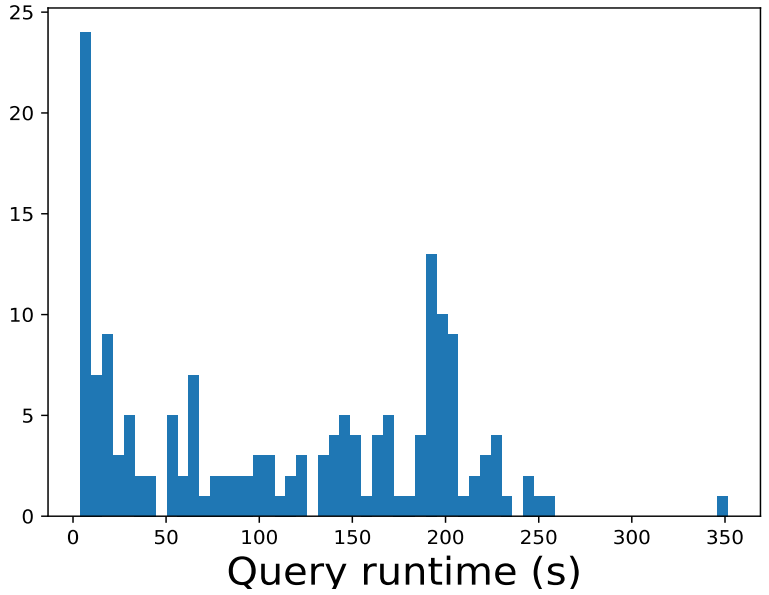
Improving DBMS Scheduling Decisions with Accurate Performance Prediction on Concurrent Queries

Ziniu Wu, **Markos Markakis**, Chunwei Liu, Peter Baile Chen, Balakrishnan Narayanaswamy, Tim Kraska, Samuel Madden

Concurrency impacts latency heavily

- Deciding when to execute a query can be crucial
- Bad scheduling decisions can have a significant impact on query latency.
- Microexperiment:
  - 4 clients issuing closed-loop queries against IMDB dataset on Postgres.
  - Look at runtime of the same query under different concurrent settings.
- Scheduling can have a large impact.



Can we build a reusable scheduler?

- We explore the design of a non-intrusive scheduler:
  - Coarser granularity (e.g. cannot schedule individual operators).
  - But can be applied to any underlying engine.

- Goal: improve query end-to-end time (queueing time + system runtime)

We need to capture resource contention

Q_0 (big join)

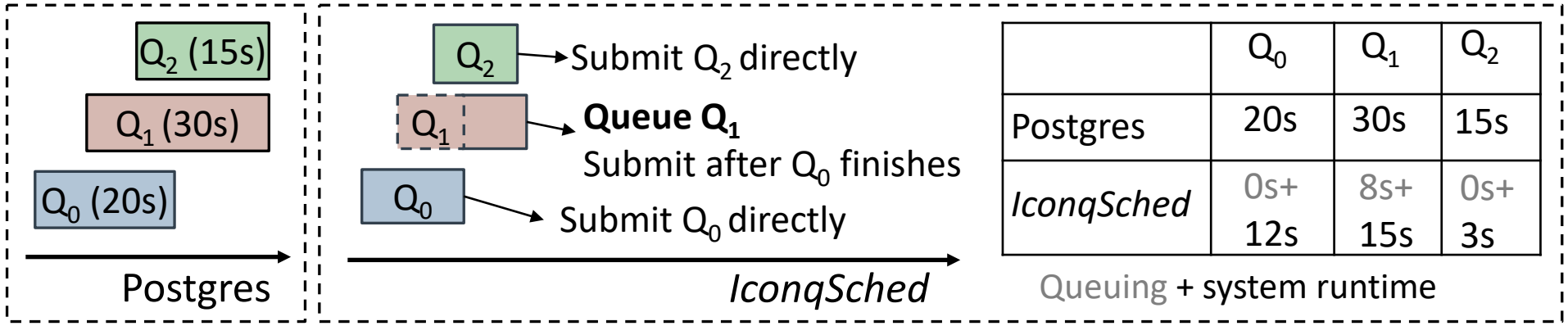
CPU: low
Memory: **high**
IO: **high**
Tables: T_1, T_2, T_3, T_4

Q_1 (big join)

CPU: low
Memory: **high**
IO: **high**
Tables: T_6, T_7

Q_2 (complex predicate)

CPU: **high**
Memory: low
IO: low
Tables: T_2, T_3, T_5



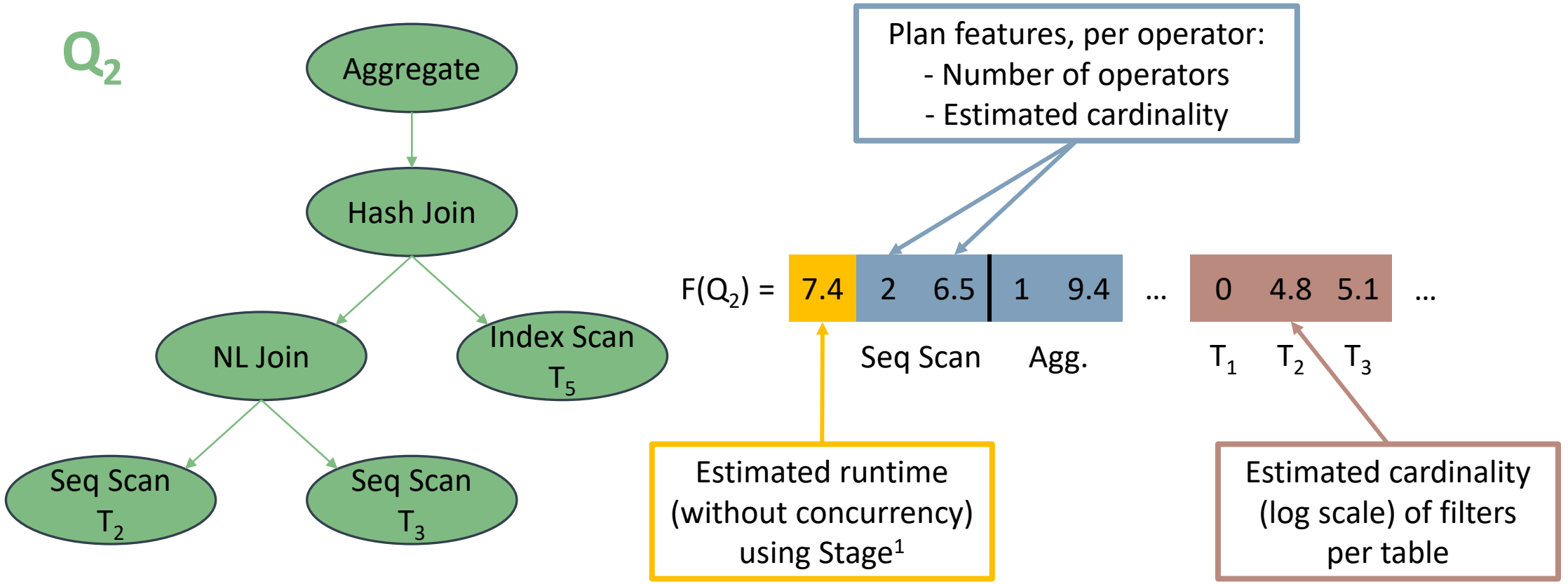
	Q_0	Q_1	Q_2
Postgres	20s	30s	15s
IconqSched	0s+ 12s	8s+ 15s	0s+ 3s

Queuing + system runtime

Key challenge: splitting past and future

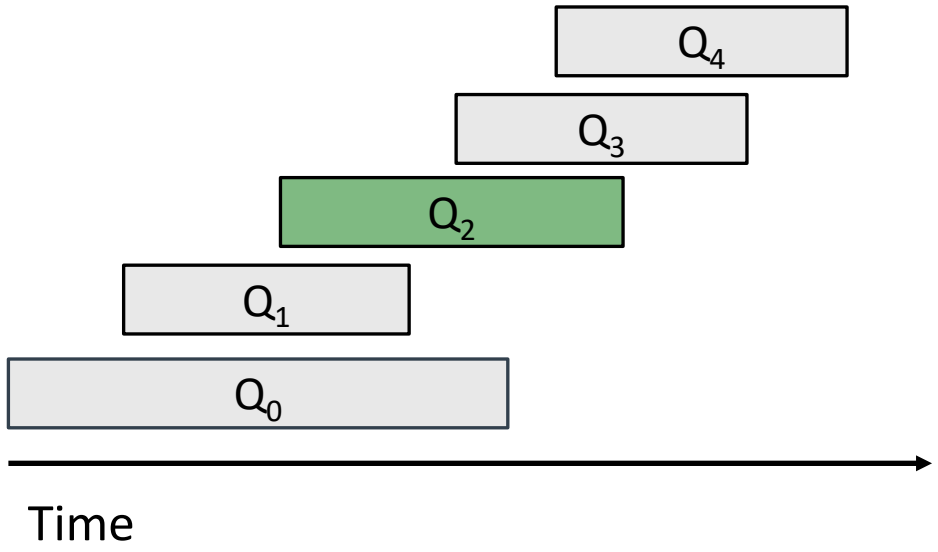
- The latency of each query is affected by both:
  - Queries that were submitted before it and are still running.
  - Queries that are submitted while it is still executing.
- The latter are not available at scheduling time.
- How can we update estimates for running queries as new ones arrive?
- Solution preview: Bidirectional LSTM

We first featurize a query using its plan

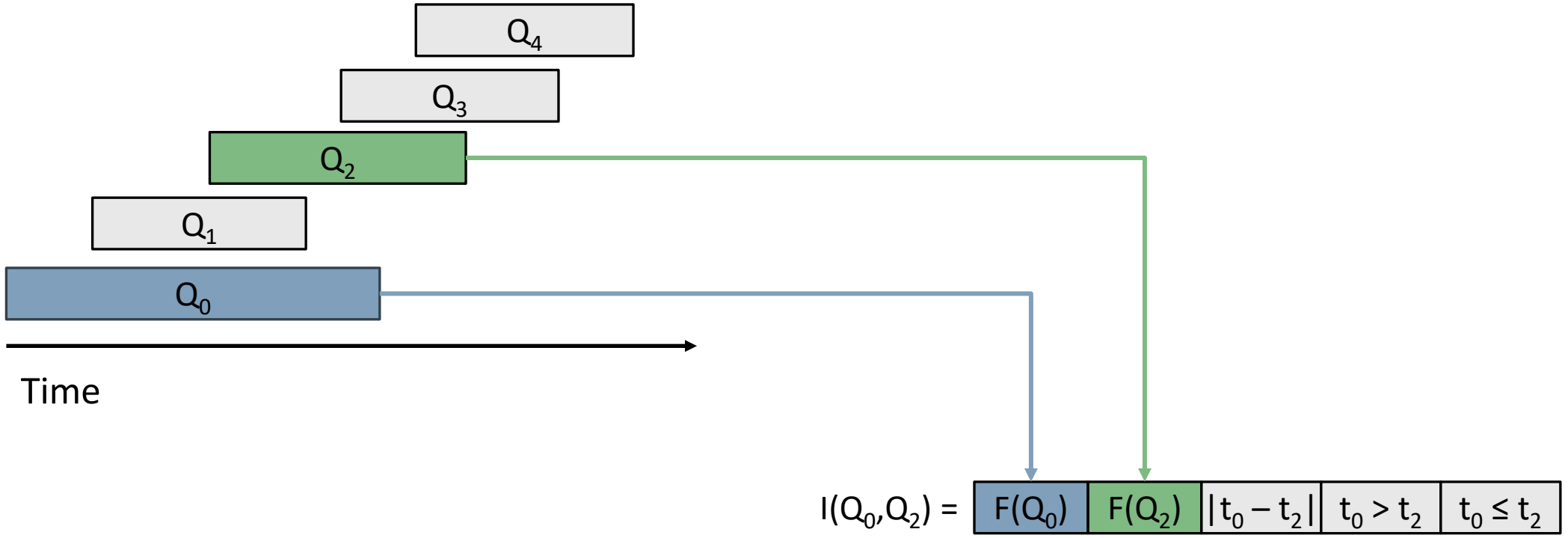


[1] Ziniu Wu, Ryan Marcus, Zhengchun Liu, Parimarjan Negi, Vikram Nathan, Pascal Pfeil, Gaurav Saxena, Mohammad Rahman, Balakrishnan Narayanaswamy, and Tim Kraska. 2024. Stage: Query Execution Time Prediction in Amazon Redshift.

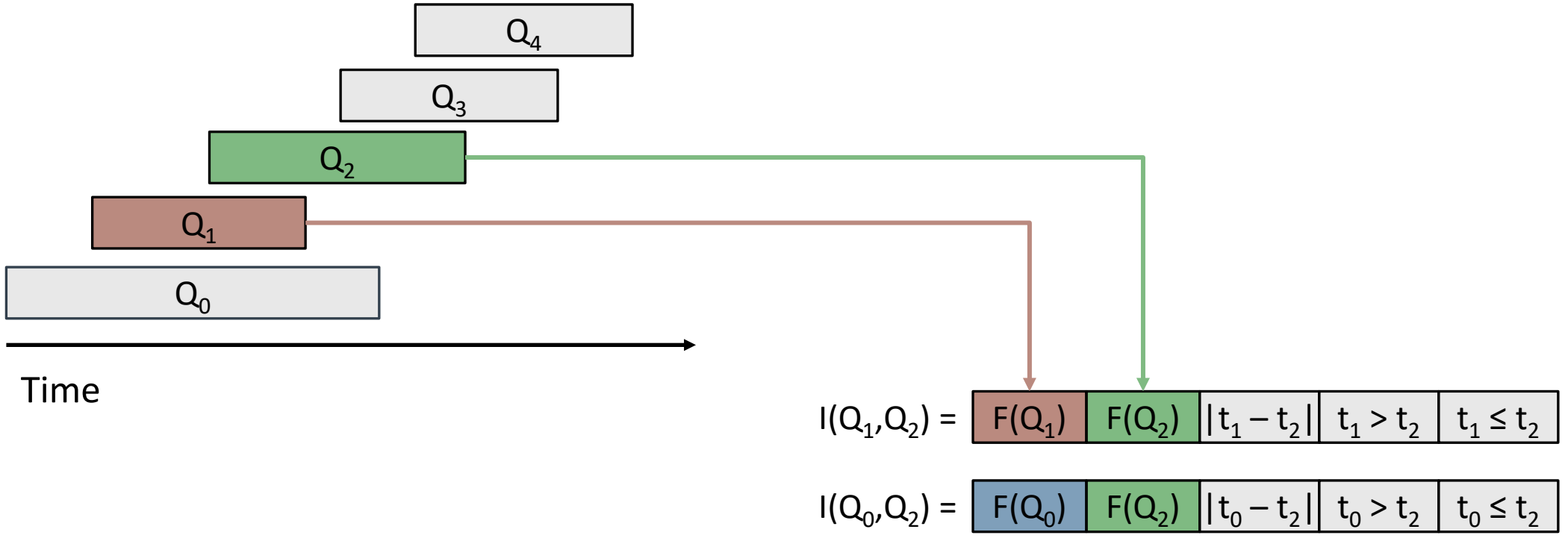
We next add query interaction details



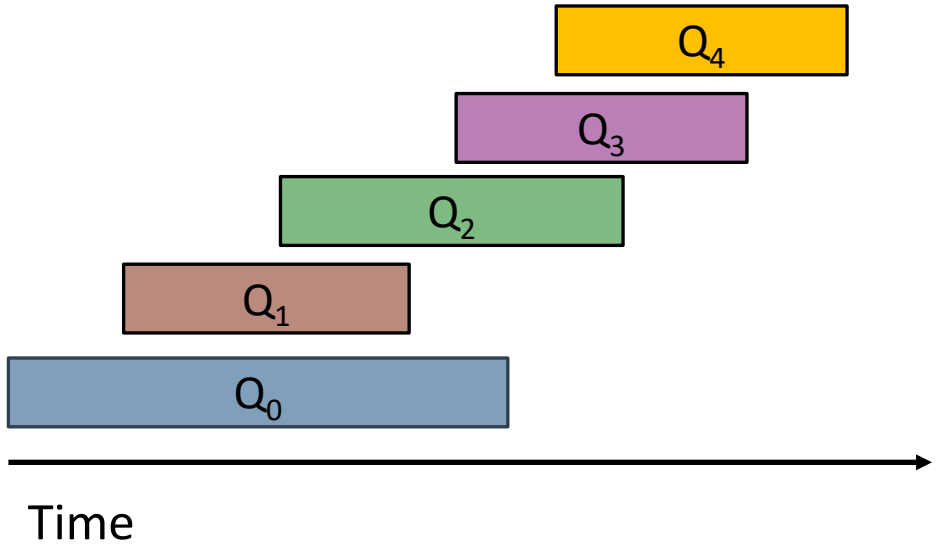
We next add query interaction details



We next add query interaction details

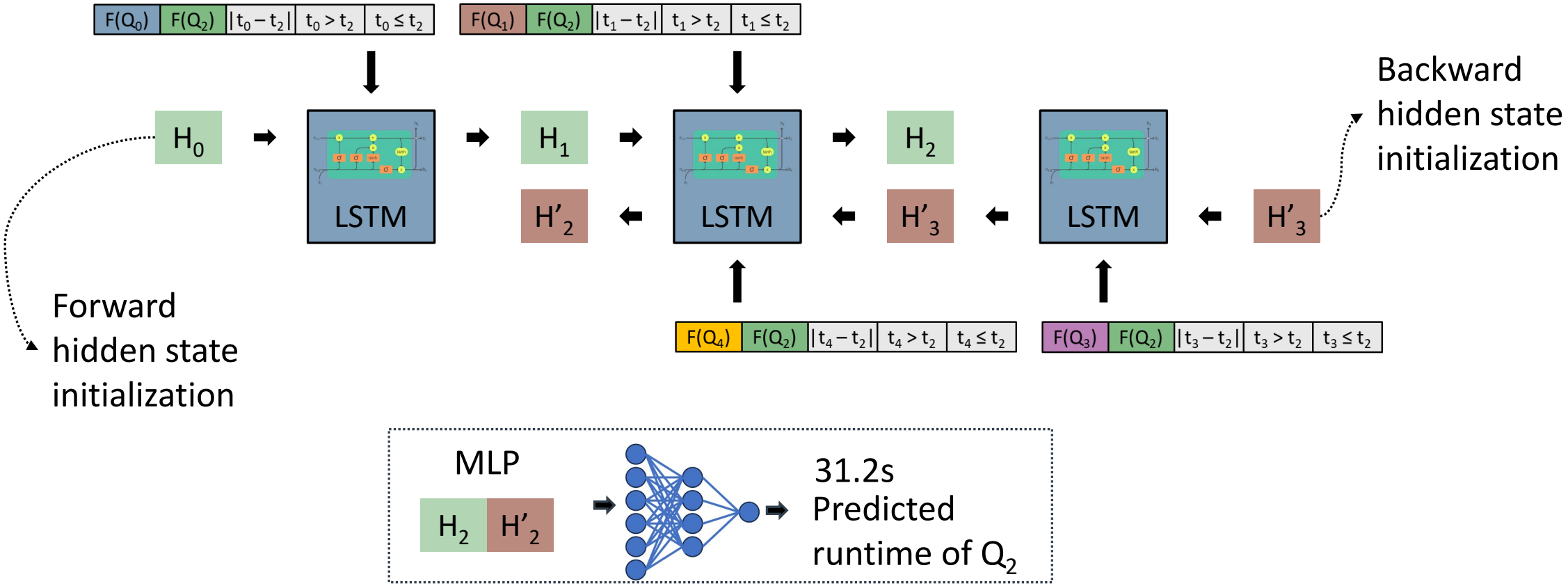


We next add query interaction details

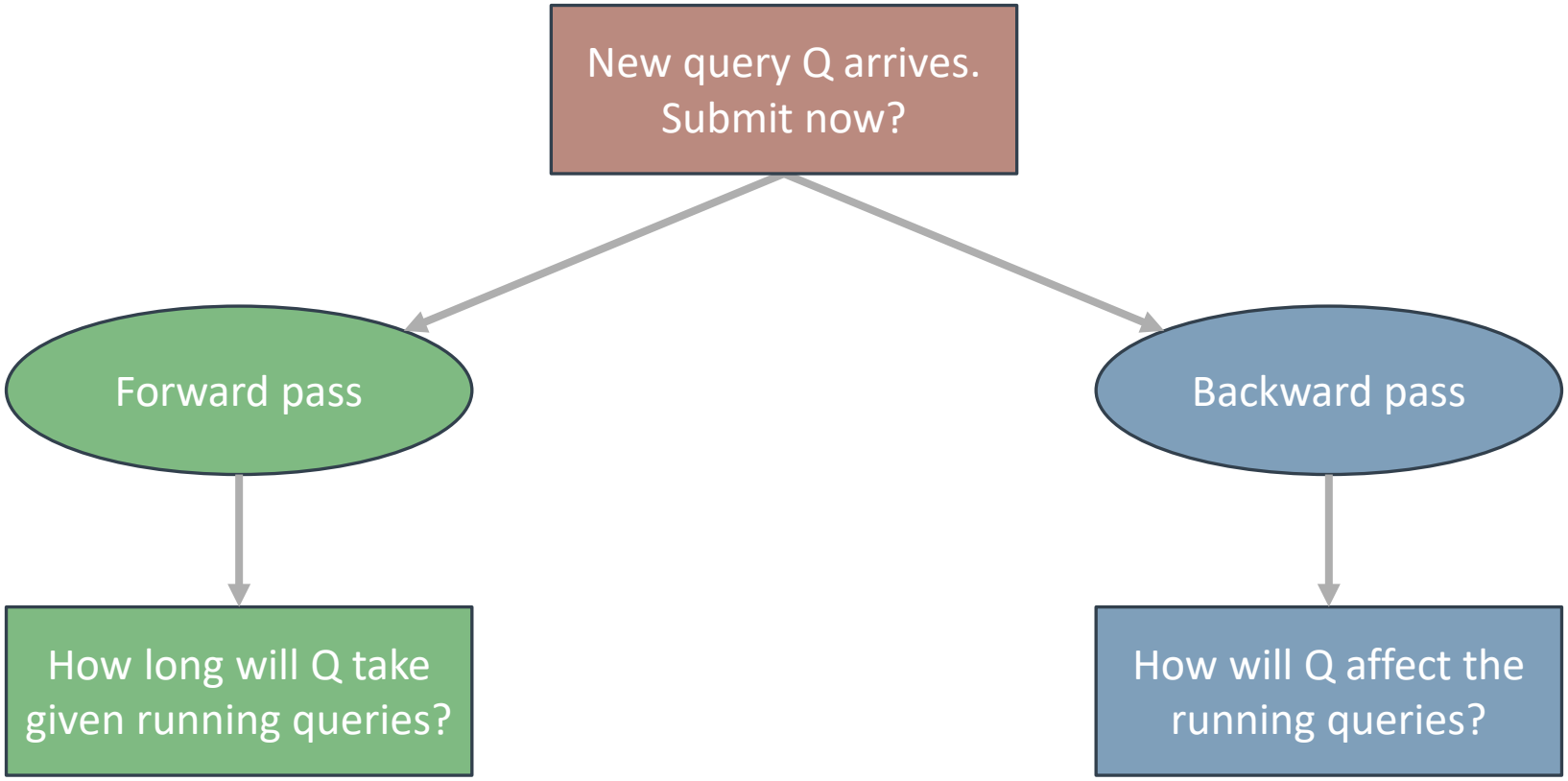


$I(Q_1,Q_2) =$	$F(Q_4)$	$F(Q_2)$	$\lvert t_4 - t_2 \rvert$	$t_4 > t_2$	$t_4 \leq t_2$
$I(Q_3,Q_2) =$	$F(Q_3)$	$F(Q_2)$	$\lvert t_3 - t_2 \rvert$	$t_3 > t_2$	$t_3 \leq t_2$
$I(Q_1,Q_2) =$	$F(Q_1)$	$F(Q_2)$	$\lvert t_1 - t_2 \rvert$	$t_1 > t_2$	$t_1 \leq t_2$
$I(Q_0,Q_2) =$	$F(Q_0)$	$F(Q_2)$	$\lvert t_0 - t_2 \rvert$	$t_0 > t_2$	$t_0 \leq t_2$

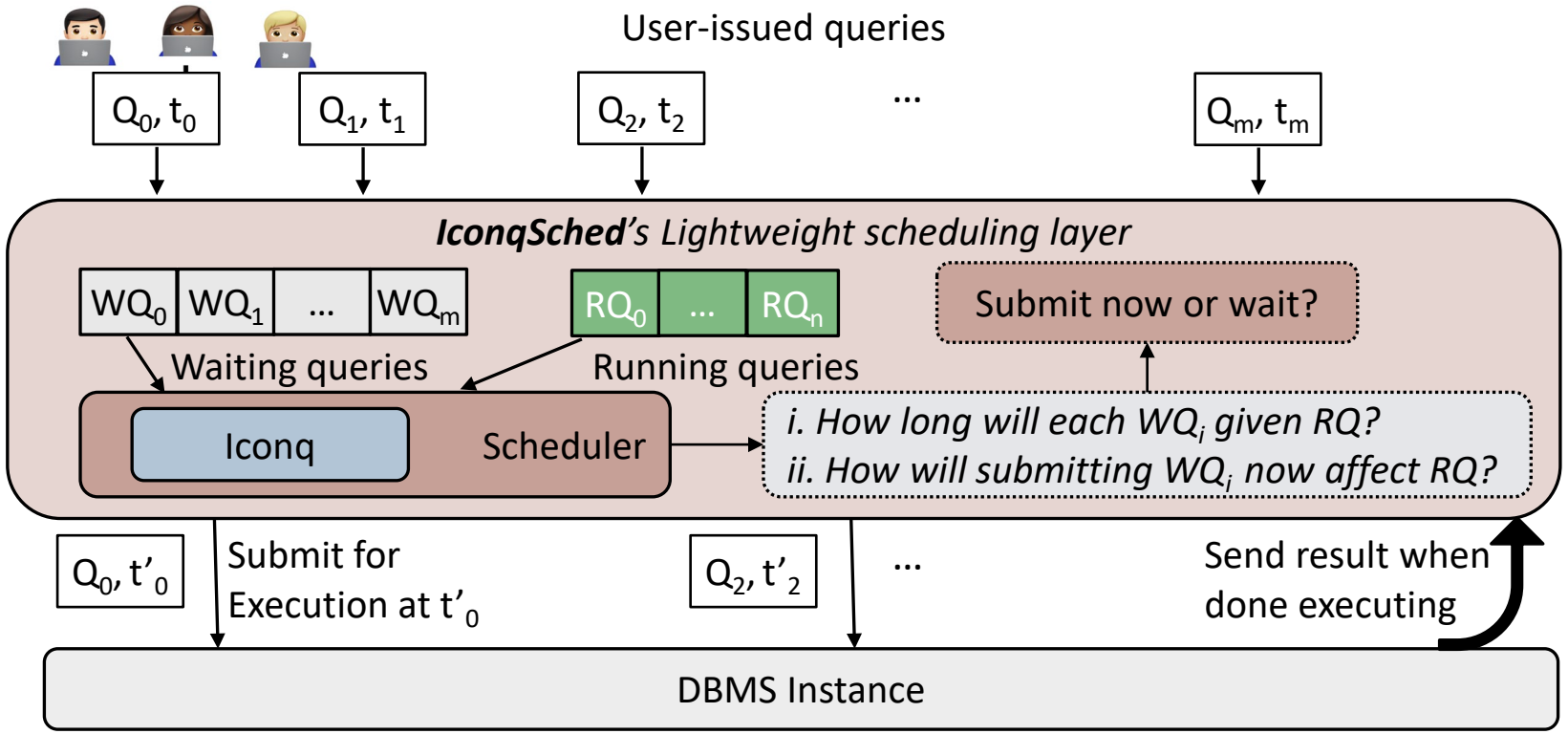
We then train Iconq: a bidirectional LSTM



At inference time, separate the passes



Iconq is at the center of our scheduler



*** For short queries (Stage[1] estimate < 5s), always submit, since Iconq requires around 100ms for inference**

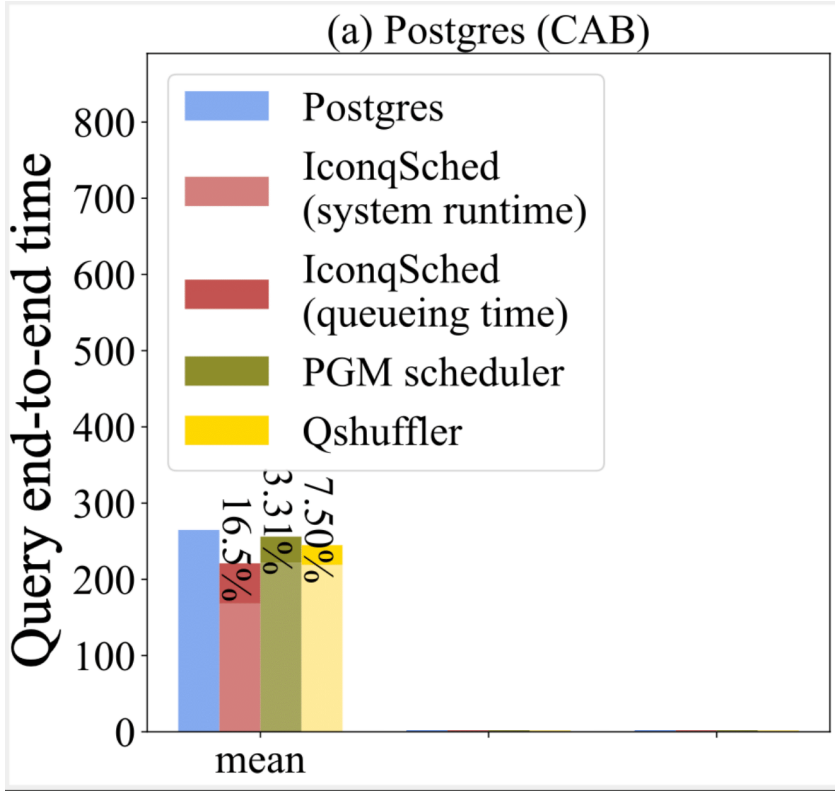
[1] Ziniu Wu, Ryan Marcus, Zhengchun Liu, Parimarjan Negi, Vikram Nathan, Pascal Pfeil, Gaurav Saxena, Mohammad Rahman, Balakrishnan Narayanaswamy, and Tim Kraska. 2024. Stage: Query Execution Time Prediction in Amazon Redshift.

Setting up the evaluation

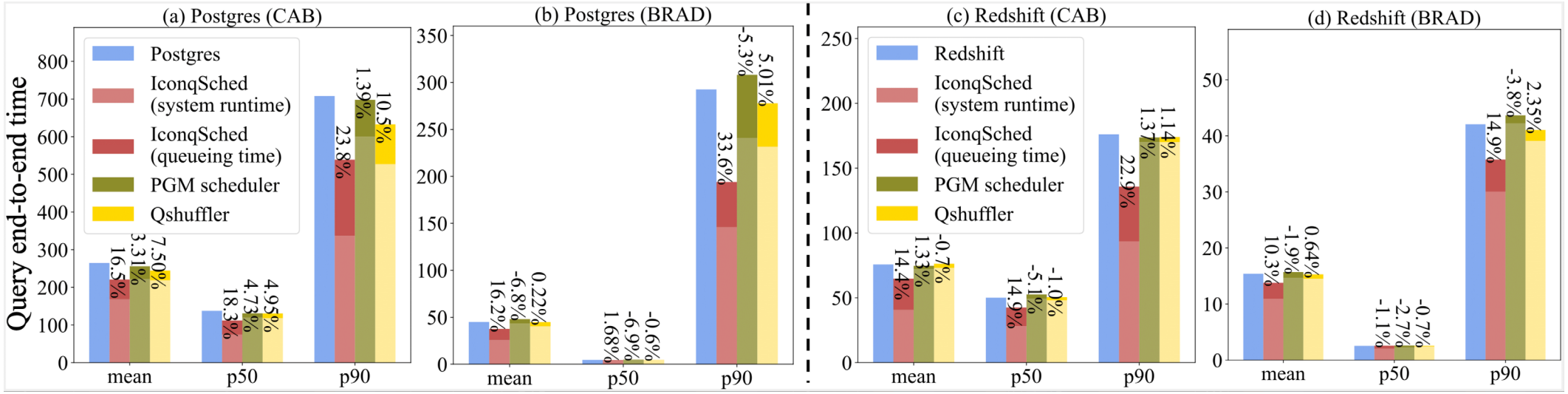
- Two different DBMS engines:
  - PostgreSQL
  - Amazon Redshift
- Two different open-source benchmarks:
  - CAB[1]: Cloud Analytics Benchmark
  - BRAD[2]: IMDB data + Snowset query arrival timestamps
- Two different state-of-the-art baselines:
  - PGM Scheduler[3]: Estimate memory and keep total bounded.
  - Qshuffler[4]: Represent system state as counts per query cluster.

[1] Alexander van Renen and Viktor Leis. 2023. Cloud Analytics Benchmark.
[2] Geoffrey X. Yu, Ziniu Wu, Ferdi Kossmann, Tianyu Li, Markos Markakis, Amadou Ngom, Samuel Madden, and Tim Kraska. 2024. Blueprinting the Cloud: Unifying and Automatically Optimizing Cloud Data Infrastructures with BRAD.
[3] Abhay Mehta, Chetan Gupta, and Umeshwar Dayal. 2008. BI batch manager: a system for managing batch workloads on enterprise data-warehouses.
[4] Mumtaz Ahmad, Ashraf Aboulnaga, Shivnath Babu, and Kamesh Munagala. 2011. Interaction-aware scheduling of report-generation workloads.

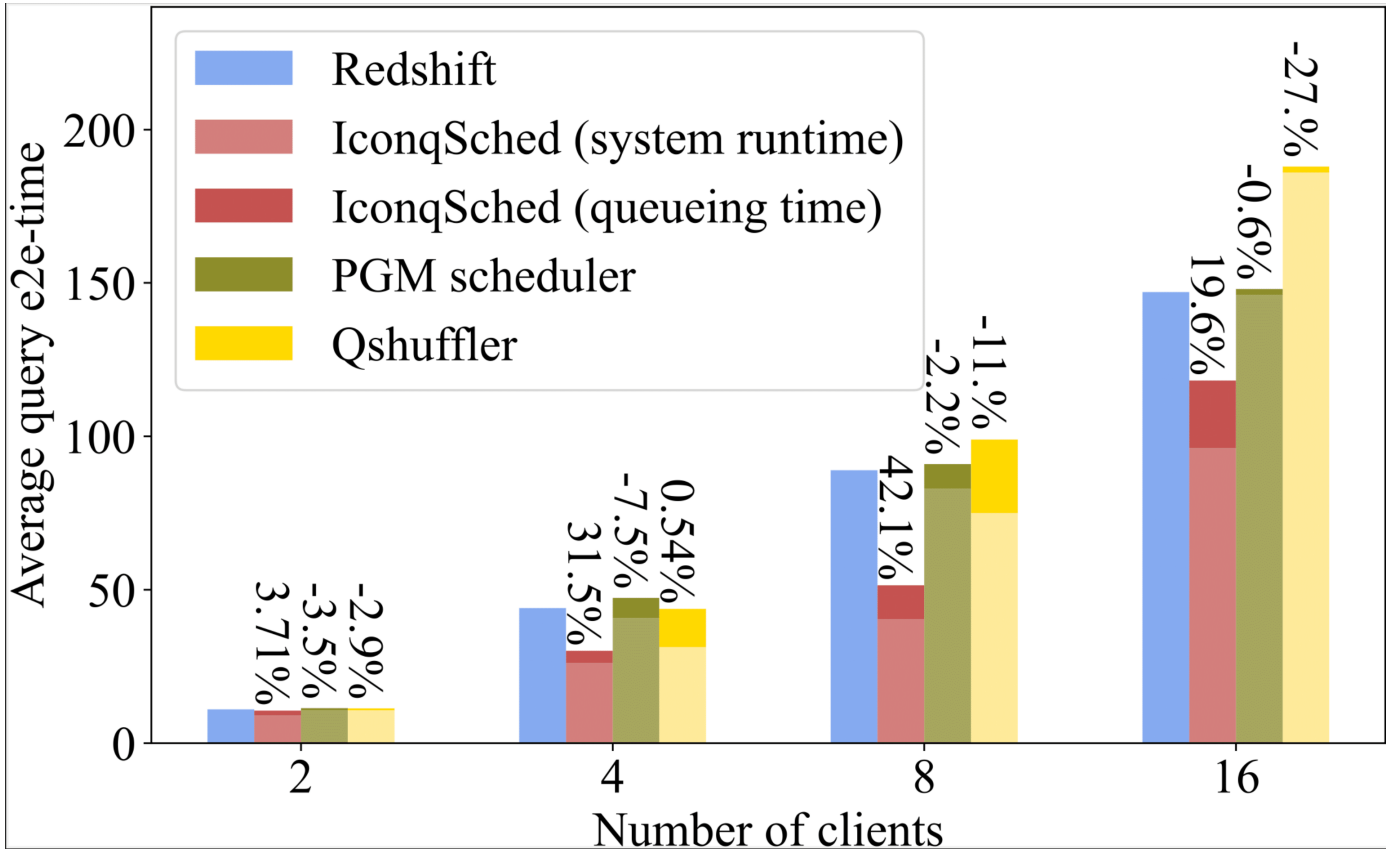
We improve query end-to-end time



We outperform baselines across settings



Iconq performs under high concurrency

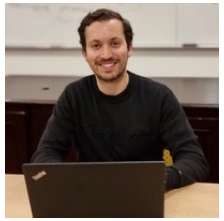
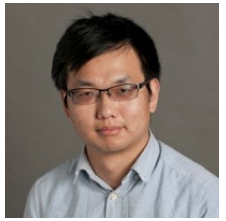


Key Takeaway

IconqSched reduces
query end-to-end time
without accessing DBMS internals

Data Systems Group @ MIT

Thank you!

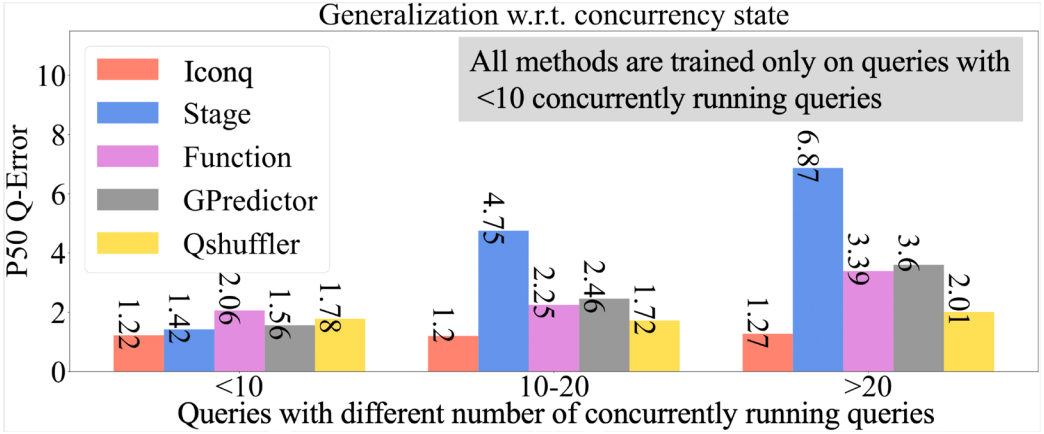




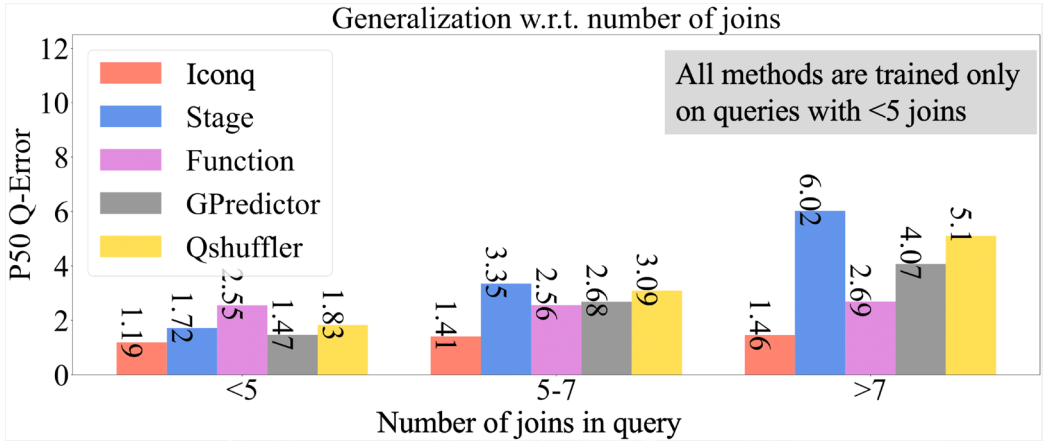


Paper: https://www.vldb.org/pvldb/vol18/p4185-wu.pdf

Questions? Reach out at ziniuw@mit.edu

Iconq generalizes well to harder contexts





$$Q_{\text{error}} = max\left(\frac{true}{pred}, \frac{pred}{true}\right)$$