Ranking of Data Race Candidates

Dependable Systems - Dirk Nowotka

Project Description

The goal of the project is to come up with one or more ranking heuristics for data race candidates and then create a prototype implementation to evaluate their usefulness.
During the execution of concurrent programs there are multiple possibilities on how each thread execution gets arranged into a global execution. A *race condition* occurs if during program execution there are at least two active threads and the behaviour of the program depends on which thread gets scheduled for execution next.
A *data race* happens when two concurrent program execution threads access a common location in memory, the accesses are not protected by explicit synchronisation, and at least one thread is writing. Data races are race conditions because a program may behave differently depending on which thread gets to access the memory location first.
Most race conditions are harmless. They happen all the time in concurrent program executions because of non-determinism in the kernel scheduler. In most cases, however, a data race can be considered a serious bug, because the programmer was not aware of its possibility. Thus, the program may misbehave in an unexpected way.
Data races are notoriously hard to find by software testing. There are at least two reasons for this. Firstly, the schedule that produces the bug may only occur rarely, because of non-deterministic scheduling. Secondly, even if the scheduling is deterministic there are too many possible schedules to test them all.
Static analysis is a technique to compute an over-approximation of the behaviour of programs without executing them. It can reveal all hidden data races but may also report false-positives, i.e. data races that would not be possible during real program executions.
A list made of data race candidates generated by a static analysis method may not be helpful if the list is too large or contains too many irrelevant entries. That would make reviewing the list take too long. A *ranking heuristic* sorts the list such that the most interesting data race candidates are at the beginning.
Work on this project consists of learning about data races in concurrent C code, generating a list of data race candidates with the static analysis tool *Gropius* (developed by Dependable Systems), developing heuristics to sort the list of data race candidates, and finally check the usefulness of those heuristics by implementing them as prototypes.

Applicable For

Bachelorstudents ☑
Masterstudents ☑

Skillset

Programming ●●●●○

Keywords

Static Analysis
Concurrent Software
Data Races

Contact

Philipp Sieweck
@ psi@informatik.uni-kiel.de