Sawmill: From Logs to Causal Diagnosis of Large Systems

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

May 23, 2024

“My queries are SO SLOW using your product, would give 0 stars if I could!”

- Alex is an **on-call engineer** at a startup offering a database-as-a-service product.
- Certain users are **somewhat dissatisfied** with the product’s latency.
- Alex has been tasked with finding what is **the best and quickest way to** deal with such complaints, so that **negative reviews and tickets stop rolling in**.
- The company could really use every human-hour available at this phase – solving the problem **correctly and with minimal effort** is essential!
- But all Alex has is **a bunch of logs:**

```
2023-11-01 17:54:53.018 EDT [ 6542c92d.1f943 3/5184 ] postgres@tpcds1 LOG:  statement: BEGIN
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/5184 ] postgres@tpcds1 LOG:  duration: 0.076 ms
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/5184 ] postgres@tpcds1 LOG:  statement: SET max_parallel_workers = 1;
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/5184 ] postgres@tpcds1 LOG:  duration: 0.062 ms
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/5184 ] postgres@tpcds1 LOG:  statement: COMMIT
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/0 ] postgres@tpcds1 LOG:  duration: 0.030 ms
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/5185 ] postgres@tpcds1 LOG:  statement: BEGIN
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/5185 ] postgres@tpcds1 LOG:  duration: 0.023 ms
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/5185 ] postgres@tpcds1 LOG:  statement: SET work_mem = '128.0';
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/5185 ] postgres@tpcds1 LOG:  duration: 0.044 ms
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/5185 ] postgres@tpcds1 LOG:  statement: COMMIT
2023-11-01 17:54:53.019 EDT [ 6542c92d.1f943 3/0 ] postgres@tpcds1 LOG:  duration: 0.026 ms
2023-11-01 17:54:53.027 EDT [ 6542c92d.1f943 3/5186 ] postgres@tpcds1 LOG:  statement: BEGIN
2023-11-01 17:54:53.027 EDT [ 6542c92d.1f943 3/5186 ] postgres@tpcds1 LOG:  duration: 0.033 ms
2023-11-01 17:54:53.028 EDT [ 6542c92d.1f943 3/5186 ] postgres@tpcds1 LOG:  statement: -- Filename: query080.sql


        with ssr as
         (select  s_store_id as store_id,
                  sum(ss_ext_sales_price) as sales,
                  sum(coalesce(sr_return_amt, 0)) as returns,
                  sum(ss_net_profit - coalesce(sr_net_loss, 0)) as profit
          from store_sales left outer join store_returns on
                 (ss_item_sk = sr_item_sk and ss_ticket_number = sr_ticket_number),
             date_dim,
```

Causal reasoning to the rescue!

- In large complex systems, **problems are a daily reality**.
- Operations teams have to **diagnose the problem from observability data** like logs and decide on the **most effective way to restore** the system.
- **Causal reasoning** can help them describe the system accurately and **draw reliable conclusions**, avoiding wasted effort.
- But no system supports **causal reasoning over log data**!

Sawmill bridges logs and causal reasoning

Explore candidate causes

Choose a variable to explore candidate causes for:

Select a variable:

duration mean

Explore Candidate Causes

Candidate cause(s) found!

Candidate Cause(s)

	Candidate	Candidate Tag
0	5da61167_14+mean	time mean
1	da4032f7_15+mean	work_mem mean
2	Date+mean	Date mean
3	8d5769b2_15+mean	seq_page_cost mean
4	9d2f8f03_15+mean	max_parallel_workers mean
5	Time+mean	Time mean
6	9eea70b1_15+mean	random_page_cost mean
7	aea309d3_23+mean	port mean
8	5da61167_26+mean	5da61167_26 mean
9	97a1388a_15+mean	maintenance_work_mem me:

ATE Treatment:

max_parallel_workers me

ATE Outcome:

duration mean

ATE Value:

-156.46574784131553

Exploration Score:

0.07397959183673469

Suggested Next Exploration:

effective_cache_size me

Decide whether to include an edge to the causal graph

Choose the endpoints of the edge you would like to decide on:

Source node: work_mem mean

Destination node: max_parallel_workers m...

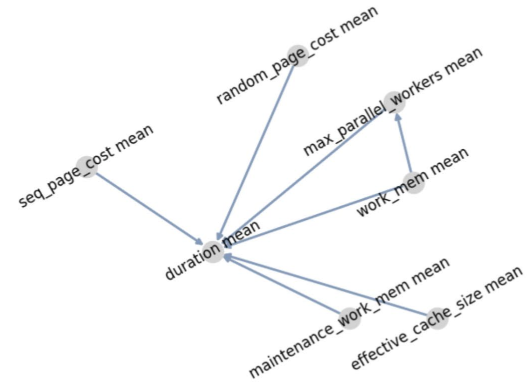
Accept

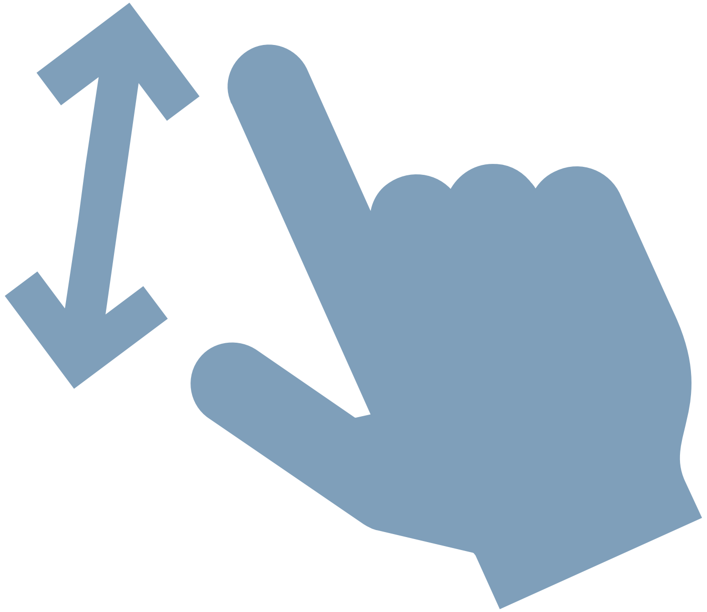
Reject

Reject Undecided Outgoing from Source

Reject Undecided Incoming to Destination

Causal graph

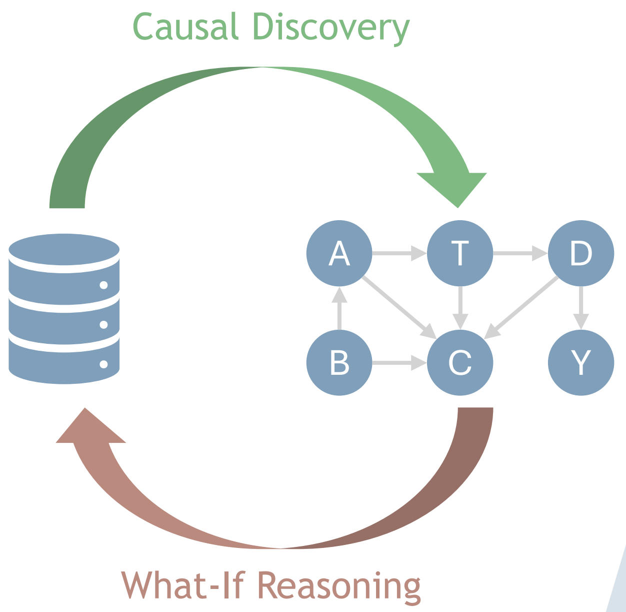




Looking under the hood

Crafting a causal graph for a complex system is a tall order

- Many applications use **causal discovery** to derive an unknown causal graph from available data.
- For system debugging, we need to go **in the opposite direction**:
  - Causal mechanism in principle known.
  - Desired data hard to collect in production.
  - Must tap whatever logs are available to evaluate the impact of potential fixes.
- But this **requires starting from a causal graph**!
- **Daunting to fully specify** for a complex system.

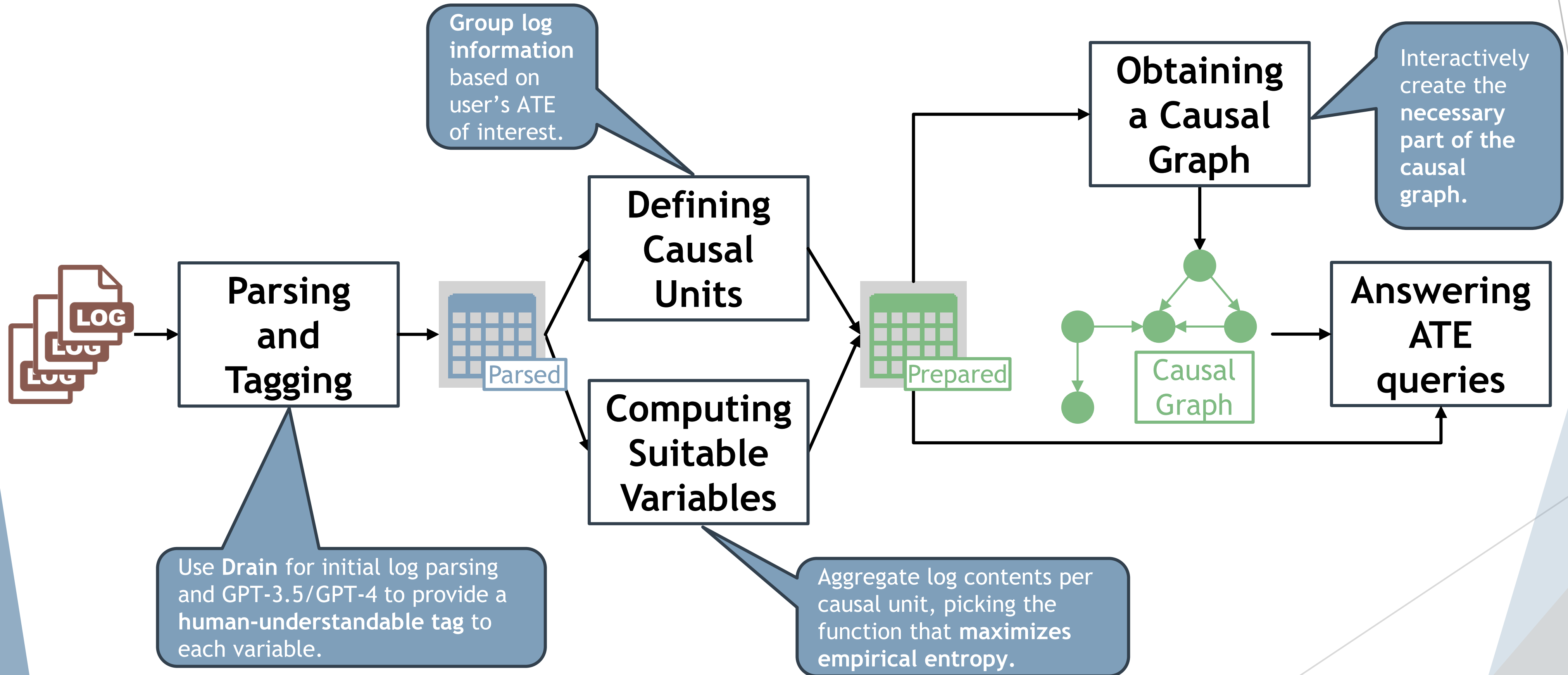


But we don’t actually need the entire causal graph!

- We are not creating a graph just for fun – we want to **use it to answer an actual question** about the system.
- In Pearl’s framework, this takes the form of an **Average Treatment Effect (ATE)** calculation.
- Correctly calculating such effects only requires **reasoning about specific paths in the causal graph.**

Key Insight:
Not every part of the graph is needed to calculate an ATE. Recover only the relevant parts and selectively tap the user’s expertise to validate them!

Sawmill's human-in-the-loop architecture bridges logs and causality



Putting Sawmill to the test

Battling confounding in real-world logs

- Collect logs from **PostgreSQL running TPC-DS**.
- **Vary performance-affecting parameters:** work_mem, seq_page_cost, random_page_cost, max_parallel_workers, maintenance_work_mem and effective_cache_size.
- **Bias parameter combinations** to trade off work_mem and max_parallel_workers.
- Not accounting for bias makes **mean latency increase for more parallelism.**

Outcome:
Sawmill helps uncover the confounding and adjust for it successfully.

Accuracy:
Relevant causes ranked highly (MRR=0.5667) compared to regression baseline (MRR=0.0476)

Human Efficiency:
Graph building requires only 5 user calls.
Dataset creation pipeline needs another 5.

Computational Efficiency:
Graph-building calls take just 4.85s.
Dataset creation needs 42.06s for 20MB.

Discerning subtle semi-synthetic effects

- Start with **real logs from a mobile application.**
- Generate similarly complex logs for 1000 users, designate a varying fraction of them (1% to 50%) as **faulty.**
- Have faulty users artificially be on a **different OS version** and have them **fail HTTP requests at varying rates** (20% to 100% of the time).
- Have non-faulty users **fail HTTP requests 10% of the time.**

Outcome:
Even when the effect is maximally subtle, Sawmill ranks the right candidate cause first.

Accuracy:
Mean MRR is 1.0000 and we recover the ATE with mean error 11.72% (14.64% for baseline)

Human Efficiency:
Graph building requires only 2 user calls. Dataset creation pipeline needs another 4.

Computational Efficiency:
Graph-building calls take just 3.21s. Dataset creation needs 240.02s for 237MB.

Overcoming noisiness in synthetic logs

- Generate synthetic logs for each of 1000 “machines” with a **varying number of variables (V in 10-1000).**
- Have V-3 of the variables take a **random value between 0-100** each time they appear.
- Set 3 special variables x,y,z such that **z confounds the effect of x on y.**
- Add Gaussian noise when drawing x and y, with a **varying standard deviation (1-10).**

Outcome:
Even when the log is maximally noisy,
Sawmill helps uncover and address confounding

Accuracy:
Mean MRR is 0.6296 and we recover the ATE with mean error 28.83% (47.88% for baseline)

Human Efficiency:
Graph building requires only 5 user calls.
Dataset creation pipeline needs another 4.

Computational Efficiency:
Graph-building calls take just 5.81s.
Dataset creation needs ≤19.56min. for ≤66MB.

Let’s chat in the poster session!

SIGMOD 2024 Demo Paper

https://people.csail.mit.edu/markakis/papers/2024_Sawmill_demo.pdf

Open-Source Implementation

https://github.com/mitdbg/sawmill