Next Steps for
Learned Query Optimization

Ryan Marcus
University of Pennsylvania

Slides: `https://rm.cab/nedb26`

Penn Engineering
UNIVERSITY of PENNSYLVANIA

This Talk

- Why **learn** a query optimizer?
- Lessons learned **deploying learned QOs**
- **LimeQO**, an offline learned query optimizer
- The end of RL for QO?

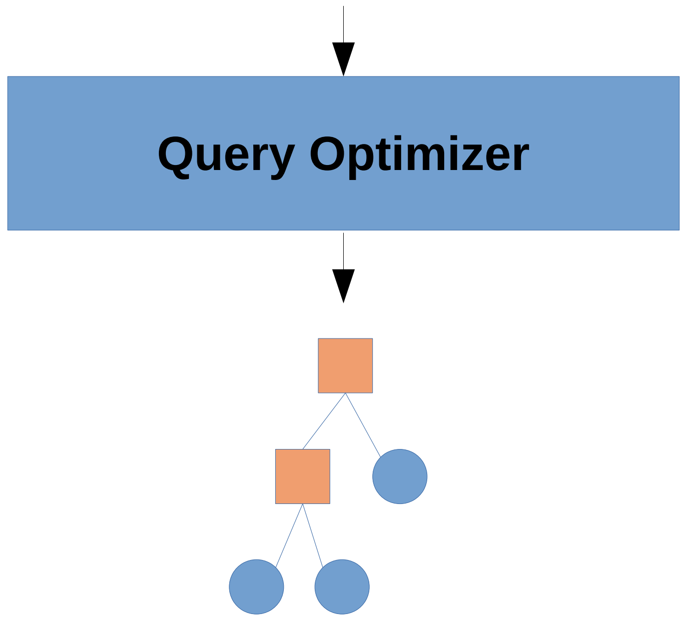
This Talk

- Why **learn** a query optimizer?
- Lessons learned **deploying learned QOs**
- **LimeQO**, an offline learned query optimizer
- The end of RL for QO?

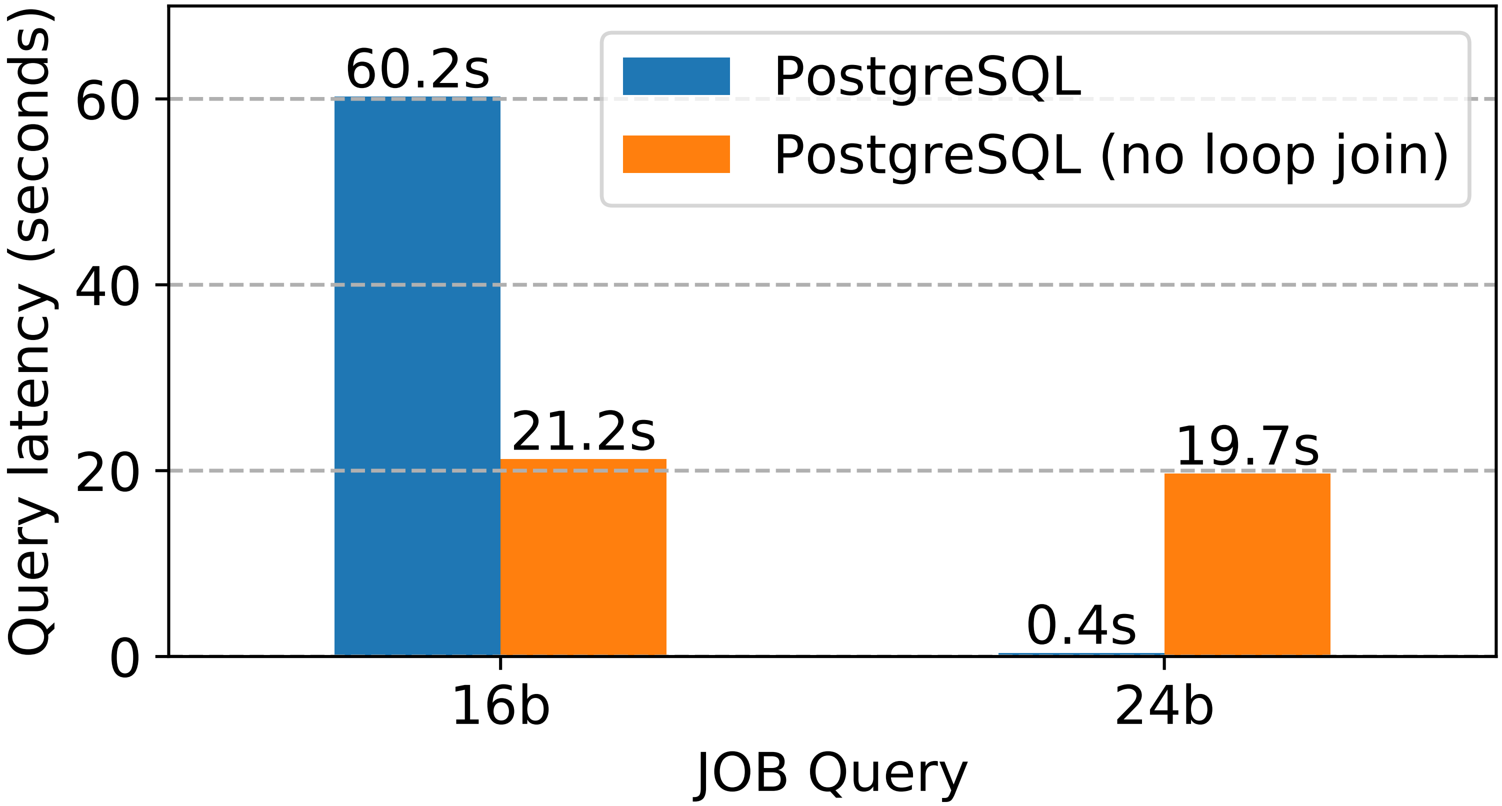
QO is a huge effort

- Transform SQL into a query plan
  - 42K LOC in PG12
  - 1M+ SQL Server
  - 45-55 FTEs, Oracle (~ $5mil/year)

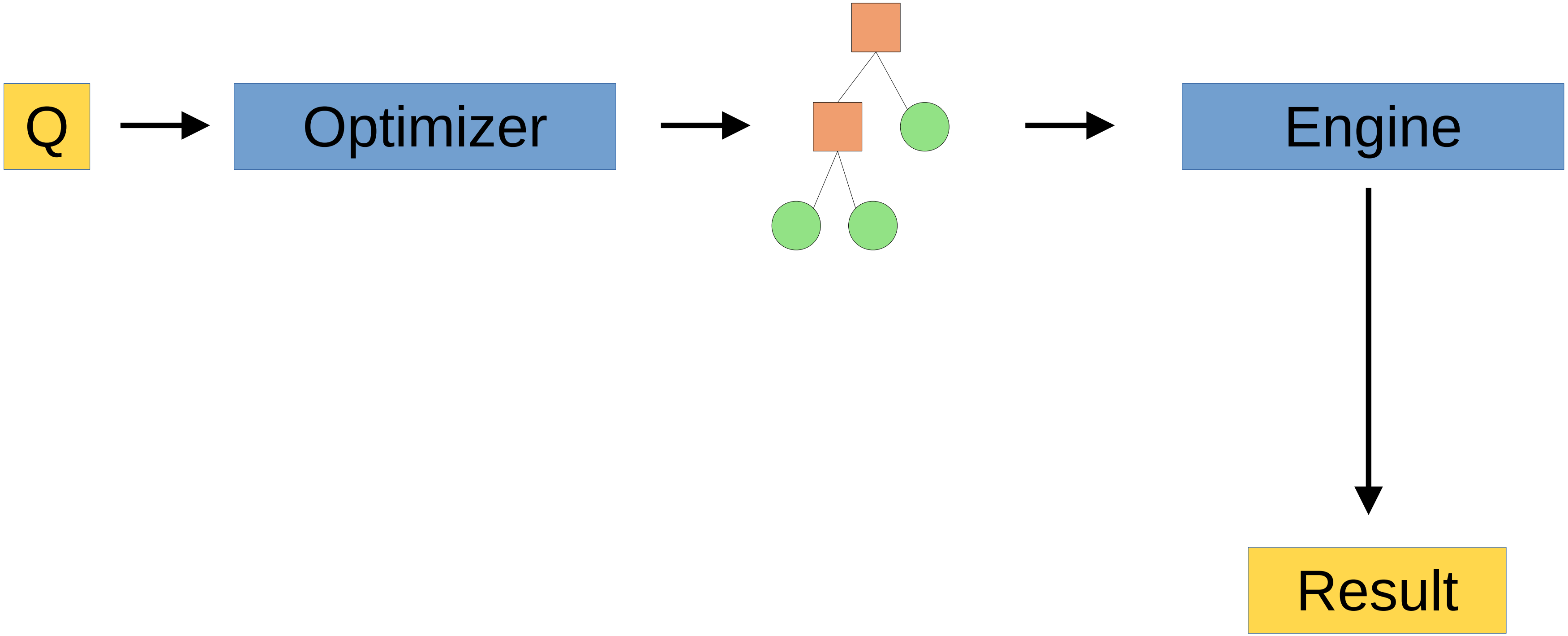
```
SELECT *
FROM t1, t2 WHERE…
```



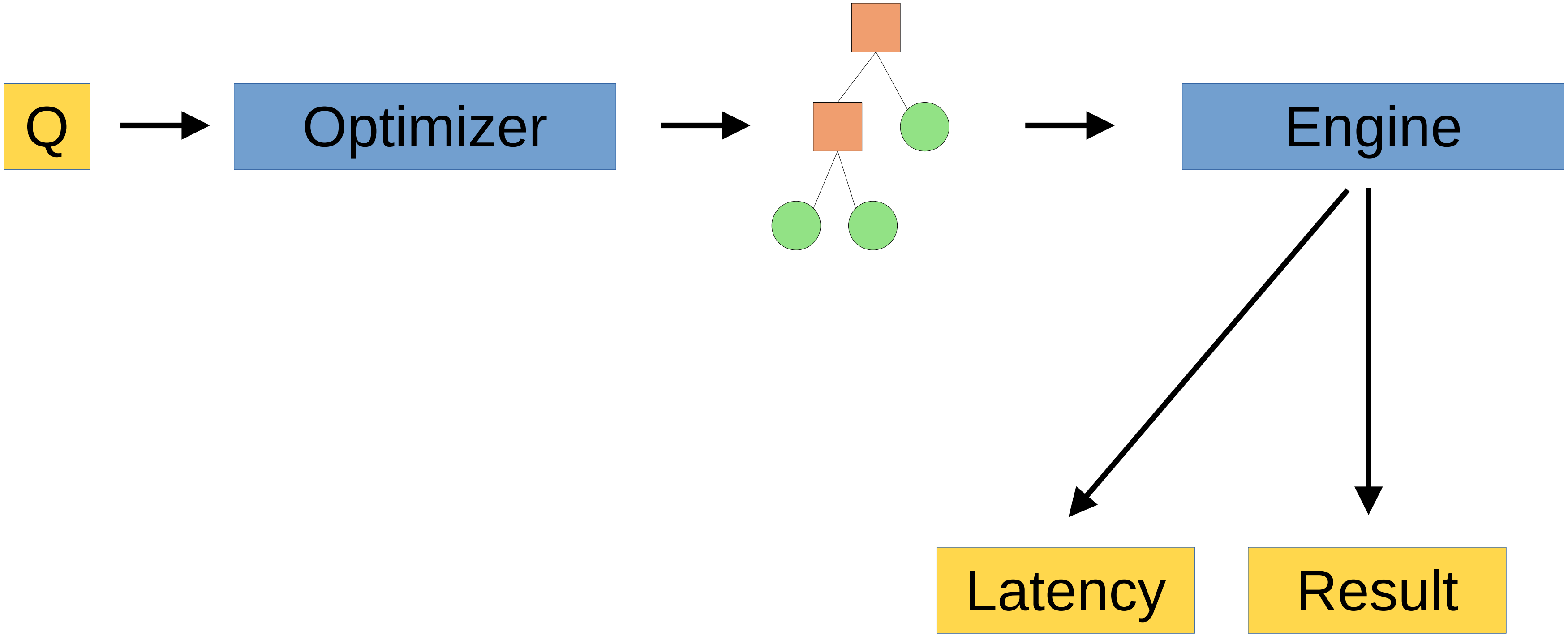
QOs aren't that good



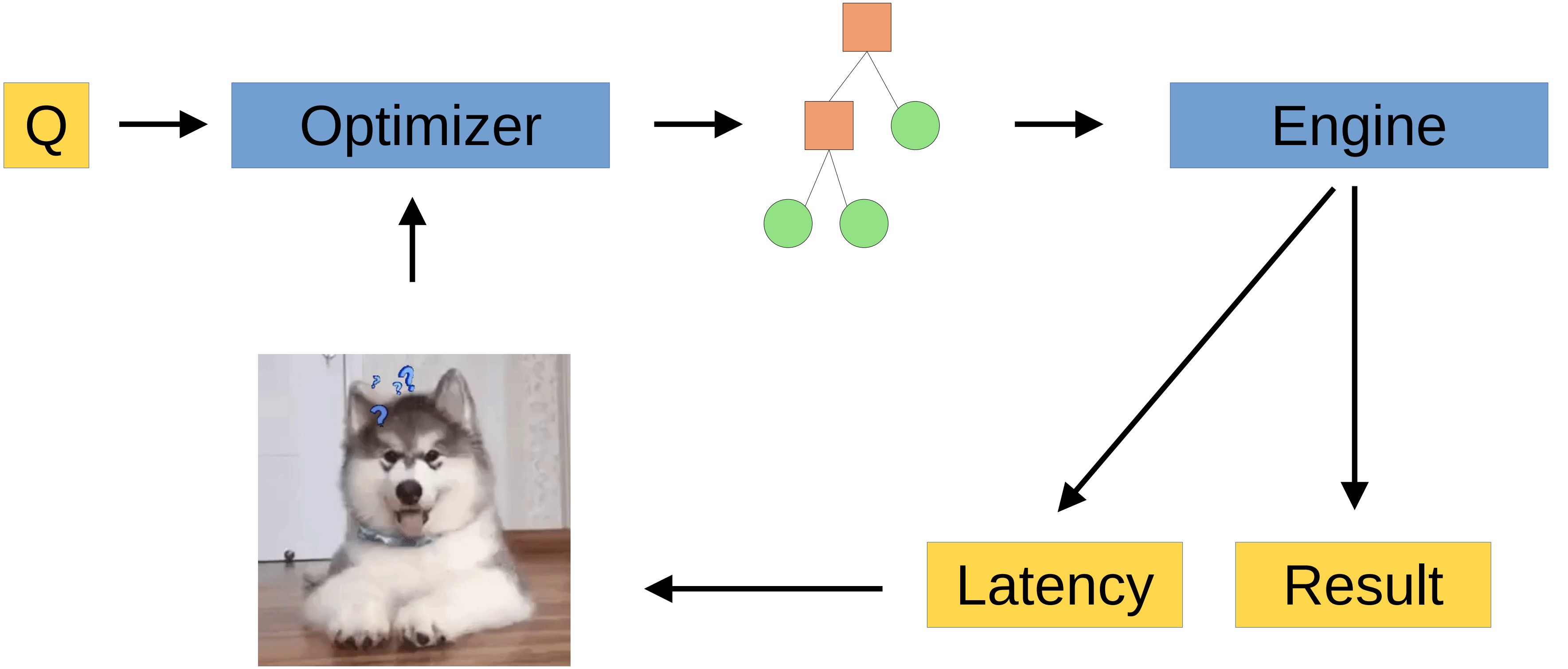
QOs are leaving info on the table



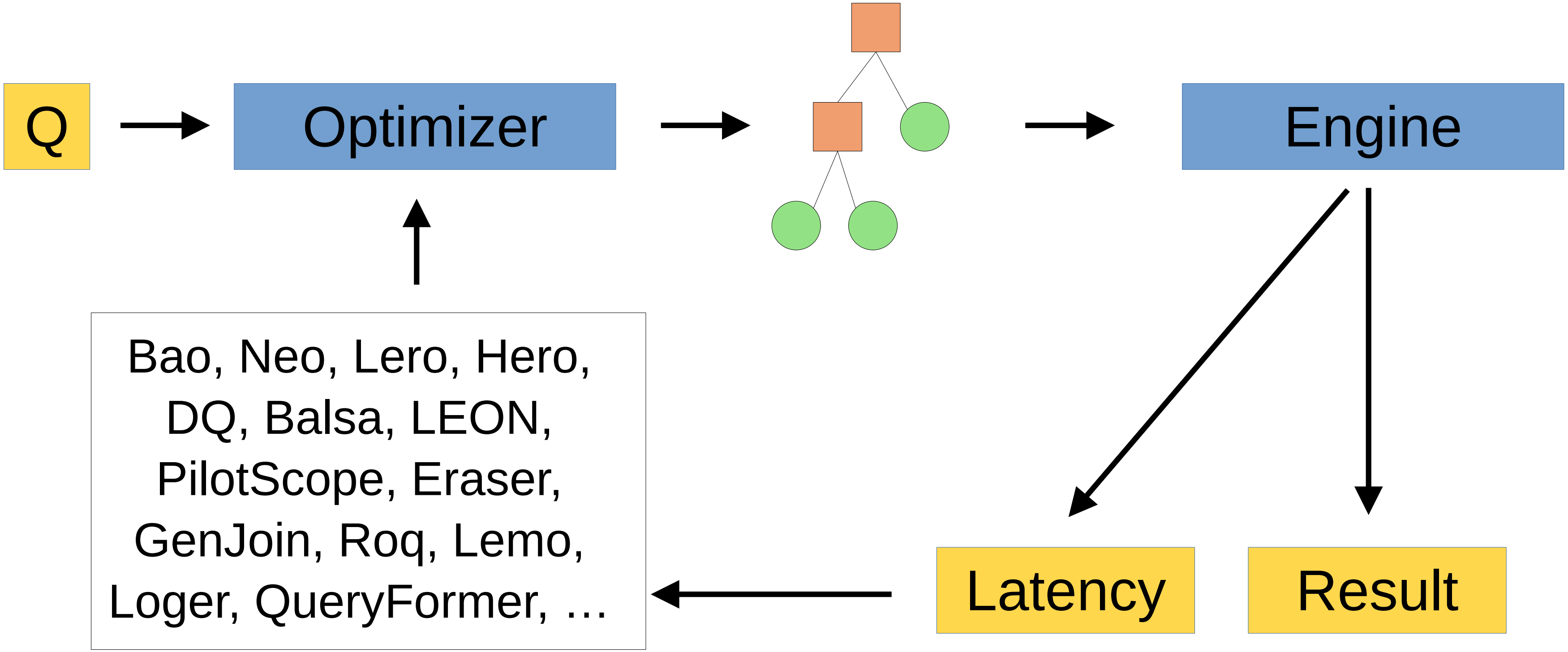
QOs are leaving info on the table



QOs are leaving info on the table



QOs are leaving info on the table



At least 29 others!

This Talk

- Why **learn** a query optimizer?
- Lessons learned **deploying learned QOs**
- **LimeQO**, an offline learned query optimizer
- The end of RL for QO?

The initial (failed) pitch

- “Put this cool RL stuff into your QO!” – Us
- “… no.” – basically everyone we talked to

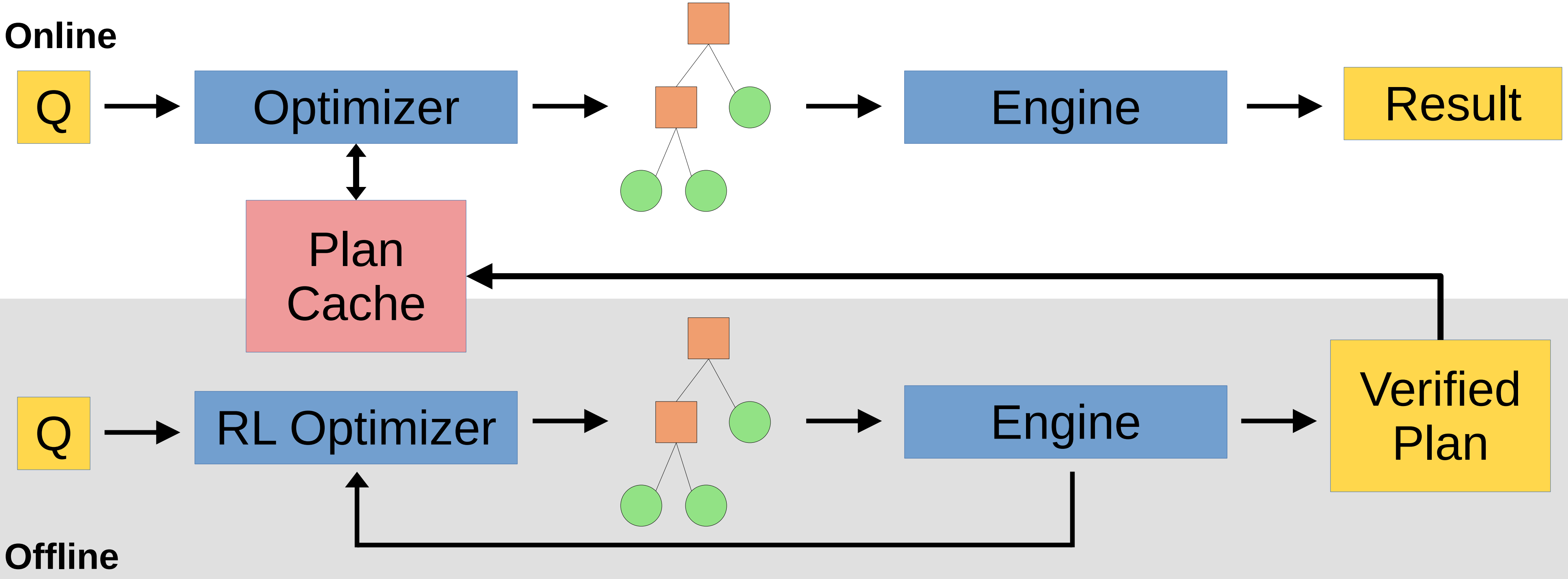
Sample Inefficiency
Even an hour of startup time kills POCs

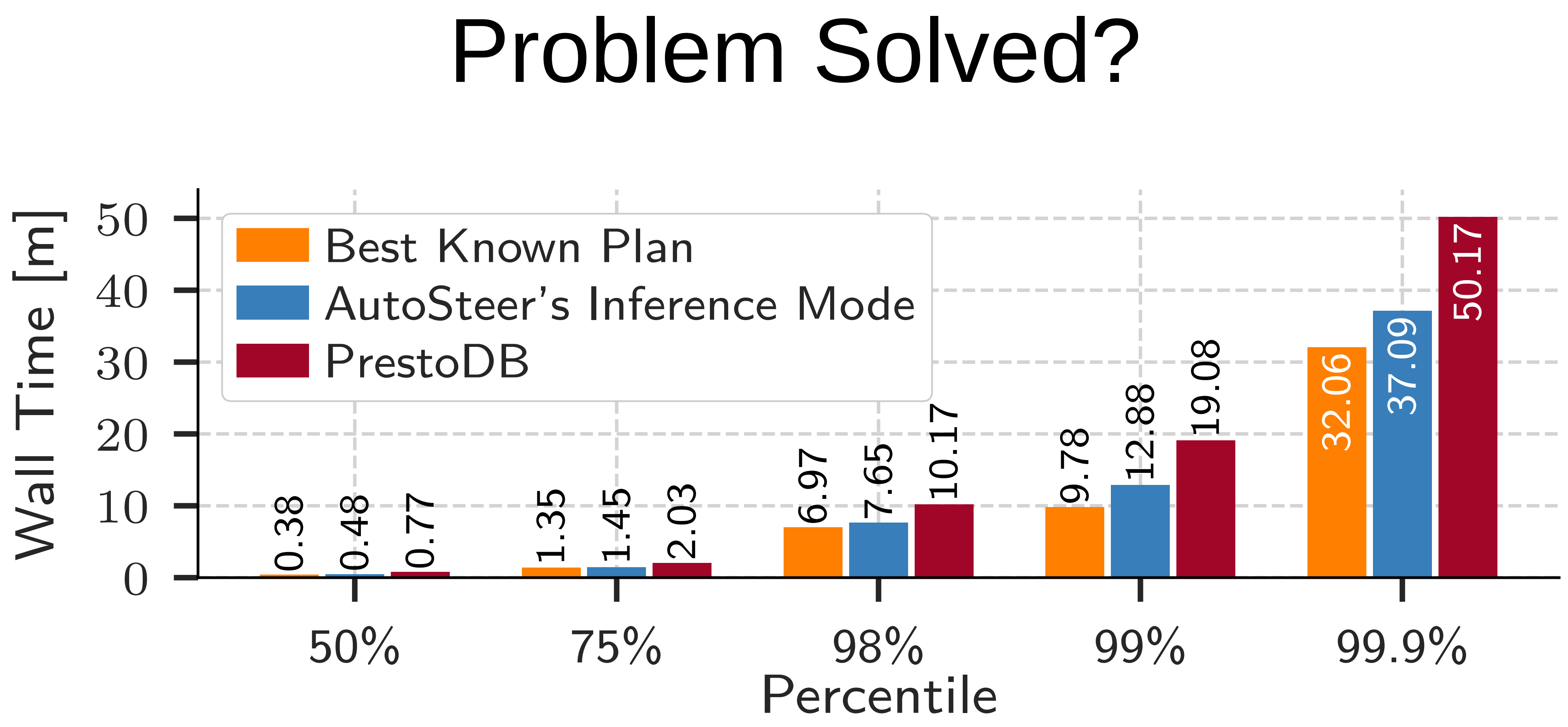
Exploration → Regressions
Unpredictable query slowdowns turn into 3AM pages.

Hot-path Complexity
Putting RL components (complex) into a QO (complex) is scary

What ended up landing

- Use RL *offline*, cache good plans, reuse live.




Problem Solved?
Wall Time [m]
50
40
30
20
10
0
Best Known Plan
AutoSteer's Inference Mode
PrestoDB
0.38
0.48
0.77
1.35
1.45
2.03
6.97
7.65
10.17
9.78
12.88
19.08
32.06
37.09
50.17
50%
75%
98%
99%
99.9%
Percentile

Problem Not Solved

- Offline execution time *is a resource*
  - … and therefore, as DB folks, we must optimize it!
- Given X hours of offline exploration time, maximize the improvement to my workload latency

This Talk

- Why **learn** a query optimizer?
- Lessons learned **deploying learned QOs**
- **LimeQO**, an offline learned query optimizer
- The end of RL for QO?



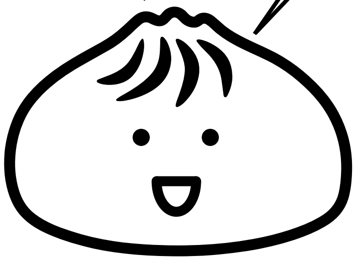
Joint work with Zixuan Yi,
Yao Tian, and Zack Ives

Default plan

	H_1	H_2	H_3	H_4	...	H_k
Q_1						
Q_2						
Q_3						
...						
Q_n						

Default plan

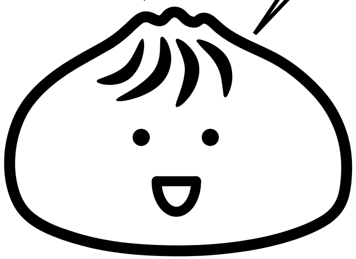
	H_1	H_2	H_3	H_4	...	H_k
Q_1	23					
Q_2	21					
Q_3	18					
...	...					
Q_n	22					



AutoSteer tests one hint per query offline

Default plan

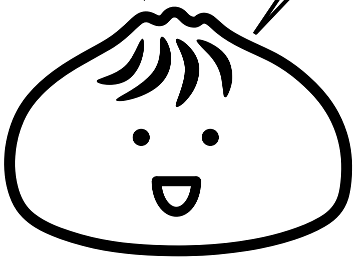
	H_1	H_2	H_3	H_4	...	H_k
Q_1	23					43
Q_2	21				8	
Q_3	18		10			
...	...					
Q_n	22	32				



AutoSteer tests one hint per query offline

Default plan

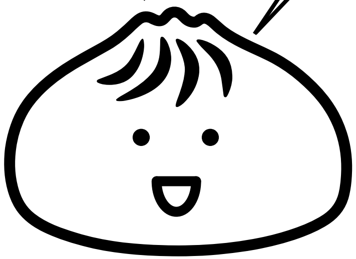
	H_1	H_2	H_3	H_4	...	H_k
Q_1	**23**					~~43~~
Q_2	21				8	
Q_3	18		10			
...	...					
Q_n	**22**	~~32~~				



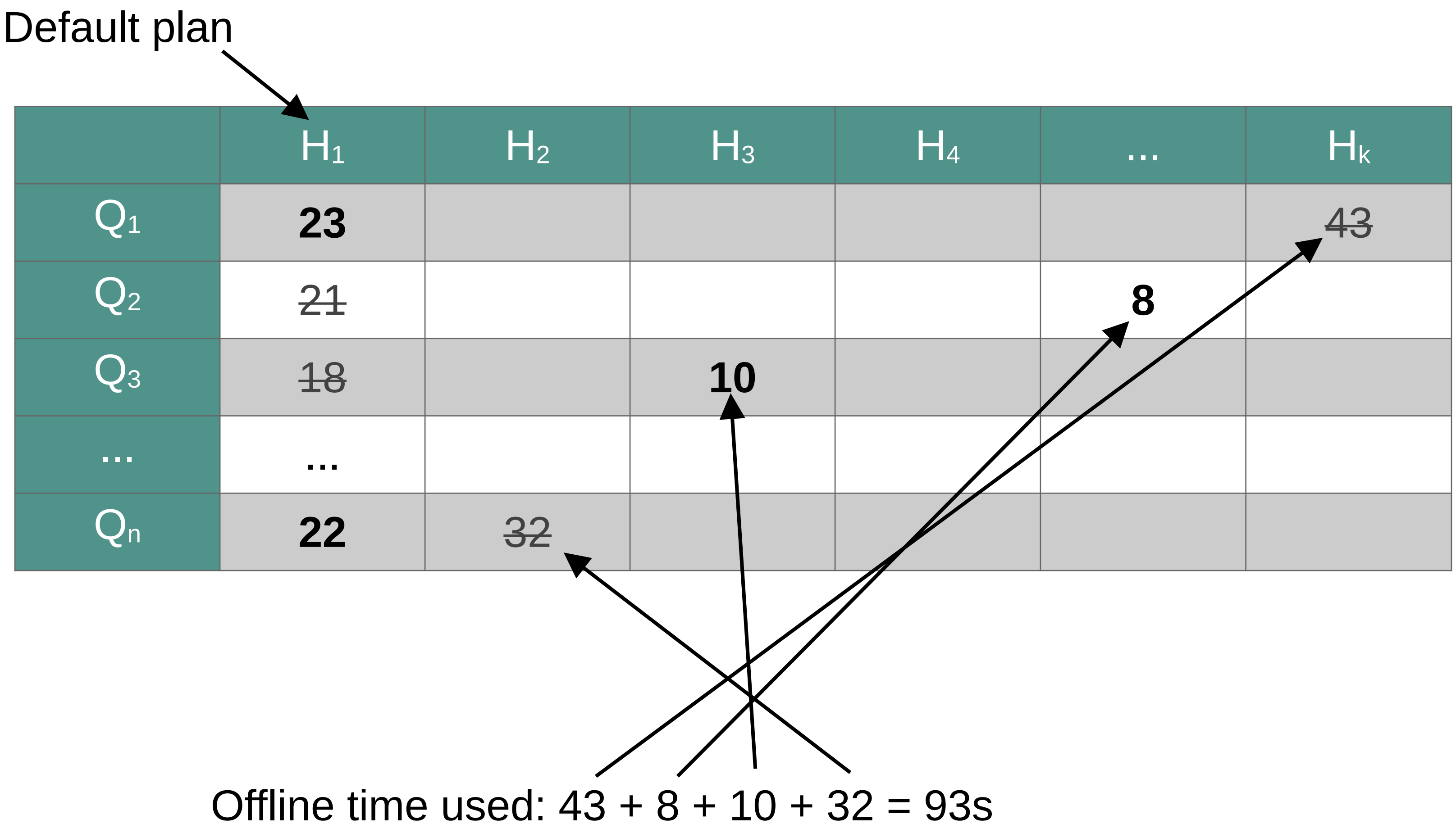
Some hints don’t work out (slower)

Default plan

	H_1	H_2	H_3	H_4	...	H_k
Q_1	**23**					~~43~~
Q_2	~~21~~				**8**	
Q_3	~~18~~		**10**			
...	...					
Q_n	**22**	~~32~~				



Some hints do work out (faster)


Default plan
H1
H2
H3
H4
...
Hk
Q1
23
43
Q2
21
8
Q3
18
10
...
...
Qn
22
32
Offline time used: 43 + 8 + 10 + 32 = 93s

Default plan

	H_1	H_2	H_3	H_4	...	H_k
Q_1	**23**					~~43~~
Q_2	~~21~~				**8**	
Q_3	~~18~~		**10**			
...	...					
Q_n	**22**	~~32~~				

Offline time used: 43 + 8 + 10 + 32 = 93s
Reduction in query time: (21 – 8) + (18 – 19) = 12s

Default plan

Key insight!

	H_1	H_2	H_3	H_4	...	H_k
Q_1	**23**					~~43~~
Q_2	~~21~~				**8**	
Q_3	~~18~~		**10**			
...	...					
Q_n	**22**	~~32~~				

We want to **minimize** offline time used and **maximize** reduction in query time!

Offline time used: 43 + 8 + 10 + 32 = 93s
Reduction in query time: (21 – 8) + (18 – 19) = 12s

	H_1	H_2	H_3	H_4	...	H_k
Q_1	**23**					~~43~~
Q_2	~~21~~				**8**	
Q_3	~~18~~		**10**			
...	...					
Q_n	**22**	~~32~~				

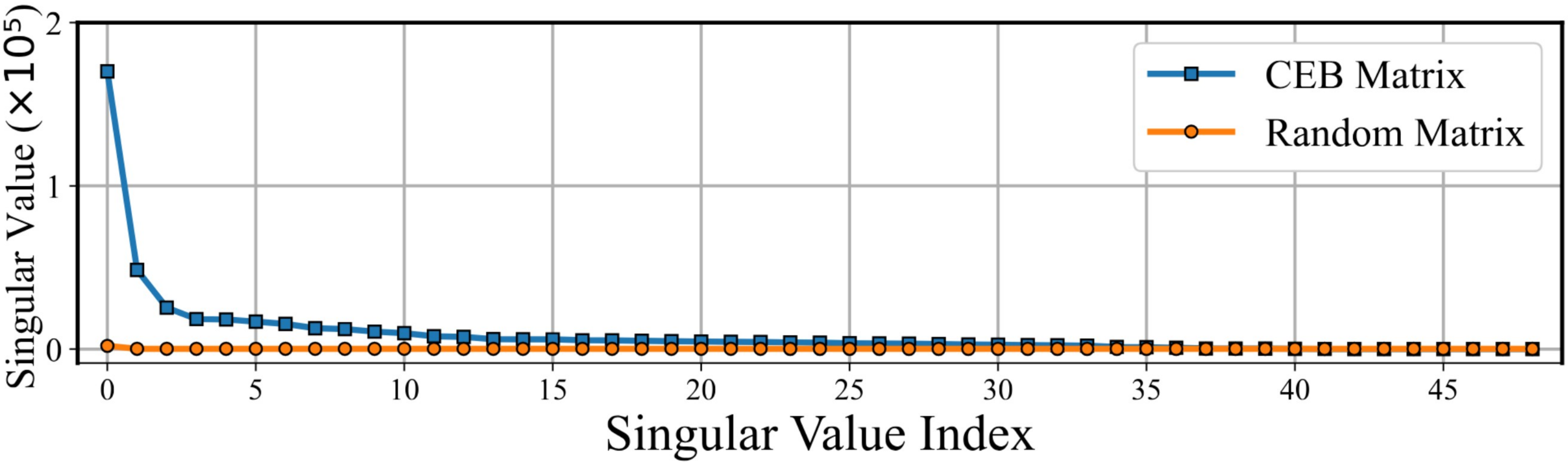
$= \hat{W}$

Workload Matrix

$\hat{W}$ is the partially observed workload matrix

W is the full workload matrix

Matrix Properties of W



→ the workload matrix has **low rank**

Matrix Properties of W

$$Q \times H = W$$

N x R   R x **K**   **N** x **K**

N is the number of queries in the workload
K is the number of possible hints

Why is W low rank?
→ Queries with similar performance on H_a are likely to have similar performance on H_b

LimeQO

- Given a partially observed $\hat{W}$, predict W

Default plan

	H_1	H_2	H_3	H_4	...	H_k
Q_1	23					
Q_2	21					
Q_3	18					
...	...	...	...	...	...	...
Q_n	22					

LimeQO

LimeQO

- Given a partially observed $\hat{W}$, predict W

Default plan → H_1

	H_1	H_2	H_3	H_4	...	H_k
Q_1	23	≈ 22	≈ 31	≈ 81	≈ ...	≈ 62
Q_2	21	≈ 9	≈ 14	≈ 43	≈ ...	≈ 11
Q_3	18	≈ 12	≈ 9	≈ 12	≈ ...	≈ 4
...	≈ ...	≈ ...	≈ ...	≈ ...	≈ ...	≈ ...
Q_n	22	≈ 19	≈ 23	≈ 4	≈ ...	≈ 16

LimeQO

Use matrix completion to guess the rest of the matrix

LimeQO

- Given a partially observed $\hat{W}$, predict W

Default plan → H_1

	H_1	H_2	H_3	H_4	…	H_k
Q_1	23	≈ 22	≈ 31	≈ 81	≈ …	≈ 62
Q_2	21	**≈ 9**	≈ 14	≈ 43	≈ …	≈ 11
Q_3	18	≈ 12	≈ 9	≈ 12	≈ …	**≈ 4**
…	≈ …	≈ …	≈ …	≈ …	≈ …	≈ …
Q_n	22	≈ 19	≈ 23	**≈ 4**	≈ …	≈ 16

LimeQO

Identify candidates for exploration (consider cost and benefit)

LimeQO

- Given a partially observed $\hat{W}$, predict W

Default plan → H_1

	H_1	H_2	H_3	H_4	...	H_k
Q_1	23				...	
Q_2	21	**≈ 9**			...	
Q_3	18				...	**≈ 4**
...	...				...	
Q_n	22			**≈ 4**	...	

Identify candidates for exploration (consider cost and benefit)

LimeQO

- Given a partially observed $\hat{W}$, predict W

Default plan

	H_1	H_2	H_3	H_4	...	H_k
Q_1	23				...	
Q_2	21	18			...	
Q_3	18				...	4
...	...				...	
Q_n	22			> 22	...	

Censored observation

LimeQO

- Given a partially observed $\hat{W}$, predict W

Default plan → H_1

	H_1	H_2	H_3	H_4	...	H_k
Q_1	23	≈ 14	≈ 11	≈ 22	≈ ...	≈ 88
Q_2	21	18	≈ 17	≈ 43	≈ ...	≈ 22
Q_3	18	≈ 11	≈ 15	≈ 16	≈ ...	4
...	≈ ...	≈ ...	≈ ...	≈ ...	≈ ...	≈ ...
Q_n	22	≈ 22	≈ 8	> 22	≈ ...	≈ 24

LimeQO

Do matrix completion again with new values

LimeQO

- Given a partially observed $\hat{W}$, predict W

Default plan → H_1

	H_1	H_2	H_3	H_4	...	H_k
Q_1	23	≈ 14	**≈ 11**	≈ 22	≈ ...	≈ 88
Q_2	21	18	**≈ 17**	≈ 43	≈ ...	≈ 22
Q_3	18	≈ 11	≈ 15	≈ 16	≈ ...	4
...	≈ ...	≈ ...	≈ ...	≈ ...	≈ ...	≈ ...
Q_n	22	≈ 22	≈ 11	> 22	≈ ...	**≈ 8**

LimeQO

Identify candidates for exploration (consider cost and benefit)

LimeQO

- Given a partially observed $\hat{W}$, predict W

Default plan → H_1

	H_1	H_2	H_3	H_4	...	H_k
Q_1	23		**≈ 11**		...	
Q_2	21	18	**≈ 17**		...	
Q_3	18				...	4
...	...	...	...	...	...	
Q_n	22			> 22	...	**≈ 8**

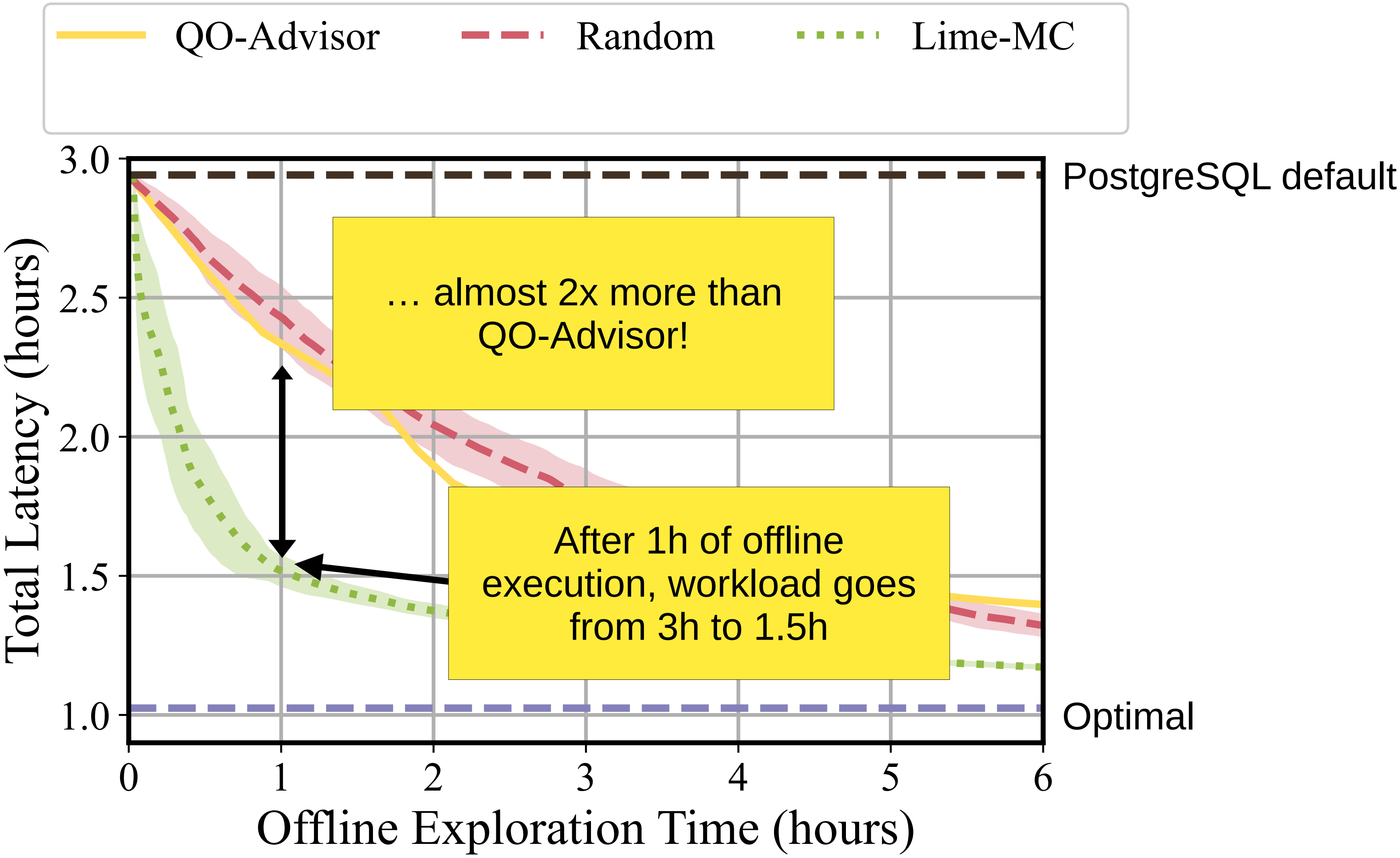
Identify candidates for exploration (consider cost and benefit)

LimeQO

- Given a partially observed $\hat{W}$, predict W

Default plan

	H_1	H_2	H_3	H_4	...	H_k
Q_1	23		> 23		...	
Q_2	21	18	12		...	
Q_3	18				...	4
...	...	...	...	...	...	
Q_n	22			> 22	...	14


QO-Advisor
Random
Lime-MC
PostgreSQL default
Optimal
… almost 2x more than QO-Advisor!
After 1h of offline execution, workload goes from 3h to 1.5h
Total Latency (hours)
3.0
2.5
2.0
1.5
1.0
0
1
2
3
4
5
6
Offline Exploration Time (hours)

Many more fun problems...

- Matrix completion with censored observations
- Which entries to explore?
  - → "acquisition function"
- Full experimental rundown
- https://rm.cab/limeqo

Joint work with Zixuan Yi, Yao Tian, and Zack Ives

What about the query level?

- LimeQO is an offline optimizer for a workload
  - Core approach: transductive learning
  - Assumes each query has a small set of options

- BayesQO is an offline optimizer for a single query
  - Core approach: Bayes opt
  - Only works on one query at a time

This Talk

- Why **learn** a query optimizer?
- Lessons learned **deploying learned QOs**
- **LimeQO**, an offline learned query optimizer
- The end of RL for QO?

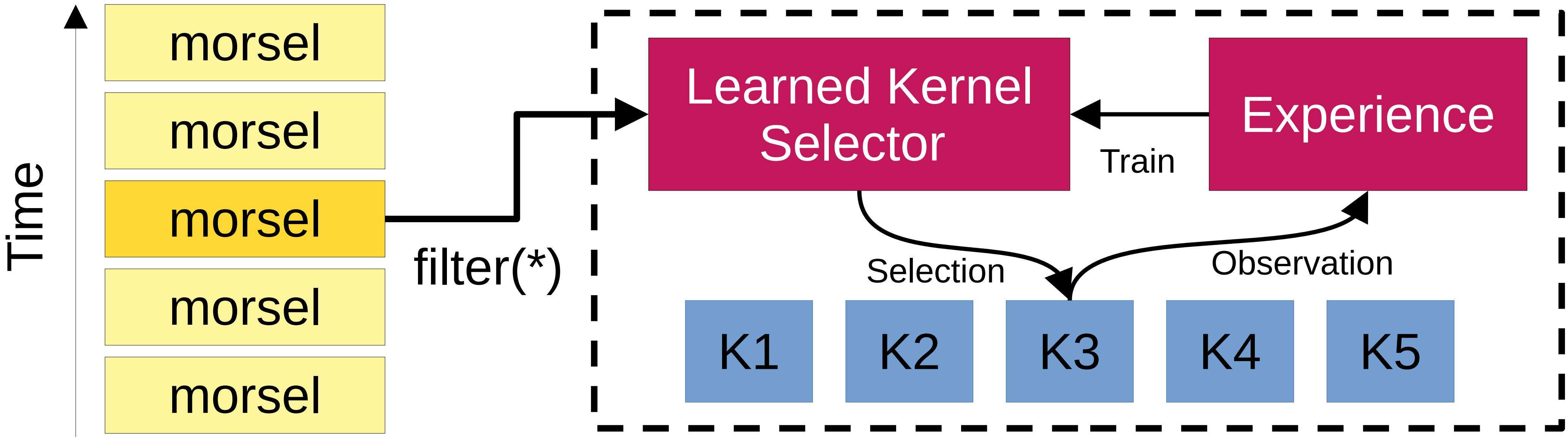
RL for QO Woes

Sample Inefficiency
Even an hour of startup time kills POCs

Exploration → Regressions
Unpredictable query slowdowns turn into 3AM pages.

Hot-path Complexity
Putting RL components (complex) into a QO (complex) is scary

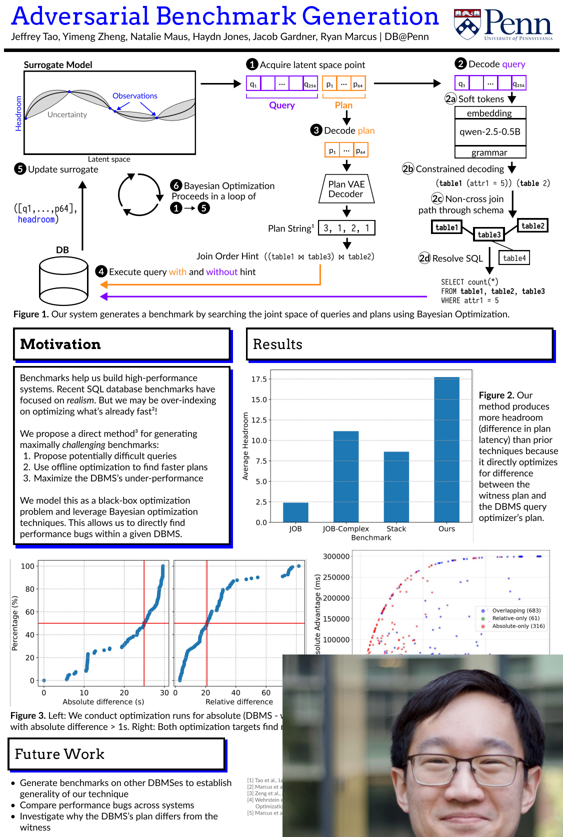
Morsel-Driven Parallelism

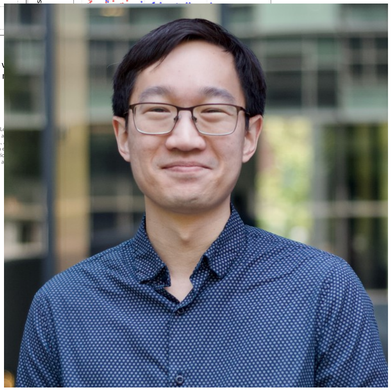


Sample Inefficiency?
1k-2k+ morsels *per query*! ✅

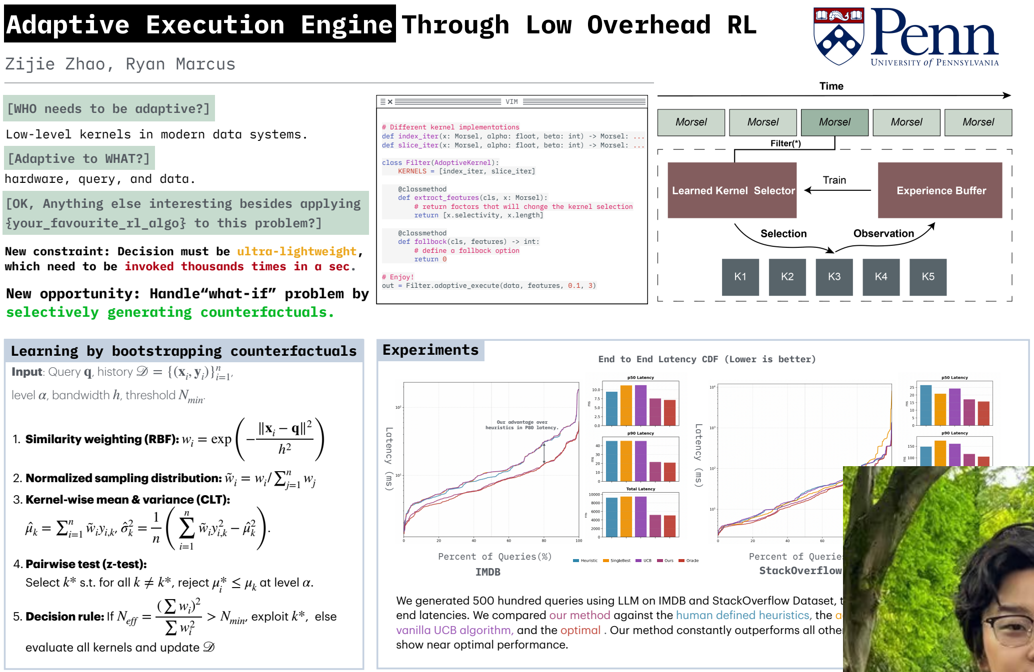
Exploration → Regressions?
Incorrect decisions average out over the course of a single query! ✅

Hot-path Complexity?
The EE is often *less* complex than the QO, but is not simple. ⚠️





Jeff Tao



Zijie Zhao

Get these slides: https://rm.cab/nedb26

My homepage: https://ryanmarc.us