Integrating MQTBench to RedQueen

Archana Ravindar,

Golang Compiler Developer, Linux Toolchain team, IBM Systems

Mentor: Matthew Treinish

Motivation

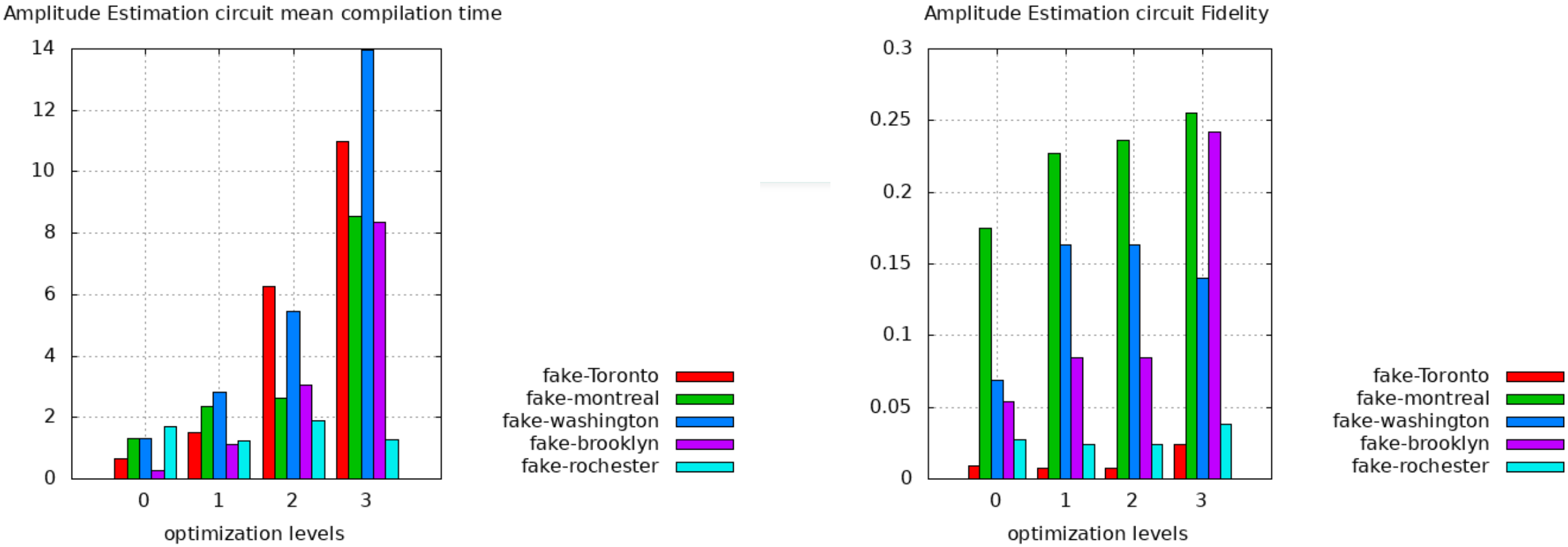
- A Q compiler translates an abstract quantum circuit to the hardware so that it can run efficiently on that hardware
  - It involves several passes to ensure the most optimal circuit is obtained as a result of translation
- Evaluation of the Q compiler is important
  - To make sure it generates the optimal circuit in most cases
  - Since Qiskit is opensource, we need to ensure from time to time that no new changes have caused a regression in performance
- MQTBench is a benchmark suite developed by TU Munich to test various compiler implementations including qiskit and includes applications from several domains - https://github.com/cda-tum/MQTBench
- This QAMP project is about integrating MQTBench with redQueen – the existing compiler test benchmark framework that uses pytest underneath

Implementation Details

- MQTBench supports creation of quantum circuits at multiple abstraction levels – for this project we select the algorithmic-level and target-independent level denoted by {0,1} in the code
- Pytest parameters: Circuit abstraction level{0,1}, circuit optimization level{0,1,2,3}, various Benchmarks{*} on various supported Backends{*}
- Tests added from the MQTBench framework:
  - Amplitude estimation, Deustch Jozsa, Grover, Ghz
  - Graphstate
  - PortfolioQAOA, PortfolioVQE
  - QAOA
  - QFT, QFTEntangled
  - QGAN (Machine learning)
  - QPEExact, QPEInexact
  - Qwalk
  - Realamprandom, su2random, twolocalrandom
  - VQE
  - Wstate
  - HHL
  - Pricingcall, Pricingput (Finance)
  - Tsp (optimization)
- Pull request: https://github.com/Qiskit/red-queen/pull/48

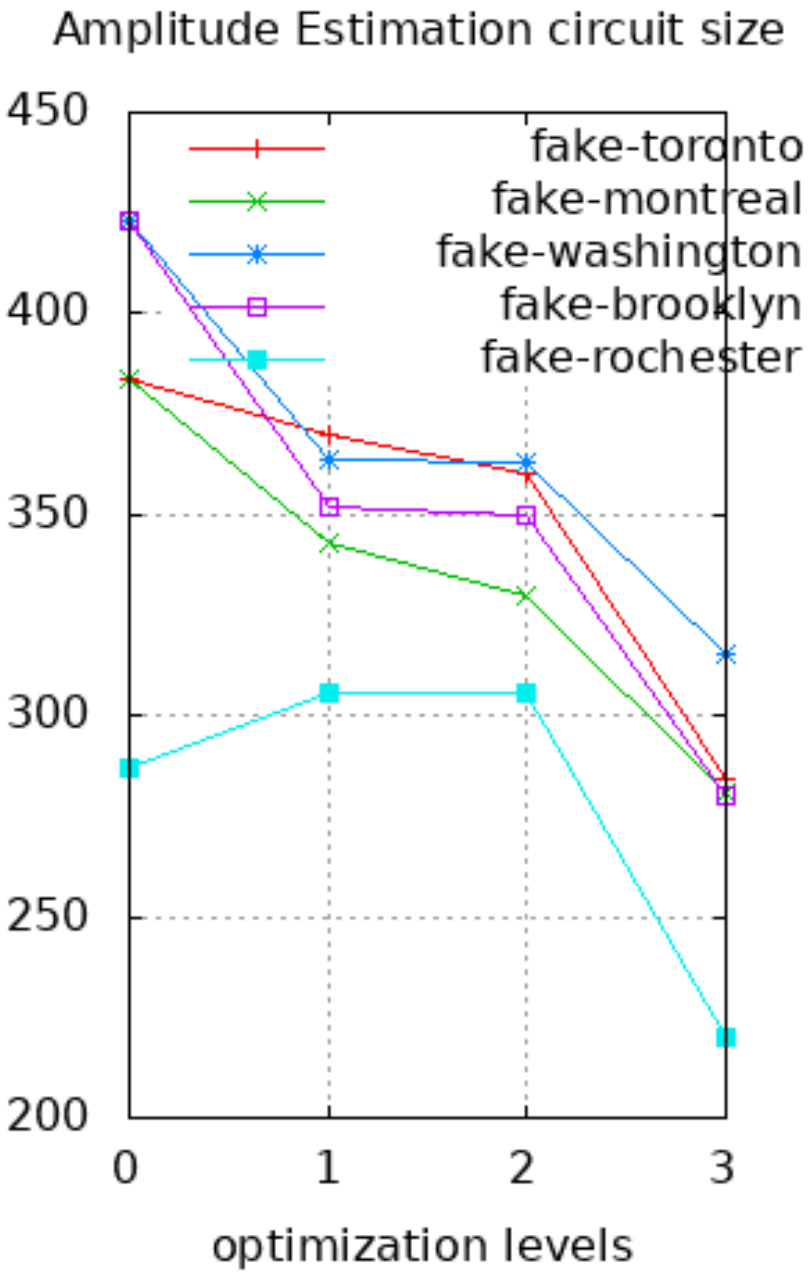
Results

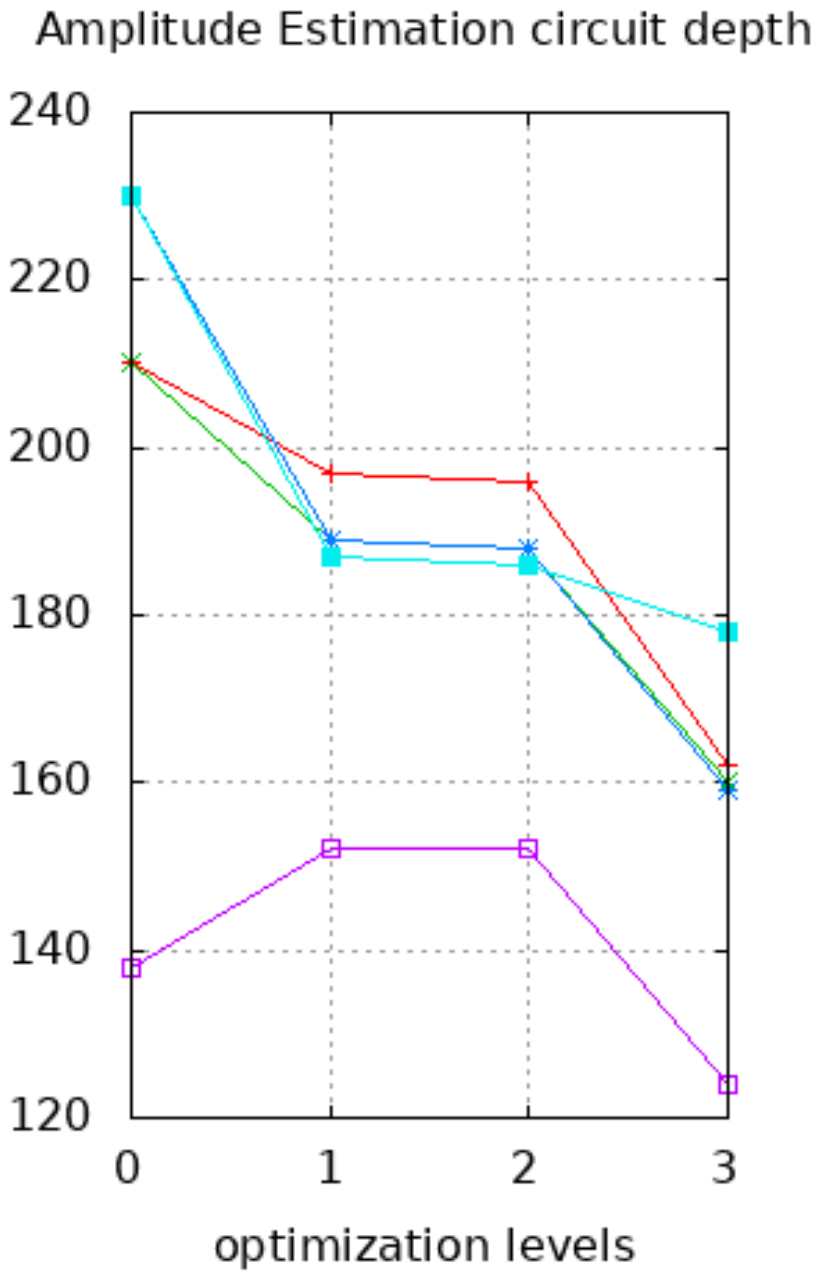
Strengthened RedQueen by adding 23 more benchmarks from MQTBench pertaining to various domains such as finance, optimization and machine learning using the same reporting interface that measures compilation time, Circuit size, depth, fidelity

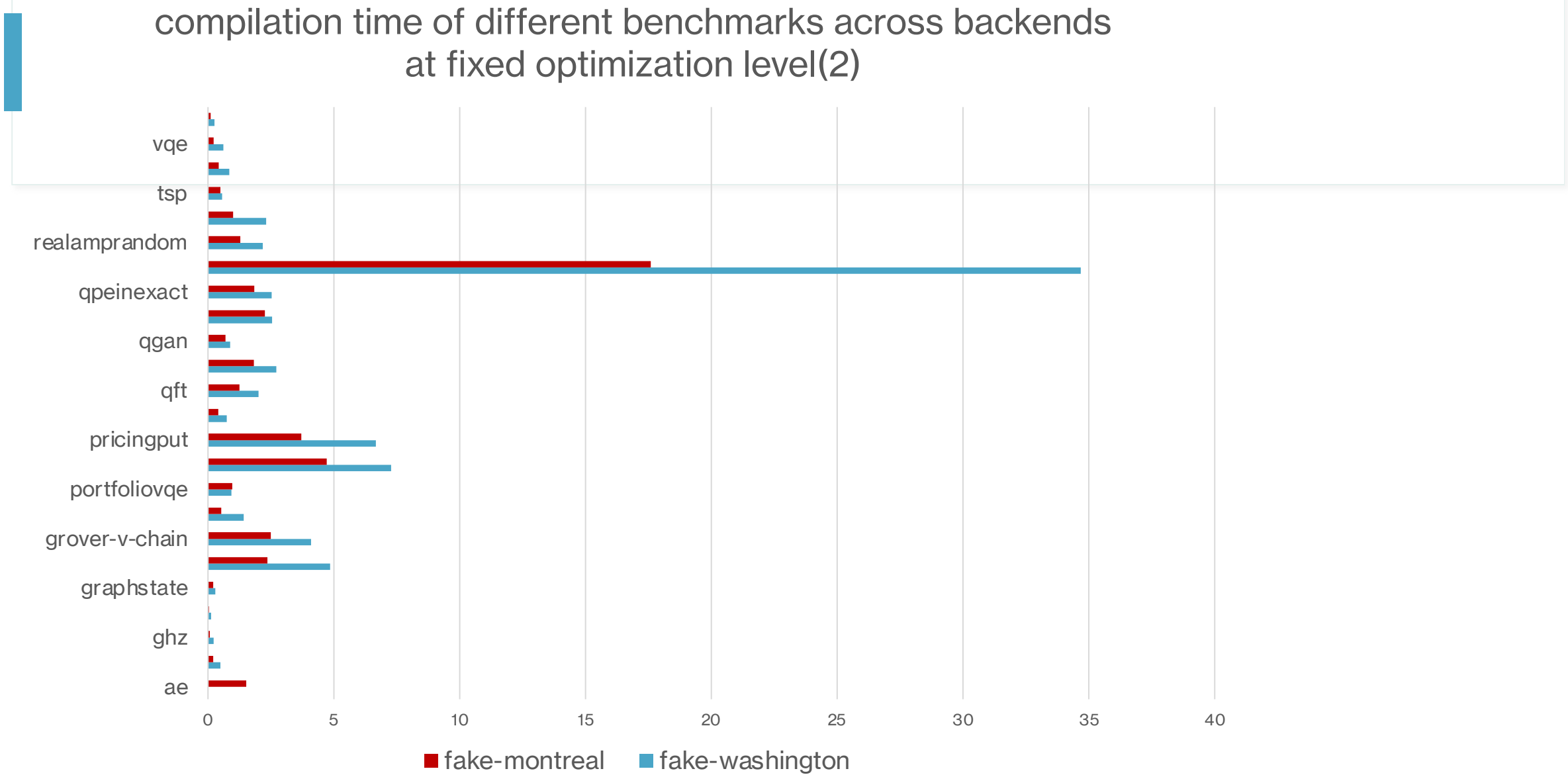


Running the following command produces the results in a table which are plotted for ease of viewing

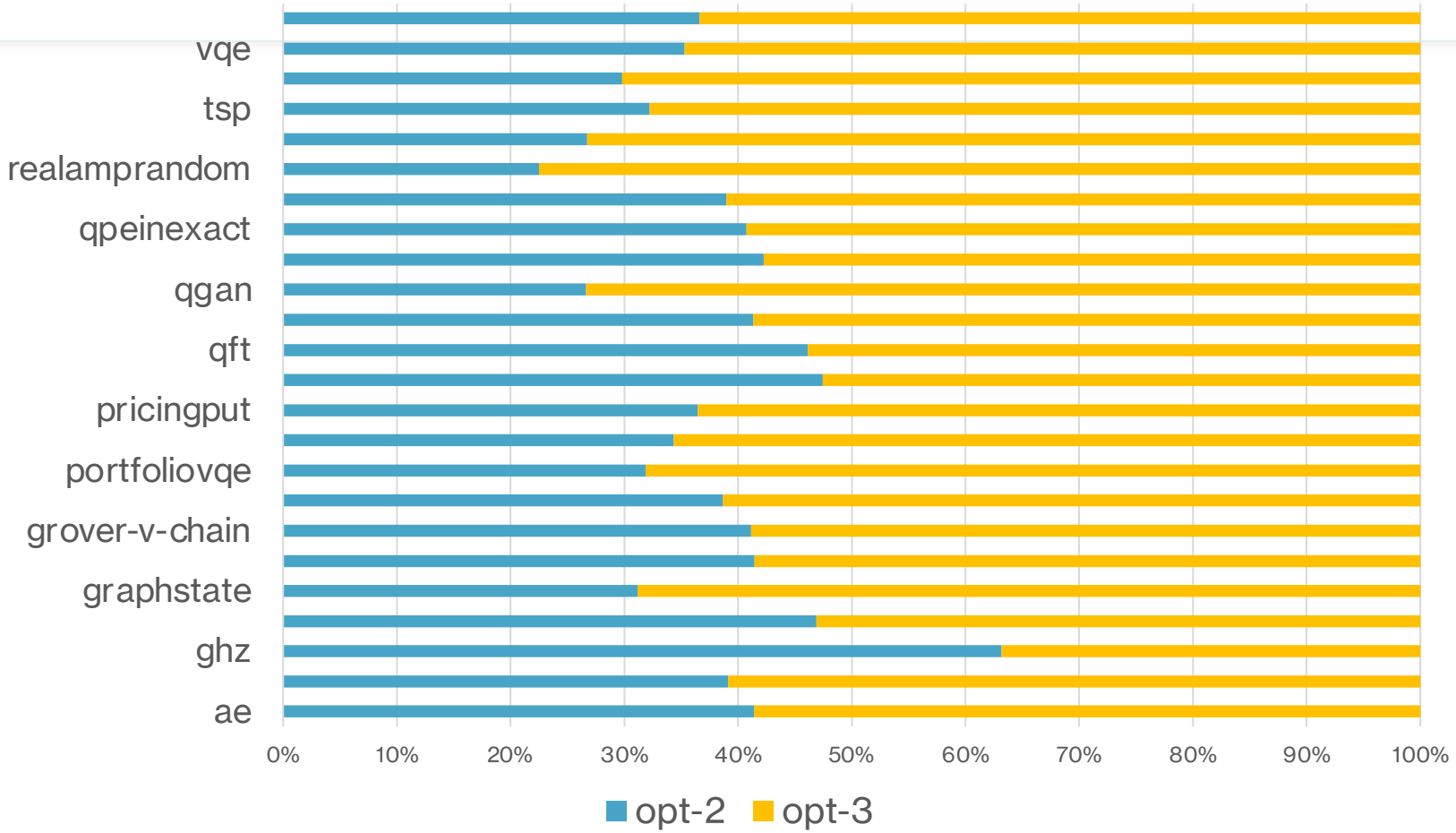
```
pytest ./red_queen/games/applications/testmqtone.py  -m qiskit —store
python3.10 -m report.console_tables --storage ./results/0001_bench.json
```


Amplitude Estimation circuit size
450
400
350
300
250
200
0
1
2
3
optimization levels
fake-toronto
fake-montreal
fake-washington
fake-brooklyn
fake-rochester


Amplitude Estimation circuit depth
240
220
200
180
160
140
120
0
1
2
3
optimization levels


compilation time of different benchmarks across backends at fixed optimization level(2)
vqe
tsp
realamprandom
qpeinexact
qgan
qft
pricingput
portfoliovqe
grover-v-chain
graphstate
ghz
ae
0
5
10
15
20
25
30
35
40
fake-montreal
fake-washington

fake-washington: Compilation time at optimization levels 2 v/ 3



By-products

- The journey was as it is exciting but we also found useful stuff along the way!
- MQTBench was pinned to qiskit version which forced us to use an older version of qiskit terra and they lose out on getting advantages of the latest and greatest updates to qiskit terra. Raised a bug in MQTBench git repo. https://github.com/cda-tum/MQTBench/issues/133 and the issue has been fixed recently

On the job Learning

- Learnt about nuances of pytest while working on this project and how one can debug issues in pytest
- We started out to generate .qasm files and then run them but some of the circuits in MQTBench are pretty big so the qasm files would be even bigger to handle so we decided to create circuits in memory and operate upon them
- Got to learn about how we can add code to compare ideal and noisy simulations. For some benchmarks with fairly big circuits (1000+ gates) Aer simulation was taking too much time to complete before the stipulated end date so we removed the Aer code for these circuits
- While encountering failures in qiskit for some benchmark combinations got to hear from Matthew on the kind of challenges quantum compiler writers encounter and they are so different from the typical issues seen in classical compilers
- Overall it has been the most enjoyable 3 months I have spent on any project! and I intend to pursue learning and contributing to quantum compiler code base

Future Work

- Three of the MQTBench circuit benchmarks deserve special treatment as the APIs to create the circuit are written specially. These were not included in the QAMP pull request and we plan to work on this as an extension of this work
- Adding better visualization in the redQueen results output

Thanks to my mentor Matthew Treinish and the Qiskit Advocate Team!!