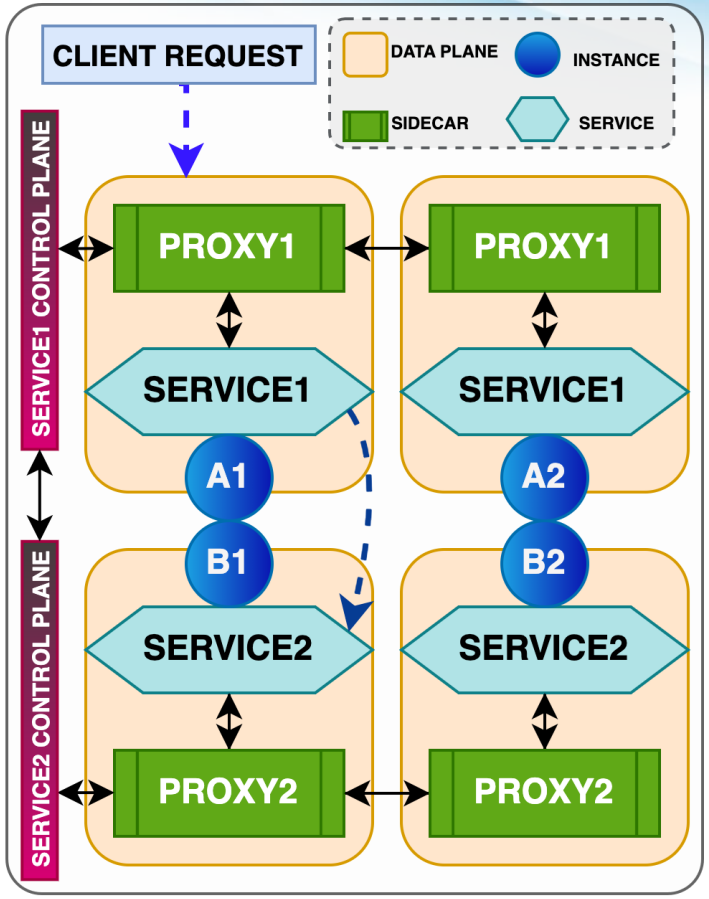
# ONLINE DIAGNOSIS OF MICROSERVICES BASED APPLICATIONS VIA PARTIAL DIGITAL TWIN

Sourav Das and Krishna Kant, Temple University
Presenter: Dr. Krishna Kant

The 22nd International Symposium on Network Computing and Applications (NCA 2024)

# Introduction to Microservices

- Small, single function modules running in separate containers
  - Intended to maximize parallelism and independence
  - Usually weak consistency, asynchronous calls, no locking
  - Broker for every resource
- Service Mesh (e.g., Istio)
  - Automated instancing for scaling.
  - Proxy based communication
  - SDN-like architecture



# Challenges in Microservices Fault Diagnosis

- Complexity of Distributed Systems:
  - Inter-service communication, data inconsistency.
- Complex Failure Scenarios:
  - Network latencies, service unavailability, cascading failures.
- Observability Gaps:
  - Difficulty in tracing across multiple services, inconsistent logging, lack of end-to-end visibility.

# Diagnosis Methods

- Traditional diagnosis methods for $\mu$S's
  - Centralized collection of service logs (e.g., using Graphana)
  - Mostly offline analysis to find problems proactively
- Online diagnosis also essential
  - Diagnose problem quickly based on observed/reported misbehavior
  - Sequential diagnosis most appropriate
    - Run a few tests ➔ Decide what to run next based on the results
- Diagnosing production systems challenging
  - No unvetted changes ➔ Cannot run "what if" tests
  - Specialized tests (read-only) may also be restricted
  - Prob

# *Partial Digital Twin (PDT)*

- Purpose: Provides reproducibility for testing in dynamic environments.
- Key Features:
  - Isolated microservice testing,
  - Handles configuration and code changes,
  - Uses Istio for managing microservice instances.
- Why not Full Digital Twin?
  - Impractical due to resource demands.
  - High synchronization traffic.

# PDT (Continued)

- Microservice Selection:
  - Dynamic, incremental construction based on diagnosis needs.
  - Cache system for frequently tested microservices.
- Query Transformation:
  - Reduces data requirements while retaining accuracy using aggregation.

# Fault Tickets

- A ticket is created for the reported fault by support team.
  - A single fault type (out of possible 8 types) and the service affected is mentioned in the ticket.

| Code | Fault Type | Symptoms Leading to User Report |
|---|---|---|
| UN | Unreachable Network | Unable to load the application or parts of it. |
| SN | Slow Network | Experiences delays in loading content. |
| US | Unreachable Service | Specific features not working with errors/timeouts. |
| SS | Slow Service | Certain actions taking longer than usual to complete. |
| DE | Data Error | Incorrect/inconsistent data displayed within the application. |
| DC | Data Corruption | Errors during data processing, or data loss. |
| UA | Unauthorized Access | Access denied when using previously accessible feature/data. |
| BA | Blocked Access | Unable to access parts of the application due to restrictions. |

# Testing Procedure

- Subset Identification:
  - Identify the microservices related to the fault.
  - PDT Check: Verify if the identified microservices are already in the PDT. If absent, replicate them in the PDT.
- Test Selection
  - Use Zero Shot Learning (ZSL) to match faults with the most relevant tests. Matching done statically
- Test Execution Order:
  - Execute tests sequentially based on fault type. Stop when misconfiguration is found.
- If no misconfiguration is found, expand subset of microservices and repeat testing.

# Assumptions

| No. | Assumption Description |
|---|---|
| 1 | **Stateless:** Microservices (MS) are stateless, with no session information retained between calls. |
| 2 | **Cascading:** Configuration changes may affect dependent MS. |
| 3 | **Query Transformation:** Queries are correctly transformed between environments without adding bias. |
| 4 | **Single Fault:** Only one fault occurs at a time within the system. |
| 5 | **No Intermittence:** Faults persist until resolved, with no intermittent behavior. |
| 6 | **Identifiability:** All misconfigurations are definitively identifiable. |

# Test Set (Total 26, named A-Z)

- nslookup <domain> : 1 iff successful DNS resolution takes <1s.
- B ping -c 4 <domain> : 1 iff ICMP latency to destination domain is <100ms in 4 attempts.
- traceroute <domain> : 1 iff destination is reachable.

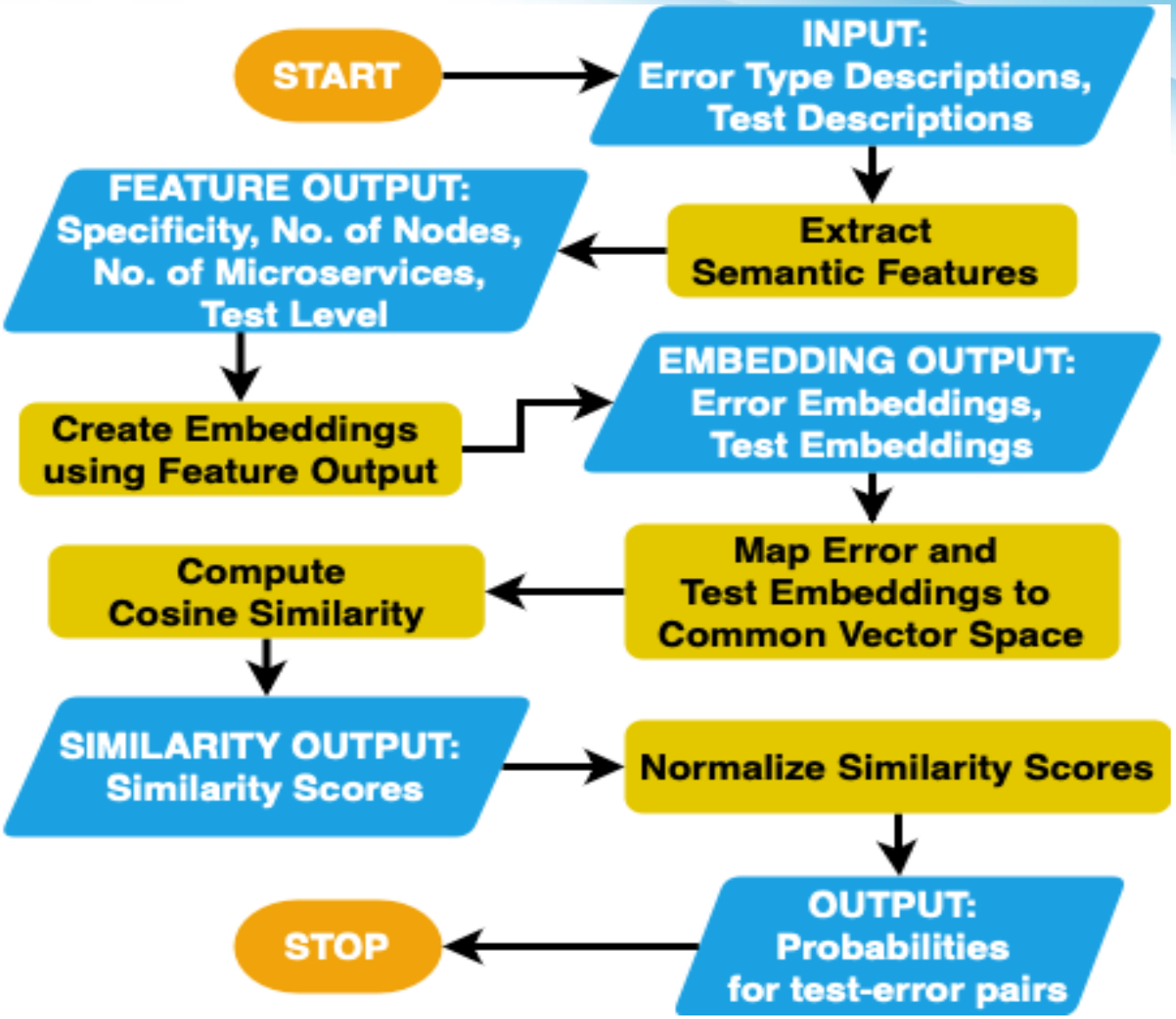
- .
- .
- .
- Z.

# Categorization of Tests into Attributes

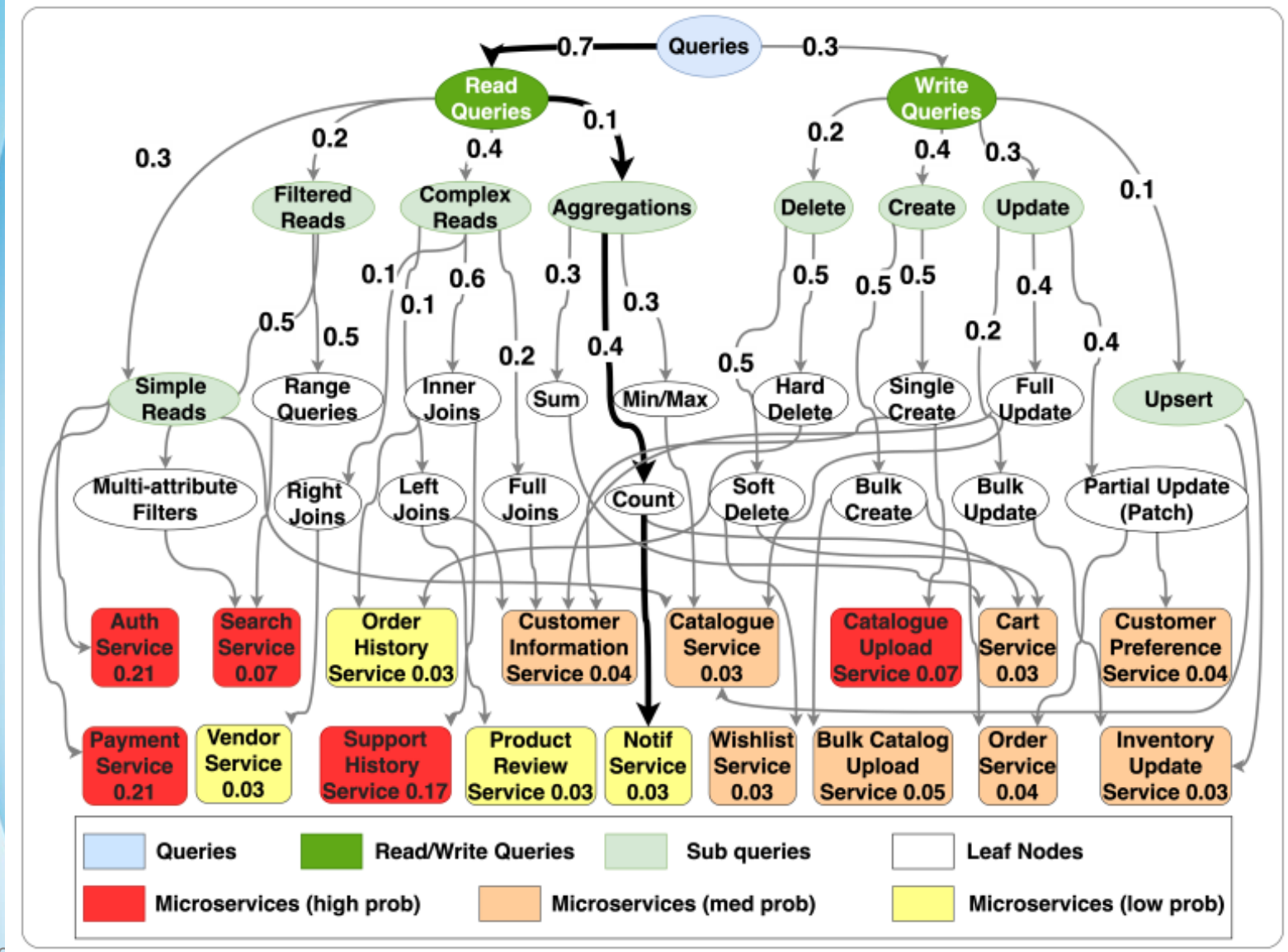
- A-E: Network bases tests which are mostly E2E.
- F-H: Security tests.
- I-Q: Infrastructure based tests in K8s and Istio configs.
- R-W: Database related tests.
- X-Z: Data corruption related tests.

| Attribute Type | Configuration Parameters |
|---|---|
| **Network** | DNS Resolution, Network Latency, Network Routing |
| **Security** | Access Control, API/JWT Keys, Authentication |
| **Infrastructure** | Service Discovery, Istio Config. (Dest. Rules, Request Routing, mTLS Encryption), Port Accessibility, Payload, Response Time |
| **Database** | DB Connectivity, DB Grants, DB Query Execution, Maximum File Size, Caching, Cron Jobs Scheduling |
| **Data** | Data Formatting, Invalid Input Handling, Environment Type Config. |

# Test Selection Infrastructure



# Test Selection Probabilities

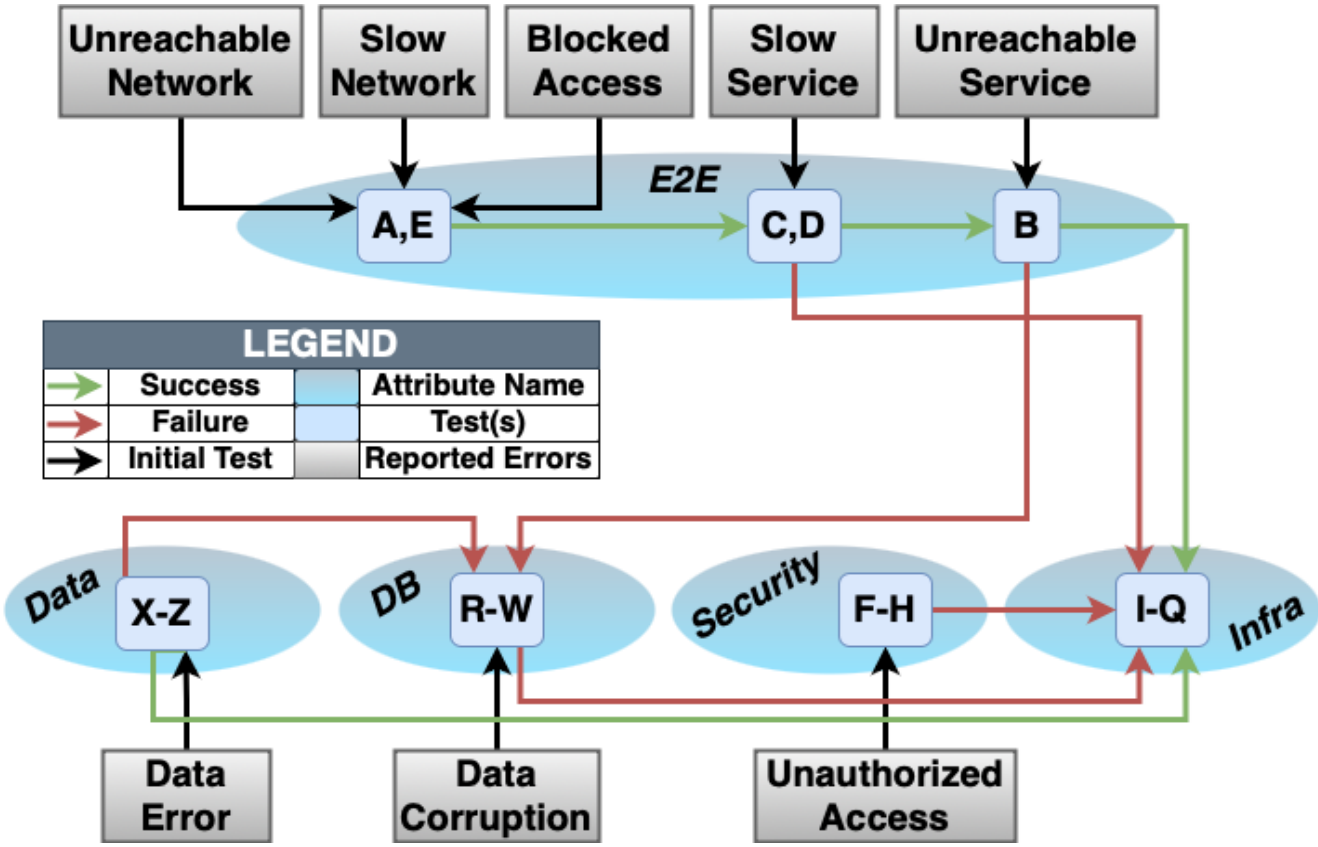


# Microservice Subset Identification

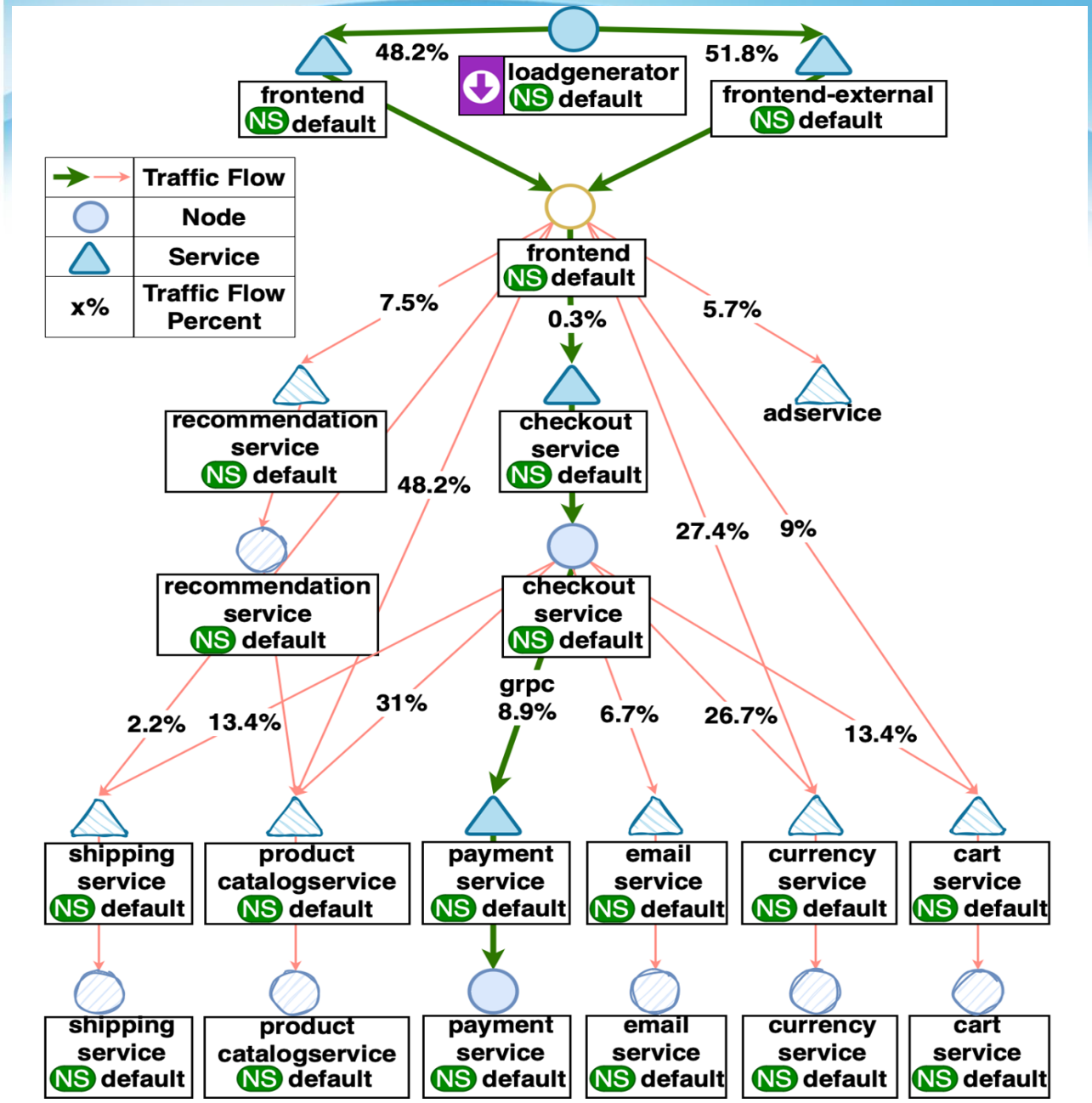
- Initialize cache with the faulty microservice
- Procedure
  - Identify microservices not in cache (A) but called by those in cache.
  - Compute fault-score using
    - historical error rate (HER),
    - call frequency (HCF), and
    - transaction call graph (TCG).
  - Add highest-scoring microservices to the list.
  - If cache is full, evict least recently used (LRU) microservices.
  - Return list with selected microservices for replication in PDT.

# Test Execution Process

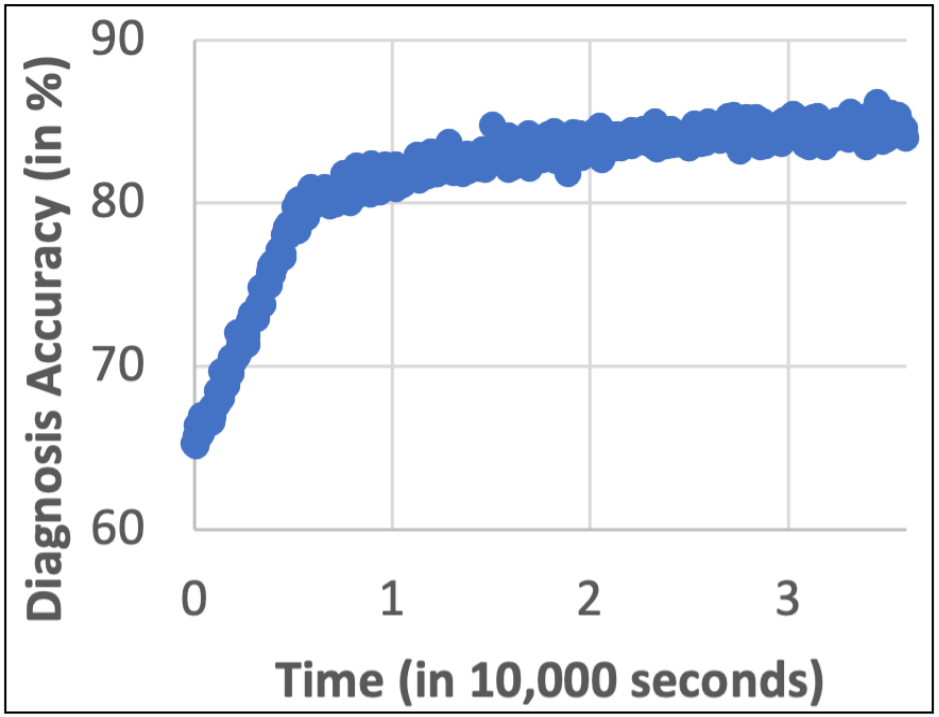
- Test flow based on relevance scores;
  - adjust based on test outcomes (e.g., for “US” or “Unreachable Service” faults: {B, R-W, I-Q} is one of the paths).

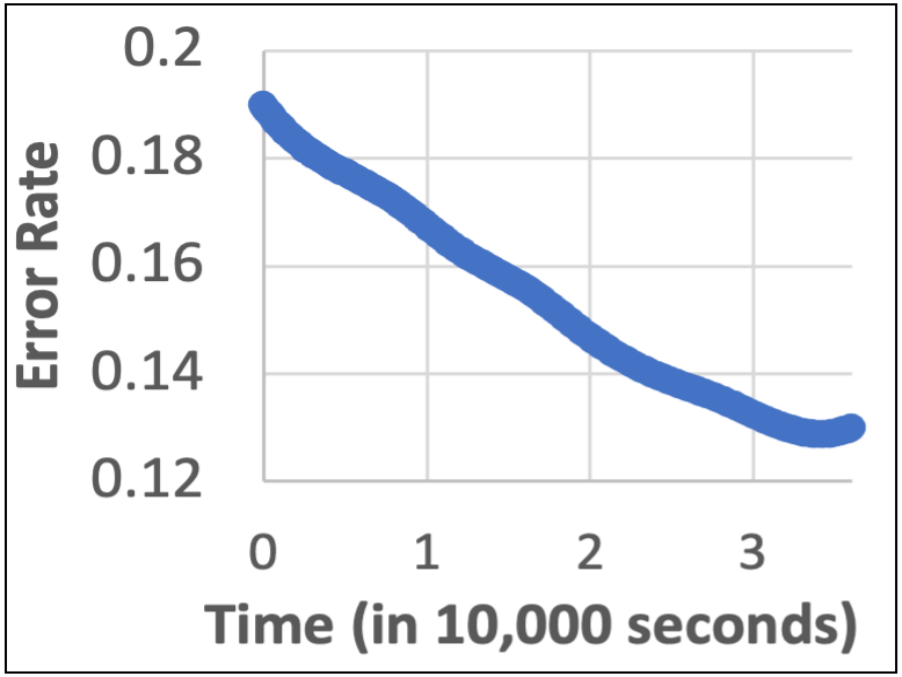


# Example shopping application

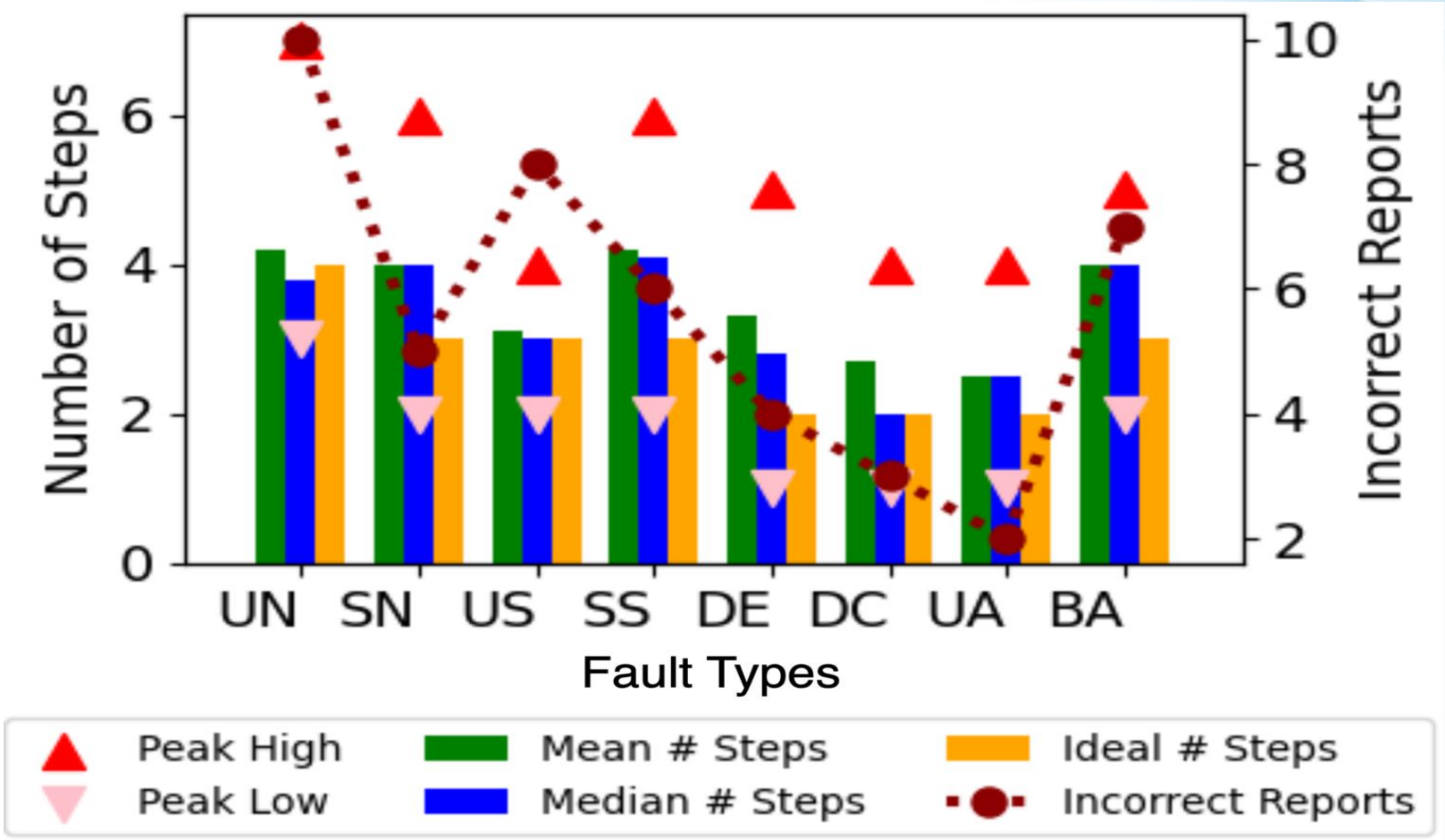


# Normal Case (Mix of Reads & Writes)

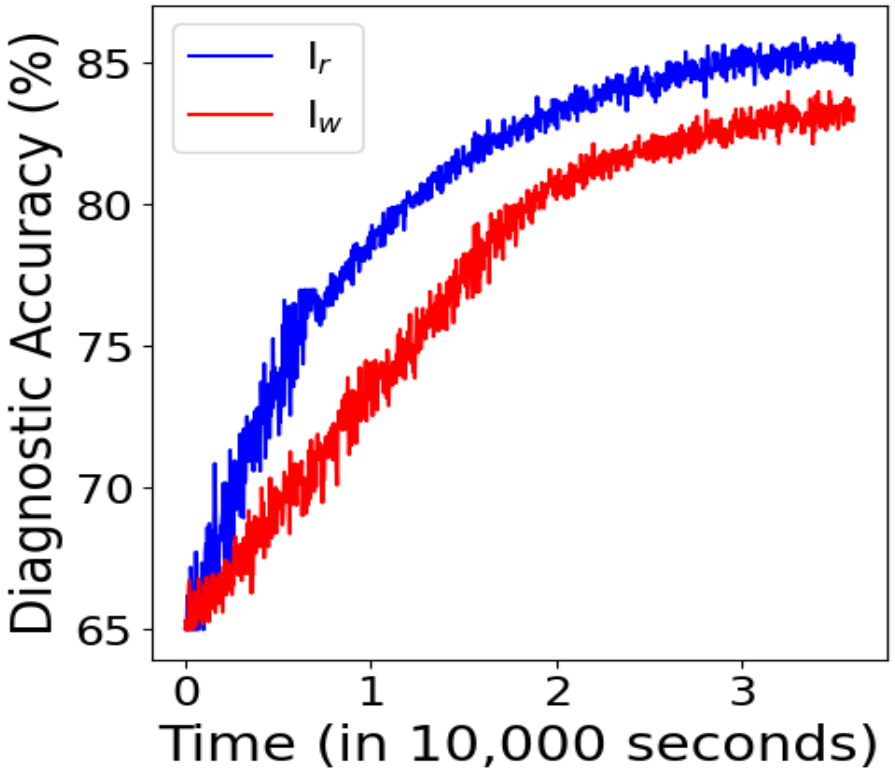


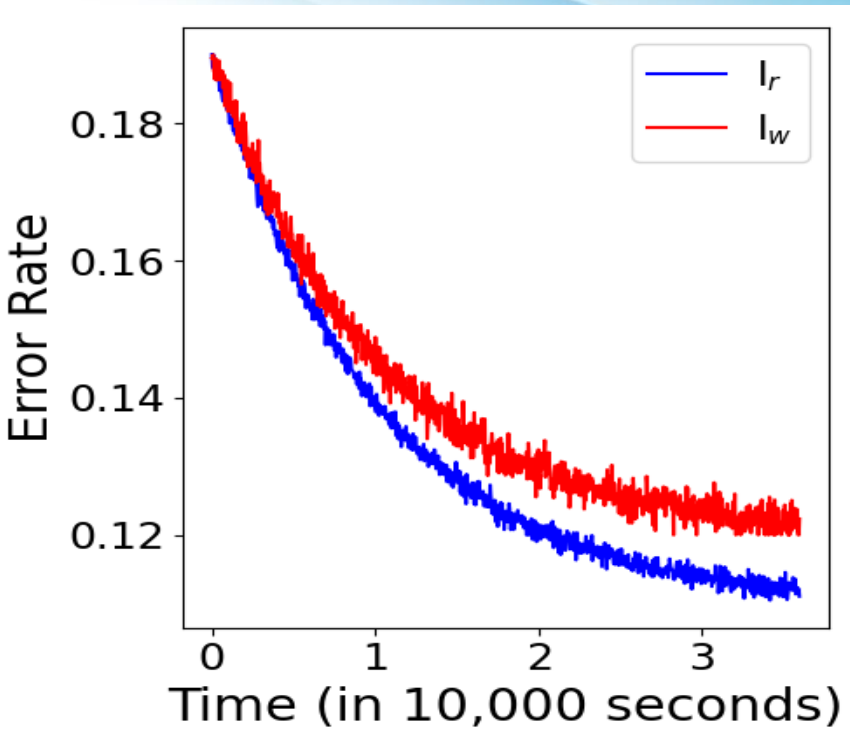



# Number of Tests Required

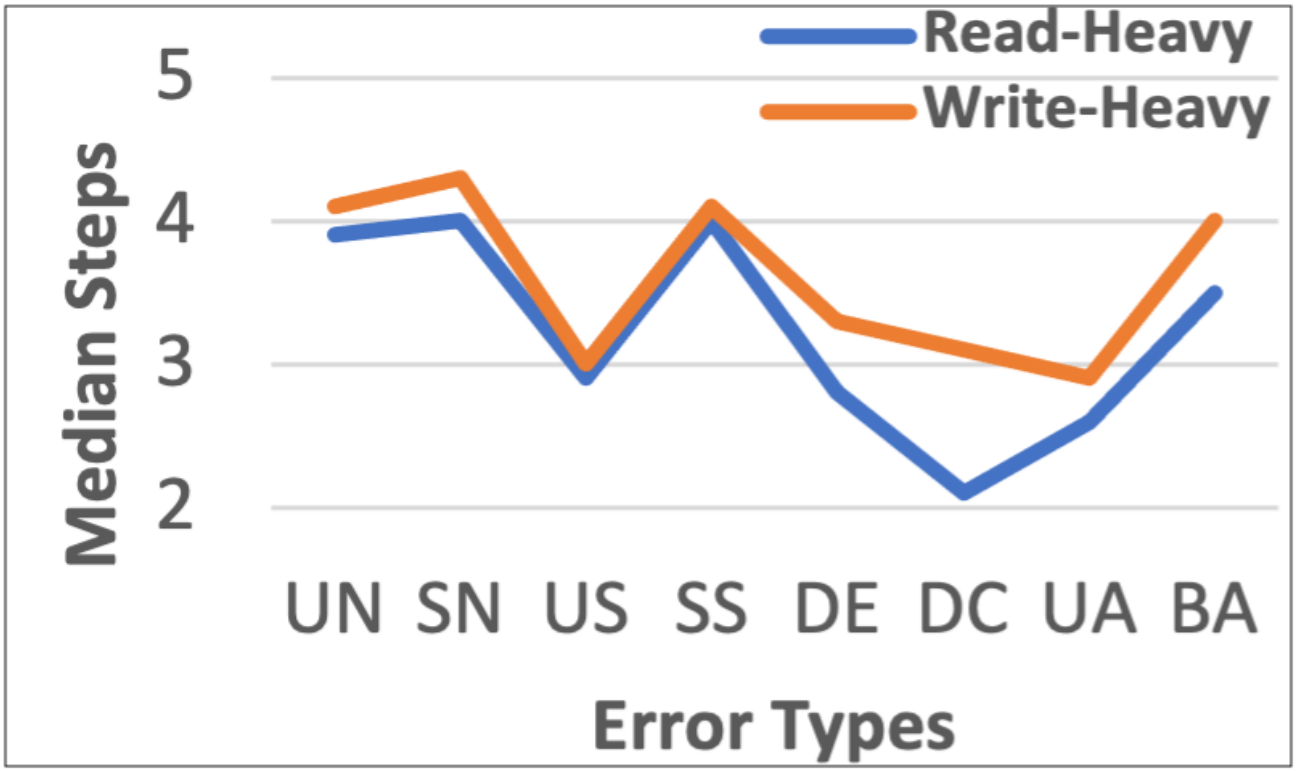


# *Mix of Read-heavy & Write-heavy Wkload*



# Mean #Tests Needed



# Conclusions

- The mean number of steps taken is approximately 22% above the ideal case for all fault types combined.

- The average median is about 7% above the ideal case.