Data Race Detection in C/C++ Concurrent Programs

CS4405 - Analysis of Concurrent and Distributed Programs

February, 2022

Project description

In this project, you will detect data races in the executions of C11 concurrent programs. [1]

Background Information

C11 concurrency

C/C++ defines relaxed memory concurrency model which is known as C11 concurrency model [1]. C11 has various kinds of accesses that affect shared memory concurrency. To begin with, it provides plain or non-atomic load and store accesses. In addition, C11 also has atomic accesses of four kinds: load, store, atomic update (RMW) such as compare-and-swap and atomic increment, and memory fence. Each atomic access is attached with a memory order from: relaxed, acquire, release, acquire-release, sequentially-consistent.

Data race

An execution has a data race if there are concurrent memory accesses on same memory locations and at least one of them is a write access. Given a data race if both memory accesses are writes then it is a write-write race. If it is between read and write accesses then we say it is a read-write race. Moreover, we may categorize data race as

1. Non-atomic-race: where at least one access is non-atomic access.
2. Relaxed-race: where at least one access is a relaxed access.
3. RA-race: where at least one access is non-SC access.

Execution

An execution consists of a set of events resulting from shared memory accesses or fences, and relations between these events. Further details are in [1, 2, 3].

c11tester

Given a C11 program the c11tester tool [2, 3] may execute the program and generate execution traces with events and relations as discussed above. Currently c11tester identify data races on non-atomic accesses. In this project you will generate the traces and identify the other types of data races.

Note: *feel free to use any other tool if you like.*

Roadmap for the project:

The project involves the following steps:

- Set up the c11tester tool.
- Write C11 test programs.

[1] The project description is subject to small changes and updates. Please contact the TA's and the teachers if you have any questions.

- Generate execution traces from C11 programs (and write it in a file).
- Develop the algorithms for data race detection on the generated trace.
- Implement the algorithm to check if an execution contains a data race. The implementation can be independent of c11tester tool.
- Evaluate on the c11tester benchmarks*.

Note:

- You may use c11tester inside the vagrant box to generate the traces, if you face difficulty in installing it from source code.
- The evaluation on the 'cdschecker-benchmarks' suffice. You may generate larger trace by changing the input.
- Some applications (e.g. firfox) require significantly more computation and memory. You may skip these.

References

[1] Mark Batty, Scott Owens, Susmit Sarkar, Peter Sewell, and Tjark Weber. Mathematizing C++ concurrency. In *POPL'11*, pages 55–66. ACM, 2011.

[2] Weiyu Luo and Brian Demsky. *C11Tester: A Race Detector for C/C++ Atomics*, page 630–646. 2021.

[3] Weiyu Luo and Brian Demsky. C11tester artifact.