Learning to localize faults using fault injection

Speakers:

Jesus Rios, Karthik Shanmugam, Qing Wang

IBM Research

Jesus Rios

Agenda

- Our approach to fault localization
- Fault Injection based causal learning
- Implementation and experiment results

Background

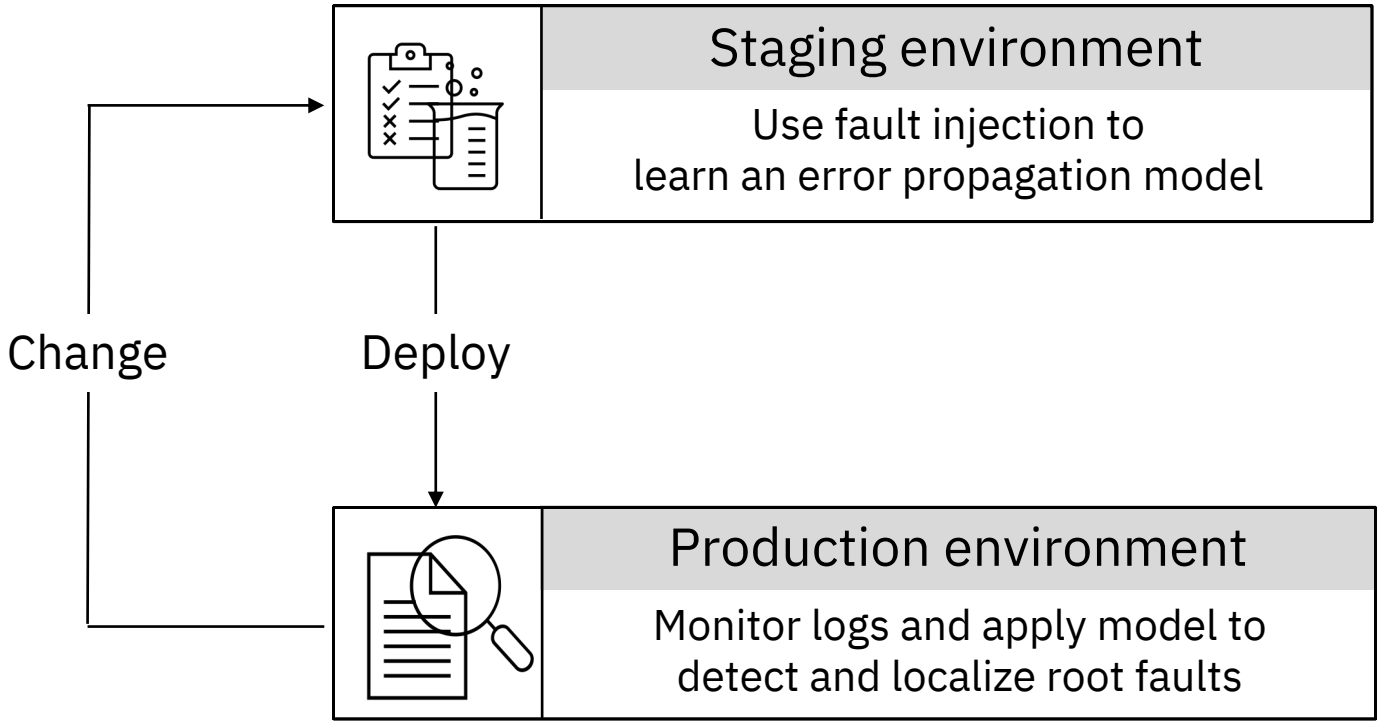
In practice

- many large distributed applications suffer from limited observability
- making very difficult to quickly find the exact location of a fault

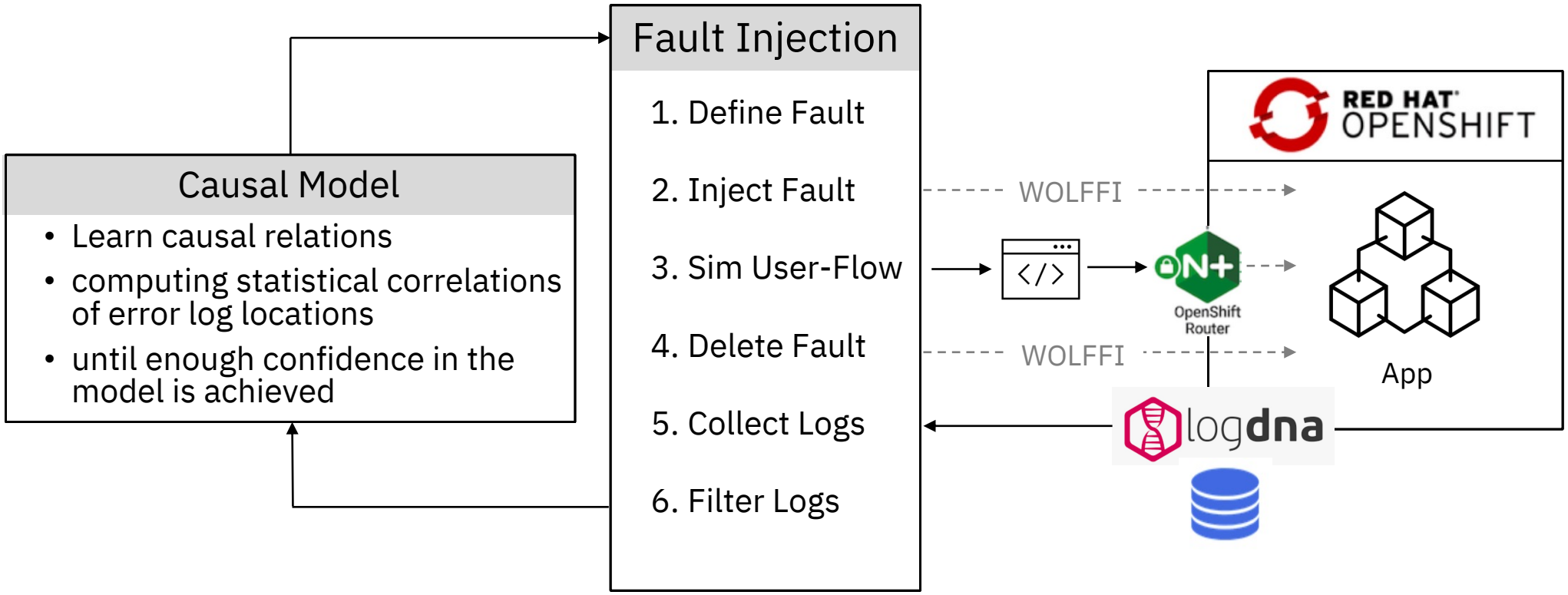
We propose

- using interventional causal learning
  - interventions = fault injections
- to automate the task of localizing the root cause of a fault
- using only application logs data

Two-Phase Cyclic Approach to Fault Localization



Learning Causal Model from Fault Injections



Empirical Demonstration

Benchmark applications

- Train-Ticket
  - user-flow simulation covered 33 of the its 41 microservices
- Day-Trader
  - 5 microservices all covered by user-flow
- Observability assumption
  - access *only* to microservice logs

Performance measures

- Learning accuracy
  - Learnt causal model vs. true underlying graph
- Localization of new injected faults
  - Accuracy
    - estimate contains true fault location
  - Informativeness -> estimate set size

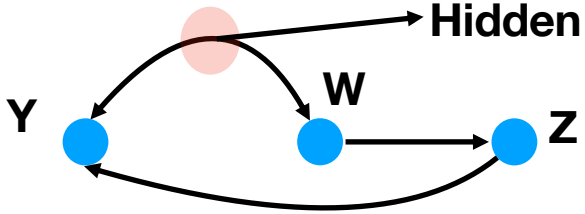
    100% -> 1

    0% -> max

Fault Injection based Causal Learning

Karthikeyan Shanmugam
IBM Research, NY

Active Learning

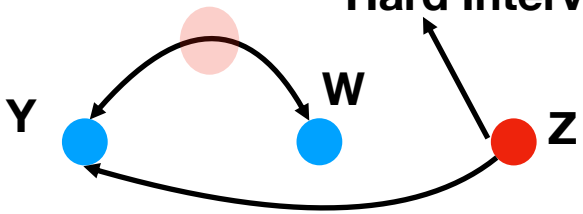


Causal Graph G with
Latent variables

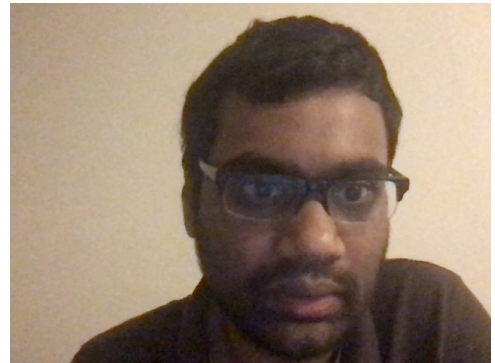
Observational Data

What is the best set of
node
interventions
to know
Causal Graph G ?

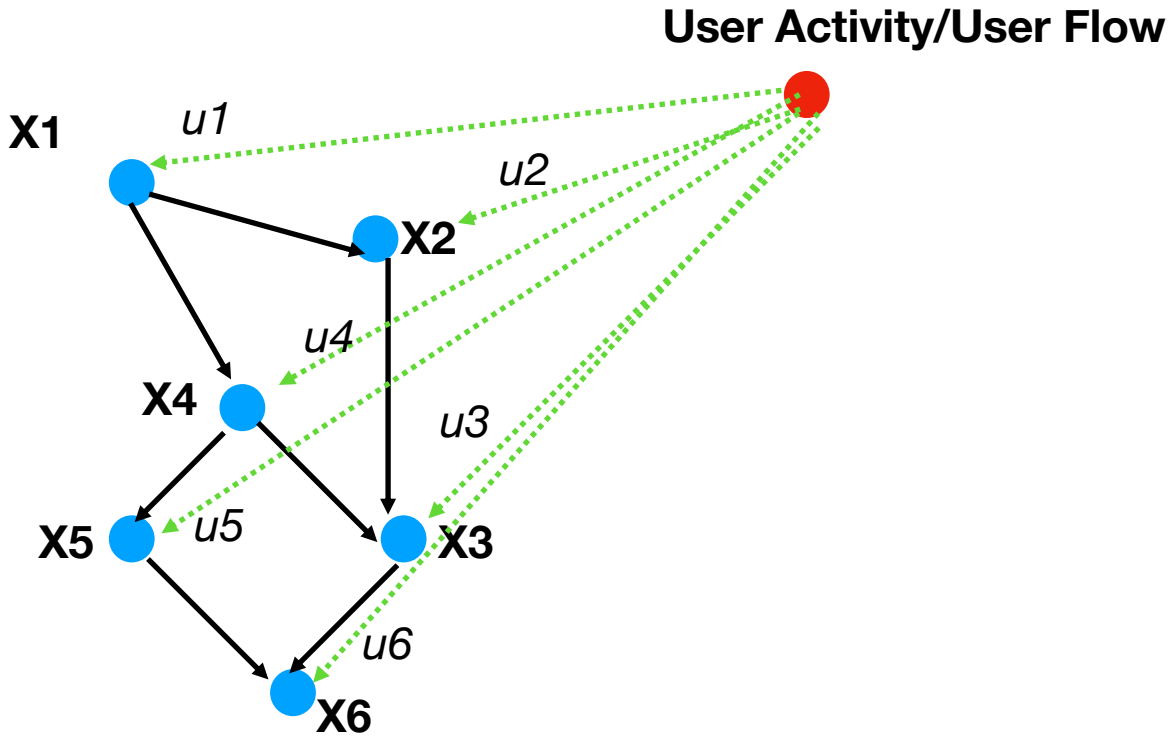
Hard Intervention



Interventional Data



Micro Services and UserFlow Model



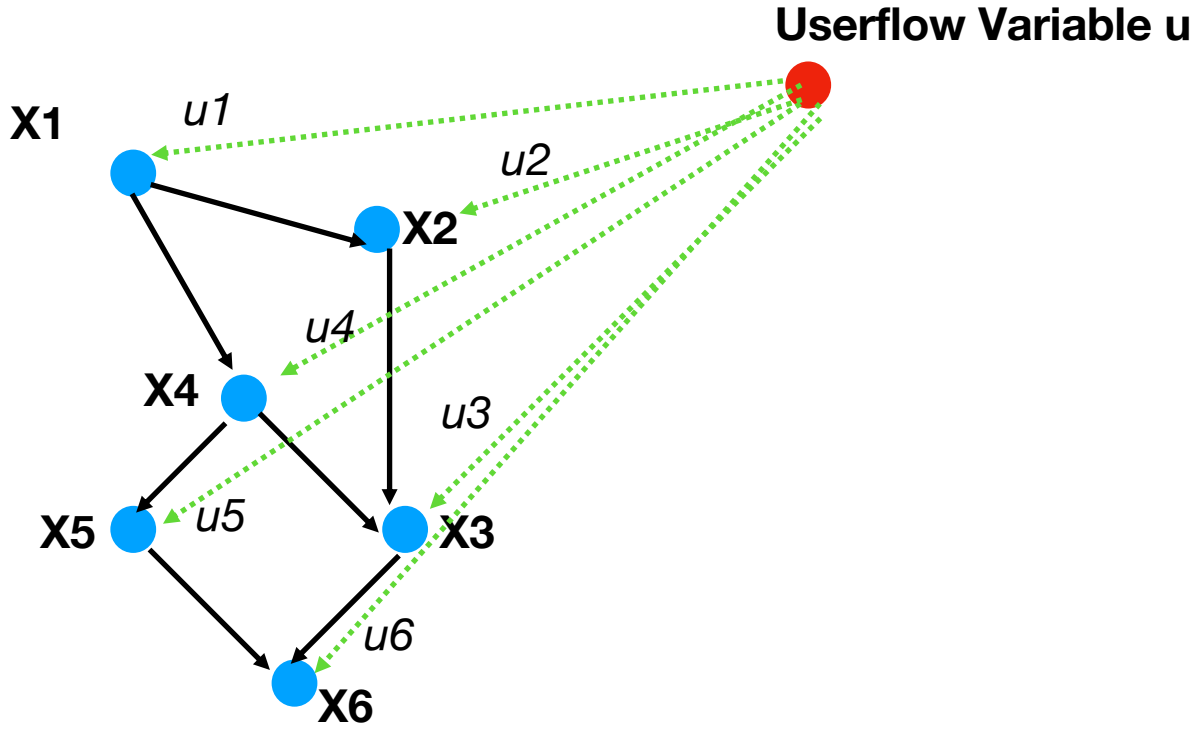
Example: x_6 - Variable that counts error logs of micro service 6

$$x_6 = f_6(x_5, x_3, u_6)$$

Error counts x_6 is a function of:

a) whether services 5 and 4 throw an error or not, and
b) whether user flow touched service 6 along with 3 or 5 or both

Micro Services and Userflow Model

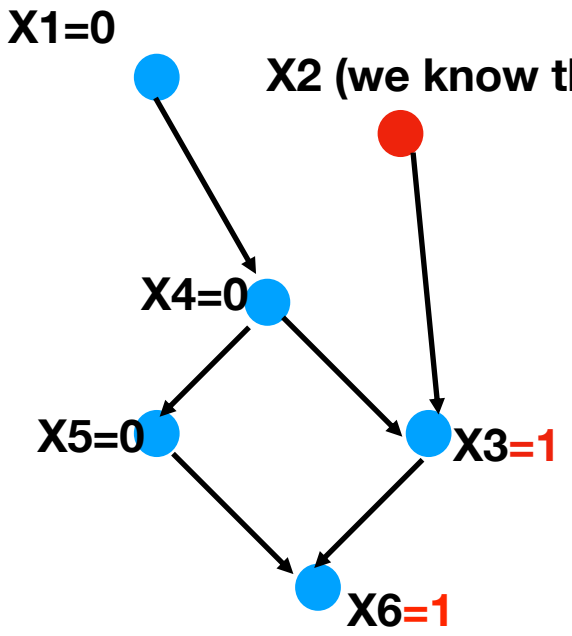


P(u) - Distribution of user flows

In general, its a Pearlian Structural Causal Model where noise variables confound multiple equations

Micro Services and Userflow Model

Intervention on X2 - stopping the service



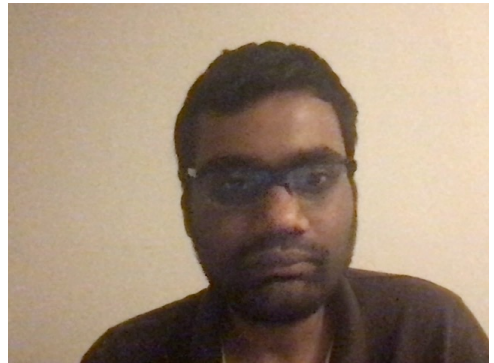
**If user flow distribution is generic,
it should roughly make services both 6 and 3
throw an error**

**E[X3], E[X6] are good simple tests
that identifies ALL descendants**

Algorithm for Finding Ancestral Graph

Condition: A Correlation test (X,Y) has sufficient strength iff X is a descendant of Y (an intervened variable)

Lemma 1. *[Pairwise Conditional Independence Test] Consider a causal graph with latents D_ℓ. Consider an intervention on the set $S \subset \mathcal{V}$ of observable variables. Then, under the post-interventional faithfulness assumption, for any pair $X_i \in S, X_j \in \mathcal{V} \backslash S$, $(X_i \not\perp\!\!\!\perp X_j)_{D_\ell[S]}$ if and only if X_i is an ancestor of X_j in the post-interventional observable graph $D[S]$.*



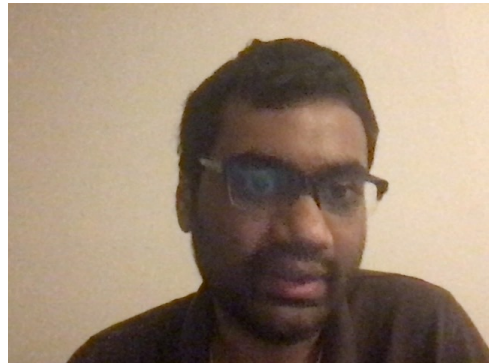
Algorithm for Finding Ancestral Graph

Condition: A Correlation test (X,Y) has sufficient strength iff X is a descendant of Y (an intervened variable)

Lemma 1. *[Pairwise Conditional Independence Test] Consider a causal graph with latents D_ℓ. Consider an intervention on the set $S \subset \mathcal{V}$ of observable variables. Then, under the post-interventional faithfulness assumption, for any pair $X_i \in S, X_j \in \mathcal{V} \backslash S$, $(X_i \not\perp\!\!\!\perp X_j)_{D_\ell[S]}$ if and only if X_i is an ancestor of X_j in the post-interventional observable graph $D[S]$.*

Algorithm 1 LearnAncestralRelations- Given access to a conditional independence testing oracle (CI oracle), query access to samples from any post-interventional causal model derived out of $\mathcal{M}$ (with causal graph D_ℓ), outputs all ancestral relationships between observable variables, i.e., D_{tc}

function LEARNANCESTRALRELATIONS($\mathcal{M}$)
  $E = \emptyset$.
  Consider a strongly sep. system of size $\leq 2 \log n$ on the ground set $\mathcal{V}$ - $\{S_1, S_2..S_{2\lceil \log n \rceil}\}$.
  for i in $[1 : 2\lceil \log n \rceil]$ **do**
    Intervene on the set S_i of nodes.
    for $X \in S_i,\ Y \notin S_i,\ Y \in \mathcal{V}$ **do**
      Use samples from $\mathcal{M}_{S_i}$ and use the CI-oracle to test the following.
      if $(X \not\perp\!\!\!\perp Y)_{D_\ell[S]}$ **then**
        $E \leftarrow E \cup (X, Y)$.
      end if
    end for
  end for
  return The transitive closure of the graph $(\mathcal{V}, E)$
end function



Algorithm for Finding Ancestral Graph

For single node Fault injections, Condition 1 is trivially true.
Injecting Faults in every node will identify all ancestral relationships

.

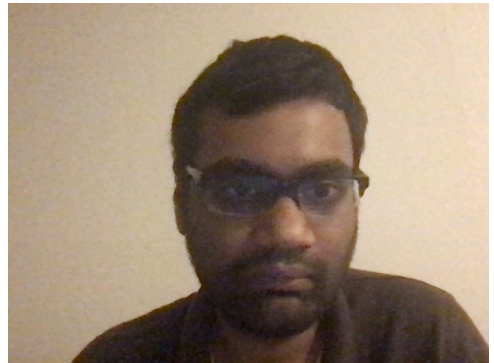
Algorithm for Finding Ancestral Graph

For single node Fault injections, Condition 1 is trivially true.
Injecting Faults in every node will identify all ancestral relationships

The graph we return: Transitive Reduction
preserves ancestry AND they are causal if correlation tests
are accurate. But we don't return all edges.

References (the above algorithm of finding the ancestral graph is common to all these references first introduced in Ref 1)

1) Murat Kocaoglu, Karthikeyan Shanmugam, and Elias Bareinboim. "Experimental design for learning causal graphs with latent variables." NeurIPS 2017.
2) Raghavendra Addanki, Andrew McGregor, and Cameron Musco. "Intervention Efficient Algorithms for Approximate Learning of Causal Graphs." Arxiv (2020).
3) Bello, Kevin, and Jean Honorio. "Computationally and statistically efficient learning of causal Bayes nets using path queries." *NeurIPS 2018*.



Experiments

Set-up

- Two benchmark microservice applications are deployed on OpenShift Container Platform (OCP).
  - DayTrader: an online stock trading system.
  - TrainTicket: a train ticket booking system.
- logDNA service is installed in the OCP cluster to collect logs.

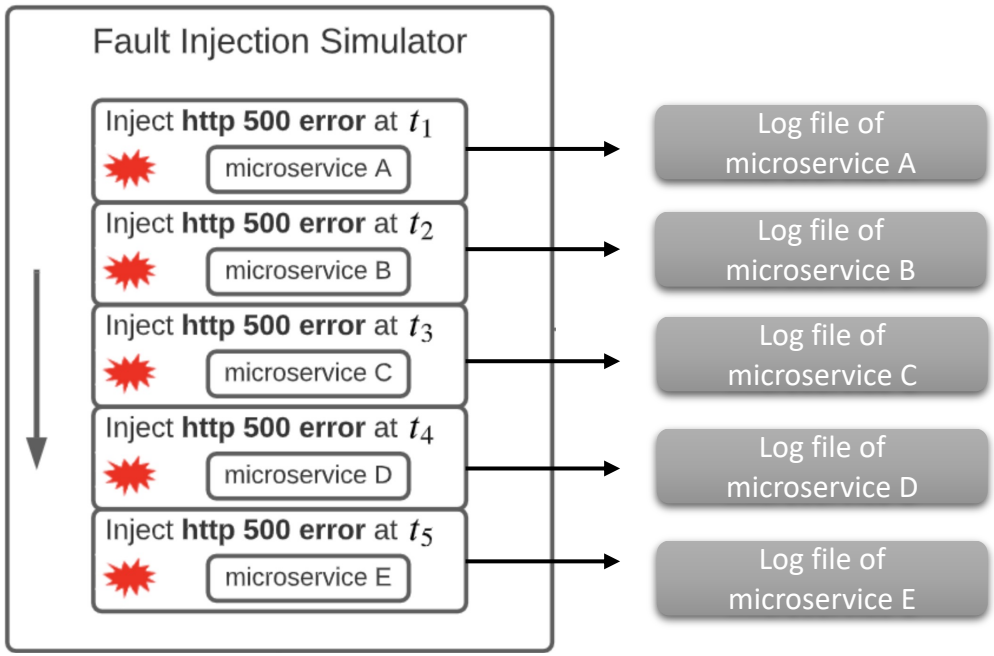
Fault Injection

- User-flow simulation
  - DayTrader
    - All five microservices can be covered by the simulated user-flow.
  - TrainTicket
    - 31 of all 41 microservices are covered.
- Fault injection framework
  - Developed in Python
  - Fault types: Http faults, latency and saturation faults, etc.

Experiments

Log Data Generation

- For both two applications, we inject one fault at a time and run the user-flow.



Log Data Preprocessing

- Label each log line by defined error patterns (e.g., “http 500 internal server error”)
  - Error logs
  - Normal logs

Log File of microservice C

_time	_line	…	_level	_app	_container
t	http 500	…	error	B	B
…	…	…	info	…	…
T	http 500	…	error	A	A

Table 1. The observed log data by injecting fault to microservice C. Microservice A and B are dependent on C.

Experiments

Causal Learning Performance

- Evaluation metrics:
  - Precision, recall, F1-score and SHD (structural hamming distance).
  - τ and bin_size

Application	τ	SHD	Precision	Recall	F_1
DayTrader	0.01	0	1.00	1.00	1.00
	0.03	0	1.00	1.00	1.00
	0.1	0	1.00	1.00	1.00
	0.3	2	0.75	0.75	0.75
	0.4	4	0.50	0.50	0.50
TrainTicket	0.01	36	0.68	0.54	0.60
	0.03	**33**	**0.73**	**0.54**	**0.62**
	0.1	37	0.71	0.44	0.54
	0.3	52	0.40	0.08	0.13
	0.4	51	0.33	0.22	0.04

Table 2. Fault Injection Causal Learning results by comparing with transitive reduction of ground truth with the setting bin_size=1s.

Fault Localization Performance

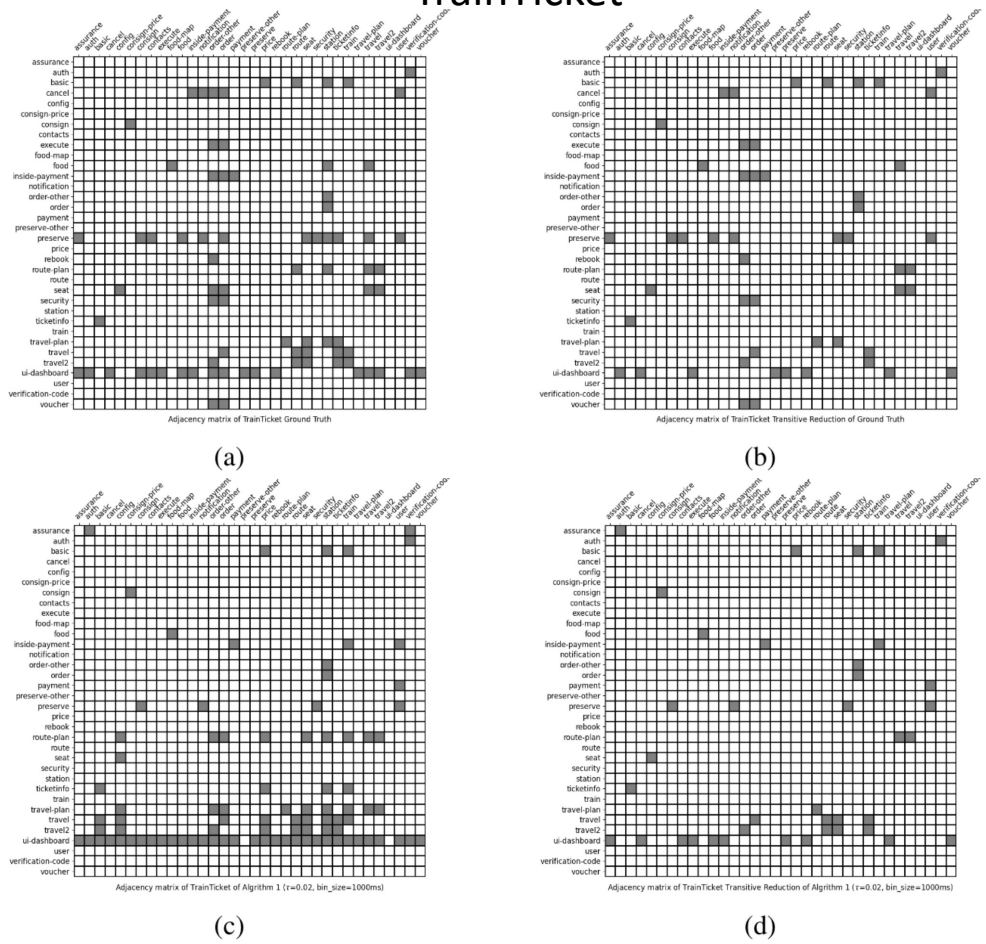
- We build the ancestral relationships using the causal learning results in the error propagation for localizing a new fault.

τ	Accuracy (%)	Informativeness (%)
0.03	**93.75**	**86.93**
0.1	90.63	79.55
0.4	15.63	21.69

Table 3. Fault localization performance on **TrainTicket** using the output of Algorithm 1 with different τ values and the fixed bin size $= 1,000$ ms in terms of Accuracy and Informativeness.

Experiments

TrainTicket

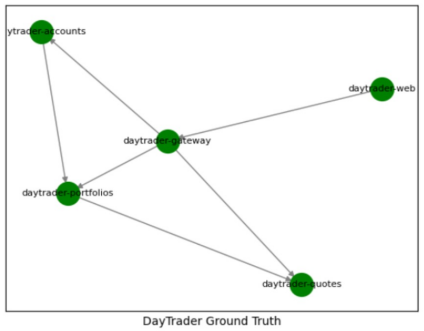


e.g. ui-dashboard -> ticketinfo

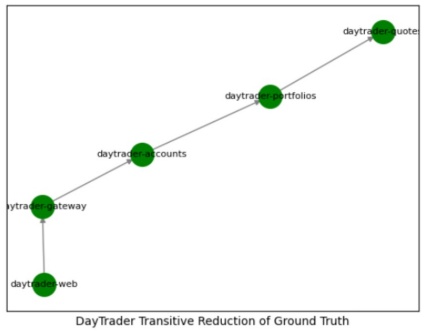
Figure 5. (a) Adjacency matrix of ground truth; (b) Adjacency matrix of transitive reduction (TR) of ground truth; (c) Adjacency matrix of Algorithm 1 (τ=0.03,bin_size=1,000ms) before Step 12; (d) Adjacency matrix of Algorithm 1 (τ=0.03,bin_size=1,000ms).

Experiments

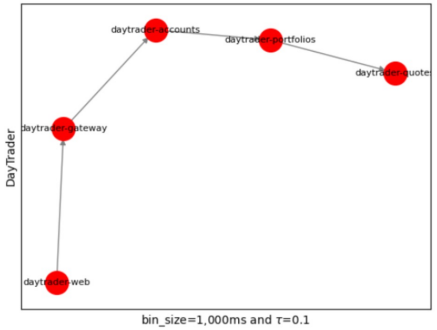
DayTrader



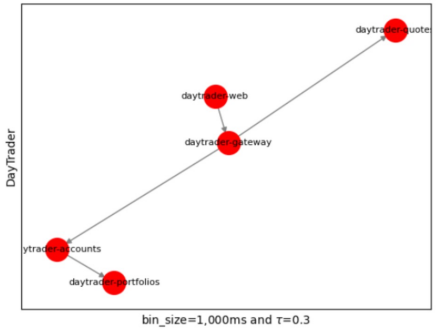
(a) Ground Truth.



(b) TR of Ground Truth.



(g) (τ=0.1,bin_size=1,000ms)



(h) (τ=0.3,bin_size=1,000ms)

Figure 4. (a) and (b) are ground truth graph and transitive reduction of it. (g) and (h) are the outputs of fault injection causal learning results with different τ values and the best bin_size=1s.

Thank you

Jesus Rios
Research Scientist
IBM Research AI
jriosal@us.ibm.com

Karthik Shanmugam
Research Scientist
IBM Research AI

Qing Wang
Postdoctoral Researcher
AI for IT Ops - Hybrid Cloud
IBM Research

Laura Shwartz
Distinguished Engineer
AI for IT Ops - Hybrid Cloud
IBM Research

Naoki Abe
Distinguished Research Scientist
IBM Research AI