Review of

Learning from Mistakes

Shan Lu et al.

What is the problem?

Multi-core systems call for concurrent programming but writing robust, sound concurrent programs is difficult. There are too many parameters to consider.

Summary

It provides a comprehensive real-world concurrency bug characteristic study. It examines concurrency bug patterns, manifestations, and fix strategies of randomly selected real world concurrency bugs. It demonstrates how designing effective techniques need a deep understanding of real world concurrency bug characteristics.

Key Insights

- This paper provides a comprehensive study of the real world concurrency bugs, examining their pattern, manifestation, fix strategy and other characteristics.
- The paper provides useful guidance for concurrency bug detection, testing, and concurrent programming language design

Strengths

- The paper delves deep into finding the real world concurrency bug characteristics and their implications for the bug detection, and testing. The table 1 in the paper does a great job of explaining this.
- They have collected real world bugs from the wild, from random and diverse open-source programs. This is a strength because this removes any bias from one type of platform and helps promote the usability of this study.

Weaknesses

- The paper discusses about how some concurrency bugs are difficult to repeat and they don't elaborate on how they solve this or how future work can address it.

Summary of Key Results

- The paper says that about 73% of the examined non-deadlock concurrency bugs were not fixed by simply adding or changing locks, and many of the fixes were not correct at the first try, indicating the difficulty of reasoning concurrent execution by programmers.
- The paper shows that around one third of the examined non-deadlock concurrency bugs are caused by violation to programmers' order intentions, which may not be easily expressed via synchronization primitives like locks and transactional memories.

- Around 34% of the examined non-deadlock concurrency bugs involve multiple variables, which are not well addressed by existing bug detection tools.

Open Questions

- How do we work on the concurrency bugs that are not reproduce-able or not easily reproduced?
- How can debugging tools be built using the knowledge from this paper?