Falcon: Fault Localization in Concurrent Programs

周昊一 MF1233055

随着基于多核处理器的平行系统的发展，并发问题导致的bug越来越常见。这类问题不但是最难发现，最难修复的，并且修复所需的时间往往需要好几天。至少有两个原因使并发执行的程序比顺序执行的程序难以测试与调试：1.并发程序有更多的非决定性行为，导致连bug的重现都有困难；2.并发程序的bug通常涉及到程序状态的改变以及多个线程的参与，难以定位和理解。

本文提出了一种基于模式的动态分析技术，用来定位并发程序中的bug。在测试过程中，这种技术检测程序实际运行过程中存取的模式，并通过分析这些模式成功与否的统计数据来计算他们的可疑程度。同时本文还介绍了FALCON，一个在Java下这种技术原型的实现。

这种新技术对于已有技术的优点在于：1.它不仅检测顺序上的错误，还检测原子性上的错误；2.它报告高优先级的真实存在的模式，而其他一些技术会报告许多良性错误，并且没有优先级；3.它在时间和空间上的效率更高。

并发错误主要有以下几种：

1. 数据竞争。当两个或更多的线程同时读写一个共享内存空间时（至少有一个是写），并且没有同步锁，就会引起数据竞争。
2. 顺序错误。两个顺序执行的线程对同一块共享内存空间进行读写操作（至少有一个是写），就是一个冲突的交错模式。
3. 原子性错误。当一个不可序列化的交错模式造成了不期的程序行为，就说发生了一个原子性错误。

文中给出了一个三线程的例子来解释这三种错误。

文中所提出的新技术在检测Java线程中的并发bug时共需要两个步骤。在第一步中，在线监测共享内存空间的读取检测出无法序列化的，以及冲突的交错模式，并将这些模式和测试用例以及结果关联起来。与已有方法不同，在这一步中并不是运行多个测试用例，而是将一个测试用例运行多次。在第二步中，对第一步中发现的模式进行可疑程度的分级，所采用的评分方式是Jaccard Index。

随后，文章介绍了技术的具体实现FALCON，并结合一些案例对其进行了评估。