Statistical Bug Isolation for Consensus Systems

How to find root causes of bugs when you logged millions of messages

Levin N. Winter, Ege Berkay Gülcan, Burcu Kulahcioglu Ozkan
Software Engineering Research Group, Delft University of Technology

TU Delft

1. Background

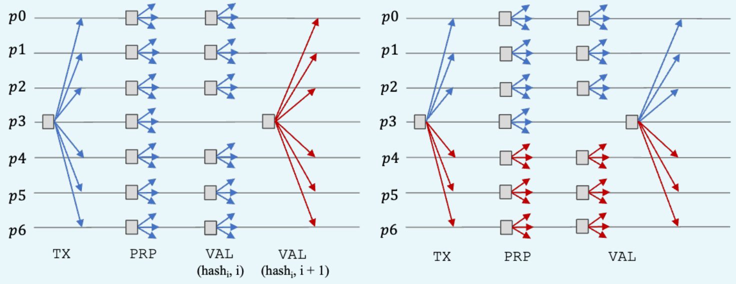
- Hard to root cause bugs in consensus systems
- Only expose under certain sequences of events
- Goal: automatically identify predicates (scenarios) that correlate with bugs

2. Research Questions

1. How well do existing methods identify bugs in consensus systems?
2. Which predicates effectively identify bugs in consensus systems?
   a. Are multiple bugs correctly discriminated?
   b. Do the predicates help to identify the underlying bug

3. Method

- Implement existing method by Libit et al. [1]
- Propose and implement our new method
- Data set of 1200 XRP Ledger runs generated by [2]
  - Contains three bugs (B_1, B_2, B_3)
  - B_2 and B_3 share a single root cause



(a) Bug B_1 (b) Bugs B_2 and B_3

Figure 1: Bugs in the XRP Ledger

4. ISOLATION Algorithm

- Developed novel predicates specific to consensus systems (Fig. 2)
  - Predicates describe message patterns
  - Model how processes in consensus systems behave
- Algorithm generates predicates based on description of protocol messages
- Use statistics to identify most important predicates
  - Modified version of statistical framework proposed in [1]

5. Results

Baseline

- Isolated a single predicate that groups all bug
- Does not discriminate between bugs with different root-causes

ISOLATION Algorithm

- Identified four predicates, P_1 to P_4, that correlate with bugs
- P_1 groups B_2 and B_3 which share the same root cause (Fig. 2b)
- P_2 isolates B_1 with root cause different from B_2 and B_3 (Fig. 2a)
- P_3 and P_4 capture similar patterns as P_2
- P_1 and P_2 have a higher precision, F_1, and $F_{0.5}$ score than baseline (Fig. 3, 4)

6. Conclusion

- Existing method cannot discriminate between bugs
- ISOLATION correctly separates bugs by their root causes and outperforms the baseline

$$2 \times \mathtt{VAL} \xrightarrow[\substack{\mathrm{ledger}(l) \neq \mathrm{ledger}(r) \\ \mathrm{seq}(l) = \mathrm{seq}(r) \\ \mathrm{time}(l) \neq \mathrm{time}(r)}]{\text{for more than 2 processes}} \mathtt{VAL}$$

(a) Predicate P_1 for B_1

$$2 \times \mathtt{VAL} \xrightarrow[\substack{\mathrm{senders}(l) \cap \mathrm{senders}(r) = \emptyset \\ \mathrm{seq}(l) \neq \mathrm{seq}(r) \\ \mathrm{time}(l) = \mathrm{time}(r)}]{\text{for more than 3 processes}} \mathtt{VAL}$$

(b) Predicate P_2 for B_2 and B_3

Figure 2: Predicates identified by ISOLATION

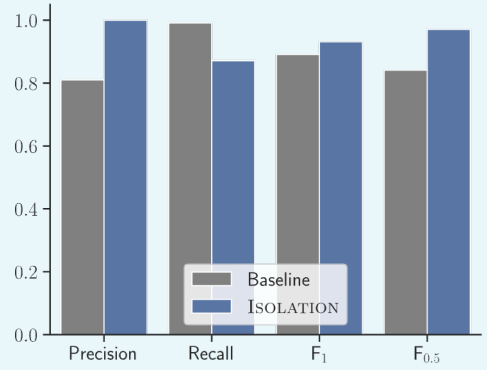


Figure 3: Performance of P_1 for B_2

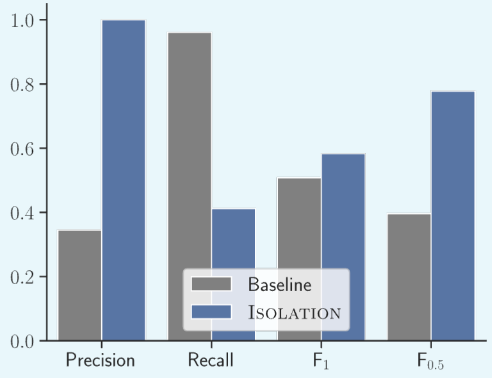


Figure 4: Performance of P_2 for B_1

References

[1] Liblit, Ben, et al. "Scalable statistical bug isolation." (2005)
[2] Winter, Levin N., et al. “Randomized Testing of Byzantine Fault Tolerant Algorithms” (2023)

Contacts

L.N.Winter@student.tudelft.nl
{E.B.Gulcan, B.Ozkan}@tudelft.nl