MTraceCheck: Validating Non-Deterministic Behavior of Memory Consistency Models in Post-Silicon Validation

Doowon Lee and Valeria Bertacco, University of Michigan

1. Memory Consistency Models
- Memory consistency models specify observable orderings of memory accesses
  - Sequential consistency (MIPS R10K), total store order (x86, SPARC), weakly-ordered models (ARM, IBM POWER), etc.
- Multi-threaded programs experience various memory ordering patterns during their program executions
  - Due to processor architectural optimizations related to out-of-order execution, coherent caches, multiple memory channels, on-chip interconnect etc.
- Memory ordering violations may cause incorrect execution results in multi-threaded programs
- Many hard-to-find bugs (errata bugs) are related to memory behaviors in multi-threaded programs

2. Prior Memory Consistency Validation
- Formal verification approach (Aigaleo/1 Lustig/17, etc.)
- Constrained-random testing approach (Hangal/04, Elver/16, etc.)

3. MTraceCheck Workflow
- Code instrumentation
- Tests generation
- Tests execution
- Test configuration
- Violation checking
- Systems under validation

4. Memory-Access Interleaving Signature
- Memory-access interleaving signature encapsulates loaded values over the execution of test program
- Step 1: Identify all reads-from relationships for each load operation
- Step 2: Assign a unique weight for each reads-from relationship
- Step 3: Accumulate weights observed at runtime

5. Collective Graph Checking
- Collective graph checking reduces result-checking computation by exploring similarity among graphs from repeated runs
- We use memory-access interleaving signature to estimate graph similarity

6. Non-Determinism in Memory Interleaving
- We ran each test 65,536 times and measured the number of unique memory-access interleavings
- High non-determinism (38% unique interleavings)
- Low non-determinism (10% unique interleavings)

7. MTraceCheck Validation Performance
- Violation checking time
- 81% reduction in graph checking time, compared to individual graph checking
- High benefit in relaxed memory model (ARM)
- Runtime overheads
- 22% / 38% runtime increase due to signature computation / sorting, respectively
- This increase is marginal (approximately 5 times) compared to time saving in violation checking

8. MTraceCheck Instrusiveness
- Memory-access perturbation
- 93% reduction in memory accesses unrelated to test execution
- Higher benefit in less contentious tests
- Code size
- 3.7x larger than original test
- Small enough to fit in L1 caches

9. Conclusions
- MTraceCheck improves the efficiency of memory consistency validation by using (1) memory-access interleaving signature (93% reduction in logging memory operations) and (2) collective graph checking (81% reduction in graph-checking time)

10. Technology Transfer
- Industry interaction and internship
- IBM Research in Israel (Summer 2015)
- Post-silicon validation research is motivated from this internship
- An extended version has been presented at the 44th International Symposium on Computer Architecture (ISCA 2017)
- Source code available at: https://github.com/leedoowon/MTraceCheck

Research goal: Improving the efficiency of memory consistency validation

Task 8.1 – Extreme scaling for design and verification

Image 168x1897 to 330x2077