Lazy Diagnosis of In-Production Concurrency Bugs

Baris Kasikci Weidong Cui
Xinyang Ge Ben Niu

UNIVERSITY OF MICHIGAN Microsoft

Problem

- It is hard to diagnose the root causes of concurrency bugs
- It is hard to perform bug diagnosis in production
- Existing attempts to solve the problem are not practical
  - High overhead
  - Assume existence of special hardware

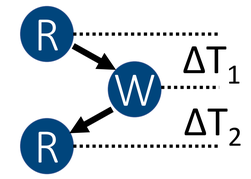
Key Insights

Coarse Interleaving Hypothesis

- Events leading to concurrency bugs are coarsely-interleaved
- Holds for real-world systems (54 bugs in 13 systems)
  - Apache httpd, MySQL, memcached, ...
- Does not hold for programs with fine-grained concurrency
  - Concurrent data structures, …

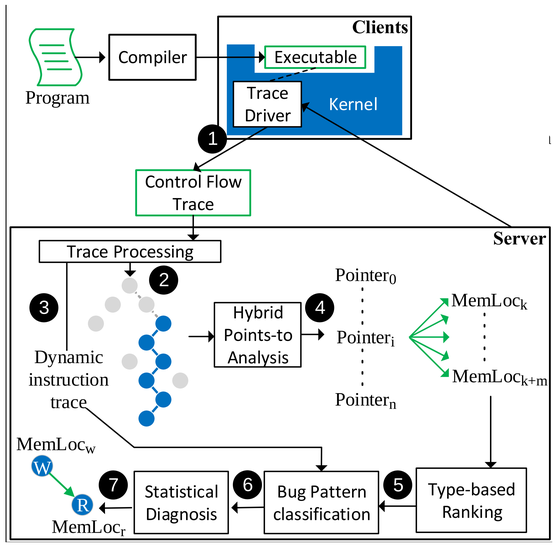
If the hypothesis holds, bug patterns can be identified efficiently

RWR Atomicity Violation



Can be measured with a coarse and low-overhead timer

How does Lazy-Diagnosis Work?



(1) Control flow trace generation

- Upon failure or on demand

(2) Trace processing

- Filters out instructions that don't execute

(3) Dynamic trace generation

- Partially ordered using Intel Processor Trace timing packets

(4) Hybrid points-to analysis

- Interprocedural, flow-insensitive, operates on executed instructions

(5) Type-based ranking

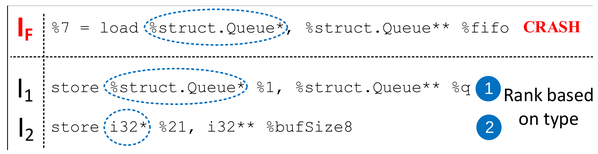
- Favors exact type match for identifying potential instructions involved in a bug

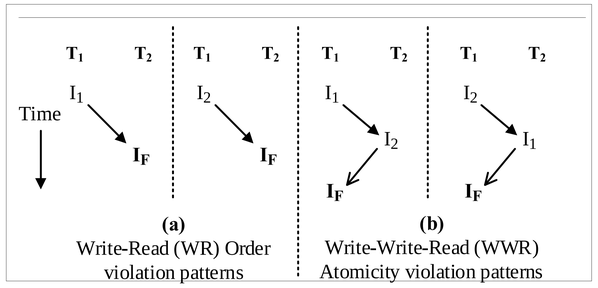
(6) Bug pattern classification

- Relies on the partially-ordered trace and type-based ranking

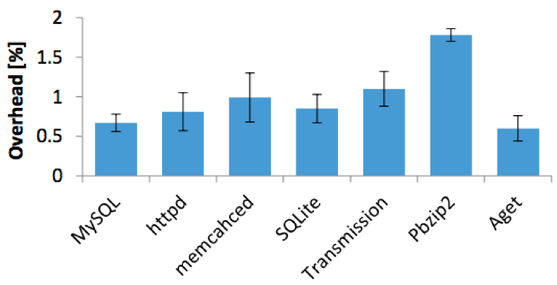
(7) Statistical Diagnosis

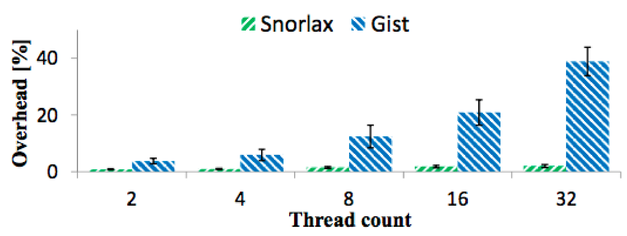
- Identifies patterns that are positively-correlated with failures





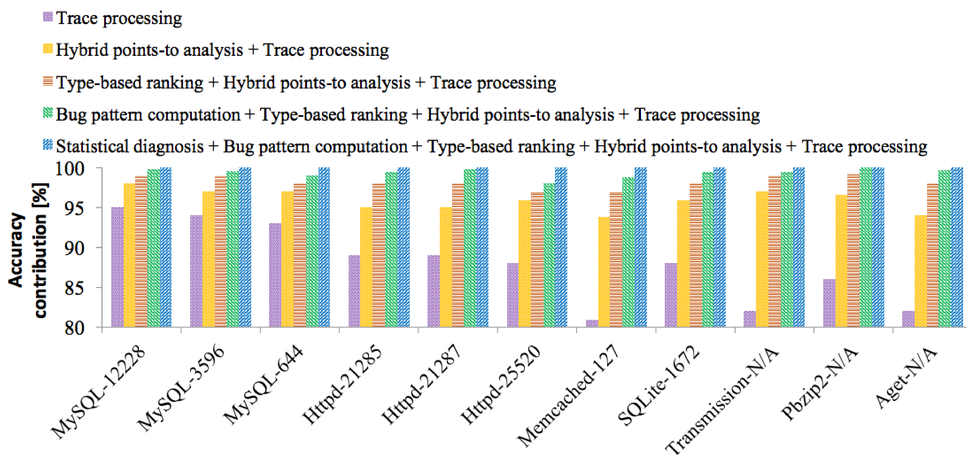
Results





Our Lazy Diagnosis prototype Snorlax:

- Has low runtime overhead (<1% on average)
- Scales better than the state-of-the art concurrency bug diagnosis technique



A combination of all the techniques is necessary for high accuracy