COBET: Effective COncurrency Bug dETector Framework for OS Kernel
Hong, Shin (advised by Prof. Moonzoo Kim)

Introduction

Motivation
As multi-core hardware becomes increasingly powerful and popular operating systems (OSes) such as Linux utilize the cutting-edge multitasking, concurrency techniques heavily to enhance performance.

However, current analysis techniques and tools for concurrent programs are not yet mature enough to support OS developers in a practical manner due to the unique characteristics of the kernel programming. In particular, the following three obstacles hinder analysis of the concurrent behavior of OSes:

• Various customized synchronization primitives
• OS developers sometimes implement their own synchronization primitives. Concurrency bug detection tools for standard synchronization mechanisms do not recognize these customized synchronization mechanisms and produce imprecise results.
• Limitations of lock-based bug detections

Most available bug detection techniques focus on low-level data races through the analysis of binary lock usages. However, OSes exploit various synchronization mechanisms for performance. In addition, high-level data races and atomic violations are more difficult to detect than low-level data races.

• Lack of scalability

A dynamic analysis often fails to uncover hidden concurrency bugs due to the exponential number of possible interleaving scenarios.

Experiments

Bug Detection Result on File Systems
We applied the five COBET bug pattern detectors to the seven Linux file systems. The result shows the number of alarms was modest. (5 bug patterns X 7 file systems result 42 warnings)

Evaluation of Semantic Analysis Techniques
To demonstrate the effectiveness of COBET semantic analyses, we measured the false alarm reduction rate through the semantic analyses and the additional time cost.

The result shows additional analyses reduce false alarms and the time cost were not burdensome.

Bug Detection Results on Device Drivers and Network Modules
To demonstrate applicability, we applied 5 pattern detectors to Linux device drivers and network modules.

COBET Framework

Overview
Most concurrency errors are caused by unintended interferences among concurrently executing multiple threads. Therefore, to detect a concurrency bug accurately, the concurrency bug pattern should be modeled with multiple code patterns, each of which captures a specific code to be executed on each thread. Furthermore, it is necessary to check whether these code patterns are possible to be executed concurrently at the same time, or not.

Pattern Detector Construction
The construction process has the three steps:
1. A user specifies the syntactic characteristics of a bug pattern in PDL.
2. The COBET synthesizer translates the PDL description to the corresponding bug pattern detector template code. The template code performs the syntactic pattern matching and calls sem_cond_checking() to check the semantic conditions.
3. The user fills out sem_cond_checking() to provide a semantic condition checking routine to complete the bug pattern detector.

Five bug patterns with semantic conditions

Further work

• Patternize conventional data race and atomicity violations

Many bug definitions used for standard lock-based concurrency bug detection techniques such as data race or atomicity violations can be represented or approximated to COBET patterns.

• Apply to Application-level programs

Large-scale application programs such as HTTP daemon or DBMS systems suffer similar difficulties to Linux kernel programming. We plan to apply COBET approach and COBET bug patterns to find bugs in these domains and compare the result to the case of Linux kernel.

• Enhance PDL to include semantic conditions specification

Currently, PDL only specifies syntactic aspects of bug patterns. We plan to extend PDL to associate essential semantic conditions to improve usability of COBET framework.