Resources Estimation for Quantum Computing Algorithms in Multiple Physical Platforms

Cesar Zaragoza Cortes

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2021

Reading Committee:
Jeffrey Wilkes, Chair
Boris Blinov

Program Authorized to Offer Degree:
Physics

University of Washington

Abstract

Resources Estimation for Quantum Computing Algorithms in Multiple Physical Platforms

Cesar Zaragoza Cortes

Chair of the Supervisory Committee:
Professor Emeritus Jeffrey Wilkes
Department of Physics

An important task in the development of quantum computing technologies is to determine the resources needed to execute a particular algorithm in a device with certain characteristics. Resources estimation allows not only to determine whether it is possible to execute an algorithm in a current Noisy Intermediate-Scale Quantum (NISQ) device, but also to experiment with the parameters of quantum hardware to find out how much it would have to scale to achieve quantum advantage. We use the Q# quantum programming language to implement the Bernstein-Vazirani algorithm, leverage simulators to perform resources estimation for trapped-ion and superconducting quantum hardware platforms, and compare the results.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Algorithms designed for quantum computers have the potential to solve some problems that cannot be efficiently solved by algorithms designed for classical computers. However, estimating how much resources are needed to execute a quantum algorithm that outperforms a classical one is a difficult task. There are many quantum programming languages and tools built around them such as Q#[4], Qiskit[5] and Cirq[2] that allow execution of quantum algorithms on simulators but out-of-the-box options to estimate resources are limited to the logical level or not existent.

1.1 The Purpose of This Thesis

This thesis aims to perform resources estimation at the physical level for trapped-ion and superconducting quantum hardware platforms. To do this, we will extend the simulators infrastructure built around Q# to calculate the maximum number of physical qubits, the total number of physical gates, and the maximum runtime required to execute a particular quantum algorithm. Additionally, the accumulated error for the computation, based on the fidelity of the physical gates, will also be calculated and analyzed to provide more information about the feasibility of obtaining reliable results from specific hardware platforms.

*ToDo: Expand on the following structure of the thesis:

1. Implement an algorithm using fault-tolerant error-corrected gates.

2. For each hardware platform do the following:

(a) Use an open-system simulator to analyze and optimize the effectiveness of fault-tolerant error-corrected gates.

(b) Use a resource estimator to determine the maximum input size for the algorithm to run on a real NISQ device.

(c) Use a resource estimator to determine the characteristics that a device should have to run the algorithm for a specific input size, and determine what would be the runtime.

3. Compare the results for each hardware platform.

*ToDo: Consider using an algotithm with more circuit depth to allow error correction to play a more important role.

We chose the Bernstein-Vazirani algorithm to perform resources estimation on because it is simple and becasue the amount of resources it demands is proportional to the size of its input. This provides the opportunity to analyze how different hardware platforms scale.

1.2 Quantum Computing Basics

*ToDo: Provide an introduction to the basics of quantum computing. (Consider making this section a chapter)

Chapter 2

THE BERNSTEIN-VAZIRANI ALGORITHM

The Bernstein-Vazirani algorithm[1] is a quantum algorithm that finds a secret string $s \in \{0,1\}^n$ given an oracle that implements a function $f : \{0,1\}^n \rightarrow \{0,1\}$ such that $f(x) = (x \cdot s)$ *modulo* 2. The most efficient classical algorithm evaluates the function n times to find s. In contrast, this quantum algorithm only needs to evaluate it once.

2.1 Algorithm

*ToDo: Mathematically describe the algorithm.

2.2 Circuit Representation

*ToDo: Show the circuit representation of the algorithm.

2.3 Q# Implementation

The following code presents a simple implementation of the Bernstein-Vazirani algorithm in the Q# programming language:

*ToDo: Make the implementation more generic by receiving the oracle as on argument to the BernsteinVazirani operation.

*ToDo: Consider creating a Q# language option for lstlisting.

```
@EntryPoint()
operation BernsteinVazirani () : Unit {
    let secret = [One, Zero, One, One, Zero];
    use (qubits, aux) = (Qubit[Length(secret)], Qubit()) {
        X(aux);
```

```
        H(aux);
        ApplyToEach(H, qubits);

        // Oracle.
        for index in 0 .. Length(qubits) - 1 {
            if (secret[index] == One){
                CNOT(qubits[index], aux);
            }
        }

        ApplyToEach(H, qubits);
        let results = ForEach(M, qubits);
        ResetAll(qubits);
        Reset(aux);
    }
}
```

Chapter 3

RESOURCES ESTIMATION FRAMEWORK

The framework we use for resources estimation is very similar to the one proposed by Soeken et al.[3]. The process is the following:

1. Implement a quantum algorithm using a high-level programming language (Q# in this case).

2. Verify the correctness of the implementation by executing the algorithm in a full state simulator.

3. Setup the simulator to estimate physical resources with the parameters that are specific to a hardware platform.

4. Analyze the results obtained from the resources estimator.

3.1 Resources Metrics

*ToDo: Describe the values obtained from the resources estimator (gate count, runtime, accumulated error), and how they are calculated.

3.2 Gate Decomposition

*ToDo: Describe why logical-level gates have to be decomposed into physical-level gates.

3.3 Limitations

*ToDo: Describe the limitations that this resources estimation has in regards to runtime (sum of the runtimes of individual gates rather than the critical path), and types of computations

(trouble with mixed states).

Chapter 4

EXTENDING Q# SIMULATION INFRASTRUCTURE FOR ESTIMATION OF PHYSICAL RESOURCES

Microsoft's Quantum Development Kit (QDK) supports the implementation of custom simulators that can be used to run Q# programs. We leverage this capability and implement a simulator that calculates the resources a quantum algorithm would require to be executed in a hardware platform with specific characteristics.

4.1 QDK Custom Simulators

*ToDo: Describe how custom simulators are implemented using diagrams and code snippets.

4.2 Software Architecture of Physical Resources Estimator Simulator

*ToDo: Describe the software architecture using diagrams and code snippets.

Source code of a working version can be found in in GitHub.

Chapter 5

TRAPPED-IONS HARDWARE PLATFORM

*ToDo: Briefly this quantum computing platform.

5.1 Native Gates and Platform Characteristics

*ToDo: Enumerate the native gates that this platform implements, its characteristics (fidelity, gate time), and how logical gates are implemented (using circuits to illustrate them).

5.2 Resources Estimation Analysis

*ToDo: Show (using tables and/or plots) how resources escalate as the size and pattern of the secret string change.

Example of ouput of resources used by the Bernstein-Vazirani algorithm using a secret string of size 5:

```
Ion Platform Resource Estimation
Bernstein-Vazirani
PHYSICAL LAYER
Total Statistics
-----------------

Qubits: 6
Gate Count: 38
Time: 1175
Error: 0.3500000000000003


Gate Statistics
```

R:

- Count: 35
- Time: 470
- Error: 0.2300000000000002

XX:

- Count: 3
- Time: 705
- Error: 0.1200000000000001

5.3 Analysis of Execution Feasibility in NISQ Devices

*ToDo: Analyze what would be the maximum size (and difficulty of pattern) of the secret string that can be used in a current NISQ device based on this platform.

Chapter 6

SUPERCONDUCTING HARDWARE PLATFORM

*ToDo: Briefly this quantum computing platform.

6.1 Native Gates and Platform Characteristics

*ToDo: Enumerate the native gates that this platform implements, its characteristics (fidelity, gate time), and how logical gates are implemented (using circuits to illustrate them).

6.2 Resources Estimation Analysis

*ToDo: Show (using tables and/or plots) how resources escalate as the size and pattern of the secret string change.

6.3 Analysis of Execution Feasibility in NISQ Devices

*ToDo: Analyze what would be the maximum size (and difficulty of pattern) of the secret string that can be used in a current NISQ device based on this platform.

Chapter 7

COMPARISON BETWEEN TRAPPED-IONS AND SUPERCONDUCTING HARDWARE PLATFORMS

*ToDo: Compare the results obtained from both hardware platforms and comment on the insights obtained.

Chapter 8

FUTURE WORK

*ToDo: Consider incorporating error correction to the resources estimation and post-analysis.

BIBLIOGRAPHY

[1] E BERNSTEIN and U VAZIRANI. Quantum complexity theory. *SIAM journal on computing*, 26(5):1411–1473, 1997.

[2] Cirq Developers. Cirq, May 2021. See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors.

[3] Mathias Soeken, Mariia Mykhailova, Vadym Kliuchnikov, Christopher Granade, and Alexander Vaschillo. A resource estimation andverification workflow in q#. *Design, Automation and Test in Europe Conference*, 2021.

[4] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. Q#: Enabling scalable quantum computing and development with a high-level dsl. In *Proceedings of the Real World Domain Specific Languages Workshop 2018*, RWDSL2018, New York, NY, USA, 2018. Association for Computing Machinery.

[5] Matthew Treinish, Jay Gambetta, Paul Nation, Paul Kassebaum, qiskit bot, Diego M. Rodríguez, Salvador de la Puente González, Shaohan Hu, Kevin Krsulich, Laura Zdanski, Jessie Yu, David McKay, Juan Gomez, Lauren Capelluto, Travis-S-IBM, Julien Gacon, Ashish Panigrahi, lerongil, Rafey Iqbal Rahman, Steve Wood, Luciano Bello, Divyanshu Singh, Drew, Joachim Schwarm, MELVIN GEORGE, Manoel Marques, Omar Costa Hamido, RohitMidha23, Sean Dague, and Shelly Garion. Qiskit/qiskit: Qiskit 0.26.2, May 2021.

Appendix A

IMPLEMENTATION OF GENERIC PHYSICAL RESOURCES ESTIMATION FRAMEWORK

*ToDo: Add source code that implements the physical resources estimation framework.

Source code can also be found in GitHub.