C-FLASH: Concurrency Faults Localized Automatically using Search Heuristics

Luisa F. Rojas Garcia • **Jeremy S. Bradbury**
Faculty of Science • **University of Ontario Institute of Technology** • **Oshawa, ON, Canada**
{luisa.rojasgarcia, jeremy.bradbury}@uoit.ca

1. Motivation

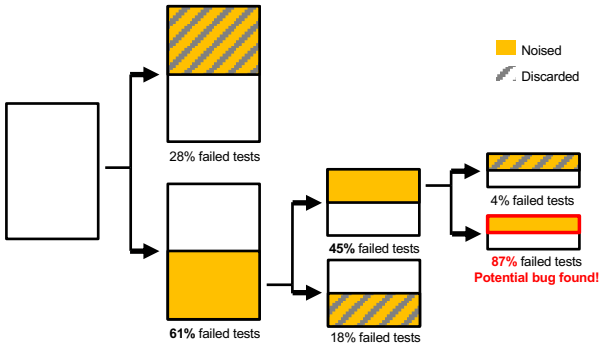
- Multicore processors have allowed for great improvements in performance – if software is concurrent!
- Unfortunately, the nondeterministic nature of concurrent programs make them challenging to develop and test [1].
- The results of concurrency testing can vary for different thread interleavings (i.e., execution schedules) [2].

RESEARCH GOAL:
Create a development tool that automatically localizes concurrency bugs, given a unit of source code and a set of tests.

2. Background

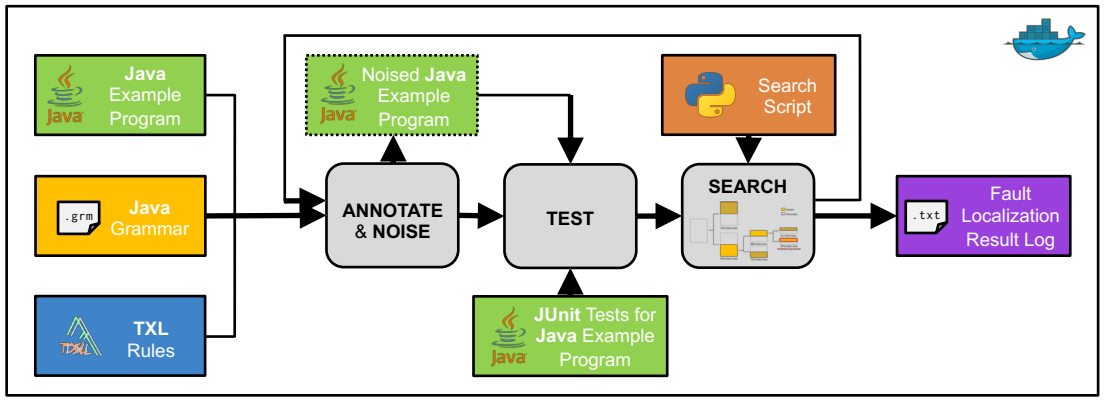
- **Concurrency bugs** can manifest themselves in some thread interleavings but not in others.
- **Concurrency testing** needs to test a concurrent program with many different interleavings.
- **Noising** is the process of inserting random delays around critical areas in in a concurrent program. By using noising, we can control thread delays and explore different execution outcomes [3].

3. Fault Localization Strategy



- Binary search approach where each section of code is divided into two sections. Each section is noised using TXL [4] and then tested separately.
- The test results for each noised section are compared and the section that exhibited a higher rate of failed tests is further split until the unit level of source code is reached.

4. Process Overview



5. Preliminary Evaluation

- To evaluate the effectiveness of **C-FLASH** we have performed bug localization on several programs from the IBM Concurrency Benchmark [5], each of which have a known data race or deadlock.
- Our preliminary results show **C-FLASH** to be effective at finding the location of a known bug in our test programs – identifying the correct synchronization block or method containing the bug.
- Next, we plan to conduct a thorough evaluation using a set of large-scale, real-life concurrent programs.

6. Conclusions and Future Work

- **C-FLASH** is a modernized and portable version of SyncDebugger, an earlier bug localization prototype [6].
- **C-FLASH** vs. SyncDebugger – the primary enhancements include:
  - Full automation of the bug localization search strategy.
  - Performance optimizations in the annotation and noising of concurrent programs under test.
  - Portability across platforms using Docker.
- The efficiency of **C-FLASH** is also of high importance in our research. In order to better optimize the bug localization process, we plan to:
  - Explore efficient search algorithms that fit our objective.
  - Parallelize our current approach, allowing for different parts of a system to be analyzed simultaneously.

7. References

[1] J. Yang, H. Cui, J. Wu, Y. Tang, and G. Hu, "Making Parallel Programs Reliable with Stable Multithreading," *Communications of the ACM*, vol. 57, no. 3, pp. 58–69, Mar. 2014.
[2] T. Yu, "TACO: Test Suite Augmentation for Concurrent Programs," *Proc. of ESEC/FSE 2015*, pp. 918–921, Aug. 2015.
[3] O. Edelstein, E. Farchi, E. Goldin, Y. Nir, G. Ratsaby and S. Ur, "Framework for Testing Multithreaded Java Programs," *Concurrency and Computation: Practice and Experience*, vol. 15, no. 35, pp. 485–499, Feb. 2003.
[4] J. R. Cordy, "TXL - A Language for Programming Language Tools and Applications," *Electronic Notes in Theoretical Computer Science*, vol. 110, pp. 3–31, Dec. 2004.
[5] Y. Eytani and S. Ur. Compiling a Benchmark of Documented Multithreaded Bugs. In *Proc. of PADTAD 2004*, Apr. 2004.
[6] J. Kwok, J. S. Bradbury. "SyncDebugger: Automatic Bug Localization in Multithreaded Programs", In *Technology Showcase at CASCON 2015*, Nov. 2015.

CASCON 2018 Technology Expo • **Markham, ON, Canada** • **October 2018**

FUNDING PROVIDED BY: NSERC CRSNG