Evaluating Benchmark Quality: a Mutation-Testing-Based Methodology

IWST 2025

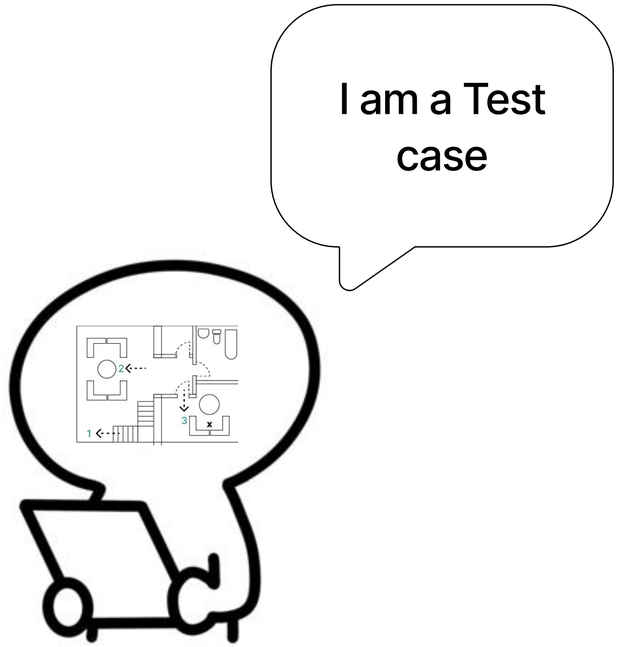
Federico Lochbaum, Guillermo Polito

Test Cases

Let’s suppose we have a broken house....

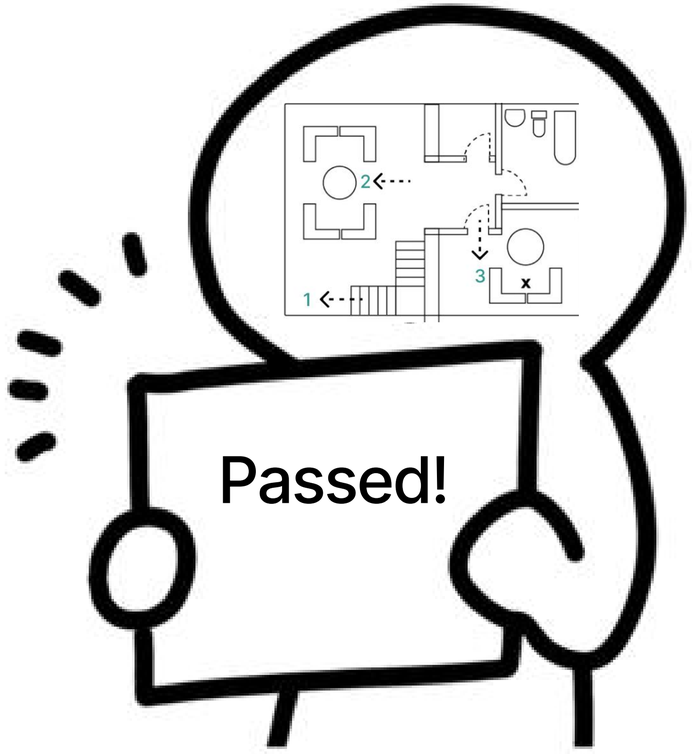
Test Cases

And we want to repair it...



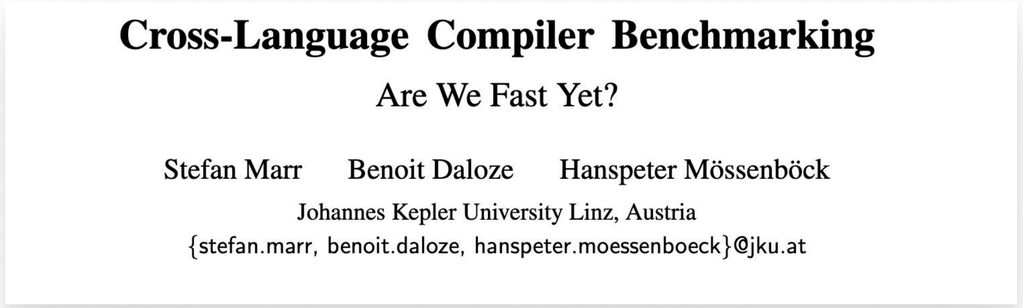
How well we did it?

How well we did it?



Benchmarks

- Measure **execution time** (wall-clock time)
- Often made by **averaging benchmarking results**, looking to reduce contextual variance
- Some work proposes using other kinds of metrics, like **energy consumption** or **memory usage**



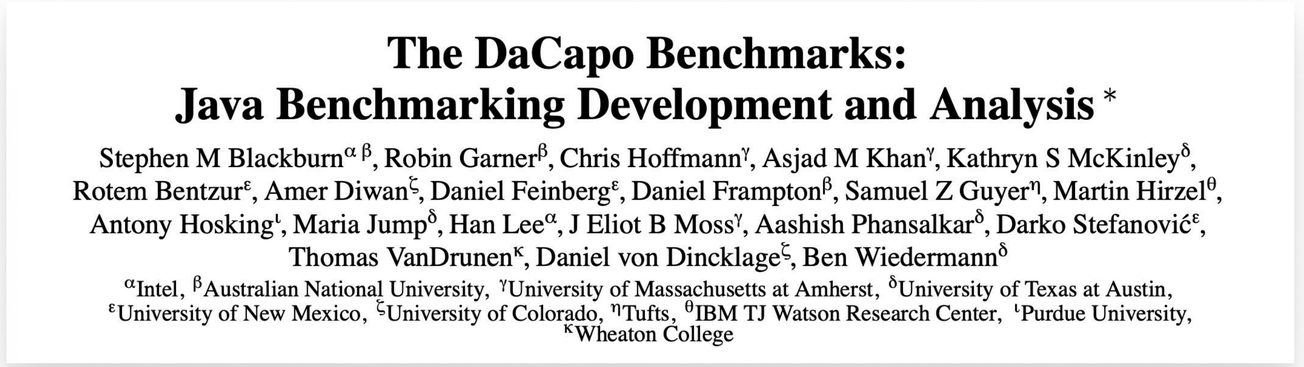
Cross-Language Compiler Benchmarking

Are We Fast Yet?

Stefan Marr Benoit Daloze Hanspeter Mössenböck

Johannes Kepler University Linz, Austria

{stefan.marr, benoit.daloze, hanspeter.moessenboeck}@jku.at



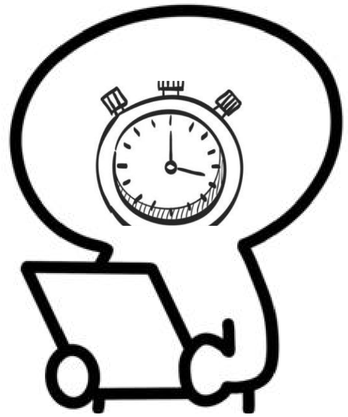
The DaCapo Benchmarks: Java Benchmarking Development and Analysis *

Stephen M Blackburn[α β], Robin Garner[β], Chris Hoffmann[γ], Asjad M Khan[γ], Kathryn S McKinley[δ], Rotem Bentzur[ε], Amer Diwan[ζ], Daniel Feinberg[ε], Daniel Frampton[β], Samuel Z Guyer[η], Martin Hirzel[θ], Antony Hosking[ι], Maria Jump[δ], Han Lee[α], J Eliot B Moss[γ], Aashish Phansalkar[δ], Darko Stefanović[ε], Thomas VanDrunen[κ], Daniel von Dincklage[ζ], Ben Wiedermann[δ]

[α]Intel, [β]Australian National University, [γ]University of Massachusetts at Amherst, [δ]University of Texas at Austin, [ε]University of New Mexico, [ζ]University of Colorado, [η]Tufts, [θ]IBM TJ Watson Research Center, [ι]Purdue University, [κ]Wheaton College

Benchmarks

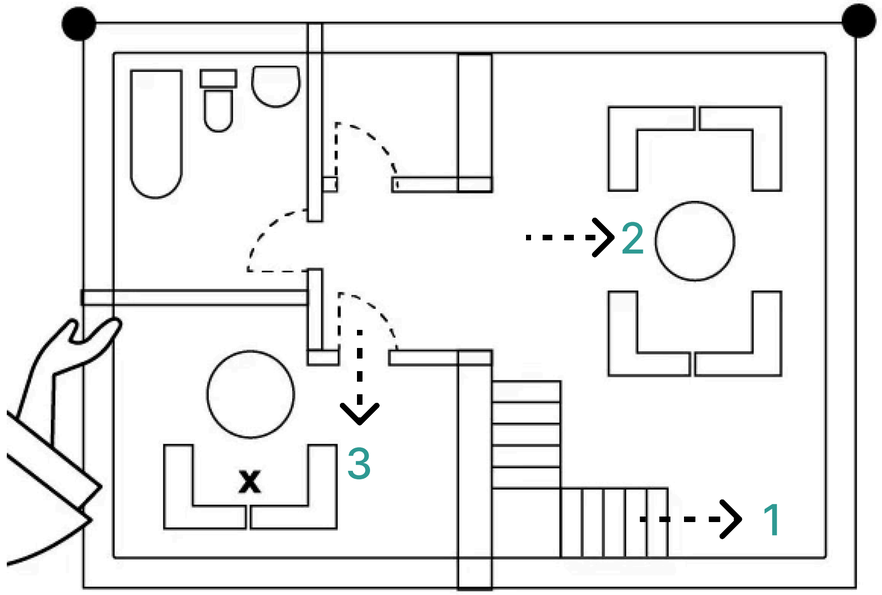
How fast we did it?



How fast we did it?

Test Cases vs Benchmarks

✔ Test Cases

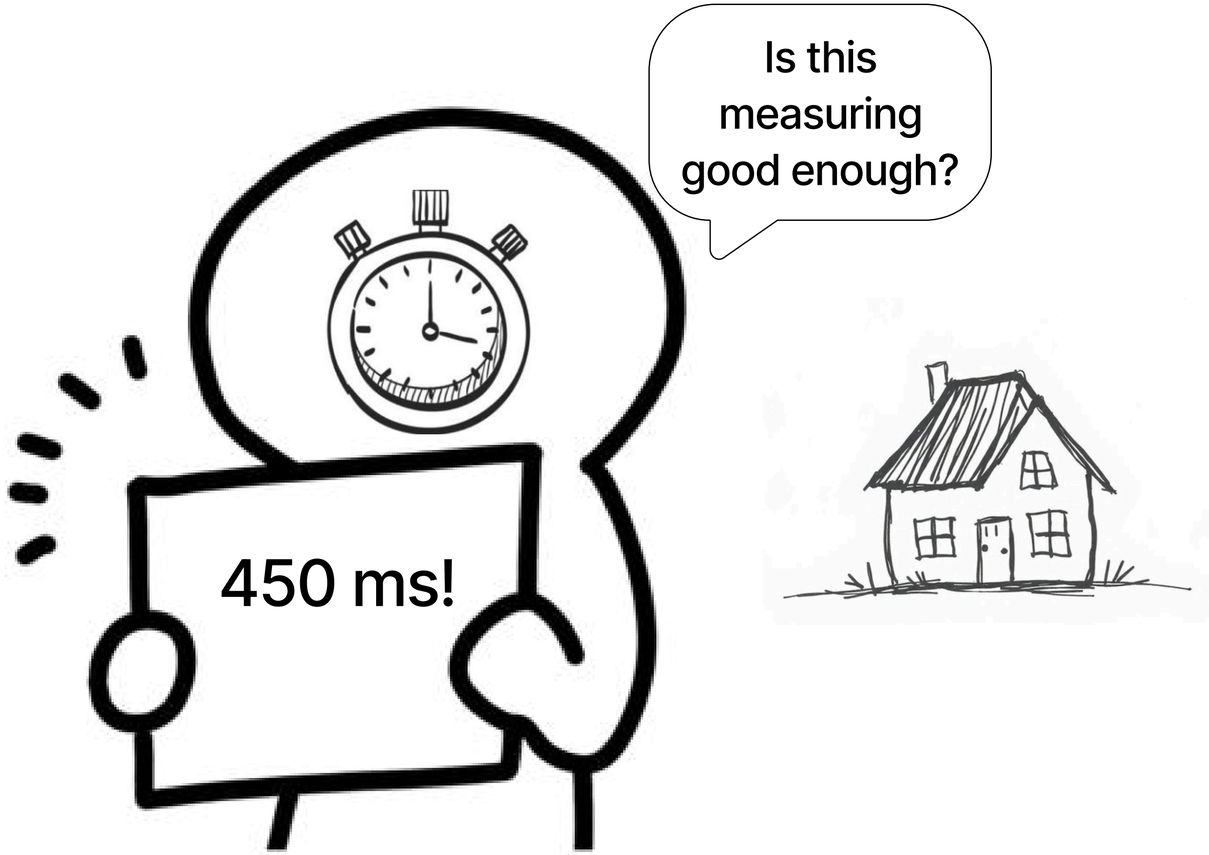


- Executes a series of steps to **validate** the program's behavior
- Check correctness (Pass / Fail)
- Self-validating
- One execution is **enough**
- Results are **architecture-independent**

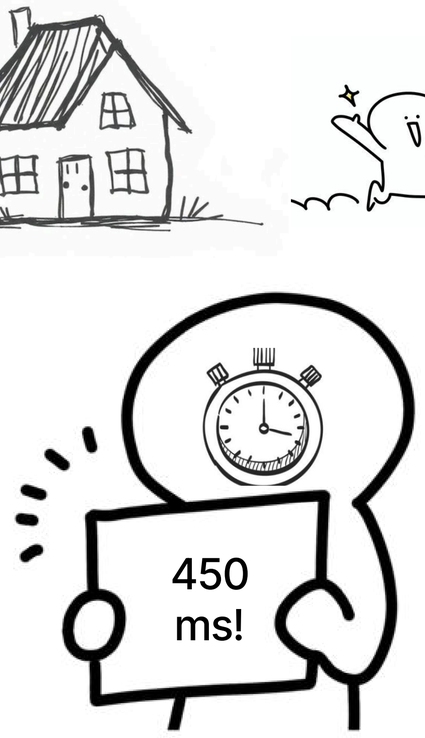
Benchmarks

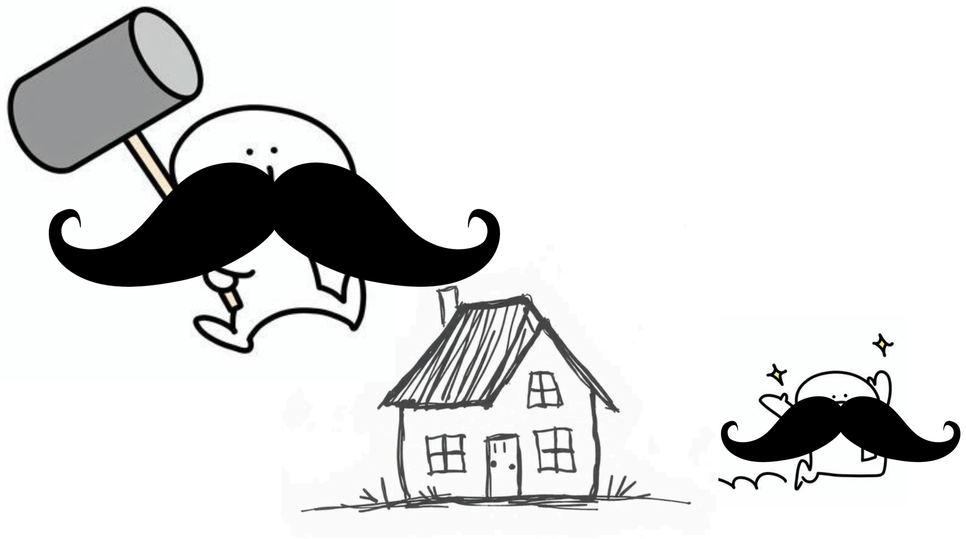
- **Stress** the program to assess performance
- Measure Performance metrics (Elapsed time / CPU)
- Not self-validating
- Require multiple runs to cope with noise
- Results are **architecture-dependent**
- **Expensive** to run

How fast we did it?



How do we know that a benchmark is “good” ?



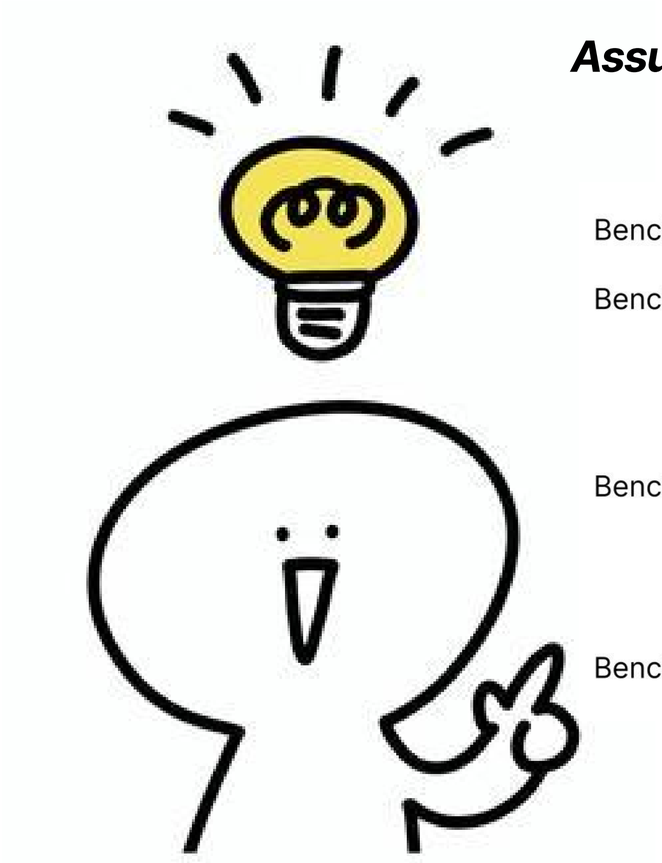


Problem: Assessing Benchmark Quality

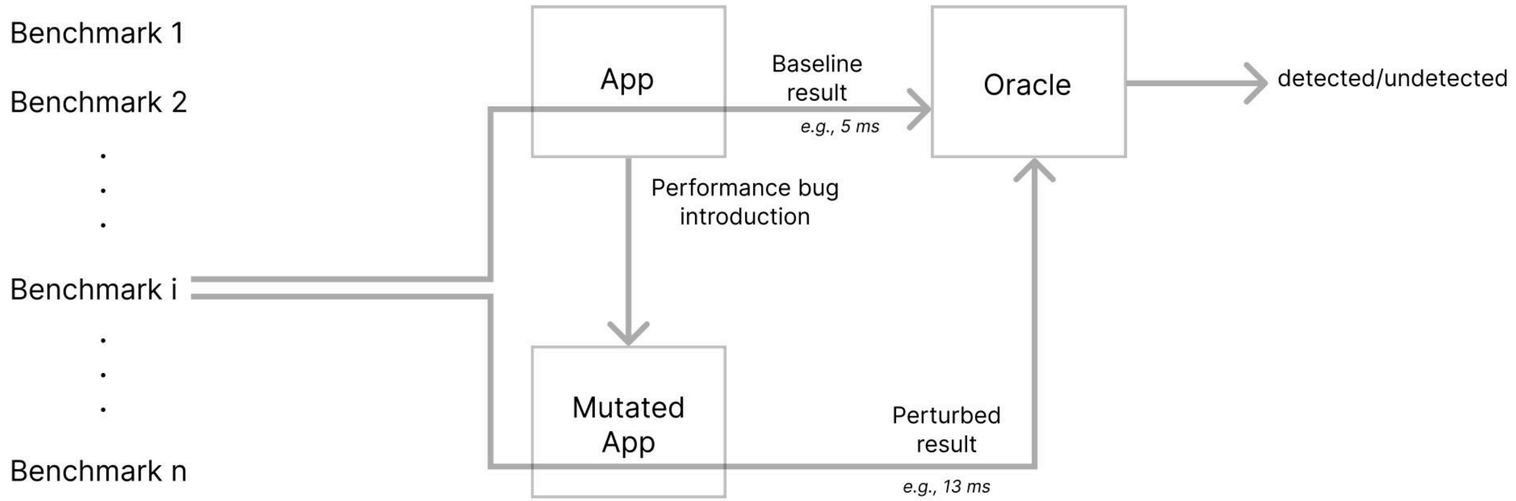
A **lack** of systematic methodologies to **assess benchmark effectiveness**

- What does it mean benchmark **quality ?**
- How to measure **benchmark effectiveness** detecting performance bugs ?
- How are introduced **performance issues** in a target program to detect them ?

Mutation Testing Benchmark Methodology - Proposal



Assumption: *A benchmark is good if it detects performance bugs*

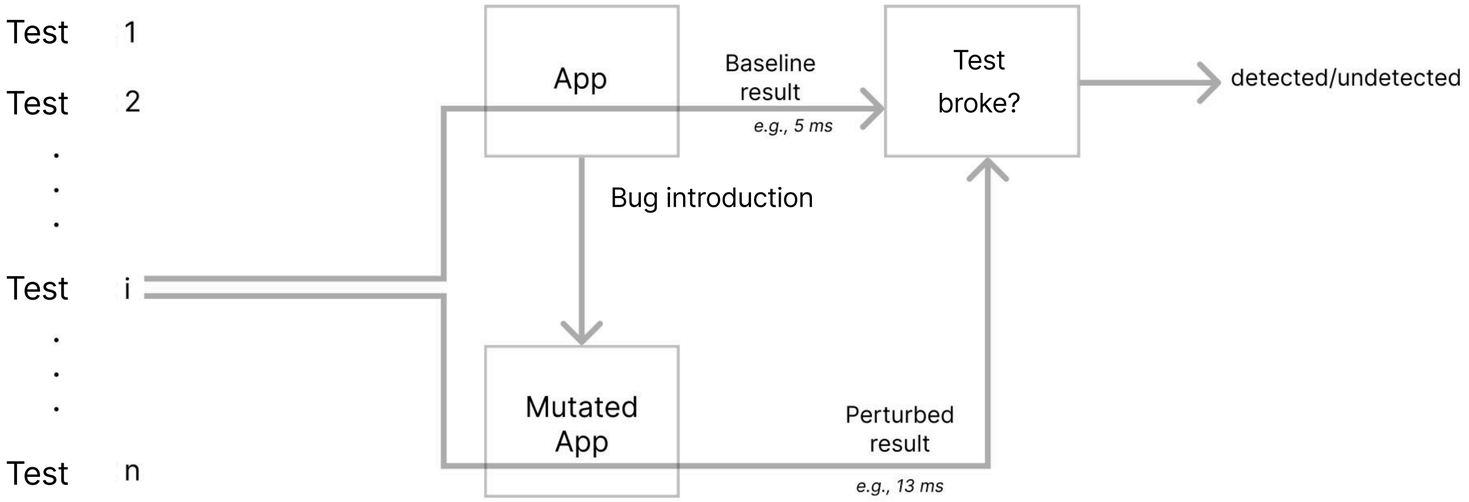


Mutation Testing ?

Mutation testing measures test quality in relation to it capability to detects bugs

It introduces **simulated bugs (mutants)** and assess if the test catches them

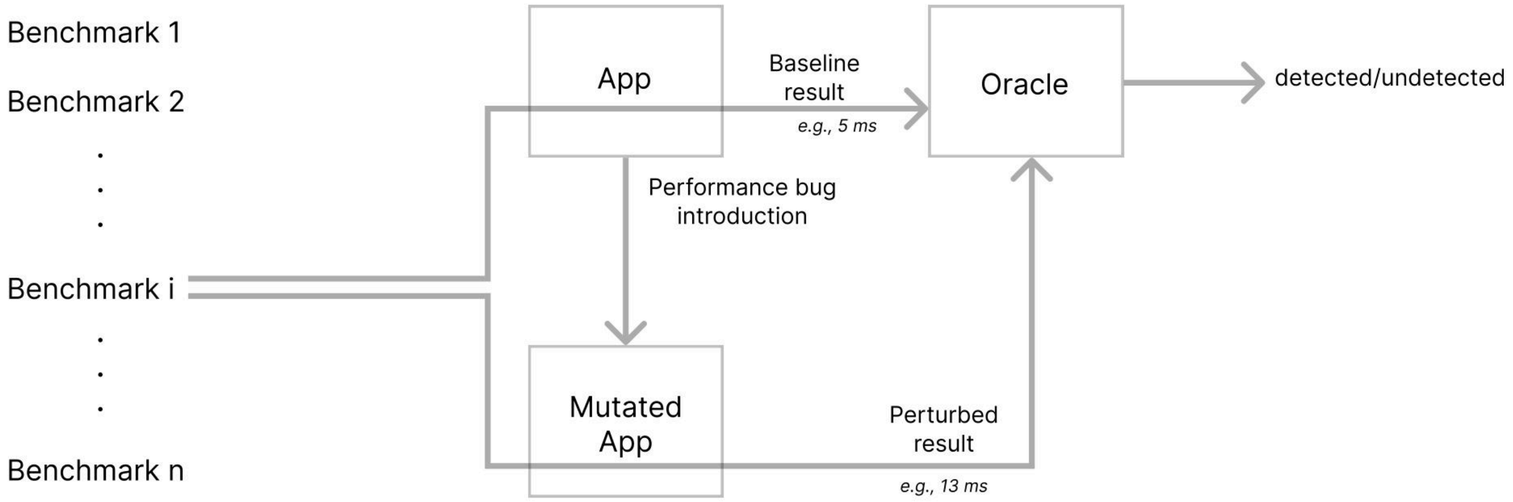
- If the test fails → the mutant was **killed (detected)**
- If not → the mutant **survived (undetected)**



Adapting Mutation Testing for Performance

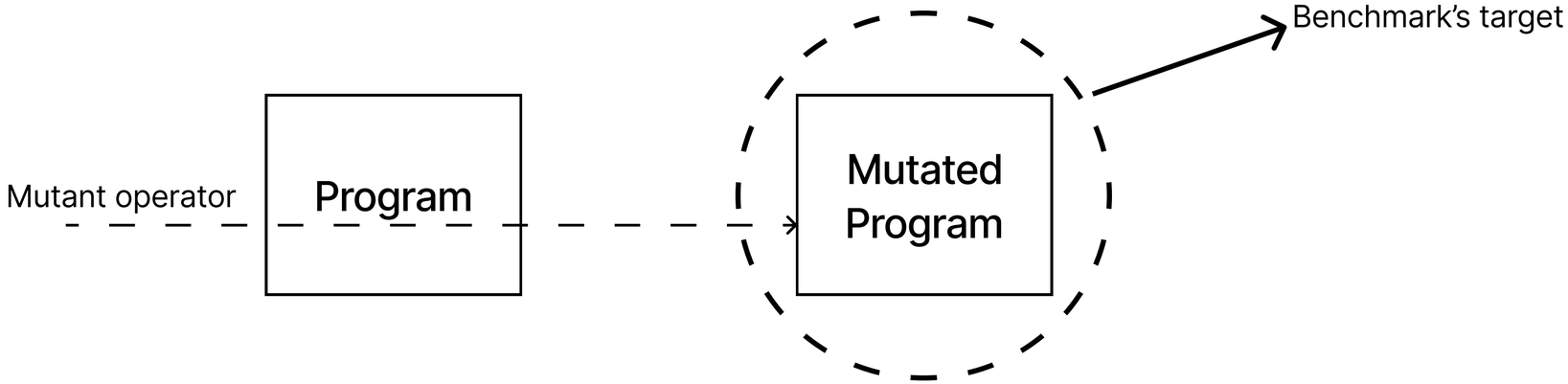
It introduces **performance bugs (mutants)** and assess if the benchmark catches them

- Is the **oracle who determines** if the benchmark kills or not the mutant



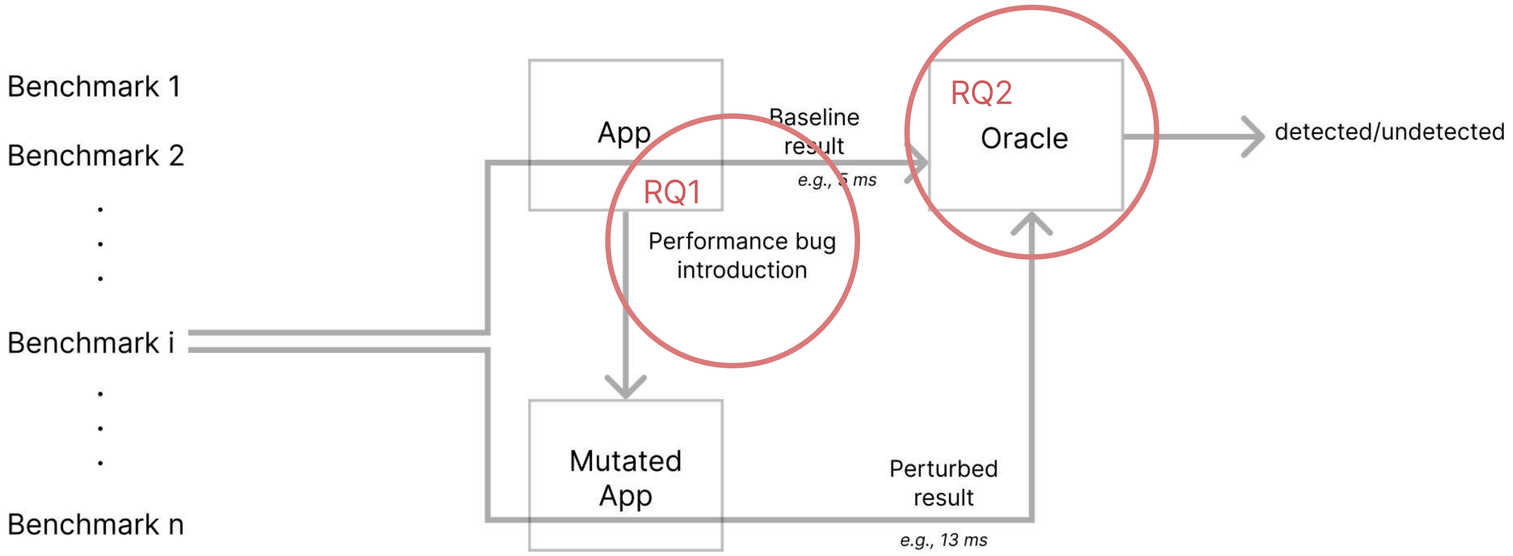
Mutation Strategy

Introduce a controlled performance bug mutant in the original program

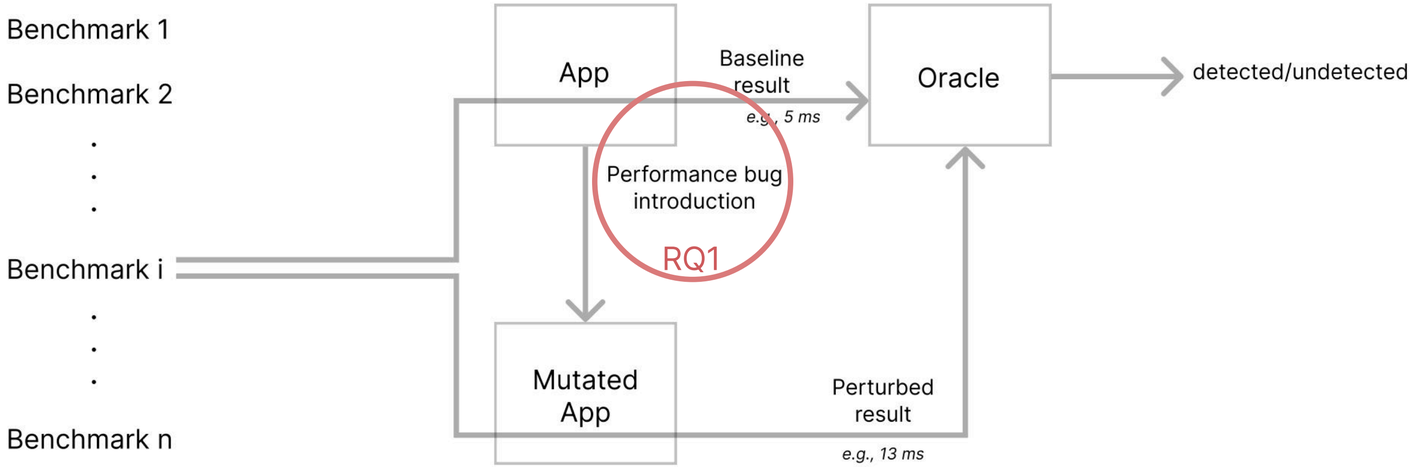


Adapting Mutation Testing for Performance

1. Introduce **performance perturbations**
2. A **performance oracle** determines if the bug had a performance impact



RQ1 - What is a Performance Bug ?



RQ1 - Performance Bug

- **Perturbation** on program execution

  (**E.g.** Latency, Locality issues)

- **Excessive** consumption of time or space by design

  (**E.g.** Long iterations, Bad implementation decisions)

- **No optimal** data structure used for a problem

  (**E.g.** Use an array instead of a dictionary to index a dataset)

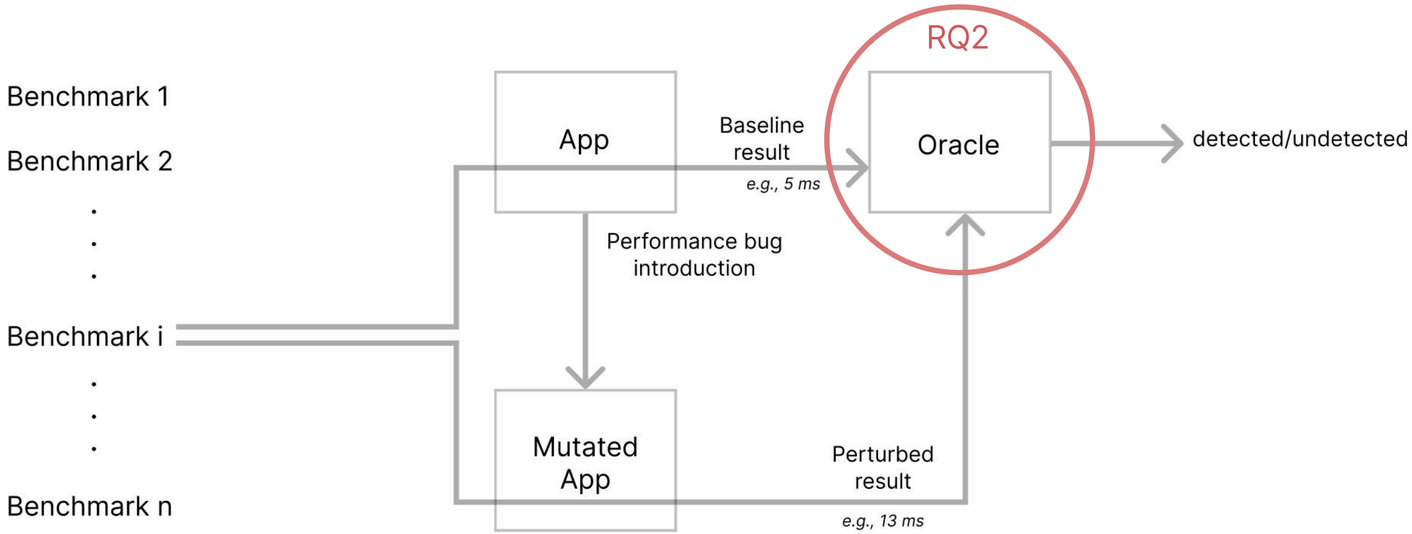
Learning from Source Code History to Identify Performance Failures

Juan Pablo Sandoval Alcocer
PLEIAD Lab
DCC, University of Chile
jsandova@dcc.uchile.cl

Alexandre Bergel
PLEIAD Lab
DCC, University of Chile
abergel@dcc.uchile.cl

Marco Tulio Valente
Federal University of Minas Gerais, Brazil
mtov@dcc.ufmg.br

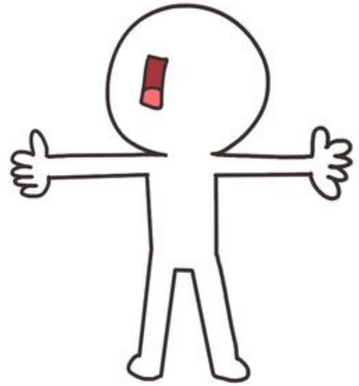
RQ2 - How do we assess a Benchmark ?



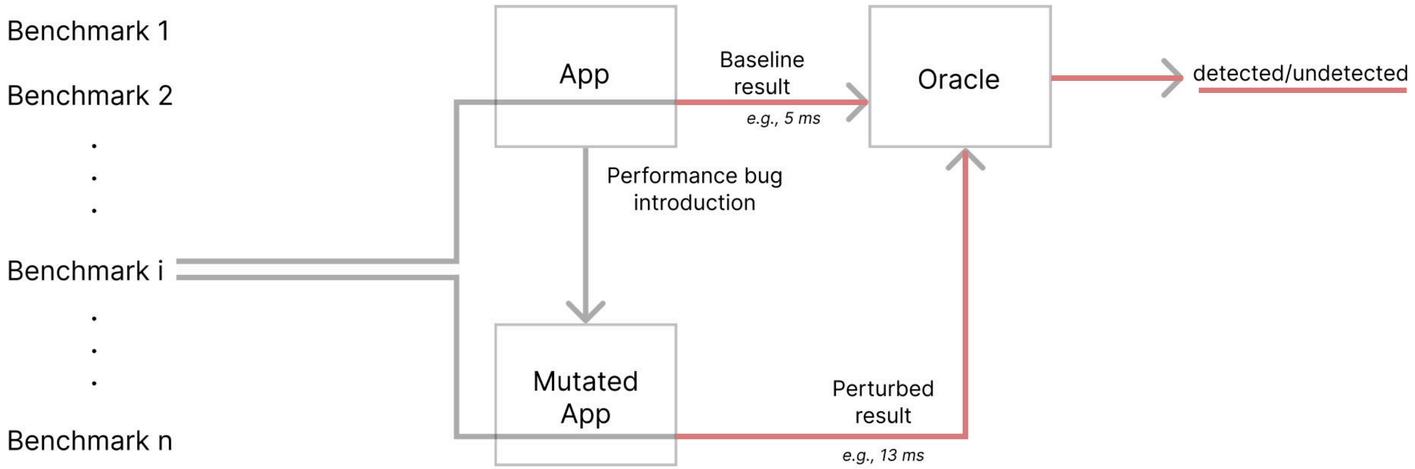
RQ2 - The Benchmark Oracle

Let's define a benchmark quality as "*How sensible is the benchmark to detect a mutant*"

What it is a benchmark sensibility ?



Where is the threshold, and what do we compare it against?



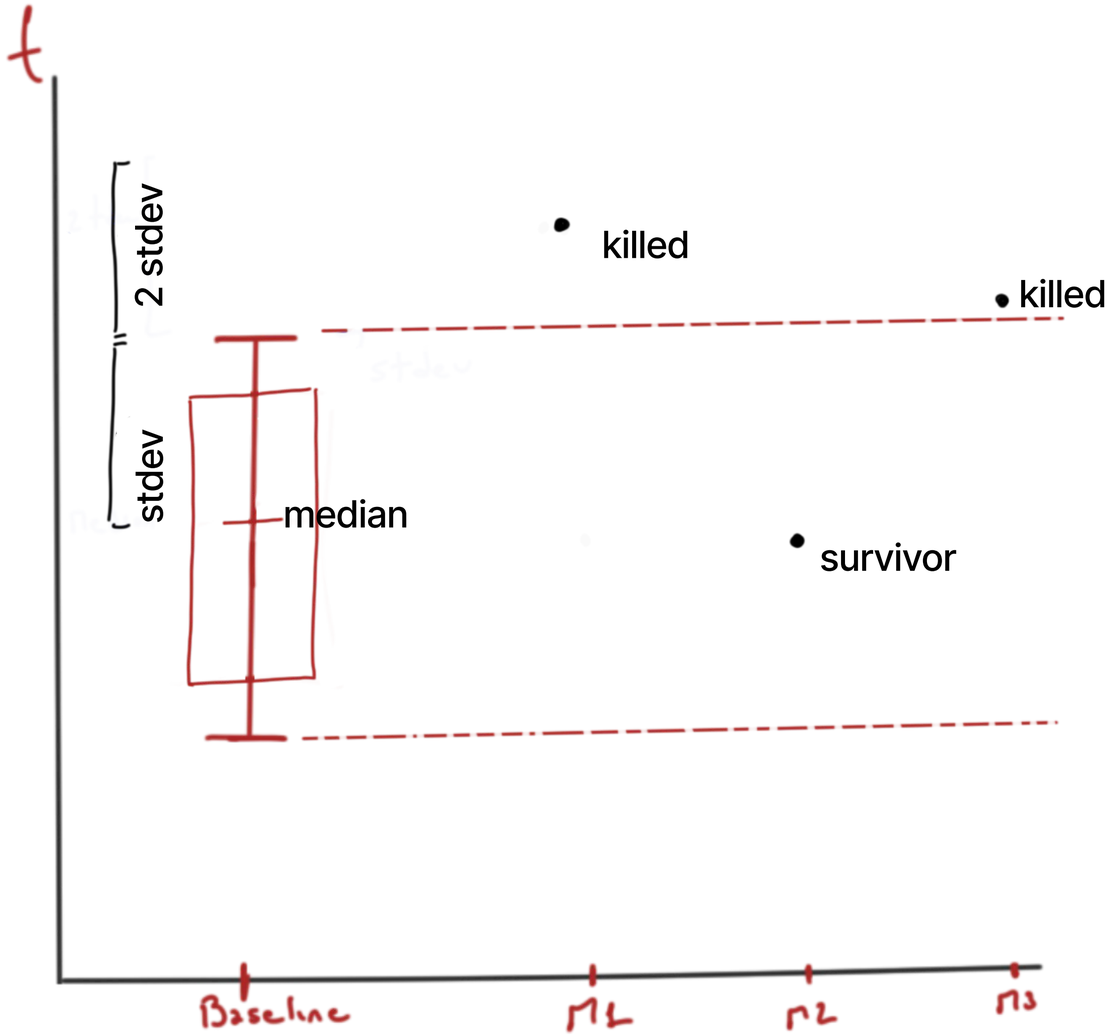
Experimental Mutants: **Sleep statements**

- *Why?* - Represents **latency**
- *How?* - Three **mutant operators**, 10, 100, 500 milliseconds
- *Where?* - At the beginning of every statement block

```
(next matchAgainst: aMatcher) ifFalse: [
    ^ alternative isNotNil and: [
        (Delay forMilliseconds: 500) wait.
        alternative matchAgainst: aMatcher ] ].
```

Experimental Oracle

1. Baseline: Average + stdev of **30** iterations to reduce external noise
2. Metric: E**xecution time**
3. Mutant detection: A mutant **is killed** if the execution time **> baseline average + stdev**



Case Study: Regular Expressions in Pharo

- **Why Regexes?** → They are well known

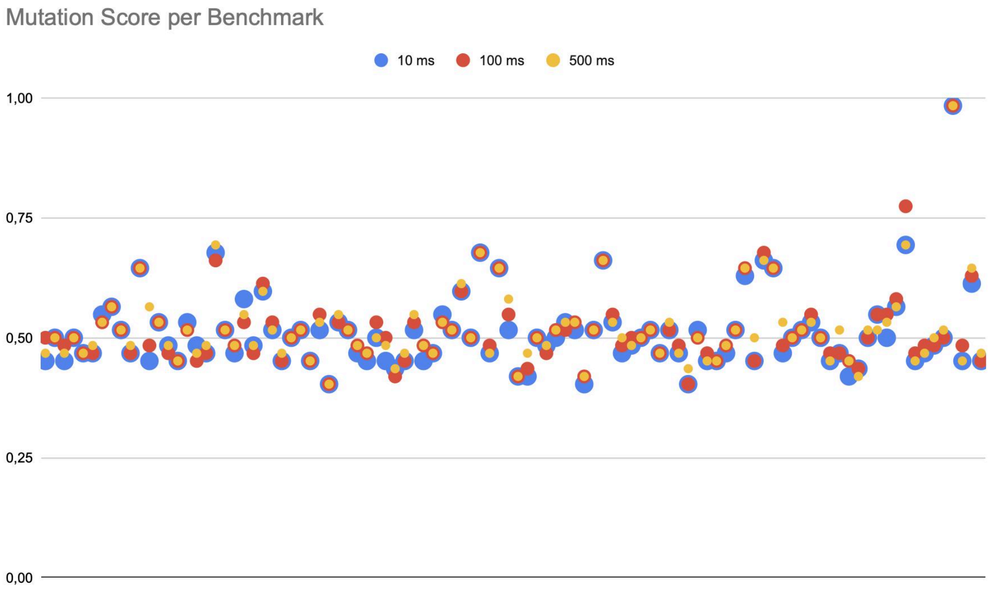
- **100** Regexes are generated via **grammar-based fuzzing** (MCTS)

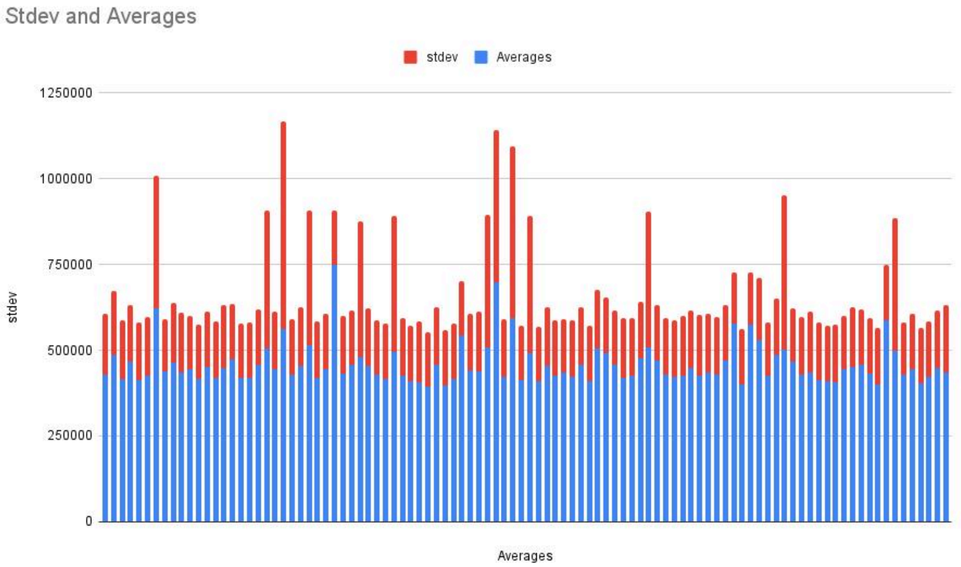
- We benchmark the regex **matches:** method with the generated regexes
  - '**b+a(b)*c+(b+c*b|b?(b+b)c?(b)*ab+a?aa)?a***' matches: 'bac'
  - '**(bb(((b)b+)))+b+|b+**' matches: 'bbbbb'

- We **assess the quality of benchmarks** to find performance bugs on **matches:** method

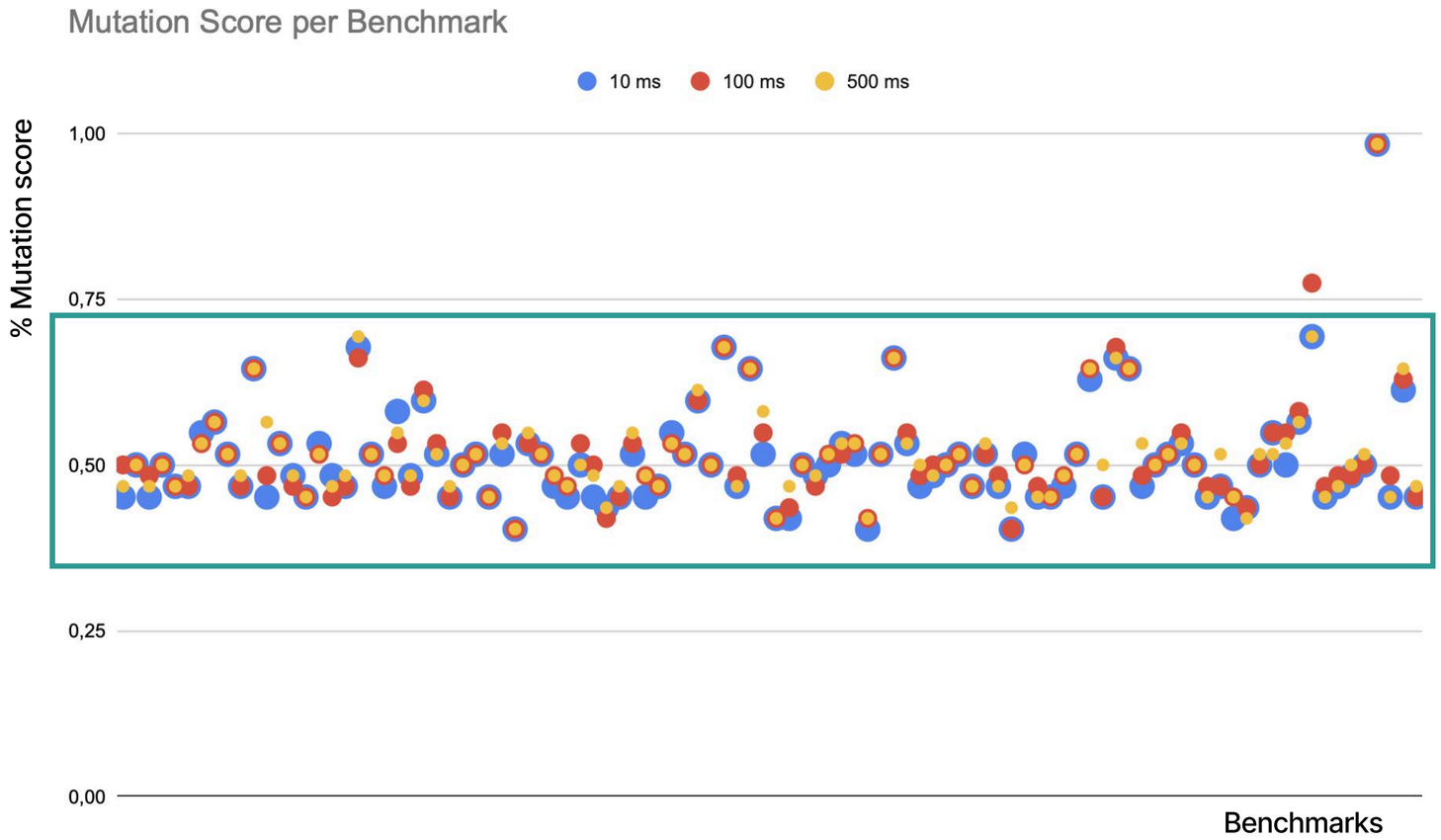
Results

- We introduce **62** mutants per mutant operator (3)
- We execute every benchmark **once per mutant** (186 times)



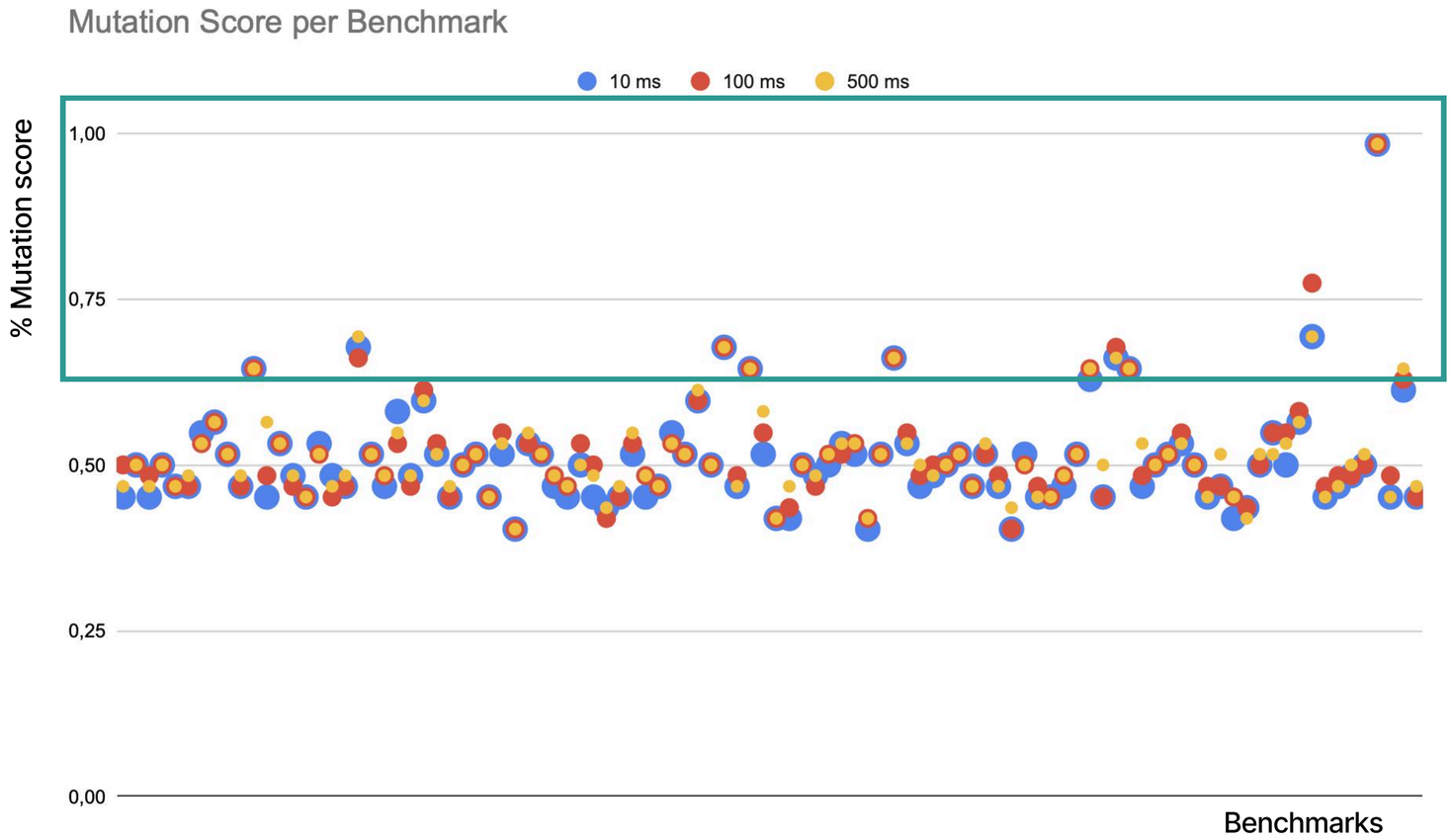


Average Benchmark Behavior



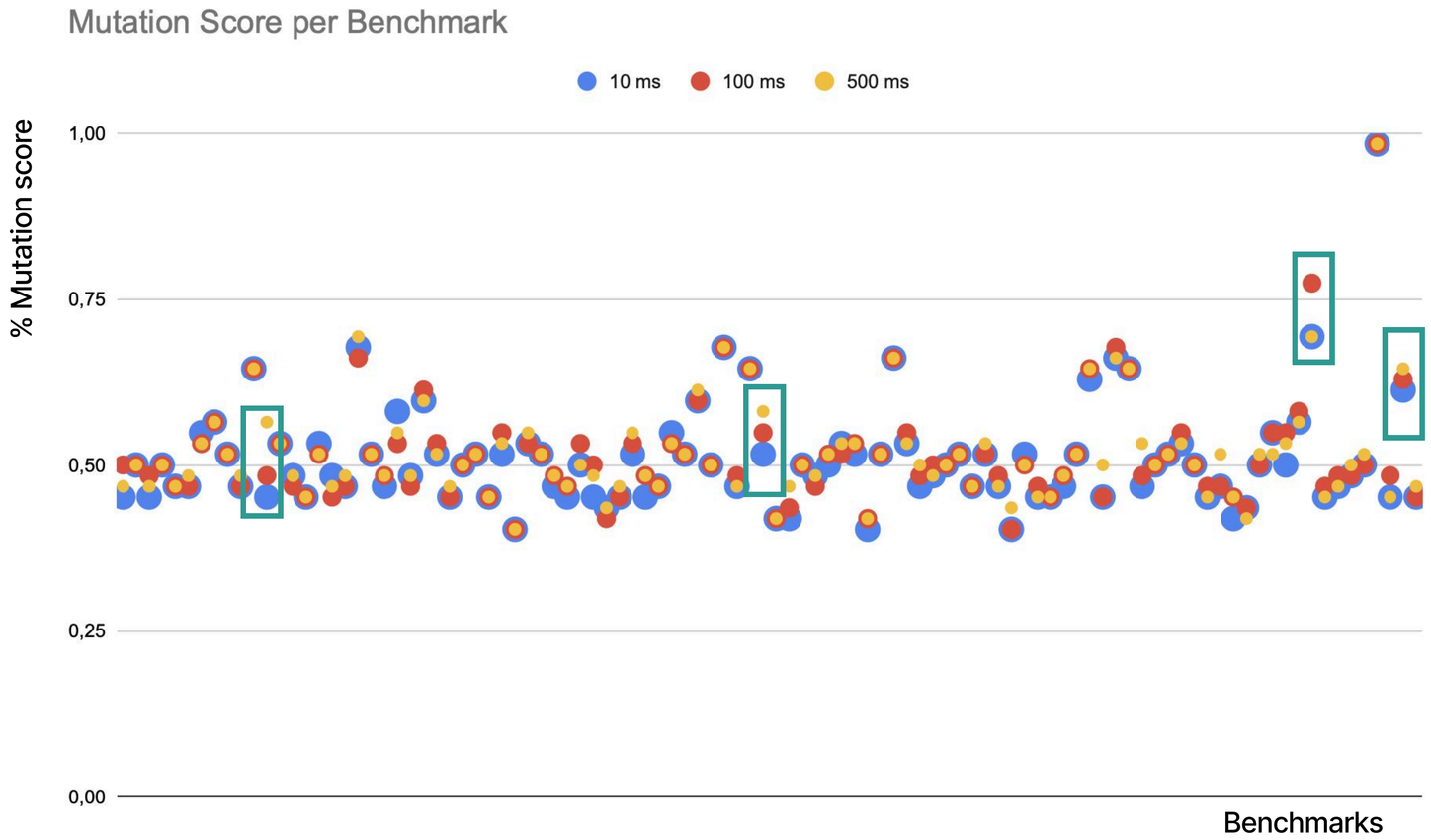
On average, the mutation score per benchmark is **51.48%**

High Score Benchmarks



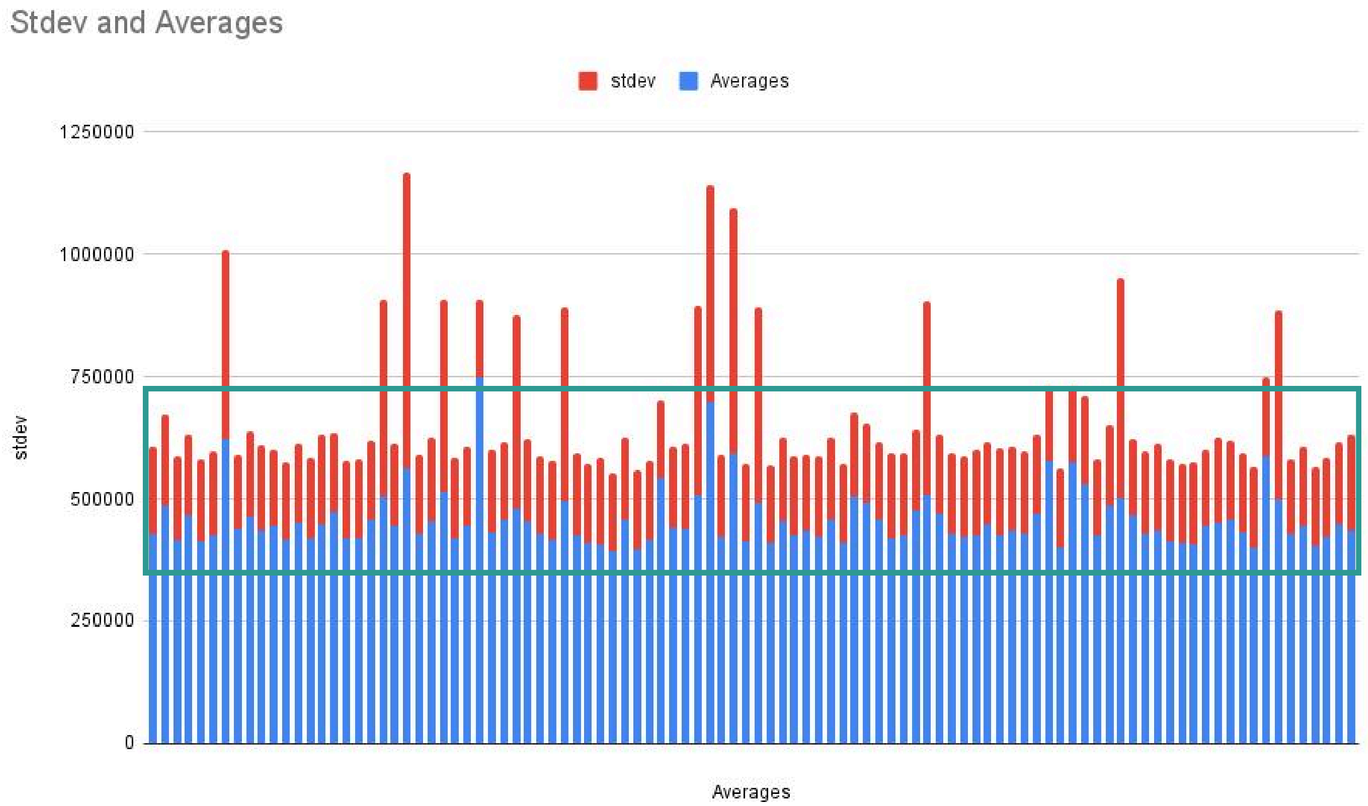
11% of benchmarks have an score **> 60%**

Performance Perturbation Sensibility



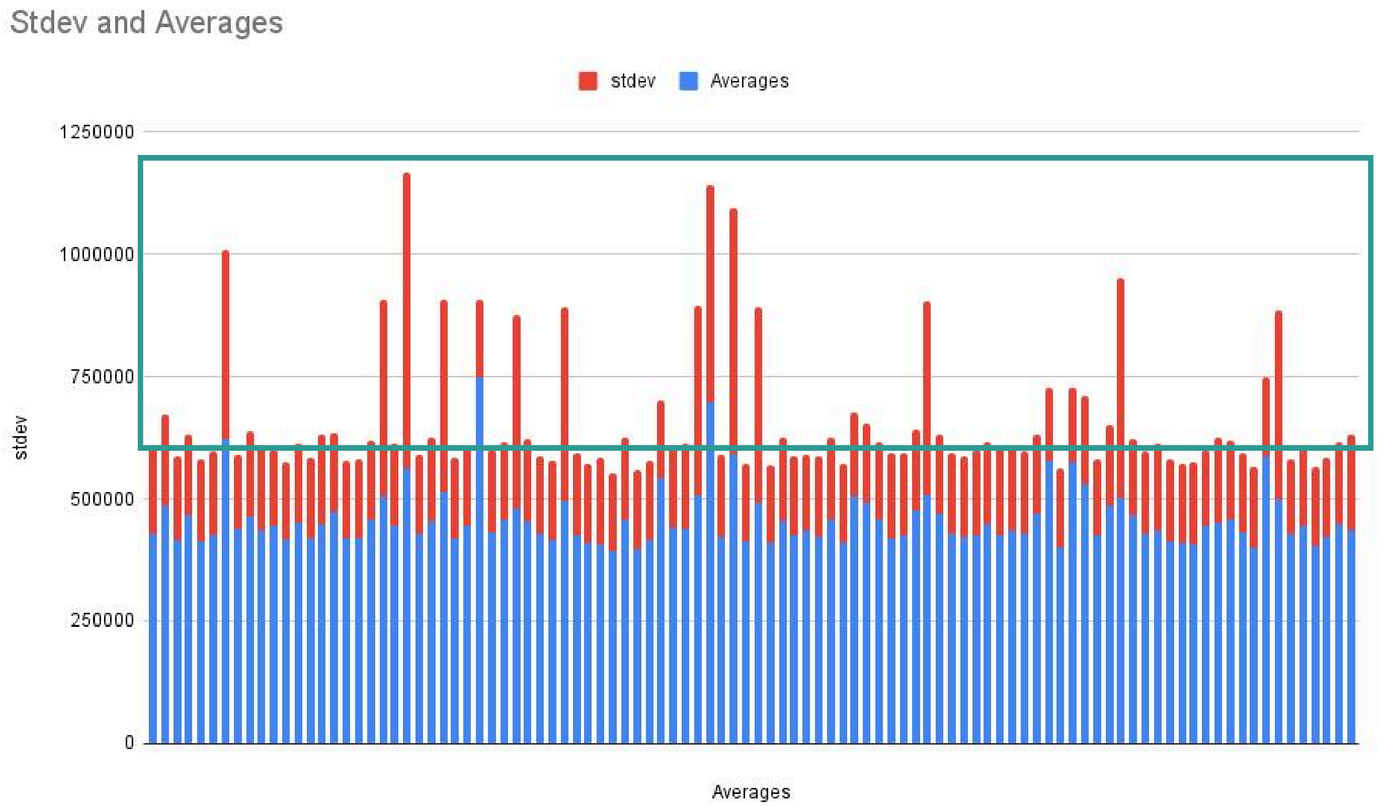
There are some benchmarks more sensible than others

Baseline Characterization



Average stdev is **42.48%** (relative)

Baseline Characterization

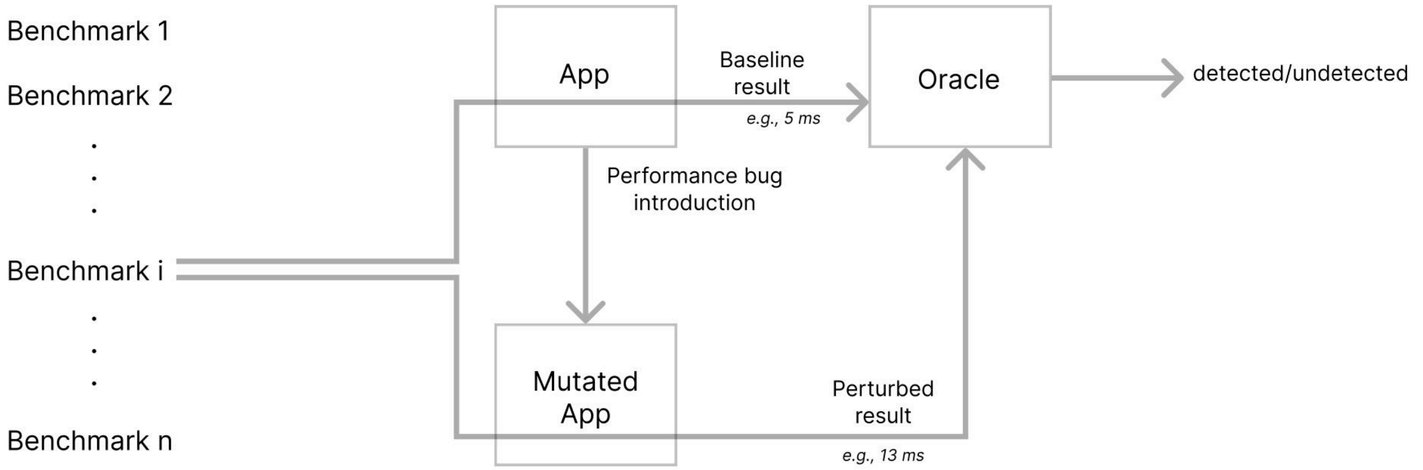


13% have high variance: **Can not detect** small perturbations

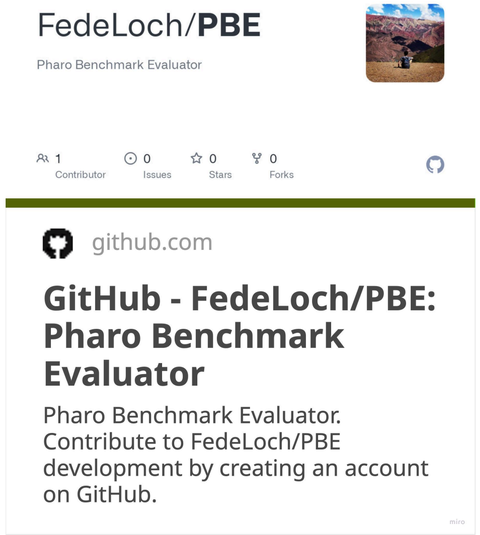
Future work

1. Improve the benchmark selection, **filtering** those with **high variance**
2. Study techniques to **minimize external noise**
3. Experiment with different **Oracle's thresholds**
4. Experiment with different **performance mutant operators**
5. Study **alternative metrics** to reduce the number of needed executions to have a **stable measure**

Conclusion



1. Systematic methodology to evaluate benchmarks's effectiveness
2. Introduce artificial performance bugs by extending mutation testing
3. Instance of the framework in a real setting showing results



IWST 2025

Federico Lochbaum, Guillermo Polito