Sawmill: From Logs to Causal Diagnosis of Large Systems

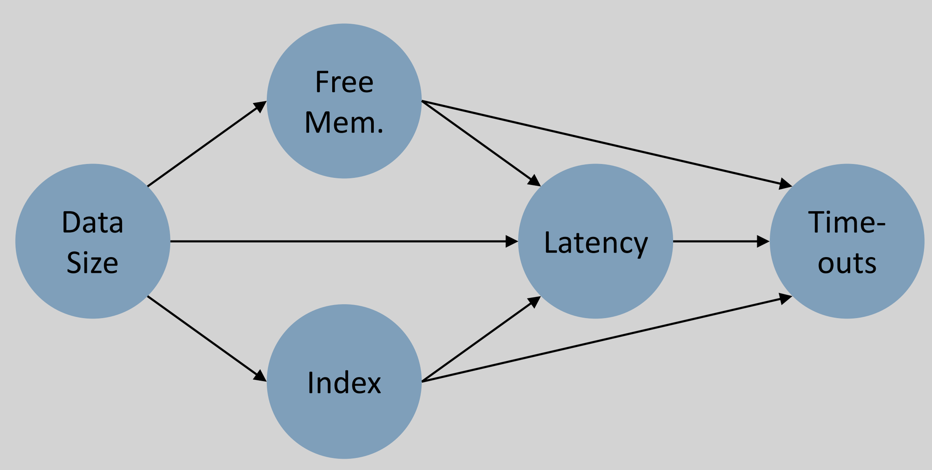
Markos Markakis, An Bo Chen, Brit Youngmann, Trinity Gao, Ziyu Zhang,
Rana Shahout, Peter Baile Chen, Chunwei Liu, Ibrahim Sabek, Michael Cafarella

MIT CSAIL

Causal Analysis on Logs Faces 3 Key Challenges

```
20:24:44 INFO u0 q34 Running CREATE INDEX midx ON metrics (id);
20:32:25 INFO u0 q35 Running SELECT * FROM metrics WHERE id=562;
20:32:26 INFO u0 q35 Ran in 607.31ms
20:32:28 INFO u0 q36 Running SELECT * FROM metrics WHERE id=555;
20:33:28 INFO u0 q36 Query timed out
```

User	M: Free Memory	I:Index Presence	L:Latency Mean (ms)	D: Data Size (GB)	T: Timeouts per day
U_0	67.80 %	1	637.02	64.41	56852
U_1	80.96 %	0	372.60	38.07	29164
...	...	...	...	...	...



Causal Analysis Can Help System Understanding

- Failures are a daily phenomenon when operating large complex systems.
- Diagnosing system problems quickly and correctly is crucial for operators.
- Causal reasoning [1] has helped scientists across domains pose, discuss and test hypotheses.

Applying Causality to Log Data is Challenging

- Pearl's framework [1] requires tabular data and a causal graph for the problem.
- However, operators often only have access to textual logs.
- Three key challenges:
  - **Challenge A – Deriving the Schema:** Logs can be parsed, but this can lead to hundreds of unlabeled variables that are hard for a human to manually label.
  - **Challenge B – Distilling the Data:** Logs contain a lot of fine-grained data. What is the best way to summarize it along the units the user cares about (e.g. machines)?
  - **Challenge C – Obtaining the Causal Graph:** The scale and dependencies in log data make automatic causal discovery [2,3] challenging. Can we tap the user's expertise intelligently?

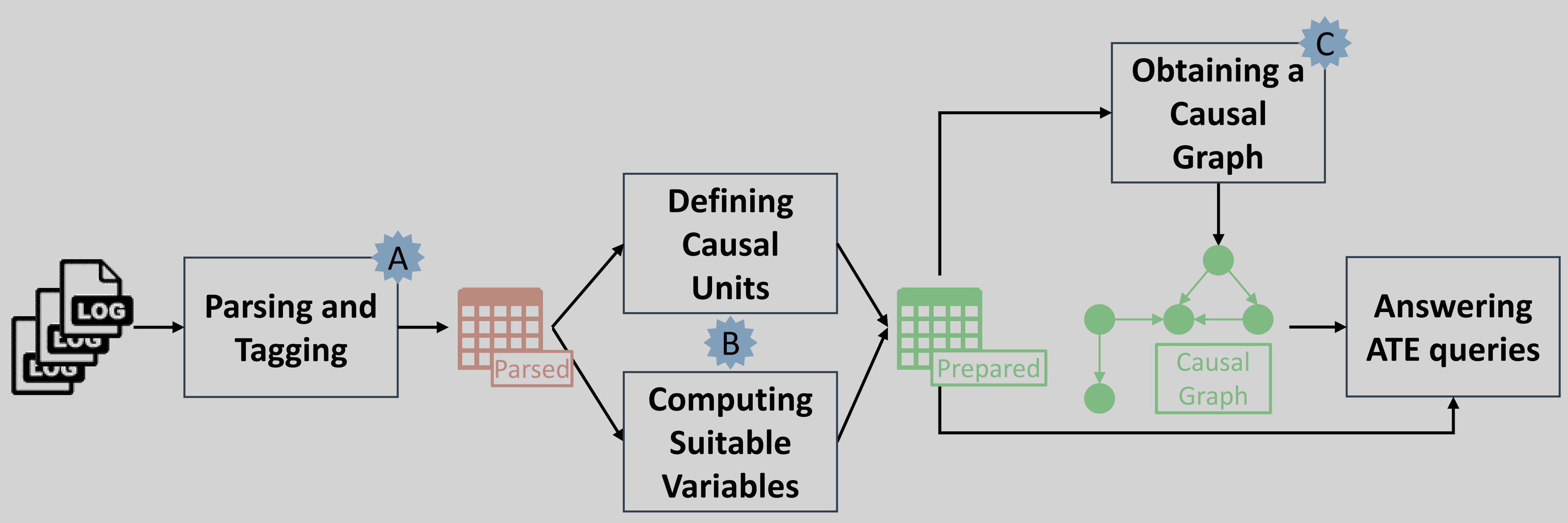
Sawmill Provides Solutions for Each Challenge

Challenge B: Aggregate Data to Maximize Entropy

- Log information can be too granular for meaningful analysis.
- Users can define **causal units** over which to aggregate log information, depending on the context (e.g. machines, users etc.).
- The best aggregate to pick for each variable can be unclear a priori.
- We pick the aggregate that **maximizes empirical entropy** between the causal units, in order to maximize downstream usefulness.

Challenge A: Utilize LLMs for Tagging

- Use Drain [4] to create the unlabeled *parsed table.*
- Leverage GPT-4 [5] to assign human-understandable tags to each variable.



Challenge C: Leverage Exploration-based Causal Discovery

- Algorithmic causal discovery faces challenges due to dependencies.
- Hand-crafting a full causal model is also daunting, but only part of it is needed.
- Use a human in the loop:
  - User provides a **variable of interest.**
  - Sawmill suggests **candidate causes.**
  - User evaluates them and revises graph.
  - The process repeats while increasing the **exploration score.**

With a Handful of User Interactions, Sawmill Uncovers Highly Accurate Effects

Dataset			True ATE	Sawmill ATE	Regression ATE	AskGPT ATE
PROPRIETARY (D2)	F=0.5	p_f=1.0	258.43	**257.47**	273.01	0.00
		p_f=0.5	114.86	**112.66**	118.04	112.66
		p_f=0.2	28.71	**27.28**	25.94	27.28
	F=0.1	p_f=1.0	258.43	**258.64**	256.01	0.00
		p_f=0.5	114.86	121.45	**119.38**	0.00
		p_f=0.2	28.71	**33.98**	35.30	0.00
	F=0.01	p_f=1.0	258.43	**258.57**	264.50	258.57
		p_f=0.5	114.86	**85.66**	84.79	0.0
		p_f=0.2	28.71	**42.64**	45.18	0.0
XYZ (D3)	V=10	R=1	2.00	**2.00**	2.00	2.00
		R=5	2.00	2.11	**2.10**	2.11
		R=10	2.00	1.97	**1.98**	1.97
	V=100	R=1	2.00	**1.96**	1.95	2.36
		R=5	2.00	**1.60**	1.58	0.00
		R=10	2.00	0.87	0.86	**0.97**
	V=1000	R=1	2.00	**1.78**	0.37	0.00
		R=5	2.00	**0.62**	-1.61	0.62
		R=10	2.00	0.12	**0.35**	0.00
Mean % Error on PROPRIETARY				**11.72%**	14.64%	67.44%
Mean % Error on XYZ				**28.83%**	47.88%	49.50%
Mean % Error				**20.27%**	31.26%	58.47%

Dataset			Sawmill MRR	Regression MRR	AskGPT MRR
POSTGRESQL (D1)			**0.5667**	0.0476	0.4815
PROPRIETARY (D2)	F=0.5	p_f=1.0	**1.0000**	1.0000	0.0000
		p_f=0.5	**1.0000**	1.0000	0.3333
		p_f=0.2	**1.0000**	1.0000	1.0000
	F=0.1	p_f=1.0	**1.0000**	1.0000	0.0000
		p_f=0.5	**1.0000**	1.0000	0.0000
		p_f=0.2	**1.0000**	1.0000	0.0000
	F=0.01	p_f=1.0	**1.0000**	1.0000	0.0714
		p_f=0.5	**1.0000**	0.0667	0.0000
		p_f=0.2	**1.0000**	0.0667	0.0000
XYZ (D3)	V=10	R=1	**0.6667**	0.6667	0.4007
		R=5	0.6111	0.5556	**0.6667**
		R=10	**0.6667**	0.5833	0.0664
	V=100	R=1	**0.6667**	0.5476	0.5000
		R=5	**0.6667**	0.5370	0.0000
		R=10	0.3889	**0.6667**	0.5000
	V=1000	R=1	**0.6667**	0.0000	0.1667
		R=5	0.6667	0.0000	**0.8333**
		R=10	**0.6667**	0.0000	0.1667
Mean on PROPRIETARY			**1.0000**	0.7926	0.1561
Mean on XYZ			**0.6296**	0.3952	0.3667
Mean			**0.8018**	0.5651	0.2730

Dataset			PARSE (s)	PREPARE (s)	EXPLORECANDIDATECAUSES (s)	**Total Time (s)**	Total Time over Log Size (s/MiB)
POSTGRESQL (D1)			37.65	4.41	4.85	46.91	2.39
PROPRIETARY (D2)	F=0.5	p_f=1.0	189.19	48.58	2.62	240.39	1.06
		p_f=0.5	189.51	48.83	3.20	241.54	1.07
		p_f=0.2	195.58	49.32	3.18	248.08	1.10
	F=0.1	p_f=1.0	194.50	49.11	3.22	246.83	1.09
		p_f=0.5	189.53	49.18	3.24	241.95	1.07
		p_f=0.2	189.58	49.50	3.28	242.36	1.07
	F=0.01	p_f=1.0	190.70	49.99	3.31	244.00	1.08
		p_f=0.5	187.09	49.47	3.38	239.94	1.06
		p_f=0.2	191.18	49.30	3.43	243.91	1.08
XYZ (D3)	V=10	R=1	72.33	4.26	2.40	78.99	1.32
		R=5	80.04	5.27	2.98	88.29	1.48
		R=10	79.69	4.70	3.09	87.48	1.47
	V=100	R=1	145.67	33.24	3.59	182.50	2.95
		R=5	144.14	33.87	3.87	181.88	2.94
		R=10	144.67	33.79	3.97	182.43	2.95
	V=1000	R=1	853.14	326.83	8.38	1188.35	18.90
		R=5	822.43	334.47	8.91	1165.81	18.54
		R=10	849.08	335.59	15.09	1199.76	19.08
Mean			260.30	82.09	4.53	346.92	4.30
Breakdown (%)			75.03%	23.66%	1.30%		

Dataset	System	PARSE	SEPARATE	SETCAUSALUNIT	PREPARE	EXPLORE CANDIDATECAUSES	ACCEPT	GETATE	REGRESS	GPT-EXPLORE CANDIDATECAUSES	**Total**
POSTGRESQL (D1)	Sawmill	1	1	1	1	2	3	1	0	0	10
	Regression	1	1	1	1	0	0	0	1	0	**5**
	AskGPT	1	1	1	1	0	3	1	0	2	10
PROPRIETARY (D2)	Sawmill	1	0	1	1	1	1	1	0	0	6
	Regression	1	0	1	1	0	0	0	1	0	**4**
	AskGPT	1	0	1	1	0	1	1	0	1	6
XYZ (D3)	Sawmill	1	0	1	1	2	3	1	0	0	9
	Regression	1	0	1	1	0	0	0	1	0	**4**
	AskGPT	1	0	1	1	0	3	1	0	2	9

D1 Battling Confounding in Real-World Logs

- Collect logs from **PostgreSQL running TPC-DS**.
- **Vary performance-affecting parameters** and **bias parameter combinations** to trade off work_mem and max_parallel_workers.
- Ignoring bias makes **mean latency increase for more parallelism.**

D2 Discerning Subtle Semi-Synthetic Effects

- Start with **real logs from a mobile application** and generate similar logs for 1000 users. Label 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.**

D3 Overcoming Noisiness in Synthetic Logs

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

[1] Judea Pearl. 2009. Causality: Models, Reasoning and Inference. Cambridge University Press.
[2] Clark Glymour, Kun Zhang, and Peter Spirtes. 2019. Review of causal discovery methods based on graphical models. Frontiers in genetics 10 (2019), 524
[3] Peter Spirtes, Clark N Glymour, and Richard Scheines. 2000. Causation, prediction, and search. MIT press
[4] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. 2017. Drain: An Online Log Parsing Approach with Fixed Depth Tree. In 2017 IEEE International Conference on Web Services (ICWS). 33–40. https://doi.org/10.1109/ICWS.2017.13
[5] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]