Automatic Benchmark Generation

December 5, 2022

Supervisor: Guillermo Polito

Emails: guillermo.polito@univ-lille.fr

Keywords: Compilers, Interpreters, Optimization, Code Analysis, Intermediate Languages

1 Context

Designing application benchmarks that are good representatives of application behaviour and are not subject to internal runtime noise is a hard task for application developers. The objective of this internship is to use automatic program generation techniques (i.e., program synthesis) that are aware of runtime noise sources as well as application domain knowledge. Noise awareness will minimize internal noise by construction.

2 Project and objectives

We will study what properties turn existing application tests into relevant benchmarks. We will use two main techniques. First, we will extract runtime profiling information to detect application hot spots and identify code portions relevant to performance. Second, we will use such profiling information to guide static code analyses on existing application test cases. Such a study will lead us to the automatic identification of benchmark candidates from existing application tests.

We will then investigate how application tests can be automatically turned into benchmarks. Identified benchmark candidates will not exhibit the same performance profile as the application at runtime because they are by design built to run fast and have few dependencies. We will design program generation techniques to produce macro benchmarks from benchmark candidates. Such program generation techniques will produce benchmarks that remain relevant and minimize internal noises.

The student will learn in this internship the following skills:

- How to write performance tests
- Using statistics to compare performance results
- Code generation to minimize measuring errors
- Extracting profiling data to obtain relevant performance tests

References:

- E. Barrett, C. F. Bolz-Tereick, R. Killick, V. Knight, S. Mount, and L. Tratt. Virtual machine warmup blows hot and cold. In OOPSLA17.
- A. Bergel. Counting messages as a proxy for average execution time in pharo. In Proceedings of the 25th European In ECOOP11.
- D. Costa, C.-P. Bezemer, P. Leitner, and A. Andrzejak. Whats wrong with my benchmark results? studying bad practices in jmh benchmarks. In IEEE Transactions on Software Engineering, 2021.
- Z. Ding, J. Chen, and W. Shang. Towards the use of the readily available tests from the release pipeline as performance tests: Are we there yet? In ICSE20.
- G. Polito, P. Tesone, and S. Ducasse. Interpreter-guided differential jit compiler unit testing. In PLDI22, 2022.
- L. Traini, V. Cortellessa, D. Di Pompeo, and M. Tucci. Towards effective assessment of steady state performance in java software: Are we there yet?, 2022.
- J. v. Kistowski, J. A. Arnold, K. Huppler, K.-D. Lange, J. L. Henning, and P. Cao. How to build a benchmark. In ICPE15.