Detecting Flaky Tests by Comparing Program Traces

Master's Thesis or Guided Research

Supervisor: Prof. Dr. Alexander Pretschner
Advisor: Fabian Leinen
Email: {alexander.pretschner, fabian.leinen}@tum.de
Starting date: flexible

Context

When testing software, we assume that a test passes if the code under test is free from failures for the input defined in the test and assume that the test fails if not. In other words, we expect a test to be deterministic, which doesn't always hold true. Think of an integration test that accesses a database which might be available 99% of the time. That means that the test passes 99 time out of 100 but also fails 1 times. This kind of tests, that fail inconsistently without changes to the code under test or the test itself are called flaky tests [1]. According to a keynote from 2017, almost 16% of all tests at Google are flaky and they spend between 2 and 16% of their compute resources only on re-running flaky tests [2]. The most common strategy to detect flaky tests is to re-run tests. Detecting a flaky test with 95% confidence requires 170 re-runs [3] which is costly, especially if these 170 runs need to be performed on every change.

An approach already exists that compares program states of different runs to identify the location of the root causes of flakiness to guide developers in fixing the issue [4]. If this approach can be used to detect flaky tests is not yet determined.

Goal

The goal of this work is to investigate whether the number and costs of re-runs can be reduced while maintaining a high detection certainty by comparing not only test outcomes but also test traces. Using the number of re-runs as a metric can guide future research on this topic, while the economical view is important for real world applications. To reach this goal, a suitable level of tracing must be chosen. Among other possibilities, these are the options:

- Annotated statement coverage using Python's trace module
- Program states by manually instrumenting code
- Flamegraphs
- System calls
- Environment data, e.g. CPU load, RAM, ...

In order to ensure comparability, the evaluation framework of [3] must be adapted. In addition, the results should be evaluated with regard to their transferability to other data, e.g. from other programming languages.

In addition, the possibility of using the developed method to check if fixing a flaky test really fixes the flakiness should be considered.

Working Plan

1. Familiarize yourself with the literature and the given dataset [3].
2. Make an informed decision about the level of tracing that's required and feasible to capture.
3. Conduct experiments and evaluate results to meet the above given goals.
4. Record your literature results, design decisions, implementation, and application results in form of a thesis.

Deliverables

- Source code of the implementation.
- Technical report with comprehensive documentation of the implementation, i.e. design decision, architecture description, API description and usage instructions.
- Final thesis report written in conformance with TUM guidelines.

Application for Thesis

Please apply for this thesis topic with your CV, grade report, and a short motivation, why you are interested in this topic. Based on these documents, we will invite some students for a personal meeting to see if the topic fits the student.

Please note that the student working on this thesis should be familiar with Python. Knowledge about software testing is highly desirable.

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17362
https://www4.in.tum.de

References

[1] Luo, Qingzhou & Hariri, Farah & Eloussi, Lamyaa & Marinov, Darko. (2014). An empirical analysis of flaky tests. 643-653.

[2] The State of Continuous Integration Testing @Google https://research.google/pubs/pub45880/

[3] Gruber, Martin & Lukasczyk, Stephan & Kroiß, Florian & Fraser, Gordon. (2021). An Empirical Study of Flaky Tests in Python.

[4] C. Ziftci and D. Cavalcanti, "De-Flake Your Tests : Automatically Locating Root Causes of Flaky Tests in Code At Google," 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2020, pp. 736-745.

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17362
https://www4.in.tum.de