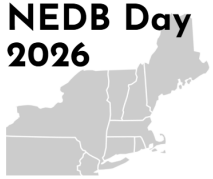
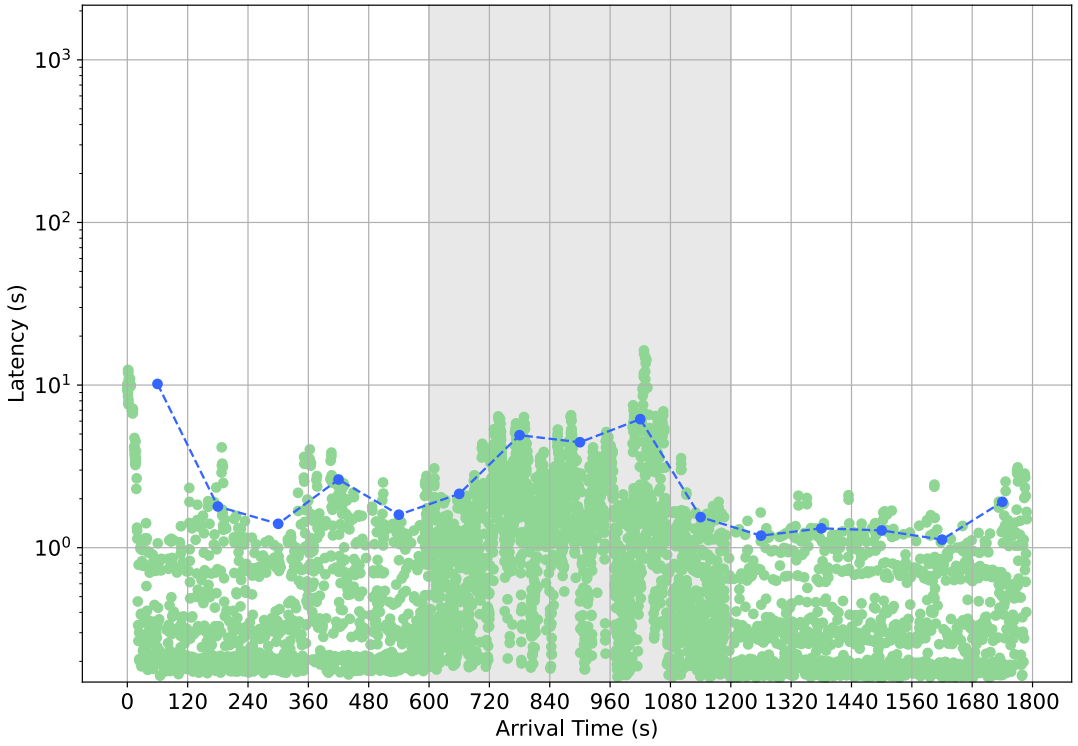
Meeting Latency SLOs on Cloud Data Warehouses

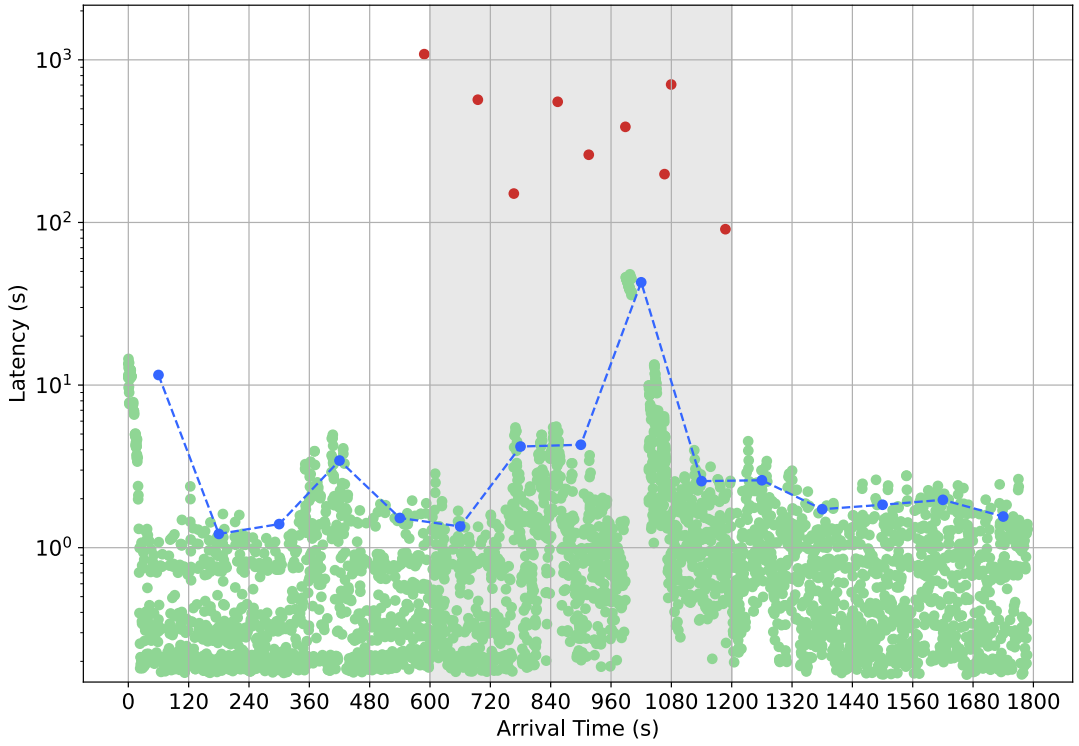
Markos Markakis, Ziniu Wu, Tim Kraska
Massachusetts Institute of Technology



I want
(at least y% of)
my queries to finish
in under x seconds
cheaply

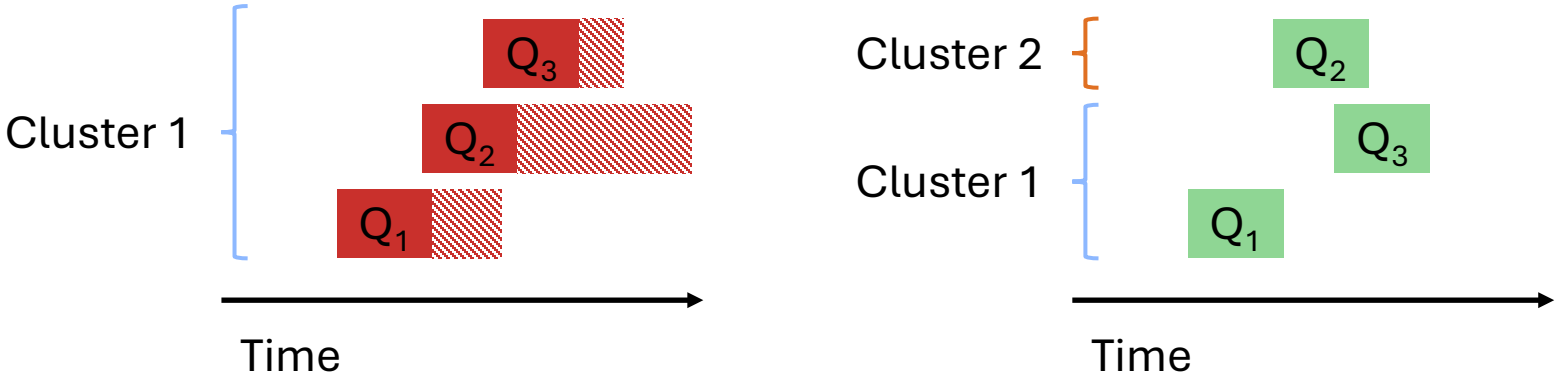
Workload variability makes latency SLOs hard





Number of clusters is a “knob” we can affect

- Snowflake-like architectures abstract away many of the traditional configuration knobs.
- But allow accessing the same data with **multiple compute clusters**, each with **independent computational capabilities.**
- We can safeguard SLOs by isolating incompatible queries.



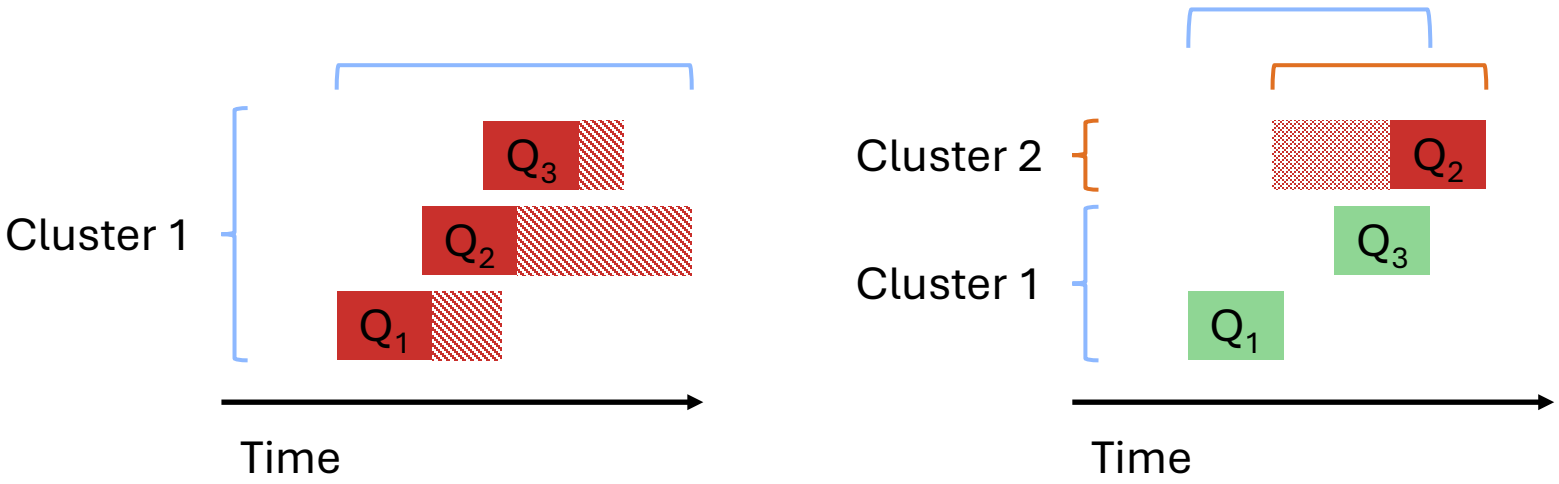
Existing work does not cover this adequately

	Past Work	Has SLOs	Multi-Cluster Cost Balancing	Models Query Interference
Single-cluster autotuning	Das et al. (2016)	✓	⊗	✓
	ResTune (2021)	✓	⊗	✓
Multi-cluster, but no contention	WiseDB (2016)	✓	✓	⊗
	SLAOrchestrator (2018)	✓	✓	⊗
	BRAD (2024)	✓	✓	⊗
Real-world, but no explicit SLO	Auto-WLM (2023)	⊗	~✓ (burst)	~✓ (queueing)
	RAIS (2024)	⊗	~✓ (burst)	~✓ (queueing)

Can we extend our work from BRAD to address this setting?

Problem: Multiple clusters don't come for free

- At the limit, could we provision a cluster for each query?
- No, because of two related costs:
  - **[Time] Startup penalty:** New clusters takes time and are cold.
  - **[Money] Per-second billing:** Marginal dollar cost is zero on active cluster.



The Problem

Our Design

Evaluation

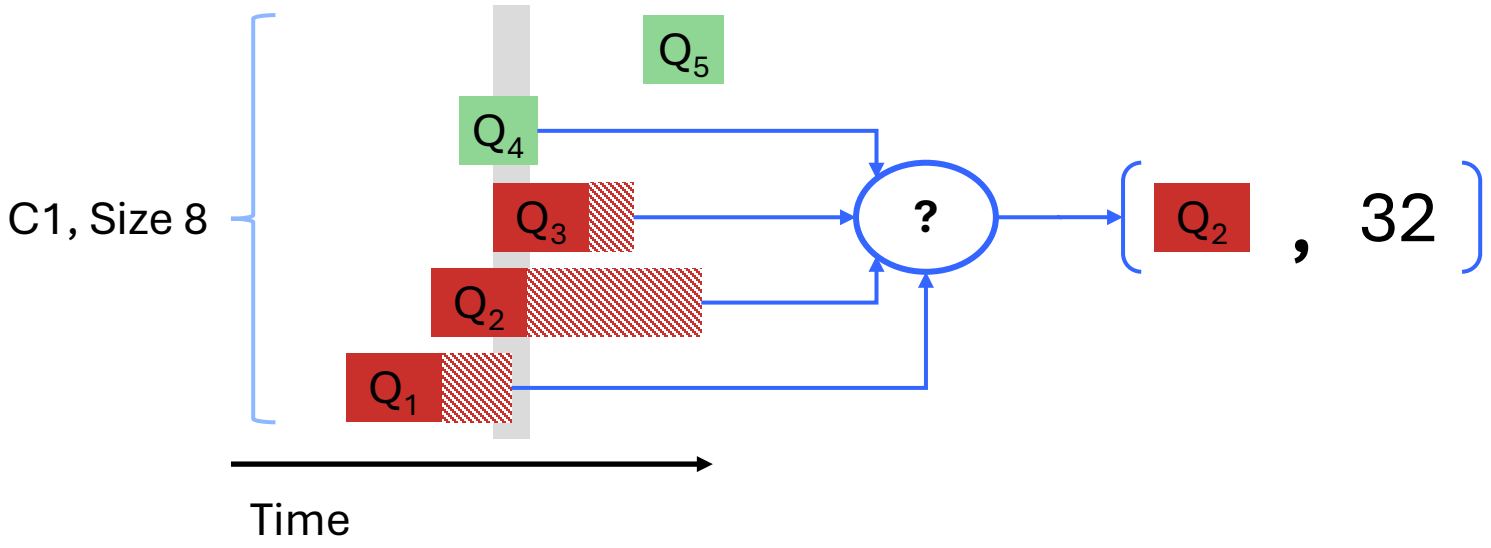
Ongoing work!

We design a practical solution based on the state-of-the-art

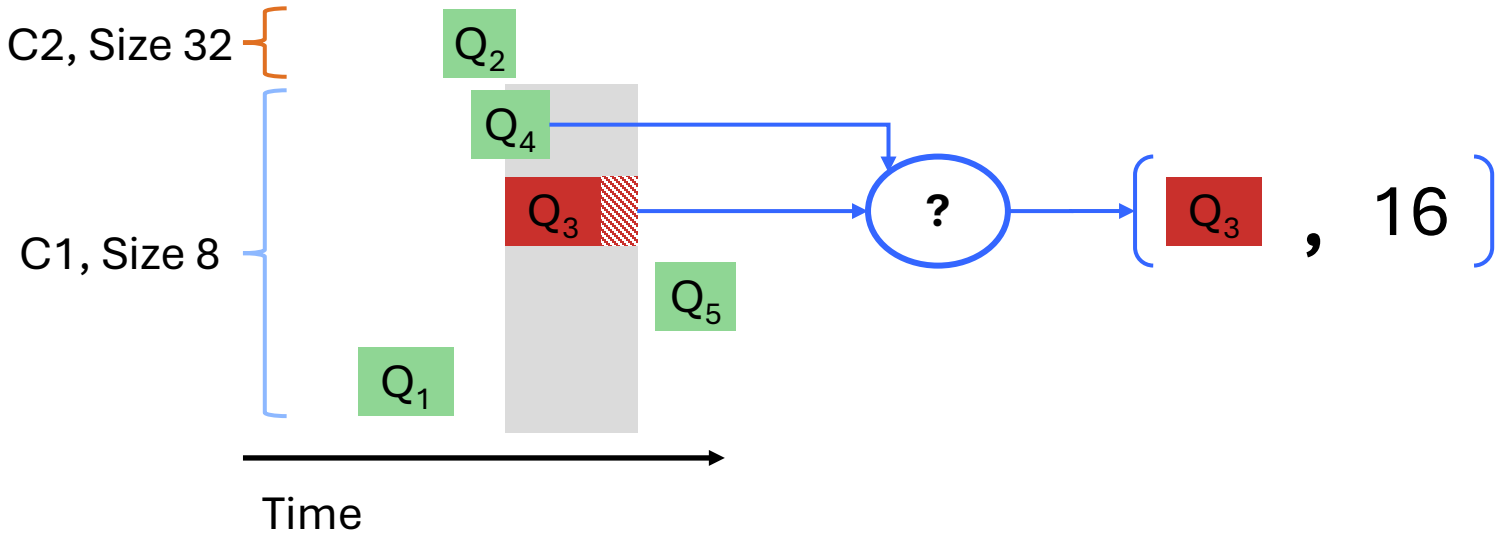
Key Challenge	Base technique	Making it practical
Modeling query interference	Iconq [1]	Consider cluster size, deal with execution-dependent overlaps
Selecting clusters to provision	Two-phase greedy packing	Add query interactions and cloud billing peculiarities
Dealing with online uncertainty	Decision tree model	Distill context-dependent decisions of expensive model

[1] Ziniu Wu, Markos Markakis, Chunwei Liu, Peter Baile Chen, Balakrishnan Narayanaswamy, Tim Kraska, and Samuel Madden. 2025. Improving DBMS Scheduling Decisions with Accurate Performance Prediction on Concurrent Queries. Proc. VLDB Endow. 18, 11 (July 2025), 4185–4198. https://doi.org/10.14778/3749646.3749686

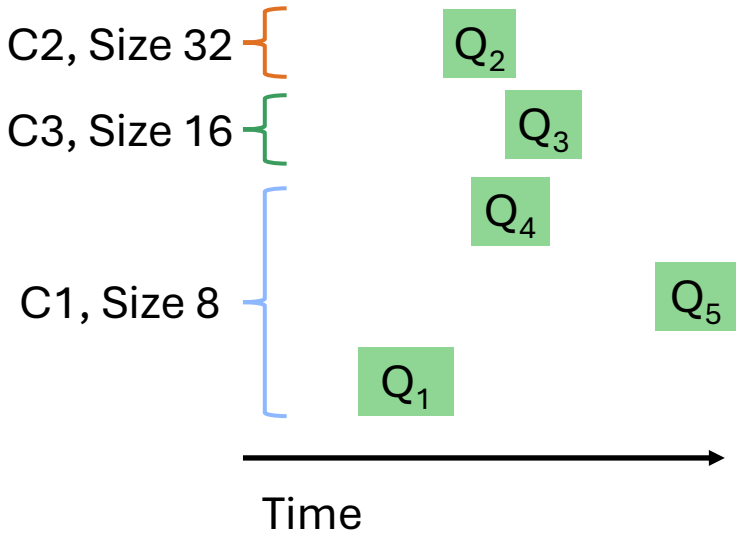
First objective: Meet the SLO on forecast



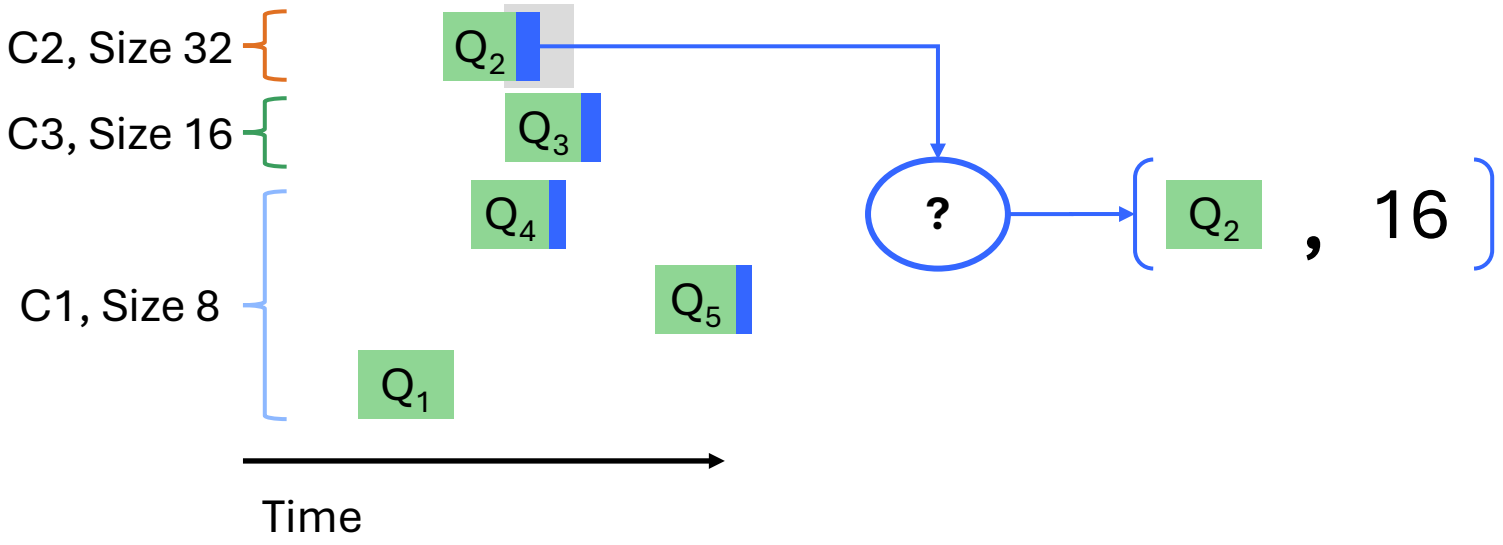
First objective: Meet the SLO on forecast



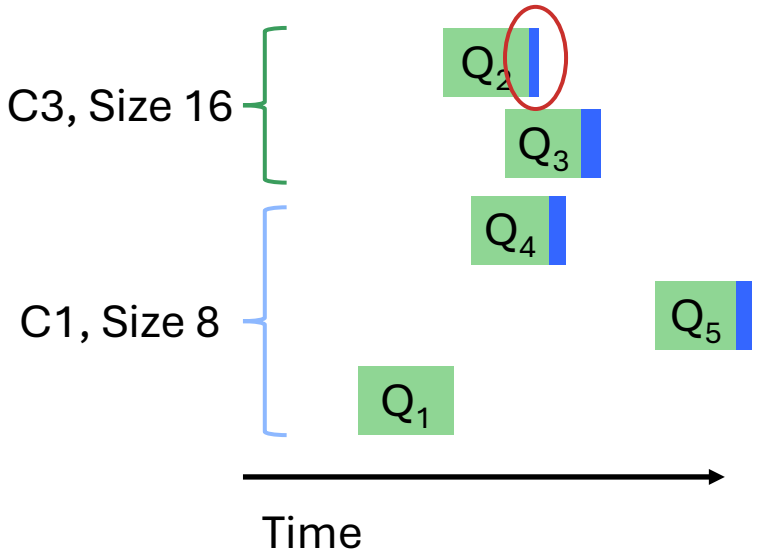
First objective: Meet the SLO on forecast



Second objective: reduce cost



Second objective: reduce cost

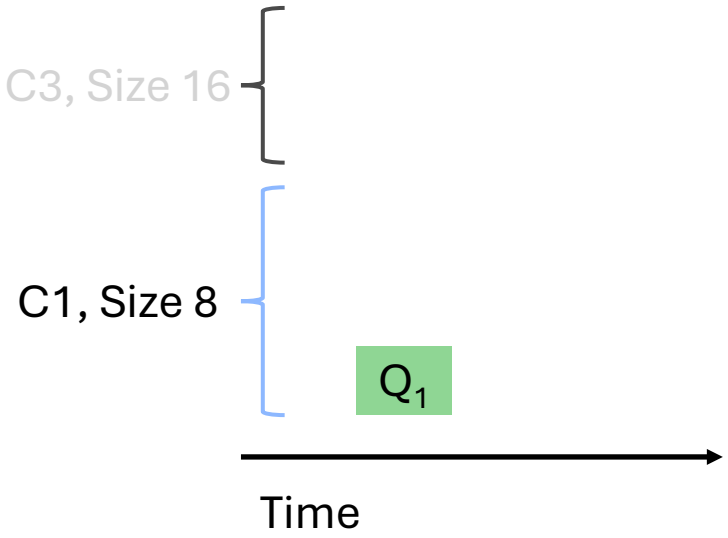


Not incurring cost yet

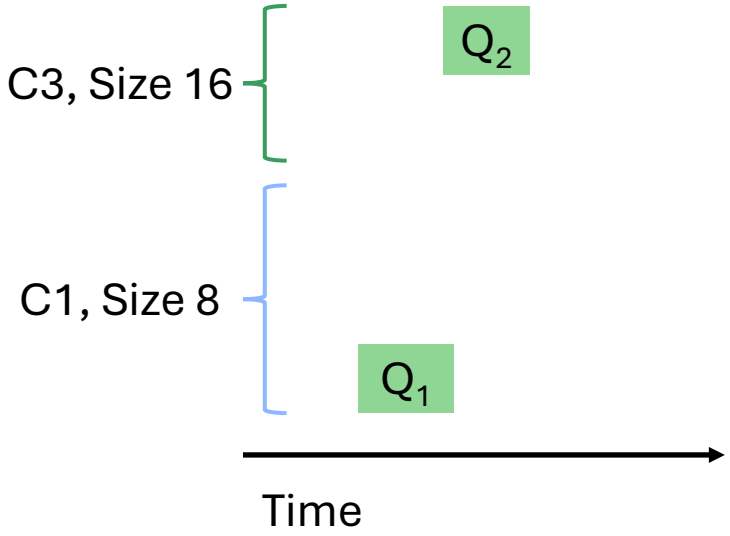
- Pre-load C1, C3
- Train XGBoost router

Could also use the latency model online, but slower

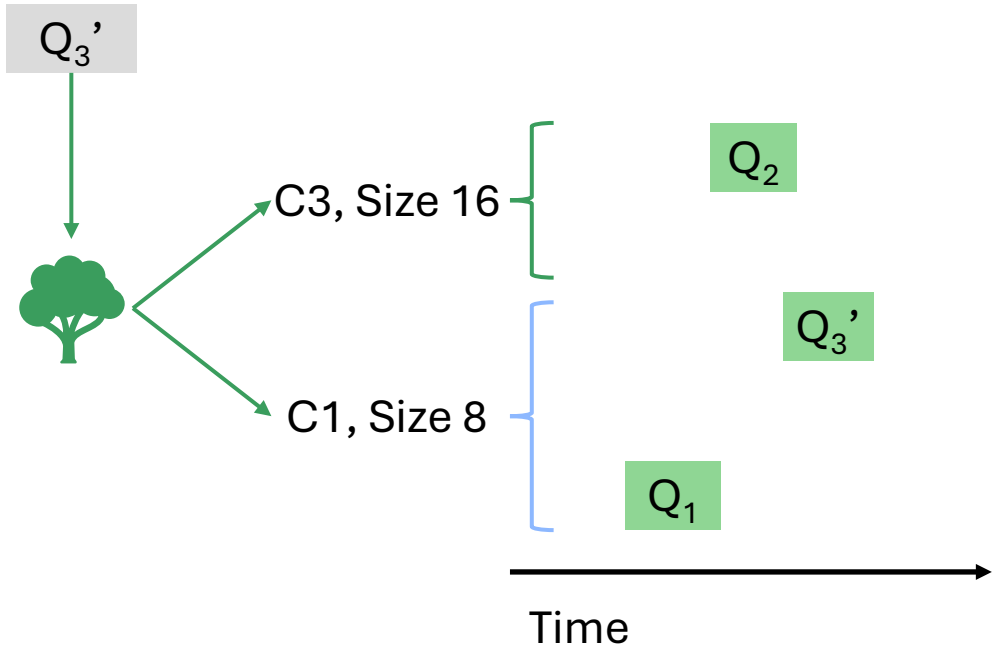
Our online phase makes decisions adaptively



Our online phase makes decisions adaptively



Our online phase makes decisions adaptively



The Problem

Our Design

Evaluation

We base our evaluation on TPC-DS *chunks*

- Benchmark TPC-DS templates in isolation on Redshift serverless clusters with 4, 8, ..., 256 RPU
- Identify **15 heavy templates:** latency >2s on every size.

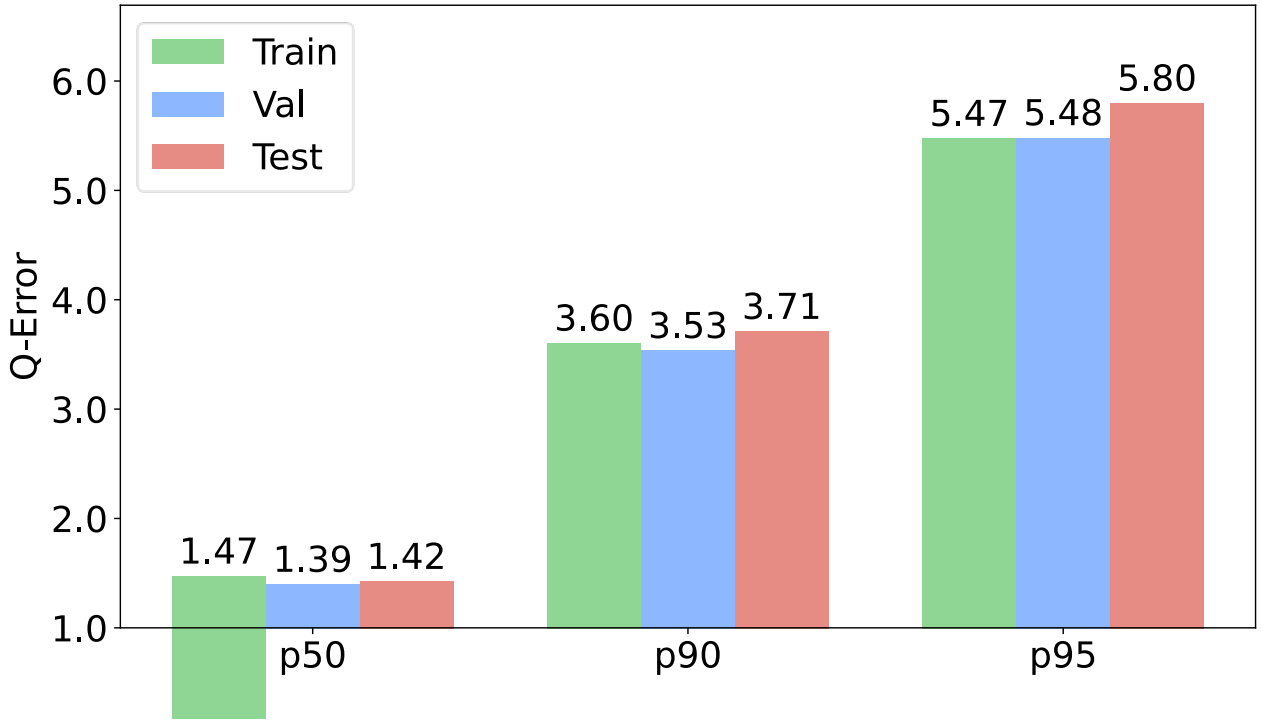
- Define 16 one-hour workload chunks:
  - Heavy queries: 0%, 10%, 25%, 50%
  - Mean interarrival time: 10s, 30s, 60s, 120s

- Run each chunk on 4, 8, 16, 32 RPU -> 64 traces
  - Result cache disabled.
  - Hold out 12 for validation and 12 for testing.

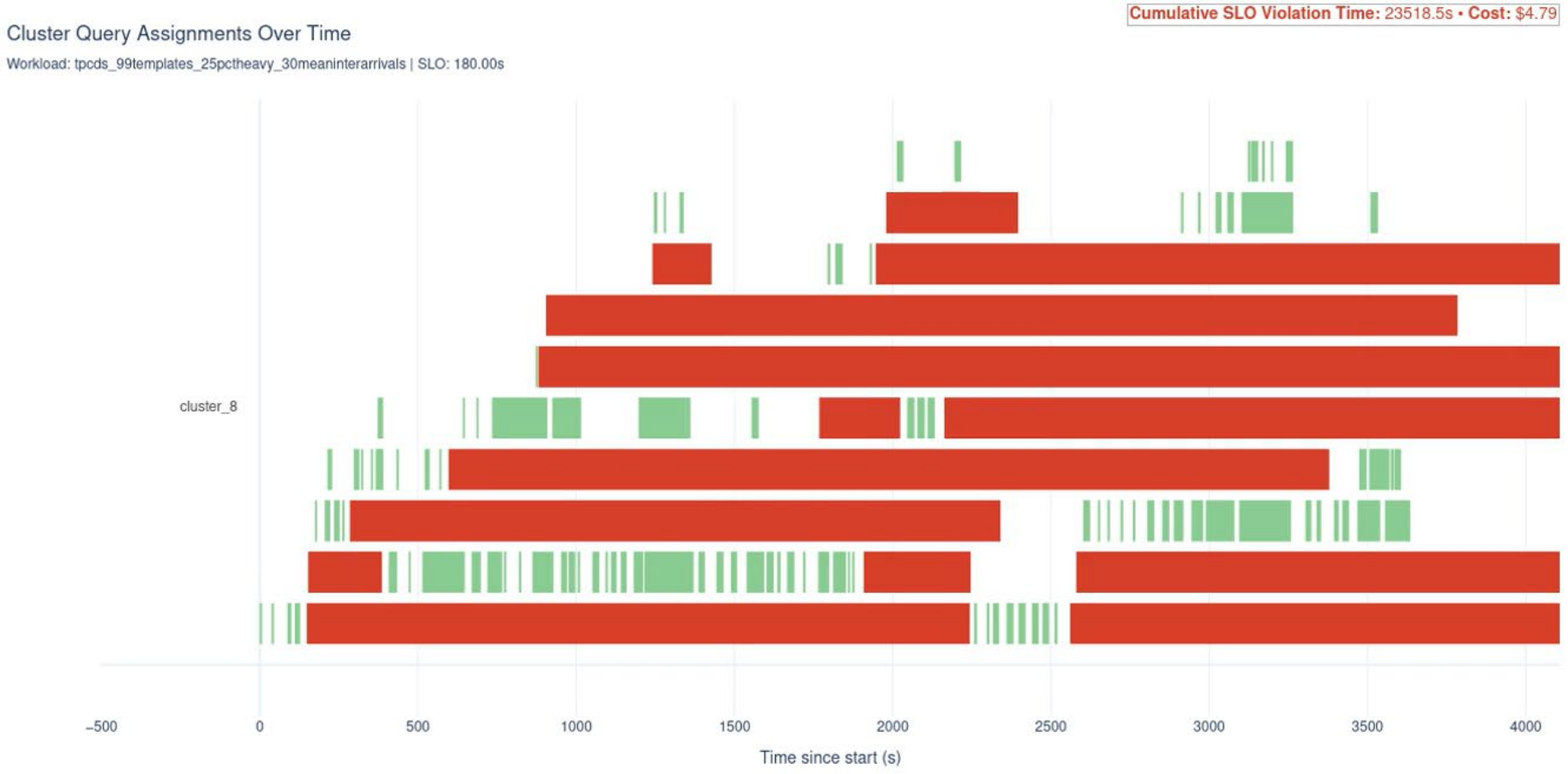
Our predictor generalizes well across chunks

We benchmark at percentiles of the Q-error:

$$max(\frac{y_{true}}{y_{pred}}, \frac{y_{pred}}{y_{true}})$$

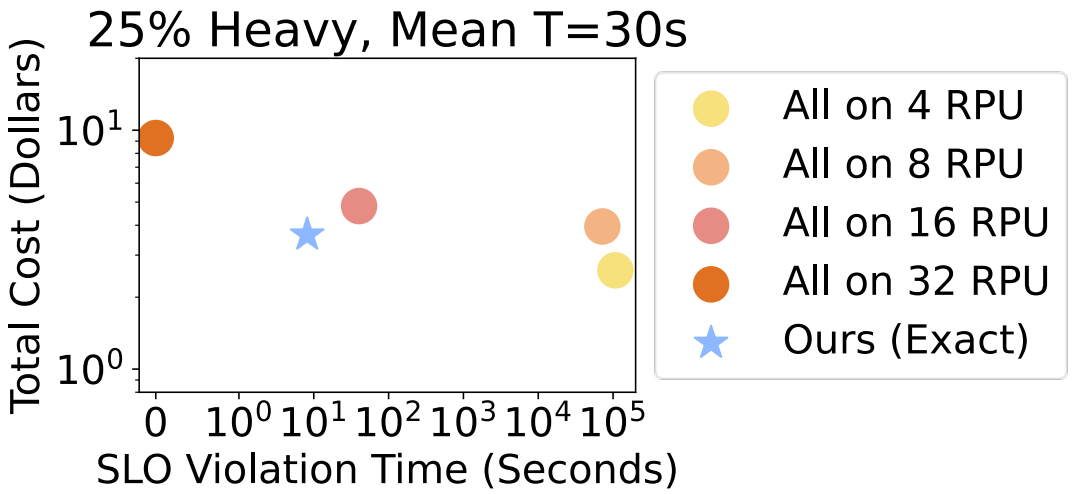


Case study on one of the chunks



Our routing plans meet the SLO cheaply

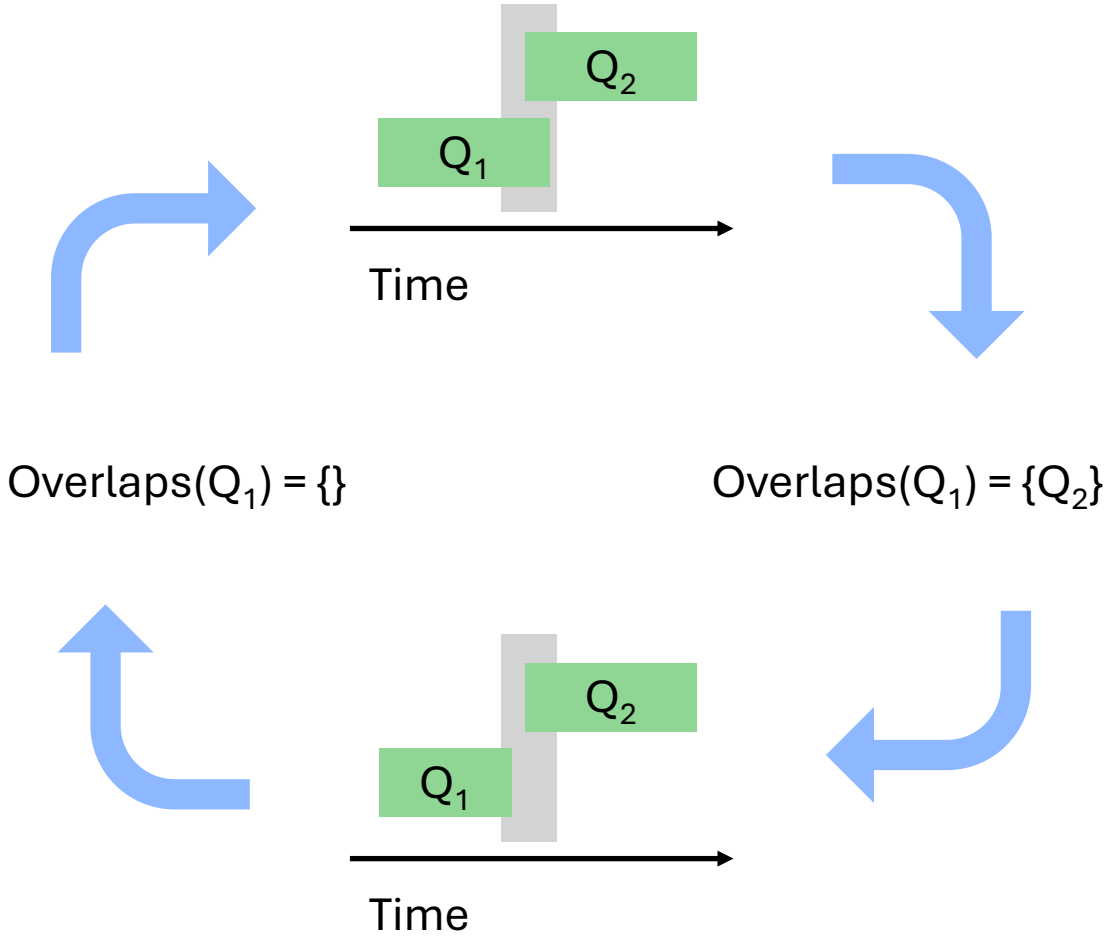
- Use 180s SLO for 119 queries.
- Use the selector to determine a cluster set and routing model.
- Re-execute and measure SLO violation rate and cost.
- Lower is better on both metrics.



	Cost vs All 32	SLO Violation vs All 16
Ours (Exact)	**2.54x lower**	**4.94x lower**

An interesting failure mode can emerge

- In Iconq, predictions rely on the **overlapping query set**
  - Online, easy to determine: what is still running?
  - Offline, we can enter an error feedback loop if not careful.
- Quick fix: later can never speed earlier up
- Better: model “nearness” without overlap?



We are exploring three future directions

Better latency model

Can we learn influence of “nearby” queries while avoiding conditioning on execution overlap?

Purely online setting

By modeling startup time, can we handle cluster additions and deletions purely online?

Probabilistic outputs

Instead of the exact latency, can we do better by learning the probability of it exceeding the SLO?

Conclusions

- Separate compute clusters can be used to avoid SLO violations.
- Cost- and interference-aware routing is a promising path.
- Current models of query interference show interesting problems.

Questions? Ideas? Reach out at markakis@mit.edu