Growing a Protocol

Kamala Ramasubramanian\textsuperscript{1}, Kathryn Dahlgren\textsuperscript{1}, Asha Karim\textsuperscript{1}, Sanjana Maiya\textsuperscript{1}, Sarah Borland\textsuperscript{1}, Boaz Leskes\textsuperscript{2}, Peter Alvaro\textsuperscript{1}

\textsuperscript{1}Disorderly Labs \quad \textsuperscript{2}Elastic

University of California, Santa Cruz
Motivation

✦ Elastic - distributed data store vendor whose products focus on real time search and analysis

✦ Data replication protocol is based on Primary/Backup

✦ Interested in a tool to determine if protocol was behaving correctly in the presence of faults as it evolves
Introduction

- Software engineering best practices:
  - Regression testing
  - Root cause analysis

- Does not work while reasoning about fault tolerance properties of distributed systems

- Associate specific inputs with system behaviors
An example

Leader Election Protocol

VERSION 1
Node with min ID is elected
x is leader

VERSION 2
Node with max ID is elected
y is leader

Error in downstream logic: If leader crashes, system exhibits unexpected behavior
Naive Solution(s)

Formal Methods
- Need to back port protocol changes to specifications

Testing
- Need to explore all fault scenarios for **EVERY** version change
Lineage Driven Fault Injection

- LDFI - Lineage Driven Fault Injection
- Uses techniques from databases, logic programming and fault injection techniques
- Reasons about how a good outcome occurred to determine why a bad outcome might occur
LDFI builds a system model beginning with the first execution being successful

Fault scenarios explored are those which falsify good outcome
Catching Bugs early …

Replica n2 is promoted to primary
Dormant bugs
A tale of two optimizations

- Specially handling certain cases to avoid expensive operations
- Re-sync mechanisms

The “simplicity” of an optimization is not a barometer in understanding if it *could* violate guarantees of the system
Avoiding expensive operations

Replica n2 is promoted to primary

Rep_W1

Rep_W2

Ack_int

Ack

CRASHED
Past & Future Work

- Concurrency bugs
  - Explore not just schedules, but reorderings as well.

- Input Generation
  - Different fault scenarios based on input data selected. Impacts bugs found.
Debugging distributed systems is hard!

Need tools which straddle verification and testing

Demonstrated that LDFI can be deployed as a tool in this space to find interesting bugs
palvaro@ucsc.edu
Questions?

kamala.ramas@ucsc.edu
https://github.com/KamalaRamas/molly