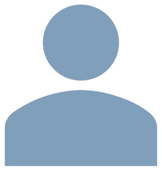
From Logs to Causal Inference: Diagnosing Large Systems

Markos Markakis, Brit Youngmann, Trinity Gao, Ziyu Zhang, Rana Shahout, Peter Baile Chen, Chunwei Liu, Ibrahim Sabek, Michael Cafarella



```
2023-11-06 16:34:40.811 EST [ 65495bf0.179a 3/2462 ] postgres@tpcds1 LOG:  statement: SET work_mem = 128;
2023-11-06 16:34:40.812 EST [ 65495bf0.179a 3/2463 ] postgres@tpcds1 LOG:  statement: SET max_parallel_workers = 1;
2023-11-06 16:34:40.812 EST [ 65495bf0.179a 3/2464 ] postgres@tpcds1 LOG:  statement: SET effective_cache_size = 262144;
2023-11-06 16:34:40.812 EST [ 65495bf0.179a 3/2465 ] postgres@tpcds1 LOG:  statement: SET maintenance_work_mem = 32768;
2023-11-06 16:34:40.813 EST [ 65495bf0.179a 3/2466 ] postgres@tpcds1 LOG:  statement: SET random_page_cost = 4;
2023-11-06 16:34:40.813 EST [ 65495bf0.179a 3/2467 ] postgres@tpcds1 LOG:  statement: SET seq_page_cost = 1;
2023-11-06 16:34:40.820 EST [ 65495bf0.179a 3/2468 ] postgres@tpcds1 LOG:  statement: -- Filename: query080.sql
2023-11-06 16:34:53.850 EST [ 65495bf0.179a 3/2468 ] postgres@tpcds1 LOG:  duration: 13029.853 ms
...
2023-11-07 08:15:48.440 EST [ 654a3884.49b7 3/3278 ] postgres@tpcds1 LOG:  statement: SET work_mem = 128;
2023-11-07 08:15:48.441 EST [ 654a3884.49b7 3/3279 ] postgres@tpcds1 LOG:  statement: SET max_parallel_workers = 1;
2023-11-07 08:15:48.441 EST [ 654a3884.49b7 3/3280 ] postgres@tpcds1 LOG:  statement: SET effective_cache_size = 262144;
2023-11-07 08:15:48.441 EST [ 654a3884.49b7 3/3281 ] postgres@tpcds1 LOG:  statement: SET maintenance_work_mem = 65536;
2023-11-07 08:15:48.442 EST [ 654a3884.49b7 3/3282 ] postgres@tpcds1 LOG:  statement: SET random_page_cost = 4;
2023-11-07 08:15:48.442 EST [ 654a3884.49b7 3/3283 ] postgres@tpcds1 LOG:  statement: SET seq_page_cost = 1;
2023-11-07 08:15:48.443 EST [ 654a3884.49b7 3/3284 ] postgres@tpcds1 LOG:  statement: -- Filename: query080.sql
2023-11-07 08:15:49.446 EST [ 654a3884.49b7 3/3284 ] postgres@tpcds1 LOG:  duration: 1003.228 ms
...
```

How can we go from this to correct quantitative conclusions?

Causal inference to the rescue!

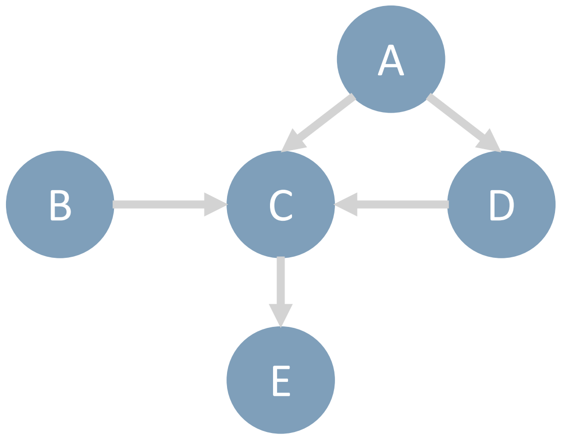
- Technical framework for drawing unbiased cause-effect conclusions.
- Ideal setting: run a controlled experiment to sidestep confounding.
- Relationships are expressed through average treatment effects:

$$ATE\,(T,Y) = E[Y\,|do(T = 1)] - E[Y|do\,(T = 0)]$$

- But logs are observational data – can't go back and "do".
- To avoid confounding, we must instead adjust for the right variables.

What to adjust for? Look at causal graph

- For each variable in the dataset, create a node.
- For each potential direct causal relationship, create a directed edge.
- Key property:
  - Each node is fully determined by its parents.
  - Conditionally independent of everything else.
- Causal discovery: automatically build a causal graph from data.



Applying causal discovery over log data is non-trivial

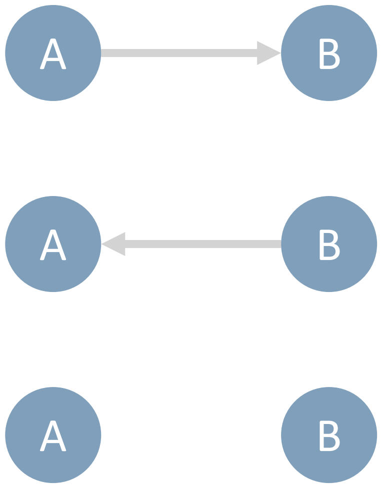
- Functional dependencies
  - The same event creates dependent log entries.
- Large number of variables
  - Many off-the-shelf algorithms scale exponentially.
- Biased data collection
  - Failure cases relatively rare, effect can be “drowned out”.

Table 1: Causal discovery on the PostgreSQL dataset. ✓ and ● indicate a non-empty and empty causal graph, respectively; ▲ indicates a 30-minute timeout; and ✗ indicates an error.

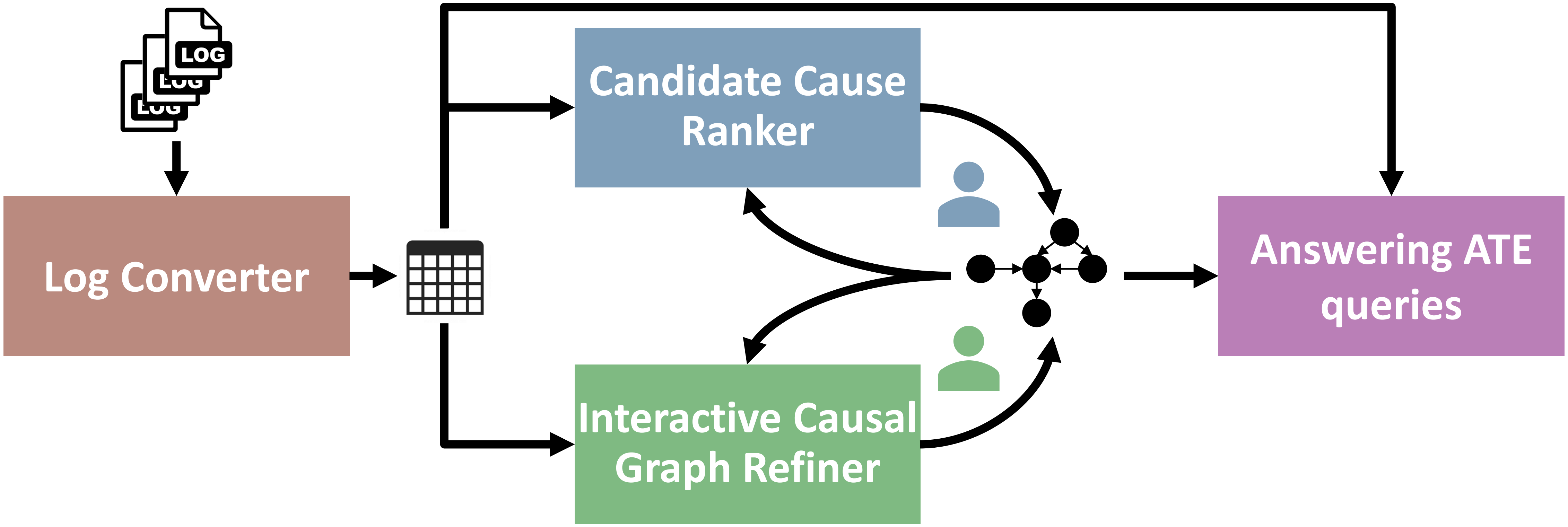
Alg.	Independence Test / Method	Result	Alg.	Scoring Function	Result
PC [93]	fisherz	✗	GIN [107]	kci	▲
	mv_fisherz	✗		hsic	▲
	mc_fisherz	✗	GRaSP [52]	CV_general	✓
	kci	▲		marginal_general	✗
	chisq	✓		CV_multi	✗
	gsq	●		marginal_multi	✗
	d_separation	✗		BIC	✗
FCI [94]	fisherz	✗		BDeu	✓
	kci	▲	GES [15]	CV_general	▲
	chisq	✓		marginal_general	✗
	gsq	●		CV_multi	▲
LiNGAM [90]		✗		marginal_multi	✗
				BIC	✗
Exact	dp [92]	✗		BIC_from_cov	✗
	astar [111]	✗		BDeu	✓

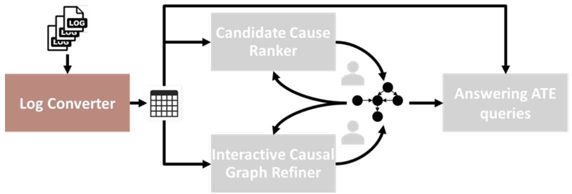
LOGos: a human-in-the-loop approach

- Manually creating a causal graph from a set of variables V would require $O(|V|^2)$ judgments.
- A judgment for outputs whether an edge should exist and in which direction.
- Our goal is to solicit judgments in order of decreasing impact on the final result.



Three main components work together



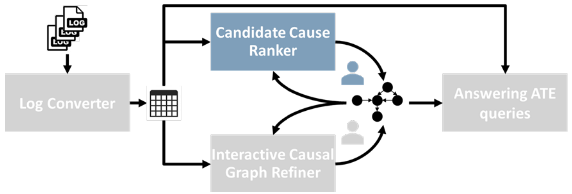


We must first obtain log data as a table

- **Causal unit:** user-defined unit of analysis.
- Aggregate variables within each causal unit.
- Pick aggregate function to maximize empirical entropy.
- Intuition: want causal units to "look different".
- More details in paper.

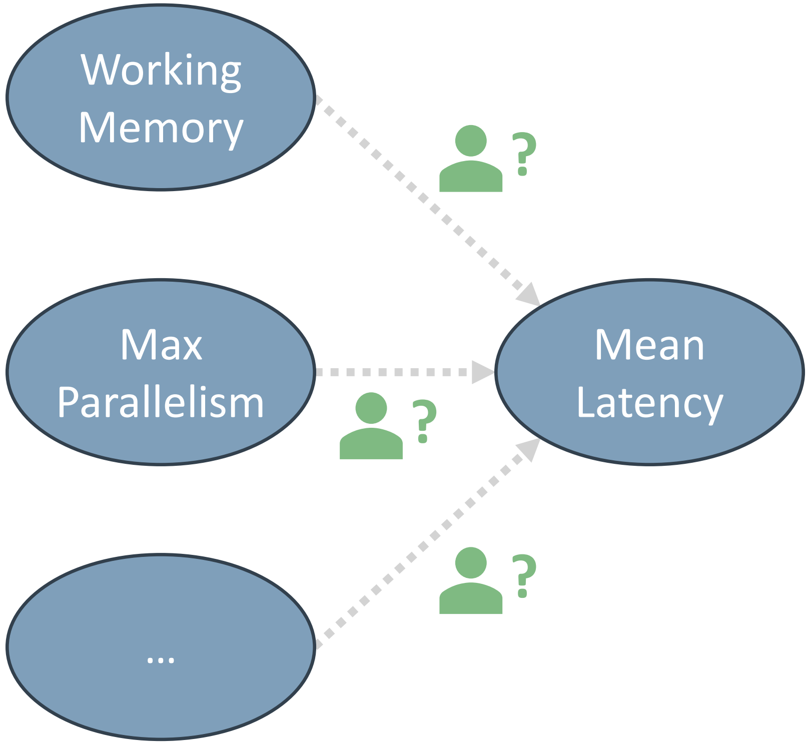
```
2023-11-06 16:34:40.811 EST [ 65495bf0.179a 3/2462 ] postgres@tpcds1 LOG:  statement: SET work_mem = 128;
2023-11-06 16:34:40.812 EST [ 65495bf0.179a 3/2463 ] postgres@tpcds1 LOG:  statement: SET max_parallel_workers = 1;
2023-11-06 16:34:40.812 EST [ 65495bf0.179a 3/2464 ] postgres@tpcds1 LOG:  statement: SET effective_cache_size = 262144;
2023-11-06 16:34:40.812 EST [ 65495bf0.179a 3/2465 ] postgres@tpcds1 LOG:  statement: SET maintenance_work_mem = 32768;
2023-11-06 16:34:40.813 EST [ 65495bf0.179a 3/2466 ] postgres@tpcds1 LOG:  statement: SET random_page_cost = 4;
2023-11-06 16:34:40.813 EST [ 65495bf0.179a 3/2467 ] postgres@tpcds1 LOG:  statement: SET seq_page_cost = 1;
2023-11-06 16:34:40.820 EST [ 65495bf0.179a 3/2468 ] postgres@tpcds1 LOG:  statement: -- Filename: query080.sql
2023-11-06 16:34:53.850 EST [ 65495bf0.179a 3/2468 ] postgres@tpcds1 LOG:  duration: 13029.853 ms
...
2023-11-07 08:15:48.440 EST [ 654a3884.49b7 3/3278 ] postgres@tpcds1 LOG:  statement: SET work_mem = 128;
2023-11-07 08:15:48.441 EST [ 654a3884.49b7 3/3279 ] postgres@tpcds1 LOG:  statement: SET max_parallel_workers = 1;
2023-11-07 08:15:48.441 EST [ 654a3884.49b7 3/3280 ] postgres@tpcds1 LOG:  statement: SET effective_cache_size = 262144;
2023-11-07 08:15:48.441 EST [ 654a3884.49b7 3/3281 ] postgres@tpcds1 LOG:  statement: SET maintenance_work_mem = 65536;
2023-11-07 08:15:48.442 EST [ 654a3884.49b7 3/3282 ] postgres@tpcds1 LOG:  statement: SET random_page_cost = 4;
2023-11-07 08:15:48.442 EST [ 654a3884.49b7 3/3283 ] postgres@tpcds1 LOG:  statement: SET seq_page_cost = 1;
2023-11-07 08:15:48.443 EST [ 654a3884.49b7 3/3284 ] postgres@tpcds1 LOG:  statement: -- Filename: query080.sql
2023-11-07 08:15:49.446 EST [ 654a3884.49b7 3/3284 ] postgres@tpcds1 LOG:  duration: 1003.228 ms
...
```

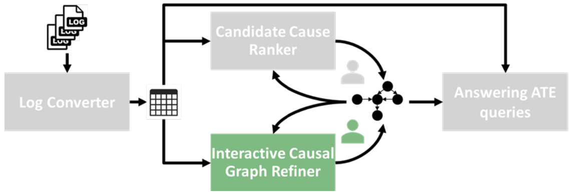
Session ID	Mode work_mem	…	Mean duration
65495bf0.179a	128	…	…
654a3884.49b7	128	…	…
…	…	…	..



LOGos ranks candidate causes to judge

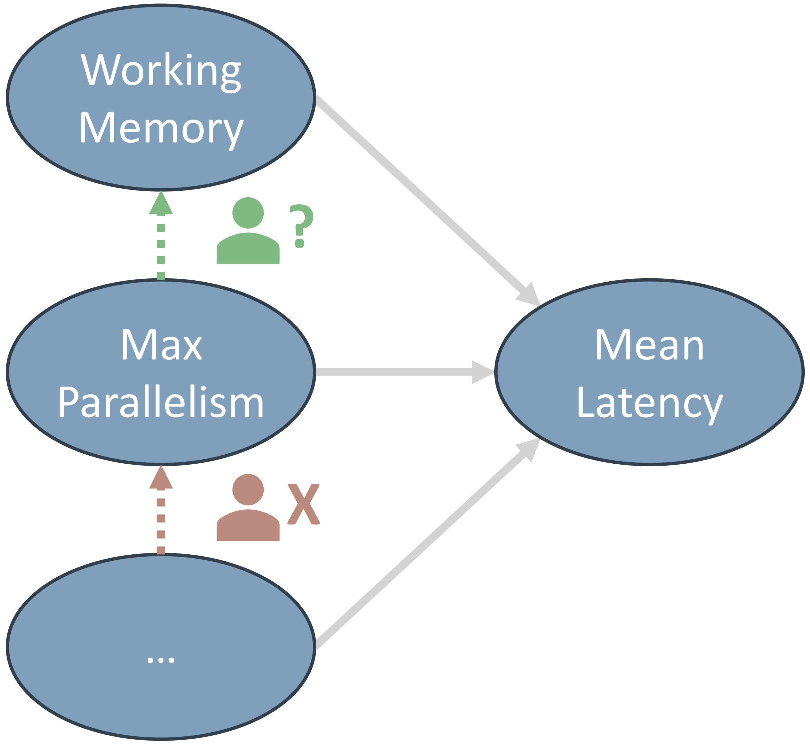
- Specify a target variable O – e.g. mean latency.
- Goal: Find candidate causes of O and solicit user judgments about them.
- Intuition: Find the few reliably related variables
- Algorithm sketch:
  - Sparsely regress O on the remaining variables using LASSO, to find candidate graph parents.
  - Compute and sort by pairwise p-values to reflect reliability.
  - Prompt user for judgments in that order.

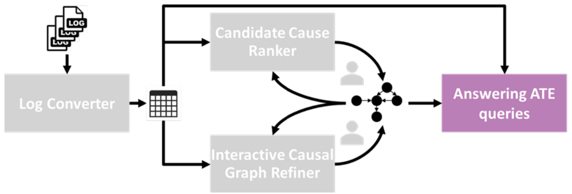




The graph can be refined interactively

- Goal: Which edge edit would most impact the ATE of interest? Have the user judge it.
- Intuition: Avoid judging graph edits irrelevant to the ATE of interest.
- Algorithm sketch:
  - From the current graph, recover the *adjustment set*.
  - For each single-variable change to the adjustment set, map it to graph edge edits.
  - Rank these alternatives by their impact on the ATE.



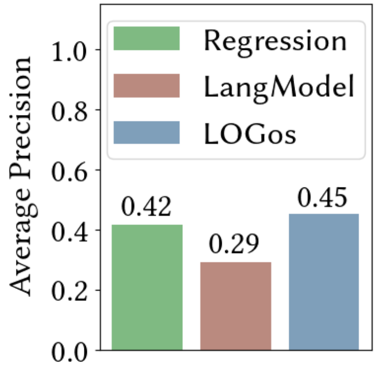


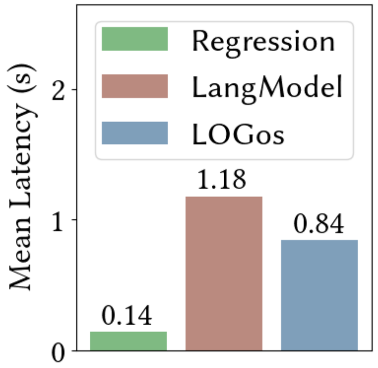
We evaluate LOGos across log collections

- No open-source log dataset with labels of what went wrong.
- We construct three datasets to evaluate on:
  - PostgreSQL (Real-world):
    - Run TPC-DS for various knob values.
  - Proprietary (Real-world, post-processed):
    - Tweak HTTP log from real application to fail with some probability.
  - XYZ (Synthetic):
    - Synthetic variables X, Y, Z among many, noisy others.

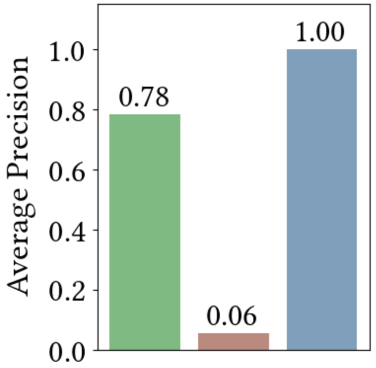
LOGos finds good candidate causes, fast

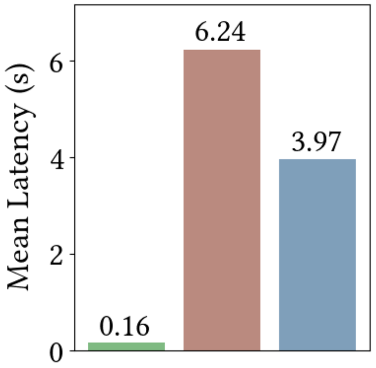
PostgreSQL



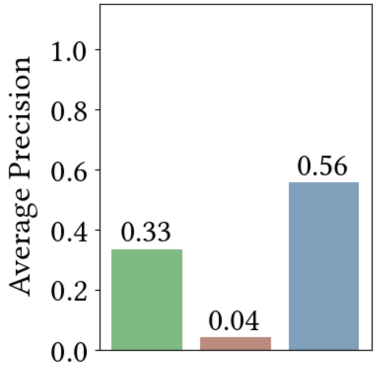


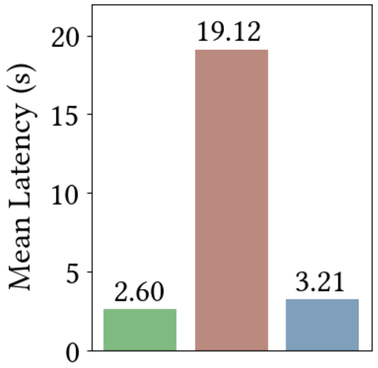
Proprietary



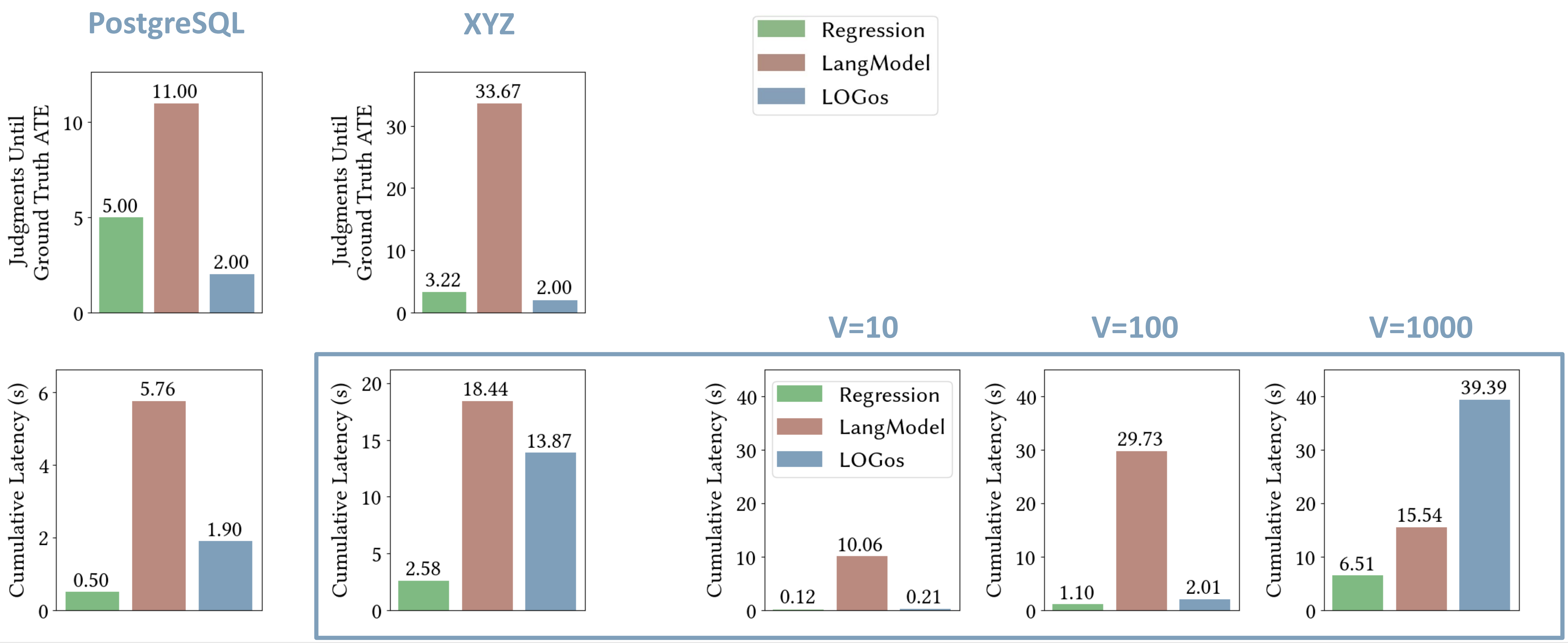


XYZ





Graph refinement reduces # judgments



Key Takeaway

LOGos can help engineers draw principled causal conclusions from logs faster

Thank you!









Paper: https://www.vldb.org/pvldb/vol18/p158-markakis.pdf

Questions? Reach out at markakis@mit.edu