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:

```sql
WITH sur AS
    (SELECT s_store_id as store_id,
        sum(s_net_sales_price) as total_sales,
        sum(coalesce(sr_net_loss, 0)) as total_loss
     FROM store_sales
     WHERE store_id = s_store_id
     GROUP BY store_id)

SELECT s_store_id as store_id,
    total_sales, total_loss
FROM sur
```
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:
  - deviation_mean
  - Explore Candidate Causes
  - Candidate cause(s) found!

**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_mean mean
  - max_parallel_workers_mean
- Destination node:
  - max_parallel_workers_mean...

- Accept
- Reject
- Reject Undecided Outgoing from Source
- Reject Undecided Incoming to Destination
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

- Parsing and Tagging
  - Use Drain for initial log parsing and GPT-3.5/GPT-4 to provide a human-understandable tag to each variable.

- Defining Causal Units
  - Group log information based on user’s ATE of interest.

- Computing Suitable Variables
  - Aggregate log contents per causal unit, picking the function that maximizes empirical entropy.

- Obtaining a Causal Graph
  - Interactively create the necessary part of the causal graph.

- Answering ATE queries
  - Prepared Causal Graph

- Obtaining A Causal Graph
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).

**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.

**Outcome:**
Even when the log is maximally noisy, Sawmill helps uncover and address confounding.
Let’s chat in the poster session!

SIGMOD 2024 Demo Paper

Open-Source Implementation


https://github.com/mitdbg/sawmill