Pike: Finding Concurrency Bugs in Large Multi-Threaded Applications

Pedro Fonseca, Cheng Li and Rodrigo Rodrigues
Max Planck Institute for Software Systems (MPI-SWS), Germany

1. Motivation

- Multi-core era
  - Applications increasingly more parallel
  - Parallel applications are prone to concurrency bugs
- Concurrency bugs
  - Non-deterministic
    - Triggered only when the OS chooses certain interleavings
    - Very hard to find
    - Have a serious impact on robustness

2. Detection of concurrency bugs

- Thread exploration techniques help (e.g., CHESS)
  - But detection relies on catching exceptions
- Data race detectors can still miss bugs
  - Particularly challenging classes of bugs:
    - Semantic: manifest by providing wrong results
    - Latent: manifest at a later time

Need a detector for concurrency bugs that are hard to find

3. Linearizability: An implicit specification

Problem: Did a concurrent execution exhibit the correct behavior?

Hypothesis: A correct concurrent execution exhibits the same behavior as one of the sequential executions

- Behavior = Output + Final state
  - Analyze the output to detect semantic bugs
  - Analyze the state to detect latent bugs
  - Comparing state is application specific
  - Need programmers to specify state summary functions

4. Finding concurrency bugs in MySQL with Pike

- We built Pike: A bug finding tool that checks for linearizability
- Pike incorporates a custom scheduler to explore different concurrent interleavings
  - Each concurrent execution is compared with every sequential execution
  - We use filters to reduce false positives
- We applied Pike to MySQL
  - Complex multi-threaded server
  - We adapted the MySQL test suite to make tests concurrent

5. Summary of results

- Pike was able to find semantic and latent concurrency bugs in MySQL
- Filters significantly reduced the number of false positives