Related Work

June 2016

Burckhardt et al. developed Line-Up[1], an automatic checker for deterministic linearizability. The tool claims to be complete, as reported violation proves the implementation non-linearizable with respect to any deterministic sequential specification. It works in two phases. In the first phase it records invocation sequences of each thread, While in the second phase it checks whether the concurrent executions are consistent with the specifications recorded in phase 1.

Intel Thread Checker[2] is a dynamic analysis tool that detects thread related defects in a multithreaded application. It instruments the source code and then executes it to detect the runtime behaviour of each thread in the program. It instruments memory and synchronization instructions. Memory instructions instrumentation records the memory address and type of access, whereas, synchronization instructions instrumentation is used to establish a happens before relation. This happens before relation is used to build a vector clock that detect data races.

In a multithreaded environment, an atomicity violation can lead to a data race. AtomTracker[3] is a comprehensive approach that can automatically infer the atomic regions and detect their violation at runtime. It has two parts. AtomTracker-I, infers the atomic regions by analysing the memory traces of the test runs of program. It greedily joins the successive references of a thread into an atomic region if it does not conflict with other threads. AtomTracker-D analyses the set of atomic regions and detects any violation at runtime.

Falcon[4] is a dynamic fault localization technique that can detect faulty access pattern of the shared variables in multithreaded program. It first analyses memory access patterns of the shared variables among the threads online, records unserializable and conflicting interleaving patterns, and associates it with a pass/fail results. second it applies some statistical technique to calculate the suspiciousness value for these detected patterns and rank these patterns according to their suspiciousness values. It detects both order and atomicity violations.

Data races are one of most common bugs in concurrent program. Lockset based approach and happens before(HB) are some standard techniques that are being used for data race detection for past two decades. Lockset based approaches may falsely alarm some data races whereas, happens before relation can miss some data races. Considering the limitations of these techniques, Smaragdakis et al. proposed a new relation, Causally precedes(CP)[5] that generalises happens-before to observe more data races and still remain sound. CP relaxes some HB edges to detect data races that HB can miss. A CP release-acquire edge between *(a) critical sections over the same lock that contain conflicting events, (b) critical sections over the same lock that contain CP ordered events.* Unlike happens-before, CP is not a reflexive relation and so some edges are relaxed.

References

[1] Sebastian Burckhardt, Chris Dern, Madanlal Musuvathi, and Roy Tan. Line-up: a complete and automatic linearizability checker. In *ACM Sigplan Notices*, volume 45, pages 330–340. ACM, 2010.

[2] Utpal Banerjee, Brian Bliss, Zhiqiang Ma, and Paul Petersen. Unraveling data race detection in the intel thread checker. In *In Proceedings of STMCS '06*, 2006.

[3] Abdullah Muzahid, Norimasa Otsuki, and Josep Torrellas. Atomtracker: A comprehensive approach to atomic region inference and violation detection. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 287–297. IEEE Computer Society, 2010.

[4] Sangmin Park, Richard W Vuduc, and Mary Jean Harrold. Falcon: fault localization in concurrent programs. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 245–254. ACM, 2010.

[5] Yannis Smaragdakis, Jacob Evans, Caitlin Sadowski, Jaeheon Yi, and Cormac Flanagan. Sound predictive race detection in polynomial time. In *ACM SIGPLAN Notices*, volume 47, pages 387–400. ACM, 2012.