Profiling of Quantum Computers and Simulators

Filippo Posta
Mathematics Division
Estrella Mountain Community College
Avondale, USA
filippo.posta@estrellmountain.edu

Professor Andreas Spanias
SenSIP Center
Arizona State University
Tempe, AZ, USA
spanias@asu.edu

Glen Uehara
SenSIP Center
Arizona State University
Tempe, AZ, USA
guehara@asu.edu

***Abstract*— Quantum Computers (QC) are a reality and evolving to fulfill their promise of revolutionizing the field of computer science. Algorithms and protocols are being developed to prepare for the day when Quantum Computers will be reliable enough to perform the calculations they are designed for. Shor's algorithm is one of these calculations and it requires the computation of Fast Fourier Transforms that may take many years on a classical computer. This work uses Quantum Fourier Transform programs to profile available Quantum Computers and simulators in an effort provide a snapshot of the current capabilities of QC.**

Keywords—quantum computers, system profiling, Braket, Qiskit, FFT

I. Project Description

Quantum Computers (QC) are a developing computational platform based on the principles of quantum physics. Their fundamental units are called Qubits. A Qubit can be on sate 0, or 1, or a linear combination of the two. This property is called *superposition* and provides a superficial reason of why QCs represent the next step in the evolution of computers [1]. The QC architecture has been constantly developed since the first (2 Qubit) quantum computer was built in 1998 and many companies have created QCs and made them available to the public via cloud systems [2,3]. In addition to QCs, many tech companies have developed QC environment simulators (QS) to provide programmers with the tools to explore the computational possibilities of QC's on classical computers [2]. The variety of cloud-based and simulator-based QC environments has created the need to profile them so that users can make educated decision of what system to use for their needs.

We aim to profile a few of the publicly available QC and QS and to do so we decided to use the Quantum Fourier Transform (QFT) and its inverse (IQFT) [4]. This choice is rooted on two factors. QCs are supposed to revolutionize cryptography by rendering the RSA Algorithm obsolete through the quick implementation of the Shor's Algorithm [5]. This quick implementation is currently unfeasible in classical computers and the bottleneck consists of the period finding step of Shor's algorithm which involves calculating Fourier transforms (FTs). The other fact is the authors' interest in machine learning and signal processing two fields that have widespread use of Fourier transforms [6].

II. Methods

Quantum Computers are a developing computational platform based on the principles of quantum physics. Their fundamental units are called Qubits. A Qubit can be on sate 0, or 1, or a linear combination of the two. The way that Quantum Programming (QP) works is to use a classical computer to prepare data, then feed the data to a quantum circuit and finally process the result on the classical computer. Most computational environments use Python for the classical computing steps, but quantum circuits are created differently. Figure 1 shows a QFT circuit for the Qiskit platform by IBM [7].

To benchmark different platforms, we choose a scenario that provided an intuitive signal as well as its FT. We choose as signal that alternates a non-zero number and zero (see Figure 1). Such signal will result in an FFT with two non-zero values at the initial and half-way position. Figures 2 and 3 represent the forward and inverse QFT circuits for such scenarios.

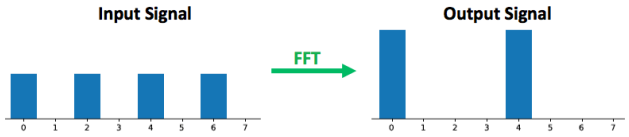

Figure 1: Signaling scenario for our analysis.

In Figure 2, the first "column" (i.e., I,H,H) represents the gates necessary to organize the qubits into the input signal for our scenario. Then, through a series of Z and Hadamard gates and a final swap between qubit 1 and 3, we arrive to the Fourier Transformed representation of the initial input. At this point we measure the state of the qubits in the circuit for data collection and analysis.

To study the accuracy of the IQFT we created the circuit depicted in Figure 3. The first two "columns" in the circuit are used to set up the configuration of the qubits to match the output of the QFT circuit. This step was only necessary to study IQFT in isolation from QFT since once we measure the qubits in the QFT circuit, those qubits cannot be used anymore as input of the IQFT.

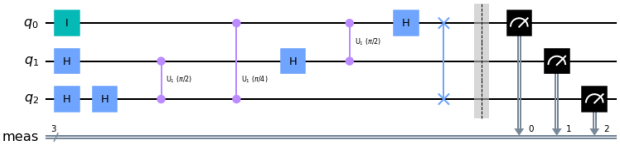

Figure 2: 3 QFT circuit for our sample problem. From left to right, the qubits are set-up to represent the input signal (1,0,1,0,1,0) and then a series of rotations leads to the Fourier Transform representation of the input signal. At that point the circuit is measured.

Conversely, when studying the serial computation of QFT and IQFT, we did not need to measure the qubits at the end of the QFT circuit, but instead we created one circuit that included QFT and IQFT with the absence of the measurement at the end of Figure 2 and the set-up gates (pinkish and purple gates) on the left of Figure 3.

RET Site sponsored by NSF Award 1953745

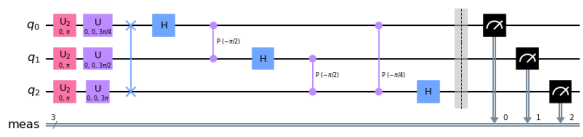

Figure 3: 3 Qubits inverse QFT circuit. From left to right, the qubits are set-up to represent the QFT of the original input and a sequence of gates is used to transform back to the original input. At which point the state of the qubits is measured.

We implemented the circuits using the IBM platform Qiskit [7], the Amazon platform Braket [8]. For the Qiskit study we used the cloud computing platform to run our codes on the Simulator and the Lima Quantum Computer. The choice of the Lima QC is due to its availability at the time we ran our first experiments. Afterwards, we used the same machine to maintain consistencies (different machines can have different error patterns due to qubits instabilities). Similarly, we used the Braket simulator as well as the Rigetti quantum machine.

We run each code from 2 to 5 Qubits configurations for each machine. 5 Qubits was the maximum allowed for the Lima machine. Each simulation consisted of 1,000 shots. A common amount to capture the error patterns of the machine without using too much computational time.

We used standard univariate methods to report and analyze the error. The next section shows our results.

III. Results

Our first analysis aimed to confirm that the precision of a QC decreases as the number of Qubits increases. This is a well-known current limitation of QCs that is due to individual Qubits instability as well as entanglement among Qubits. We run the QFT, IQFT and the I-QFT codes on both IBM-Lima and Rigetti QCs and confirmed that precision decreases as the number of Qubit increases. The results for the QFT code run IBM-Lima are shown in Figure 4. In this case precision is intended as the percentage of correct shots. Our results show that QC are viable up to 4 Qubits since we can take the most common outcome from the 1,000 shots as the correct answer. At 5 Qubits the calculation becomes so noisy that the correct calculation can be different from the most occurring one.

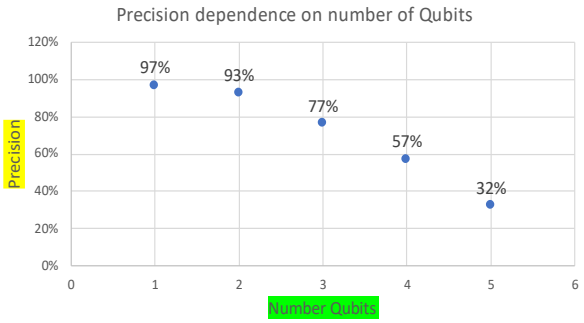

Figure 4: Precision behavior as function of number of Qubits for the QFT code run in IBM-Lima QC.

Our next analysis focused on the comparison between simulators and actual Quantum machines (QPU) and its results for the Qiskit platform are shown in Figure 5. The results show a clear separation between the simulator and the actual QPU (IBM-Lima). This separation is consistent for all of our three sample codes and similar results were observed in the Braket study. We should not that for this analysis we used the Sum of Squared Error as a measure of computational performance. As a result, there is an increasing trend with the number of Qubits rather than the decreasing trend observed in Figure 4. The results are consistent since the increasing trend in Figure 5 shows an increase in error as the number of Qubits increases.

The difference in precision between the two machines is due to decoherence that is not observed, and it is hard to account for in the simulators. Decoherence manifests as Qubits states instabilities and it is due to environmental factors that are hard to predict and quantity. For example, small changes in electromagnetic fields surrounding the QC could be the source of these instabilities. Because this process is very complicated and not well understood, it becomes impossible to represent it within the simulators [9].

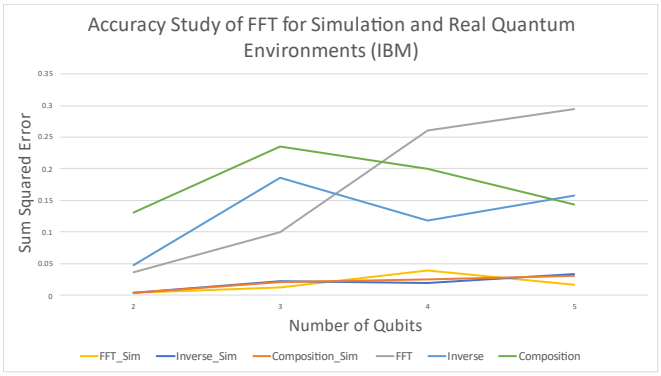

Figure 5: Performance analysis on Qiskit comparing simulator and QPU as function of Qubits. Performance is measured using the sum of squared error, thus the increasing trends.

The results that were observed for the IBM simulator were also observed in the Braket simulator. A comparison is shown in Figure 6, and it includes an interesting improvement in accuracy for the forward QFT when compared with the inverse and the composition. This could be due to the initial Qubit configuration of the QFT circuit in Braket that requires the use of less gates than the other codes and implementations.

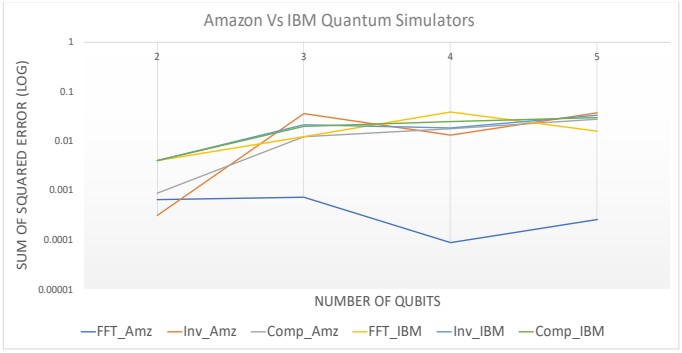

Figure 6: simulation for Braket and Qiskit are similar except for FFT.

Our last analysis compared two QPUs performance on the QFT codes for various Qubits configuration. The results are shown in Figure 7. The error's trendline shows a slightly better stability by the Lima machine for lower number of

Qubits, but this advantage disappears as the number of Qubits increases. For practical purposes the machines perform the same since the most-likely outcome for the machine is the same for all Qubits configurations.

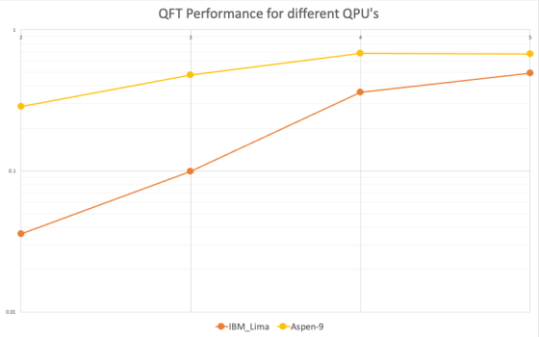

Figure 7: Performance analysis of QPUs for the QFT code. IBM_Lima was accessed using Qiskit and the Rigetti Aspen 9 QPU was accessed using Braket cloud services.

IV. Conclusions

From our analysis we observe that the decrease in precision as the number of Qubits increases is still a common issue. Additionally, the decrease monotonic in the machines that have been able to test. That is a very troublesome issue that highlights the architectural instability of QPUs. A better performance pattern would be characterized by a flat precision curve for low Qubits followed by a decrease, such a patter would imply an ability to stabilize entanglement and decoherence issues.

Decoherence is also the main differentiator between simulators and QPUs. This leads to overly optimistic (in terms of precision) results that do not translate to actual QPUs. However, the use and development of simulators is of the utmost importance since they provide a platform to test circuits quickly and inexpensively, before they are run on the QPUs.

We did not observe any major differences between the IBM and Amazon simulators and the Lima and Rigetti QPUs in terms of performance. At the same time, we created our codes to the specifications of each machine. A better practice could involve the use of TKET [11], a platform that is aimed at taking a circuit and optimize it for various Quantum computing platforms, including Qiskit and Braket. This approach could make the profiling more robust and portable to other platforms.

Computer profiling is an important element of quantum computing. Quantum machines are constantly evolving and periodical checks on the hardware will help their development and performance.

V. Aknowledgements

This work was supported by NSF Award 1953745 and the SENSIP center at ASU.

References

[1] Resch, Salonik, and Ulya R. Karpuzcu. "Quantum computing: An overview across the system stack." *arXiv preprint arXiv:1905.07240* (2019).

[2] J. Hidary, Quantum Computing: An Applied Approach. Champ: Springer, 2019.

[3] Devitt, Simon J. "Performing quantum computing experiments in the cloud." *Physical Review A* 94.3 (2016): 032329.

[4] Weinstein, Yaakov S., et al. "Implementation of the quantum Fourier transform." *Physical review letters* 86.9 (2001): 1889.

[5] Terhal, Barbara M. "Quantum supremacy, here we come." *Nature Physics* 14.6 (2018):530-531.

[6] P. Wittek. Quantum Machine Learning: What Quantum Computing Means to Data Mining. Academic Press, 2014.

[7] Qiskit Development Team “Learn quantum Computing Using Qiskit”. URL: https://qiskit.org/textbook/preface.html (last accessed July, 2021).

[8] Amazon Braket. URL: https://aws.amazon.com/blogs/aws/amazon-braket-get-started-with-quantum-computing/ (last accessed July, 2021).

[9] Rigetti Quantum Computers. URL https://aws.amazon.com/braket/hardware-providers/rigetti/ (last accessed July 2021).

[10] Grurl, Thomas, Jürgen Fuß, and Robert Wille. "Considering decoherence errors in the simulation of quantum circuits using decision diagrams." In *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1-7. 2020.

[11] TKET. URL https://cambridgequantum.com/tket/ (last accessed July 2021)