Fault Localization in Concurrent System

Chiyu Cheng

What is Fault Localization and why is it important.

Testing is one of most critical and expensive task in the development of software. The process of detecting and locating bugs in software system is called fault localization. Manually fault localization is acceptable for the small software system but unbearable for the large scale system. Therefore, an automated fault localization technique is necessary for the real-world complex software system.

What is Concurrent System and why is it hard to perform fault localization technique

A **concurrent system** is one where a computation or operation can be executed without waiting for all other operations to complete. It generally involves the coordination of multi-thread/process/cores to run a program. The concurrency fault is difficult to find because an concurrent system may have large number of thread interleavings, which makes it is hard to reproduce a fault in one specific thread interleavings.

Current State of Arts

Falcon: Fault Localization in Concurrent Programs.

http://delivery.acm.org/10.1145/1810000/1806838/p245-park.pdf?ip=169.234.98.40&id=1806838&acc=ACTIVE%20SERVICE&key=CA367851C7E3CE77%2EE385B6E260950907%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1556657396_4af76bac3e605d754a5ae091da05238a

This paper provides a dynamic fault-localization technique that can pinpoint faulty data-access patterns in multi-threaded concurrent programs. It monitor memory access among multiple threads and try to catch the data-access pattern while the program fails or success.

Cooperative Crug Isolation *

A. Thakur, R. Sen, B. Liblit, and S. Lu. Cooperative crug isolation. In WODA, pages 35–41, 2009.

This approach has the similar idea with the previous one except that it collects the shared variable accesses. For example, it design predicates to see whether a previous variable access is thread-local or thread-remote. CCI can compute the suspiciousness scores based on the pass/fail result of the predicates.

More possible technique

“testing based on random execution interleavings “

S. D. Stoller. Testing concurrent Java programs using randomized scheduling. In Proc. Second Workshop on Runtime Verification (RV), volume 70(4) of Electronic Notes in Theoretical Computer Science. Elsevier, July 2002

exhaustive context-bounded testing

M. Musuvathi and S. Qadeer. Iterative context bounding for systematic testing of multithreaded programs. PLDI., 42(6):446–455, 2007.

Race-directed random testing

K. Sen. Race directed random testing of concurrent programs. In PLDI, 2008.

Evaluation Framework

In this project, I want to develop an evaluation framework to study and evaluate the current state of art in finding the concurrency fault. The plan is surveying several papers in this program and evaluating those methods under the same standard. The basic criteria are feasibility, efficiency, and accuracy. We will test if each method is doable, and record the performance of each method on the real system with concurrency bug. Like the SPLASH-2 programs

Can we locate the fault in the large scale system easily?