies in multi-tenant and micro services settings.

My Conclusion

Traditional systems and Existing performance-aware systems

Such profiling has three limitations

This informs two requirements for this work

To address these requirements,we introduce Cilantro: Cilantro is an online learning mechanism

The design of Cilantro is informed by the following two key insights

Cilantro Architecture: Cilantro is composed of two key components
Policies: Online learning policies in Cilantro:

Cilantro’ s operation under various adversarial conditions that may occur in deployment

The Cilantro scheduler is implemented in 7600 lines of Python code

We evaluate Cilantro in two settings described

Conclusion

敬请批评指正

汇报人丨许树旺　　时间丨2023年12月