An Empirical Study of Flaky Tests in Python

Martin Gruber[1] Stephan Lukasczyk[2] Florian Kroiß[3] Gordon Fraser[4]

Abstract: This is a summary of our work presented at the International Conference on Software Testing 2021 [Gr21b]. Tests that cause spurious failures without code changes, i.e., *flaky tests*, hamper regression testing and decrease trust in tests. While the prevalence and importance of flakiness is well established, prior research focused on Java projects, raising questions about generalizability. To provide a better understanding of flakiness, we empirically study the prevalence, causes, and degree of flakiness within 22 352 Python projects containing 876 186 tests. We found flakiness to be equally prevalent in Python as in Java. The reasons, however, are different: Order dependency is a dominant problem, causing 59 % of the 7 571 flaky tests we found. Another 28 % were caused by test infrastructure problems, a previously less considered cause of flakiness. The remaining 13 % can mostly be attributed to the use of network and randomness APIs. Unveiling flaky tests also requires more runs than often assumed: A 95 % confidence that a passing test is not flaky on average would require 170 reruns. Additionally, through our investigations, we created a large dataset of flaky tests that other researchers already started building on [MM21; Ni21].

Keywords: Flaky Test; Python; Empirical Study

1 Methodology

To investigate the diffusion and nature of test flakiness in Python, we scanned the Python package index (PyPI) for repositories with `pytest`-compatible tests, collecting a set of 22 352 projects containing a total of 876 186 test cases. We identified flaky tests within these projects and made a first distinction between order-dependent and non-order-dependent flakiness by executing each project's tests 200 times in default order and 200 times in random order. Splitting the test executions further into 20 batches of 20 reruns each allowed us to also detect infrastructure related problems, which manifest in the passing and failing of entire batches (which are executed on the same machine). Lastly, to retrieve a more fine-grained distinction within non-order-dependent flaky tests, we manually classified 100 of them along previously established root causes. To aid this process, we searched the execution traces of these tests for keywords which indicate specific types of flakiness.

[1] BMW Group | University of Passau, Germany martin.gr.gruber@bmw.de
[2] University of Passau, Germany stephan.lukasczyk@uni-passau.de
[3] University of Passau, Germany kroiss@fim.uni-passau.de
[4] University of Passau, Germany gordon.fraser@uni-passau.de

2 Results

Through repetitive reruns we encountered 7 571 flaky tests in 1 006 projects. 59 % of them are flaky only when executed in a certain test order, therefore being caused by order-dependencies. 28 % are related to infrastructure issue, and the remaining 13 % of non-order-dependent flaky tests are mostly caused by the usage of network and randomness APIs. While these results show that flaky tests are equally prevalent in Python as they are in Java (about one in 200 tests is flaky), the inspection of their root causes reveals strong differences, specially concerning non-order-dependent flakiness: While previous studies found concurrency, especially asynchronous waiting, to be the most common cause of flakiness, we find only little evidence for these in Python. On the other hand, networking and randomness—which are regarded as minor issues by other investigations—are causing 79 of the 100 non-order-dependent flaky tests we inspected. We suspect that these differences are caused by the type of software typically written in Python, which includes scientific software and web application.

Besides the diffusion and the nature of flaky tests in Python, we also investigated the number of reruns needed to unveil a test's flakiness. We abandoned the practice of basing this decision on the number of reruns needed to reveal a test's flakiness once, due to its instability and bad reproducibility. Instead, we developed our own technique by creating a probabilistic model. Our results suggest that in order to be 95 % sure that a passing test case is not flaky, on average at least 170 reruns are required. On the other hand, only one rerun is required to gain the same confidence that a test failure is not caused by flakiness.

3 Data Availability

Our published dataset [Gr21a] includes the test outcomes of all 400 test executions of each test case we investigated as well as the results of a manual classification of the root causes for 100 non-order-dependent flaky tests.

Literatur

[Gr21a] Gruber, M.: An Empirical Study of Flaky Tests in Python, Zenodo, Jan. 2021, URL: https://doi.org/10.5281/zenodo.4450434.

[Gr21b] Gruber, M.; Lukasczyk, S.; Kroiß, F.; Fraser, G.: An empirical study of flaky tests in python. In: 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST). IEEE, S. 148–158, 2021.

[MM21] Mjörnman, J.; Mastell, D.: Randomness as a Cause of Test Flakiness, 2021.

[Ni21] Nilsson, J.: Possibilities of automatic detection of „Async Wait“ Flaky tests in Python applications, 2021.