# Revelio: ML-Generated Debugging Queries for Finding Root Causes in Distributed Systems

Pradeep Dogga

UCLA

# Collaborators

Karthik Narasimhan†

Anirudh Sivaraman‡

Shiv Kumar Saini*

**George Varghese**δ

**Ravi Netravali**†

† PRINCETON UNIVERSITY

‡ NYU

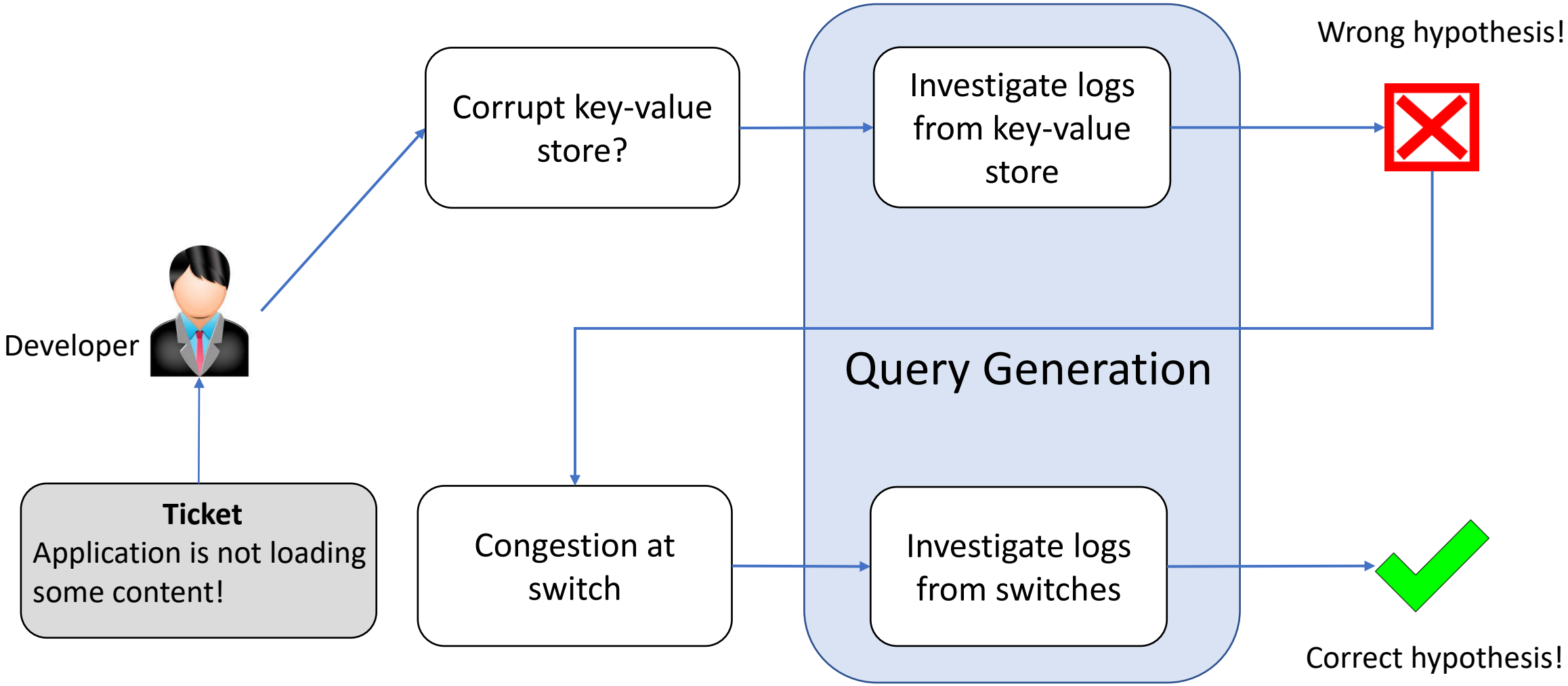
*

δ UCLA

# What Revelio is about

- **Problem:** Difficult to understand **which** debugging tool to use, **when** and **how** (i.e, which debugging query?)
- **Solution:** Leverage patterns in historical debugging data to *auto generate debugging queries* using ML
- **Ideas:** Leverage unstructured reports and structured logs, modularity and abstraction (e.g., stability in rank ordering)
- **Opensource Testbed:** Enables debugging experiments and data collection by others

# Today's Root Cause Diagnosis – Painful and buggy




Largely manual and error-prone

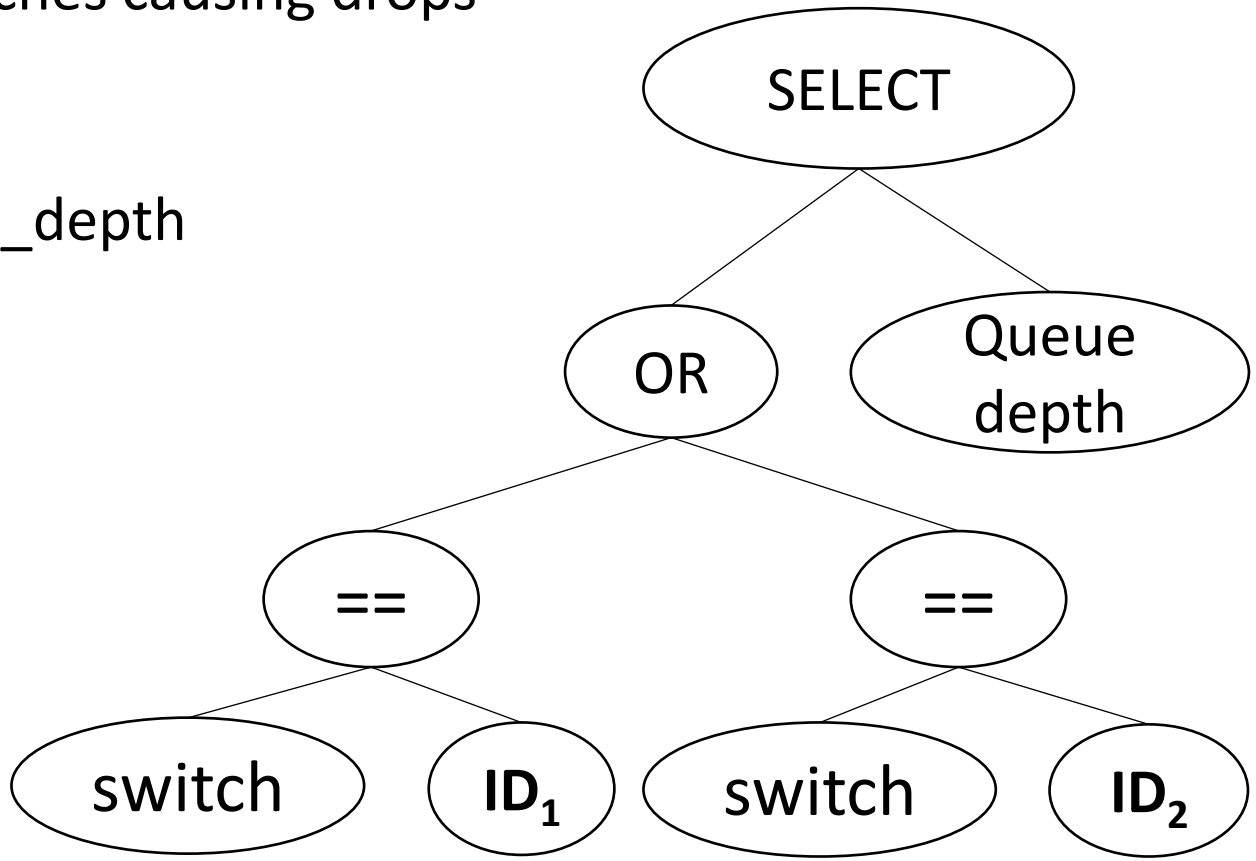
# Debugging Query

Query issued on logs to validate root cause hypothesis for debugging.

**Example hypothesis:** “Congestion at switches causing drops”

**Debugging Query:**

```
SELECT * FROM Queue_depth
WHERE
switch = ID1 OR
switch = ID2
```



## Debugging remains difficult

- Not lack of tools
  - Too many! - Modularity, features, expenses, etc.
  - Painful to learn using all these tools.
- Large search space of hypotheses
  - Which subsystem and which metrics to investigate?

# Tons of logs everywhere!

Distributed tracing at Pinterest wit
new open source tools

Pinterest Engineering Follow
Feb 14, 2017 · 7 min read

**The New Stack**
@thenewstack

When you find a new type of performance issue, the temptation is to add a new set of metrics to a dashboard. Most of the time this is not a good idea. Overly busy dashboards can quickly lead to cognitive overload — Google's @lizthegrey on #microservices debugging #qconnyc

**The New Stack**
@thenewstack

To monitor its thousands of services, Facebook captures about billion traces a day (about ~100TB collected), a dynamic sampling of the total number of interactions per day — @Facebook's Haozhe Gao and Joe O'Neill #QConNYC

Ticketmaster Traces 100 Million Transactions per Day with Jaeger

Orate Follow
May 6 · 6 min read

## Developer Survey

- Scope: 7 services, serve 83 million requests per day.
- **Multiple Tools used:** Splunk, Datadog, CloudWatch, Lightstep, New Relic, Pingdom, Icinga, etc.
- **Alerting Monitors:** Hard to maintain, neglected due to false alerts.

## Manual Ticket Analysis

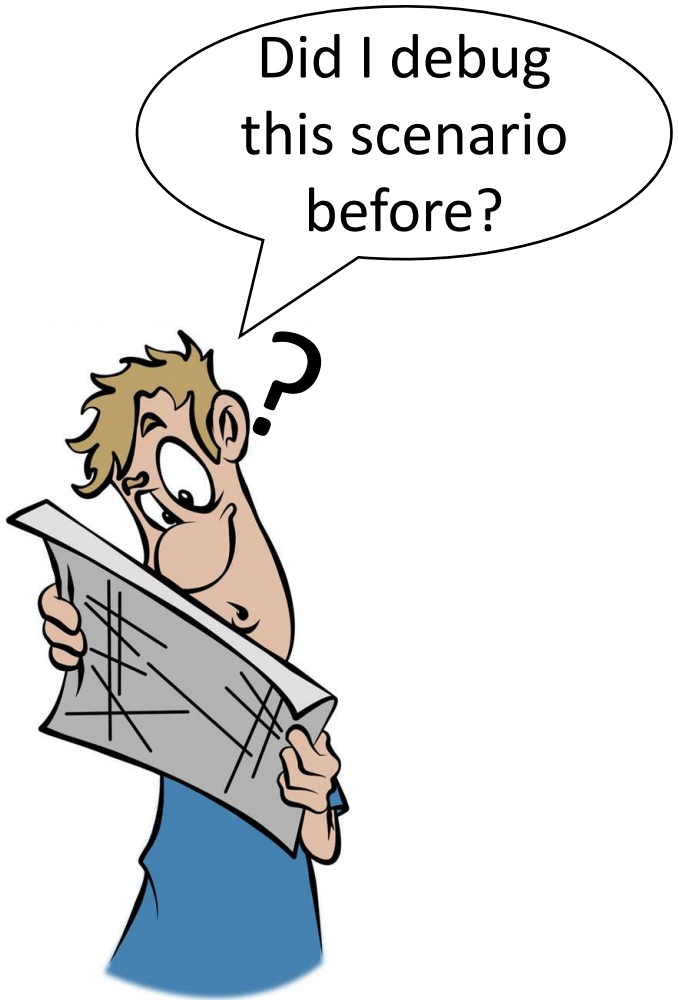
- Scope: 176 tickets (4-month period).

- 94% of the root-causes could be clustered to 7 categories

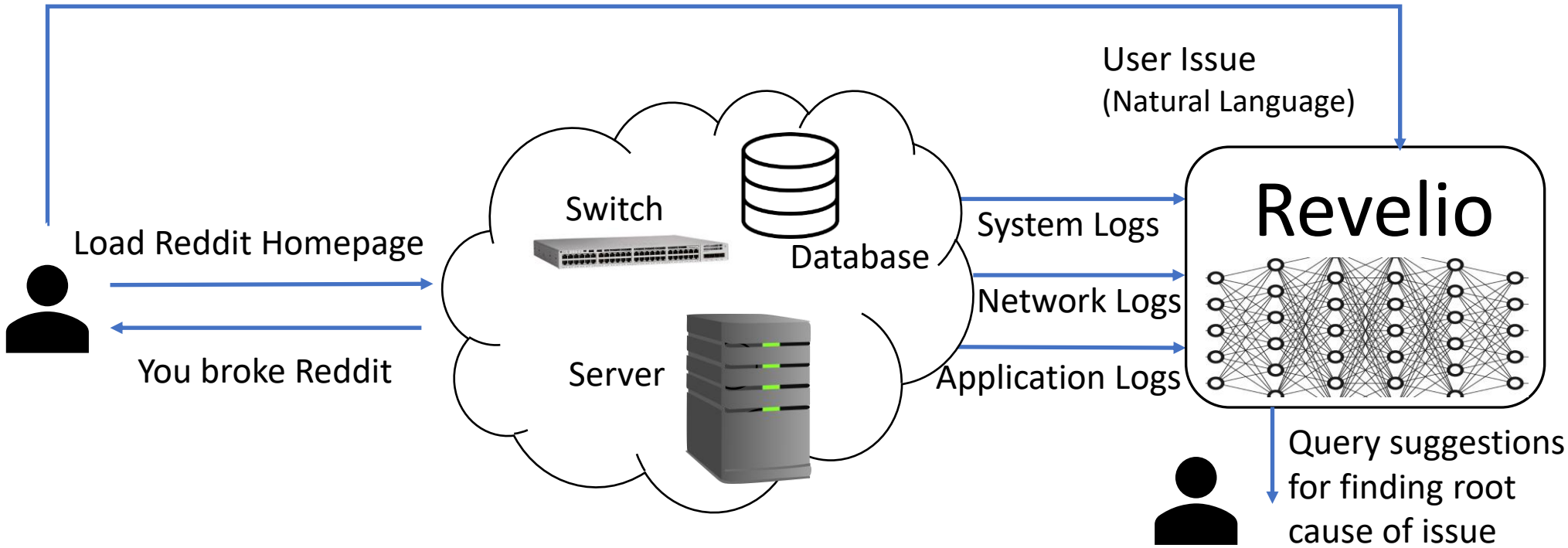
- Resource under-provisioning
- Component failures
- Subsystem misconfigurations
- Network-level misconfigurations
- Network congestion
- Source-code bugs
- Incorrect data exchange

## Takeaways

- **Hard-won** debugging intuitions guide developers.
- Debugging queries are used to interface with **multiple tools**.
- But bottleneck remains which tool and which query parameters to use.
- But, root-causes of several bugs share **common** characteristics. Leverage?

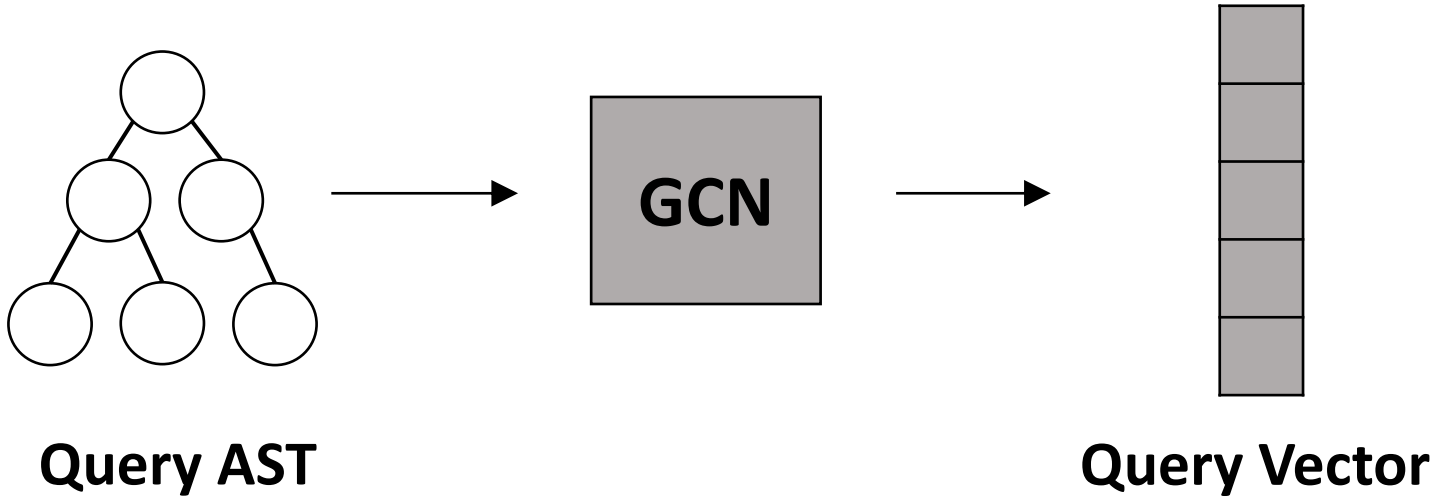


# Revelio: ML-Generated Debugging Queries



# Challenges – Predicting Queries

- Highly structured output space.
- **Solution**: Leverage the inherent tree structure of queries using GCNs.



# Challenges – Scaling

- Large search space of queries.
- **Solution:** Leverage *Modularity* – decompose query to a template and parameters that fill the template.

# Modularity – Template and Parameters

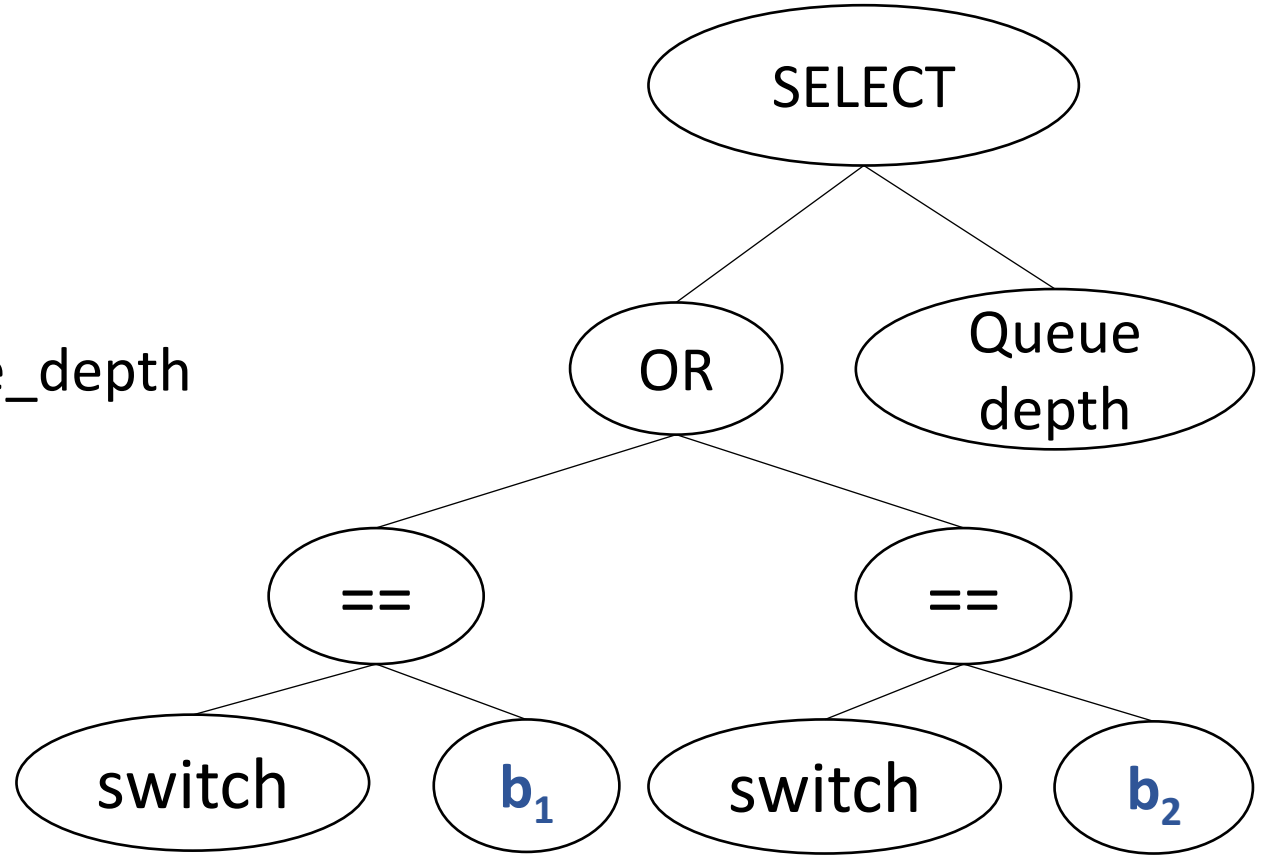
**Debugging Query:**

```
SELECT * FROM Queue_depth
WHERE
switch = ID1 OR
switch = ID2
```

**Template:**

```
SELECT * FROM Queue_depth
WHERE
switch = b1 OR
switch = b2
```

**Parameters:** ID1, ID2



# Challenges – Scaling

- **Template Prediction:**
  - Motivated by repetitiveness of bugs – small number of templates.
  - Use User reports and system logs.
  - Shrinks output space regardless of scale!

- **Parameter Prediction:**
  - Use system logs – Shrinks input space!
  - User reports rarely contain mentions of specific components.

# Challenges – Generalization to new fault locations

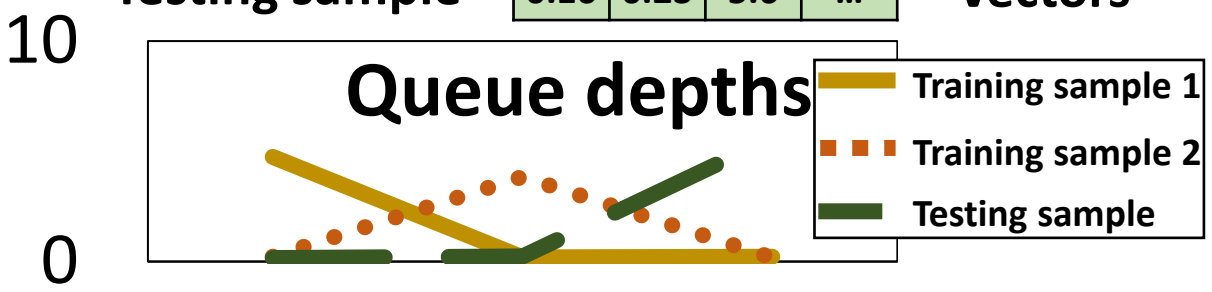
- Infeasible to gather training data capturing all faulty locations in the system.
- **Solution:** *Abstraction* – Rank Ordering of components based on their features better than ordering by IDs.

# Stability in Feature Rank Ordering – Better Predictability

| Parameter list | Avg. Queue depth (Training sample 1) | Avg. Queue depth (Training sample 2) | Avg. Queue depth (Test sample) |
|---|---|---|---|
| Switch 1 | 4.75 | 0.21 | 0.16 |
| Switch 2 | 0.18 | 3.85 | 0.23 |
| Switch 3 | 0.21 | 0.17 | 5.6 |
| … | … | … | … |

**Training sample 1 =** | 4.75 | 0.18 | 0.21 | … |
**Training sample 2 =** | 0.21 | 3.85 | 0.17 | … |
**Testing sample =** | 0.16 | 0.23 | 5.6 | … |

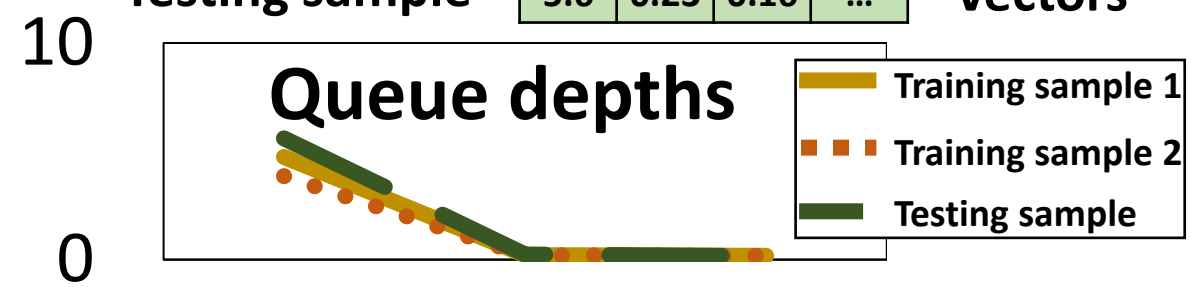
**Unordered Feature vectors**




Queue depths from train set **don't correlate** with those from the test set

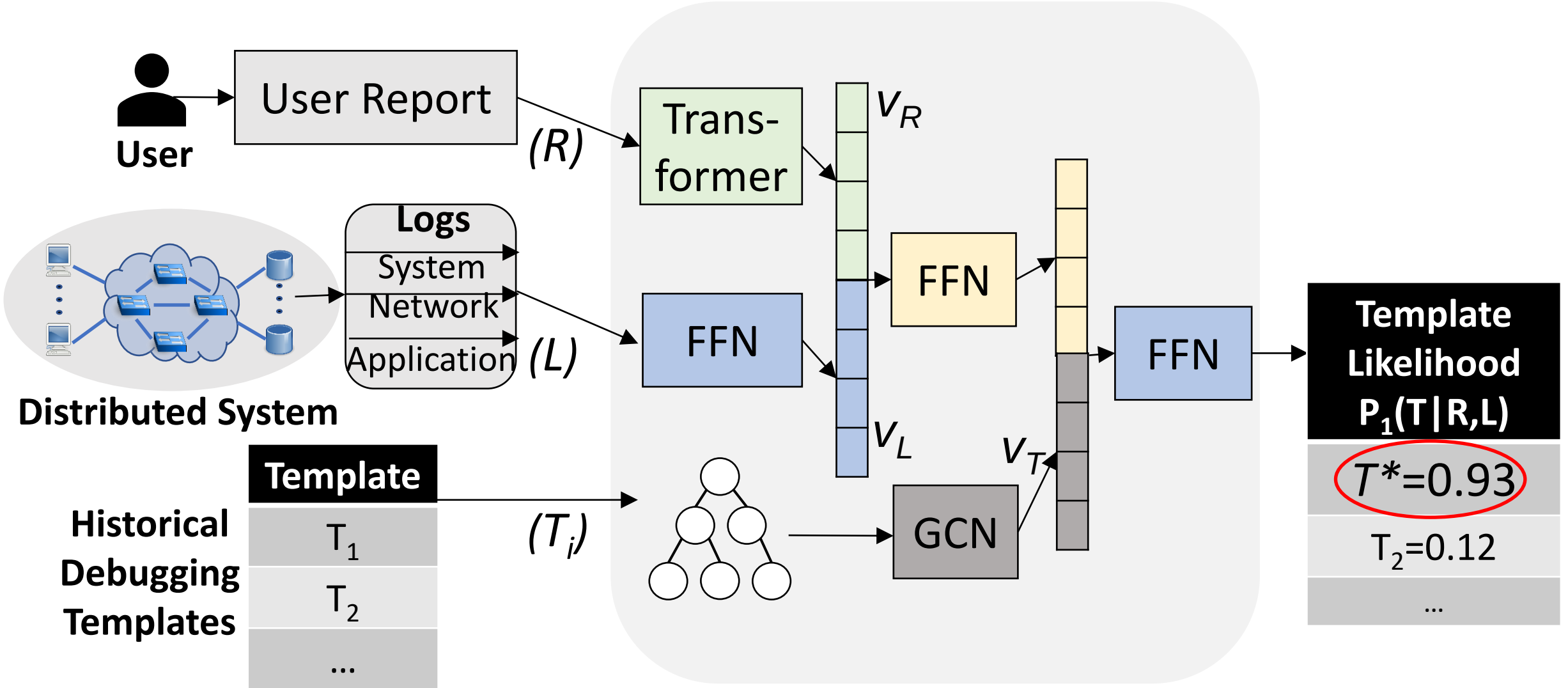
**Training sample 1 =** | 4.75 | 0.21 | 0.18 | … |
**Training sample 2 =** | 3.85 | 0.21 | 0.17 | … |
**Testing sample =** | 5.6 | 0.23 | 0.16 | … |

**Ordered Feature vectors**

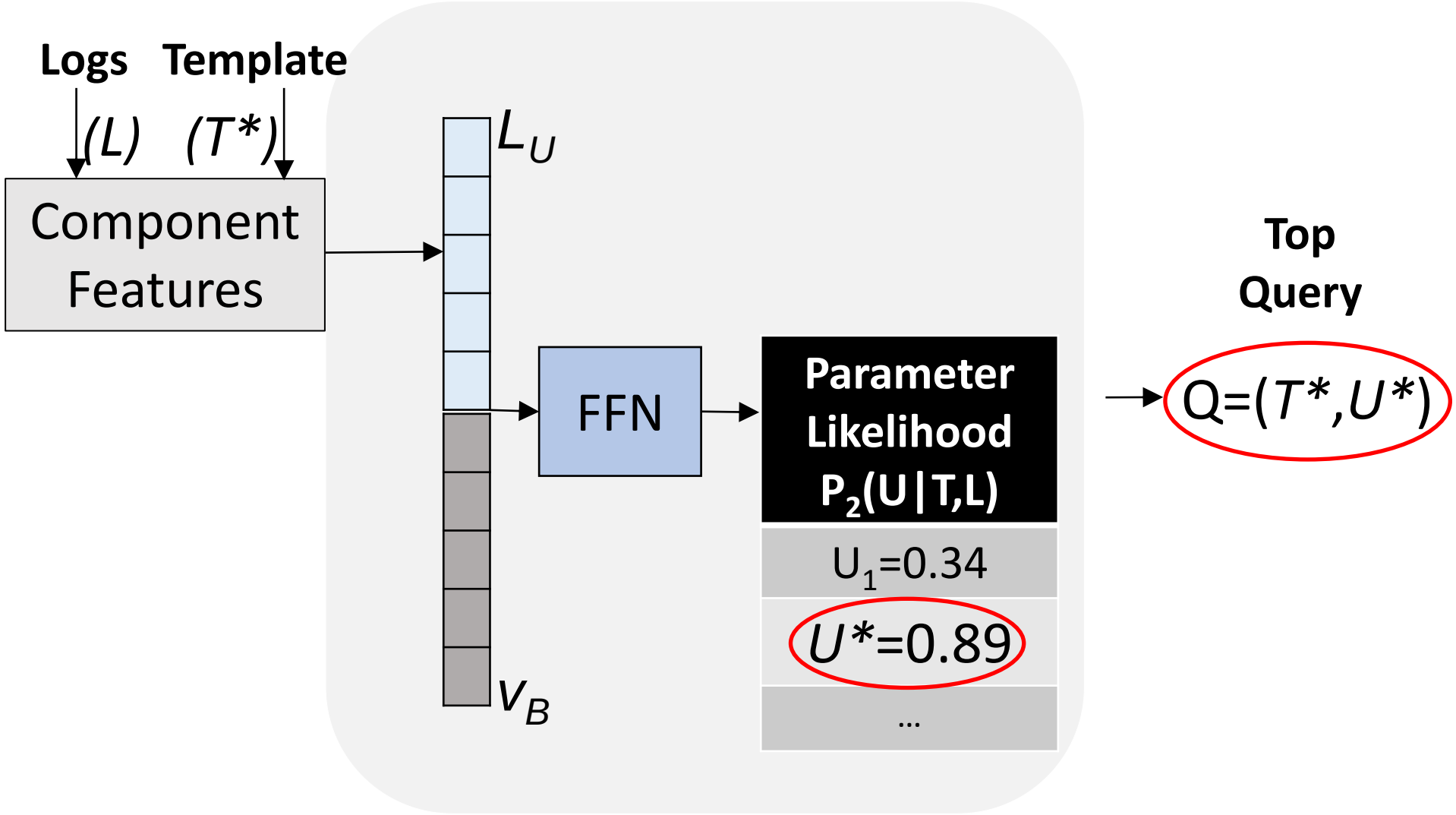



Queue depth order statistics from train set **correlate** with those from the test set

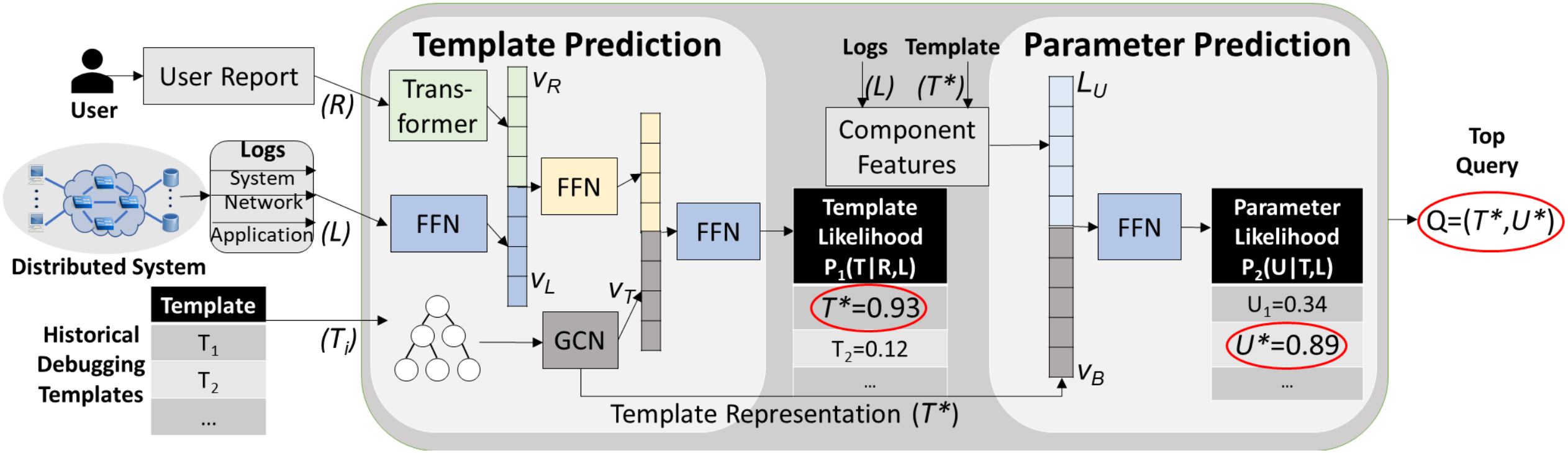
# Design - Template Prediction



# Design - Parameter Prediction



# Design - Debugging Query Generation



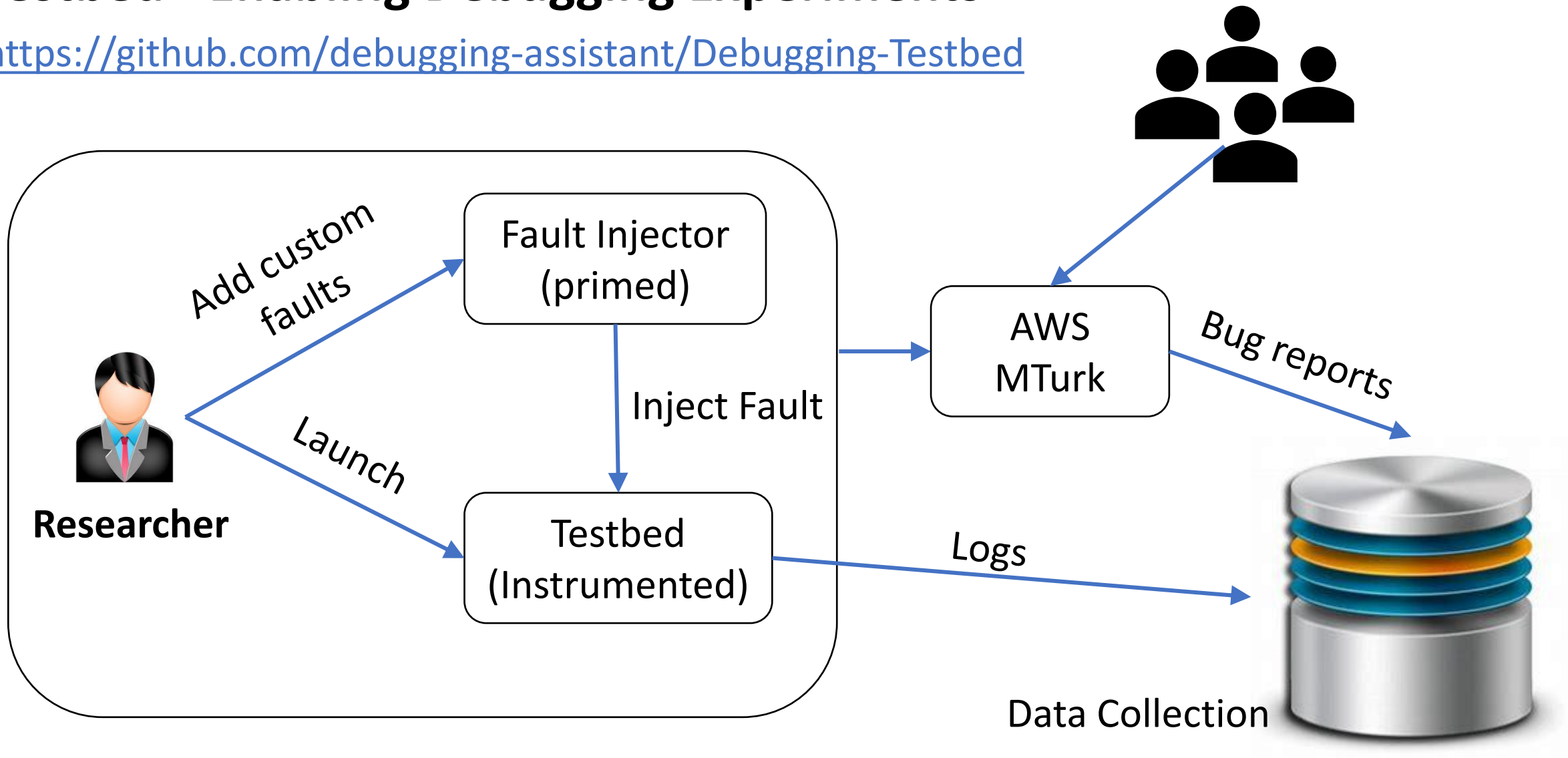
# Distributed Systems Debugging Testbed – Open Source

https://github.com/debugging-assistant/Debugging-Testbed

- **Real-World Applications:** Reddit, Sock Shop, Hipster Shop.
- **Logging Frameworks:** Jaeger (OpenTracing), cAdvisor, Marple, TCPDump.
- **Fault Injector:** Motivated by production study and primed with faults.
- **Mturk Interface:** Expose your testbed to real users.
- **Virtualized Topology:** Deployed on a single machine.

# Testbed - Enabling Debugging Experiments

https://github.com/debugging-assistant/Debugging-Testbed



# Testbed – Logging Metrics

https://github.com/debugging-assistant/Debugging-Testbed

- **Marple[1]:** Programs P4 switches in the network and collects queue depth, packet metadata.
- **Jaeger:** Collects application function execution times, tags and exceptions.
- **cAdvisor:** Reports CPU, memory, disk and network utilization metrics for each host in the testbed.

[1] Language-Directed Hardware Design for Network Performance Monitoring – SIGCOMM ’17

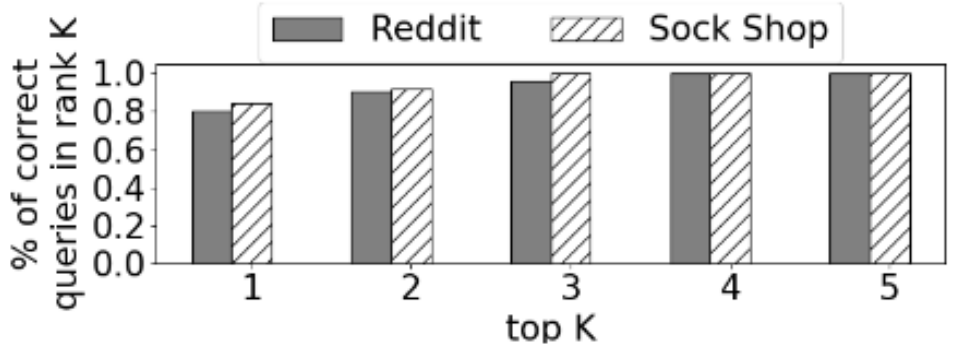
# Dataset – Evaluating Revelio

| Metric | Reddit | Sock Shop |
|---|---|---|
| # Unique Faults | 76 | 102 |
| # Unique Queries | 118 | 320 |

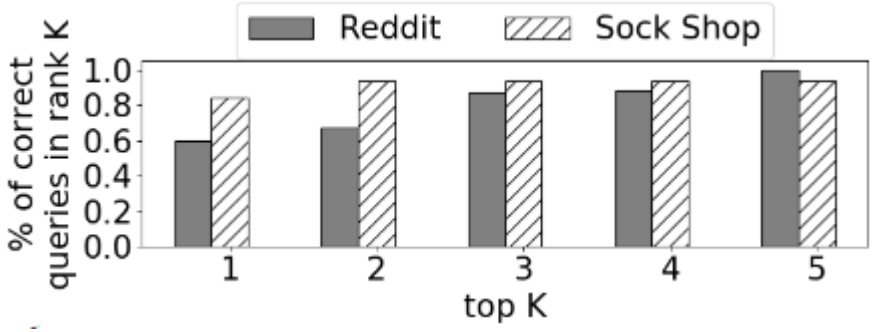
| Dataset Split | Percentage |
|---|---|
| Training | 53% |
| Validation | 13% |
| Test_repeat | 17% |
| Test_generalize | 17% |

| App | Dataset Size |
|---|---|
| Reddit | 694 |
| Sockshop | 346 |

# Evaluation – Revelio's Performance




- Revelio's performance on 'repeat' faults




- Revelio's performance on 'unseen' manifestations of faults

# Evaluation – Impact of design choices

| Scenario | test_repeat | test_generalize |
|---|---|---|
| User report+system logs | 1.33 (100%) | 1.97 (100%) |
| Only system logs | 1.86 (100%) | 2.29 (90.2%) |

Table 6: **Impact of different input sources on Revelio's performance. Results list avg rank (% in top-5) and are for Reddit.**

| Model | test_repeat | test_generalize |
|---|---|---|
| Revelio | 1.33 (100%) | 1.97 (100%) |
| Revelio_monolithic | 17.5 (15.1%) | 22.4 (18.5%) |
| Revelio_no_rank_order | 1.29 (100%) | N/A |
| Revelio_classifier | 2.41 (88.7%) | 2.69 (86.9%) |

Table 7: **Comparison with simpler ML approaches. Results list avg rank (% in top-5) for Reddit.**

# Evaluation – Impact of available training data

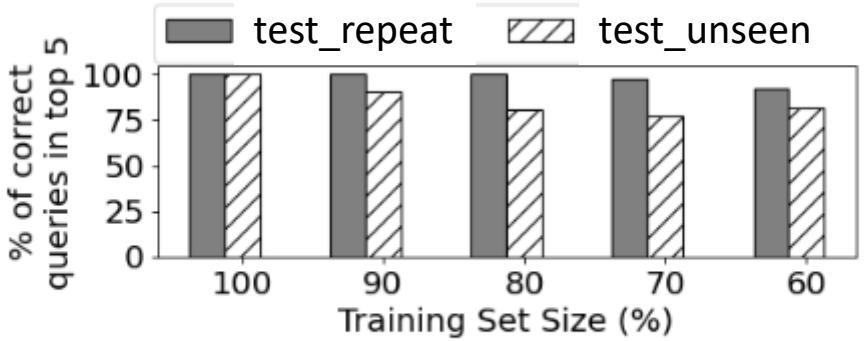



Figure 7: **Top-5 query accuracy when training Revelio on random subsets of the data. Results are for Reddit.**

# Summary

- **Problem:** Difficult to understand **which** debugging tool to use, **when** and **how** (i.e, which debugging query?)
- **Solution:** Leverage patterns in historical debugging data to *auto generate debugging queries* using ML
- **Ideas:** Leverage unstructured reports and structured logs, modularity and abstraction (stability in rank ordering)
- **Opensource Testbed:** Enables debugging experiments and data collection by others

# Thank you!

**Join Us:** https://github.com/debugging-assistant

Contact: dogga@cs.ucla.edu
http://web.cs.ucla.edu/~dogga