TECHNISCHE UNIVERSITÄT BERLIN

Efficiently Detecting Performance Changes in FaaS Application Releases

Martin Grambow | Mobile Cloud Computing Research Group | Berlin | Dez 11 ’23

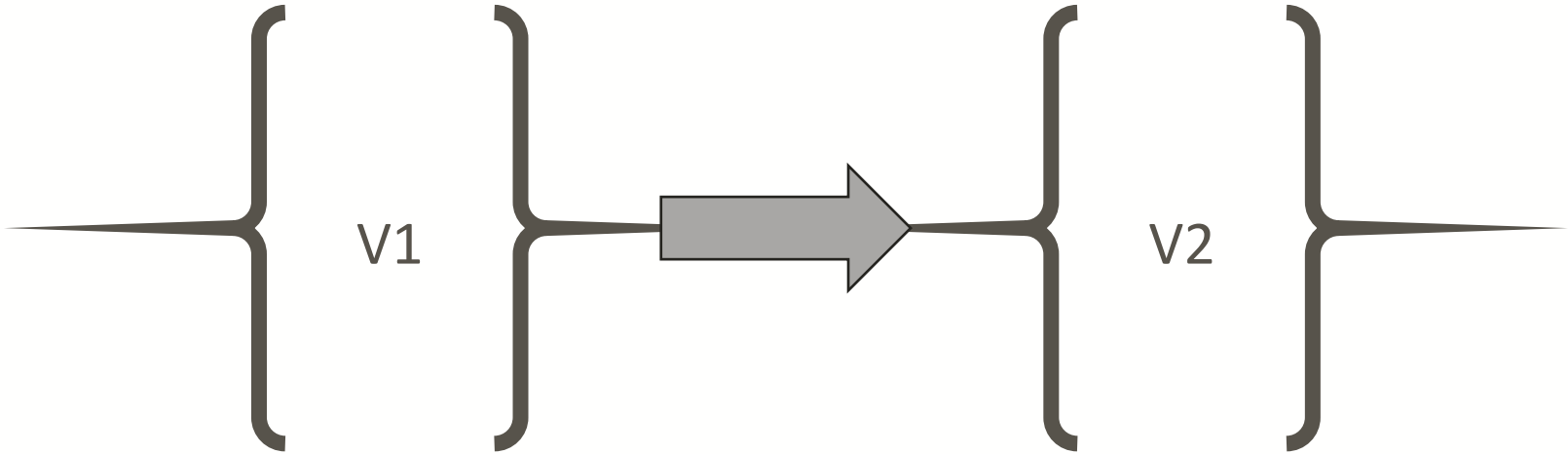
grambow@tu-berlin.de

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

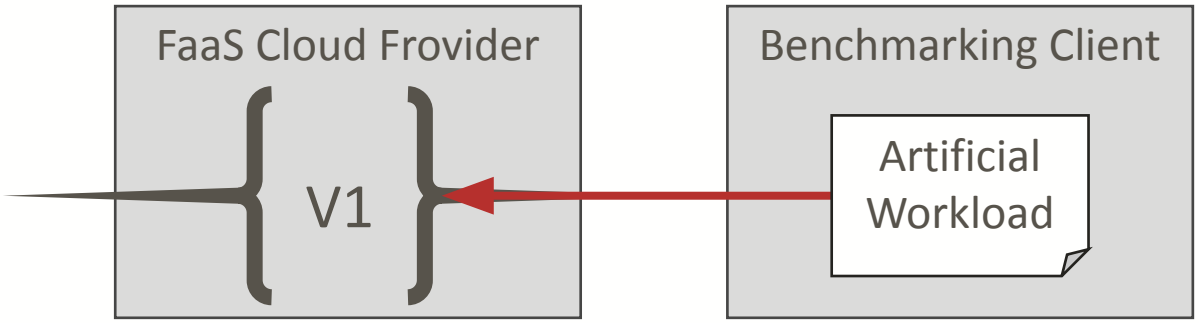
There are code changes

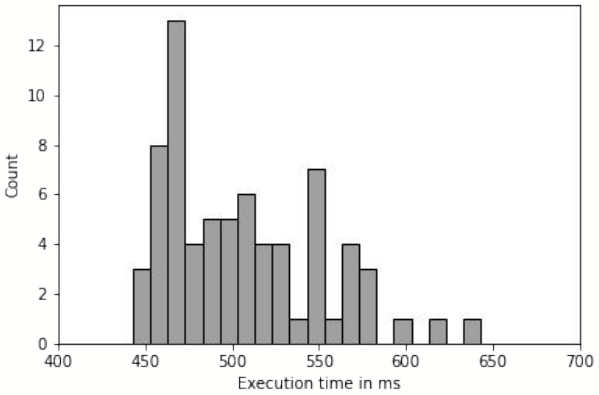
- Bug fixes
- New library versions
- New features
- …



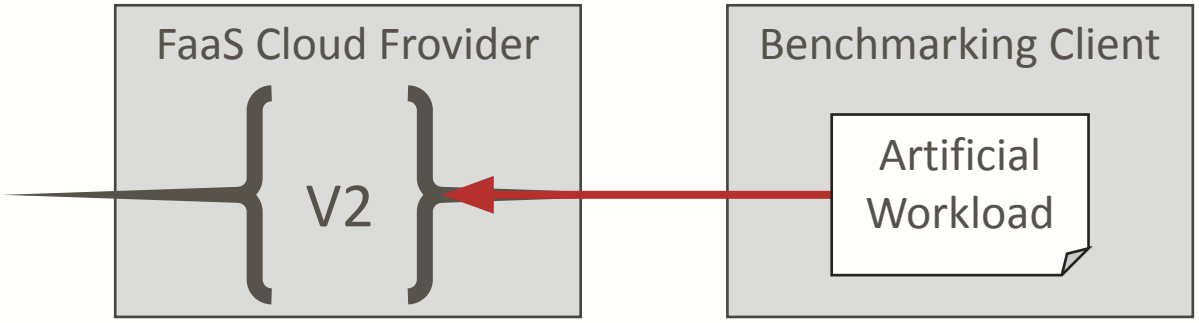
How would you measure and quantify the performance change?

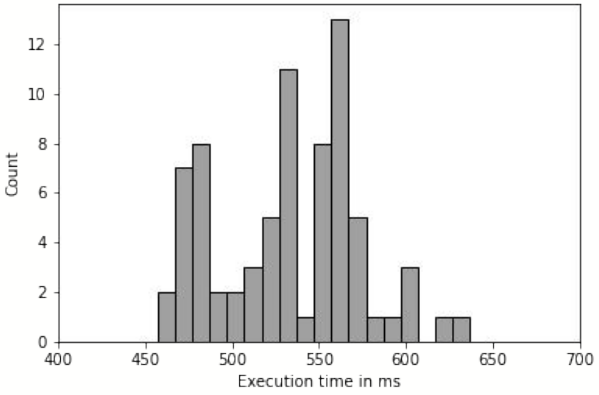
The traditional benchmarking approach





Somewhere between +1.6% and +12.3%



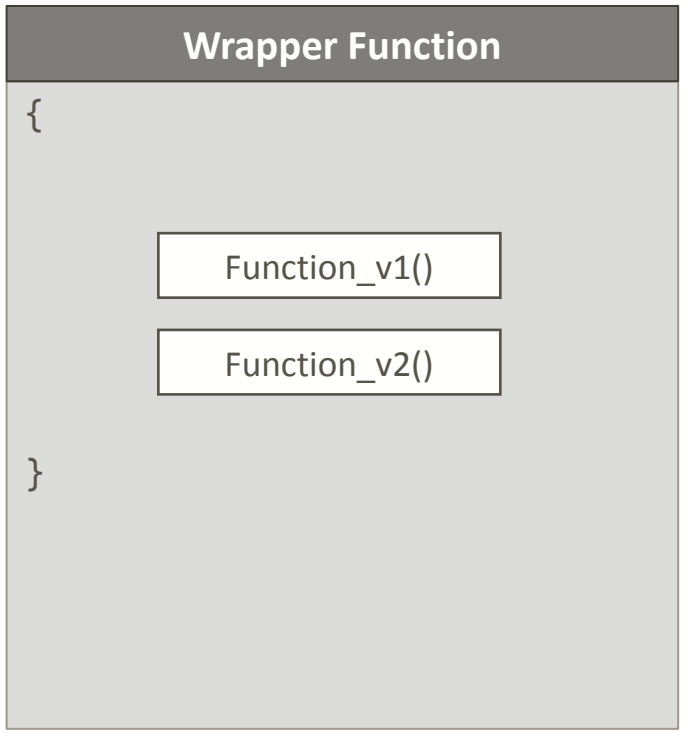


Idea: Run both function versions on the same instance

- Same compute unit
- Same environment
- Same random fluctuations
- …

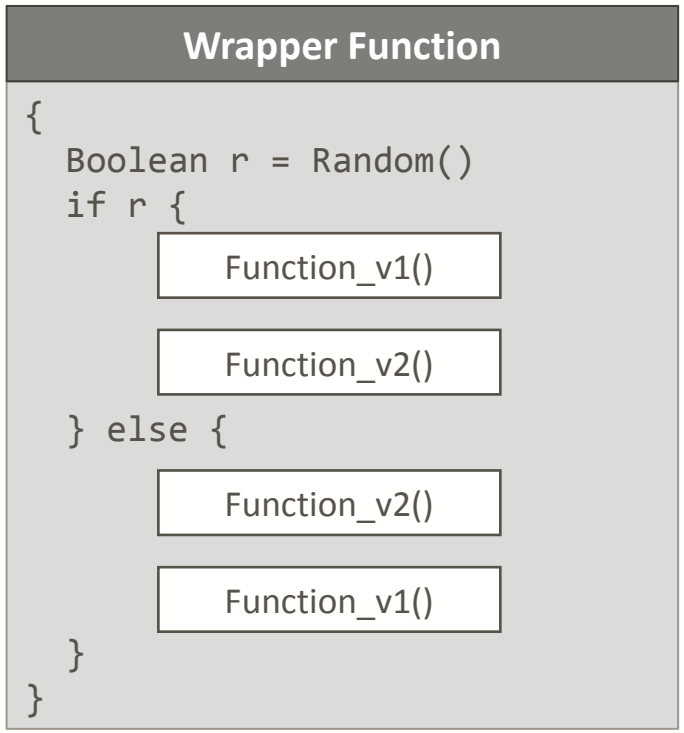
This should reduce the confidence interval width and lead to more accurate results!

Idea: Merge source code into a wrapper function



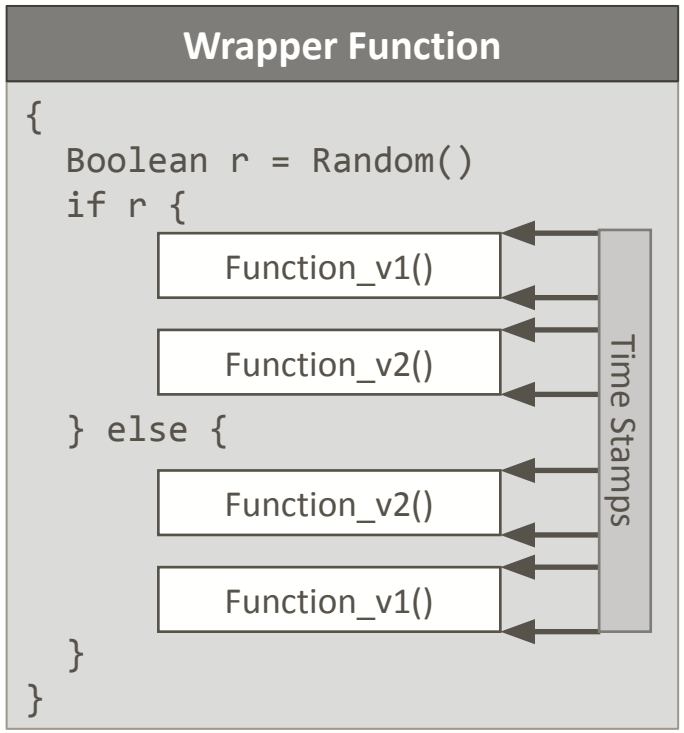
- Execute both functions successively

Idea: Merge source code into a wrapper function



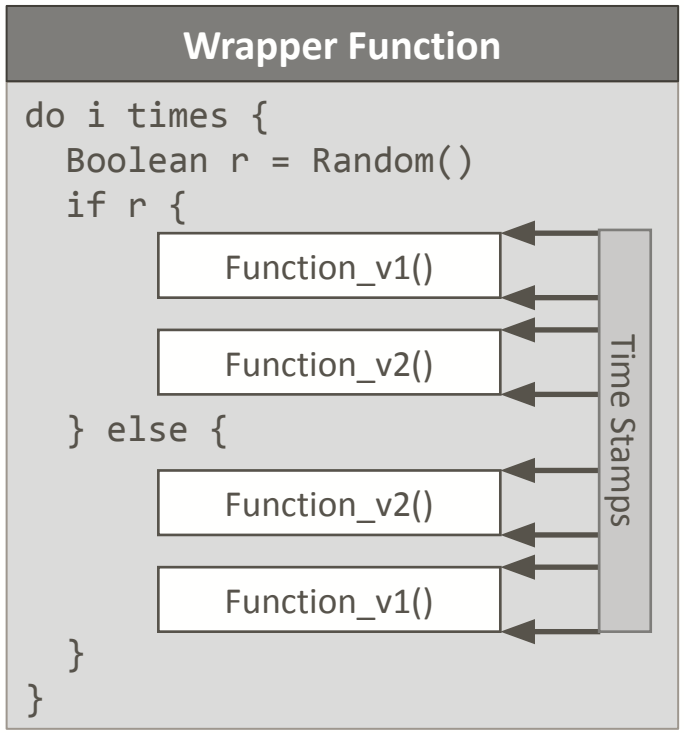
- Execute both functions successively
- Either version 1 or version 2 first

Idea: Merge source code into a wrapper function



- Execute both functions successively
- Either version 1 or version 2 first
- Track the time before and after each call

Idea: Merge source code into a wrapper function

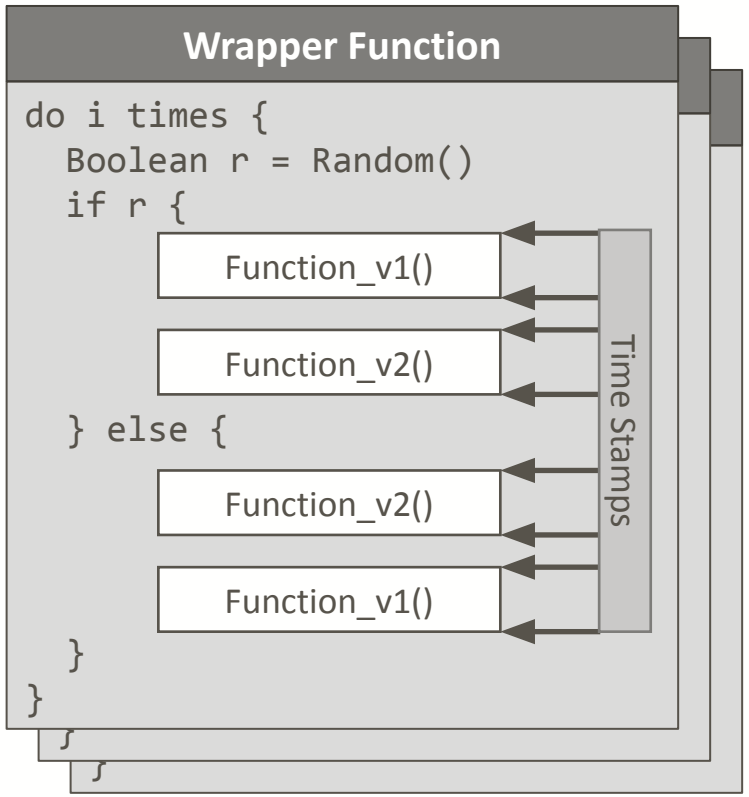


- Execute both functions successively
- Either version 1 or version 2 first
- Track the time before and after each call
- Repeat the calls several times

Randomized Multiple Interleaved Trials (RMIT) [1]

[1] - Ali Abedi and Tim Brecht. 2017. Conducting Repeatable Experiments in Highly Variable Cloud Computing Environments. In Proc. of ICPE '17. ACM.

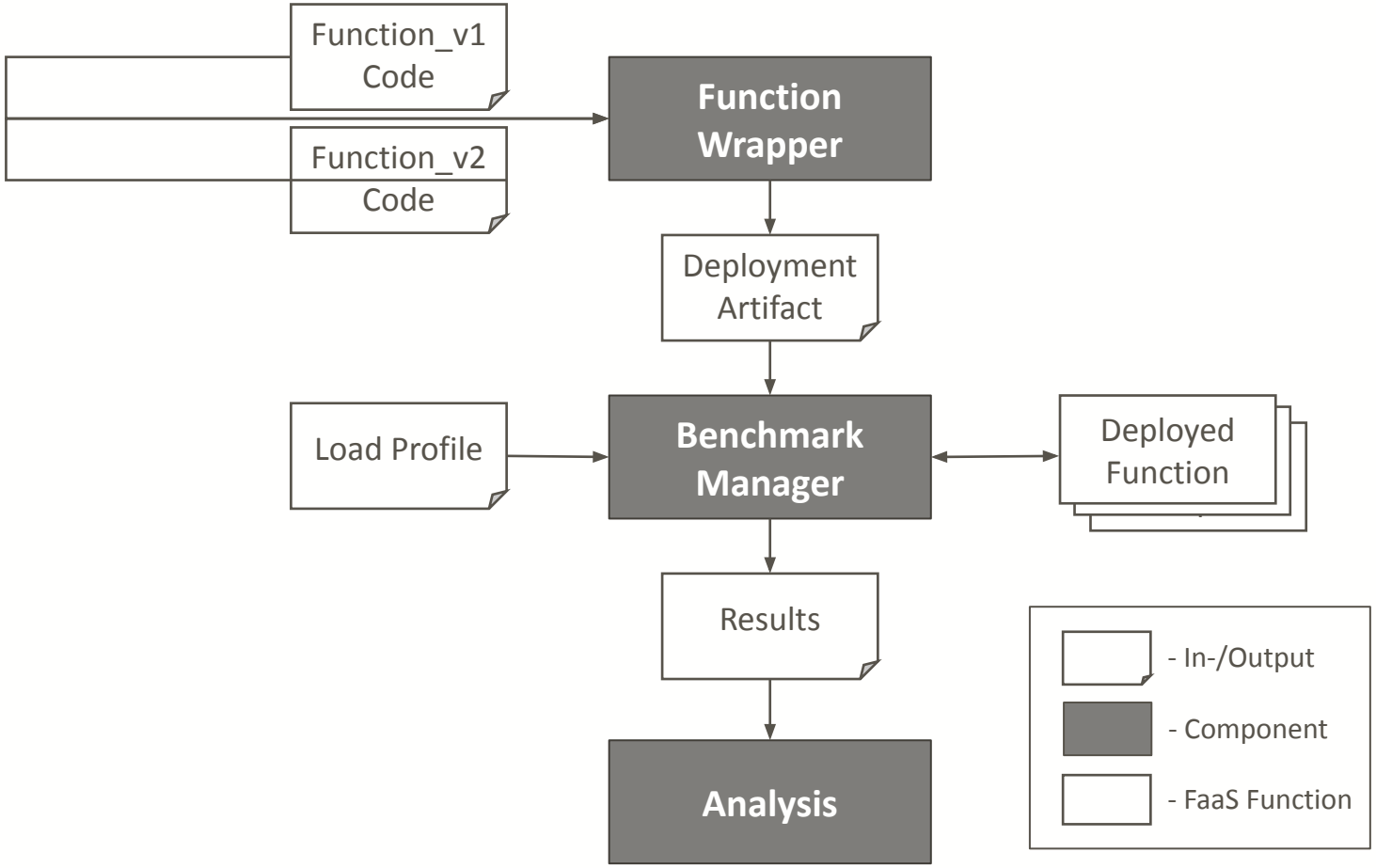
Idea: Merge source code into a wrapper function



- Execute both functions successively
- Either version 1 or version 2 first
- Track the time before and after each call
- Repeat the calls several times [1]
- Deploy Wrapper function several times

[1] - Ali Abedi and Tim Brecht. 2017. Conducting Repeatable Experiments in Highly Variable Cloud Computing Environments. In Proc. of ICPE '17. ACM.

Open source prototype: faasterBench

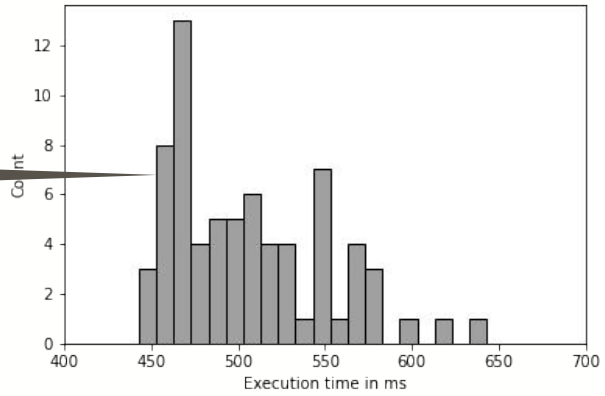


https://github.com/martingrambow/faasterBench

Experiments on AWS and GCP

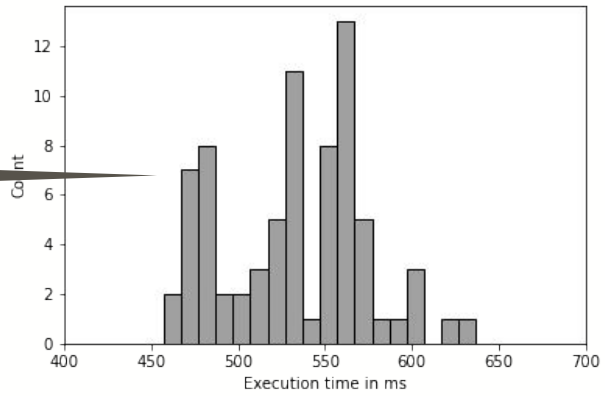
Traditional

V1

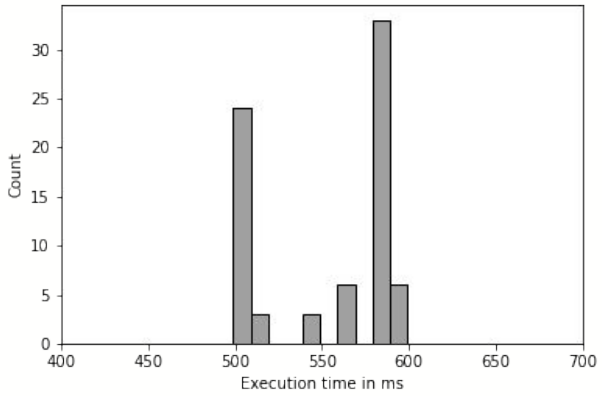


Somewhere between +1.6% and +12.3%

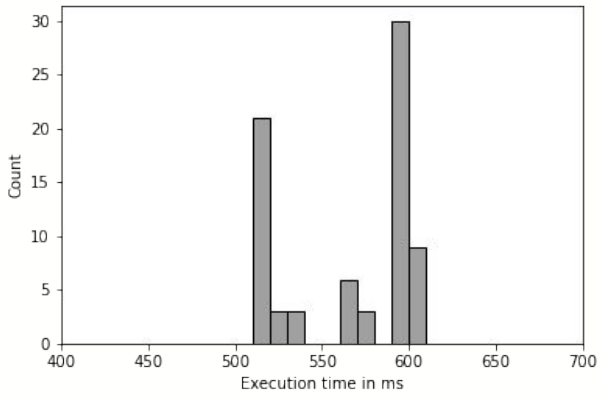
V2



faasterBench

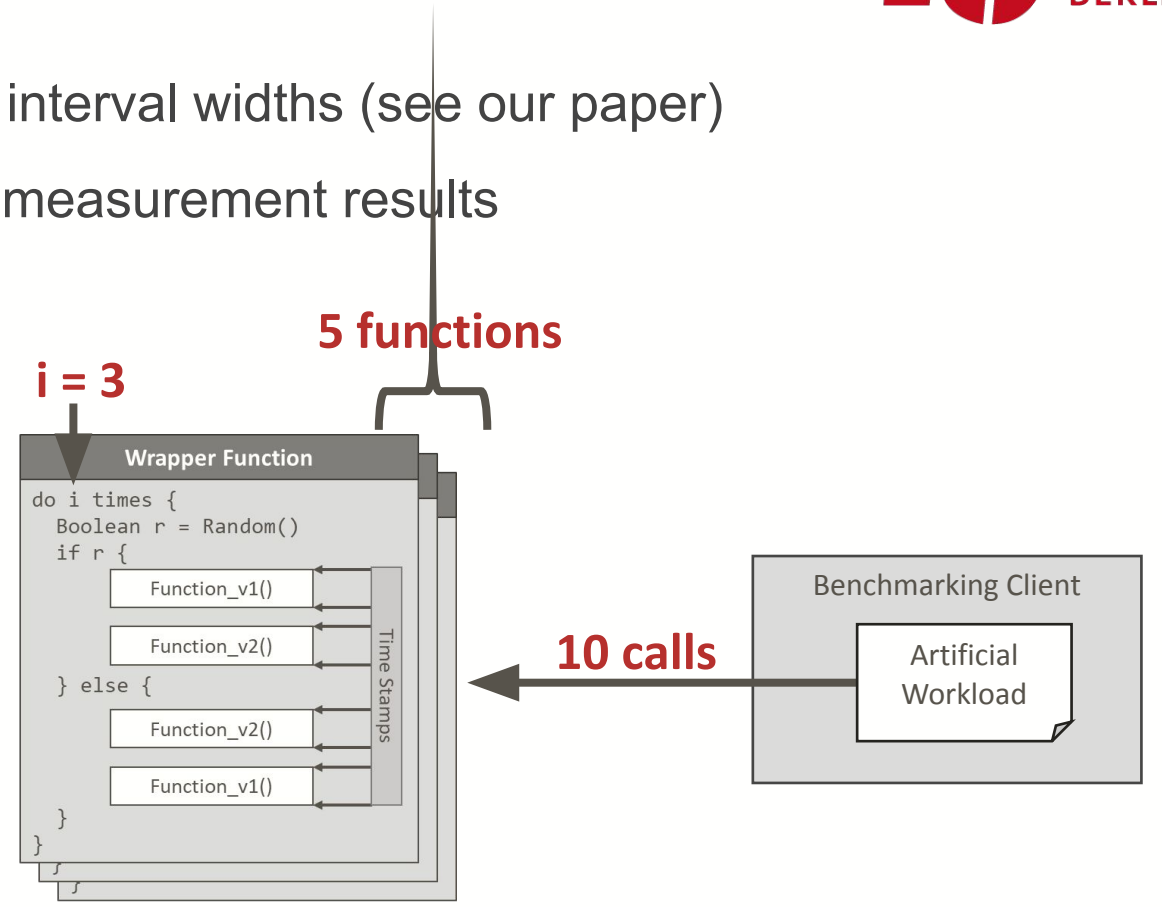


Somewhere between +2.2% and +2.6%



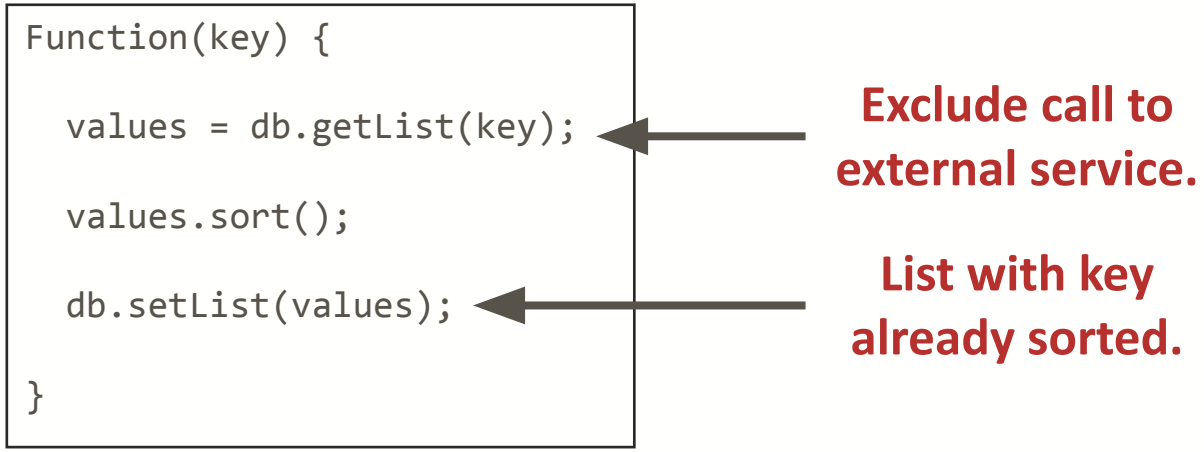
Findings and implications

- RMIT drastically reduces the confidence interval widths (see our paper)
- FaasterBench significantly improves the measurement results
- A good setup:
  - 3 iterations
  - 5 deployed wrapper functions
  - 10 calls per wrapper function
  - (150 measurement pairs in total)



Next steps

- Support more FaaS platforms
- Consider external calls
- Consider function parameters



TECHNISCHE UNIVERSITÄT BERLIN

Any Questions?