A High Performance Computing Infrastructure for the Efficient Execution of Hybrid Quantum-Classical Algorithms

Thesis submitted by

Steven Oud

under the guidance of

Harold Meerwaldt, QuTech
Damian Podareanu, SURF
Richard Versluis, TNO

in partial fulfillment of the requirements for the degree of

Bachelor of Science

A High Performance Computing Infrastructure for the Efficient Execution of Hybrid Quantum-Classical Algorithms

Steven Oud*
500776959

Faculty of Computer Science, Information Technology, Business IT and Management
Software Engineering

Advisor: Marten Teitsma

Amsterdam University of Applied Sciences
February 21, 2021

*Tel.: +31621451016

Abstract

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Contents

List of Figures

List of Tables

1

Introduction

As an introduction to the work that was done during this thesis, this chapter starts with presenting a brief overview on the relevance of quantum computing, and describing the motivation behind this work. After that, the objective of this work is described and an outline of this report is given.

1.1 Motivation

Quantum computing is a new model of computation that promises to solve certain problems more efficiently than classical computers by making use of quantum mechanical phenomena such as superposition and interference. The idea of quantum computing originated from Benioff [1], who proposed a quantum mechanical model of the Turing machine in 1980. This idea was later extended as Manin [2] and Feynman [3] independently suggested that quantum computers have the potential to solve certain computational problems intractable by classical computers. Since then, researchers have been searching for applications for quantum computing. Some noteworthy developments in the field of quantum computing include Shor's algorithm for factoring integers [4] and Grover's algorithm for unstructured database search [5]. These quantum algorithms promise an exponential and quadratic speedup respectively over their best-known classical counterparts. The finding of such speedups have catalyzed research towards quantum computers, and more applications have since been found in fields including chemistry [6], cryptography [7], and machine learning [8].

Until recently, quantum computing had been a mainly theoretical field. These days, however, thanks to recent technological advances, various quantum devices are being actively developed. Furthermore, technological giants such as IBM, Microsoft, Intel, and Google are investing heavily in the development of quantum computers. The applications of these current quantum devices are very limited however, due to their inability to store information for a long time and their sensitivity to errors. For instance, Fowler, Mariantoni, Martinis, and Cleland [9, Appendix M] estimate that to factor a 2000-bit number using Shor's algorithm would require a quantum computer with about a billion physical qubits, and error rates below 4×10^{-13}. In contrast, devices today have about 50 to 100 physical qubits with error rates above 0.1.

These small and error-prone quantum devices, referred to as noisy intermediate--scale quantum (NISQ) devices, may be unfit to run quantum algorithms like Shor's algorithm, but they may still prove to be useful and perform tasks intractable by classical computers [10]. To deal with the limitations of NISQ devices, hybrid quantum--classical algorithms (HQCAs) are being actively researched. These HQCAs combine classical and quantum computations as visualized in Figure 1.1. Hybrid quantum-

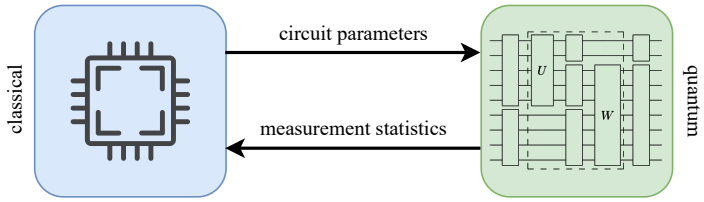

Figure 1.1: General structure of a hybrid quantum-classical algorithm. The data interchange between the classical and quantum part is often repeated many times.

-classical algorithms typically involve a small quantum computation inside of a classical optimization loop, greatly reducing the amount of quantum resources needed. This makes them suitable to run on NISQ devices, and are expected to be one of the first useful applications for quantum computing [11]. It is important to note that NISQ devices running HQCAs will likely not be revolutionary by itself. Instead, it should be seen as an important stepping stone towards more powerful quantum devices and algorithms.

Research towards HQCAs often involves executing quantum circuits on actual quantum chips or through quantum circuit simulation. To support this kind of research, classical and quantum computing facilities are needed. Furthermore, these facilities need to be connected and able to interchange data in a timely manner for this kind of research to be feasible. This is especially challenging given that both quantum and classical resources are shared with other users.

1.2 Objective

The purpose of this thesis is to propose an infrastructure for the efficient execution of hybrid quantum-classical algorithms. This work is in collaboration with TNO, QuTech, and SURF, and is focused on the efficient execution of HQCAs using QuTech's quantum computing platform Quantum Inspire [12] and SURF's high performance computing (HPC) center. To allow for the efficient execution of HQCAs, the SURF HPC and Quantum Inspire job schedulers should be synchronized to minimize the execution of the algorithm. A key part in this is figuring out sources of overhead and current bottlenecks. In a hybrid setup like this, overheads such as quantum and classic scheduling wait times, data transfers, and resource initialization can quickly increase the run time of HQCAs.

1.3 Outline

This report is structured as follows. Chapter 2 provides the reader with a background on computational complexity, quantum information, and quantum computation. Chapter 3 takes a deeper look into well-known HQCAs and their working. Chapter 4

demonstrates the practical execution of HQCAs using Quantum Inspire and SURF's HPC center.

2

Background

This chapter is focused on making the reader familiar with concepts used throughout this report. First, an introduction to computational complexity is given to establish a mathematical framework to describe the efficiency of computer algorithms. Second, the basic ideas of quantum information theory are presented. Finally, an overview of quantum computation is given.

2.1 Computational Complexity

In computer science, there seems to be a fundamental limit to what problems we can solve. Some problems seem to be inherently uncomputable: there exists no general solution that does not go into an infinite loop for certain inputs [13, 14]. This report will not go further into what problems are computable and uncomputable. Rather, it will look at the computational efficiency of certain algorithms: how many resources are required to solve a problem?

2.1.1 Big-O Notation

The time and space required by an algorithm generally grows as the size of the input grows. Because of this, it is traditional to describe the efficiency of an algorithm as a function of the size of its input [15]. This function describes the number of primitive operations it performs for a given input size. The notion of input size here depends on the context of the problem. For example, when computing the discrete Fourier transform, the input size refers to the dimension of the input vector. When talking about a problem like integer multiplication, however, it is more fitting to talk about the input size as the number of bits needed to represent the input in binary.

When analyzing the efficiency of algorithms, we look at the asymptotic growth for a given input size. Consider an algorithm that given input size n takes n^2 primitive operations to run and another algorithm that takes $500n^2 + \log n$ primitive operations to run. In big-O notation, both these algorithms are said to run in $O(n^2)$ time. That is, the number of primitive operations it performs scales quadratically with the input size. Constant factors are ignored as they become negligible as $n \to \infty$. While they are practically significant — an algorithm that runs in $O(n/2)$ runs twice as fast as an algorithm that runs in $O(n)$ — they are not relevant to asymptotic analysis.

Formally, if we have functions $f(n)$ and $g(n)$ such that f eventually grows slower than some multiple of g as $n \to \infty$, we say $f(n) = O(g(n))$. For example, given $f(n) = 200n^2$ and $g(n) = n^3$, f begins to grow slower than g when $n > 200$. Thus, g bounds f from above, and $f(n) = O(g(n)) = O(n^3)$. Some common big-O run times are shown in Table 2.1, along with their written name and an example. Throughout this report, algorithms that are bounded above by a polynomial (i.e. all run times until polynomial in Table 2.1) will be referred to as polynomial-time algorithms, and algorithms that are not bounded above by a polynomial will be referred to as superpolynomial-time algorithms.

Notation	Name	Example
$O(1)$	Constant	Accessing single element from array
$O(\log n)$	Logarithmic	Binary search
$O(n)$	Linear	Unstructured database search
$O(n \log n)$	Linearithmic	Fast Fourier Transform
$O(n^2)$	Quadratic	Insertion sort
$O(n^k)$	Polynomial	Gaussian elimination
$O(k^n)$	Exponential	Graph coloring
$O(n!)$	Factorial	Brute-force search traveling salesman problem

Table 2.1: Common big-O run times from fast to slow.

2.1.2 Turing Machines

The previous section described the measurement of computational efficiency as the number of primitive operations it performs for a given input size. This abstract definition can be extended by choosing a computational model in order to define what a primitive operation means. The standard computational model used for this is the Turing machine. It is chosen as computational model for the analysis of computational efficiency because of its simplicity and because it is able to simulate most physically realizable computational models with little overhead [16].

A Turing machine is an abstract machine that manipulates symbols from a work alphabet on a finite amount of one-way infinite length tapes divided into cells [14] (Figure 2.1). Along these tapes runs a tape head that can read and write one symbol at a time. The machine has a finite set of states, which the machine executes one at a time by loading them into the state register. At any time, the machine can be in one of the finite states. A state can be thought of as a rule with the following form:

$$(q_i, a) \mapsto (q_j, b, H), \tag{2.1}$$

where q_i and q_j are states, a and b are symbols from the work alphabet, and $H \in \{L, S, R\}$ decides how to move the tape head: one cell to the left (L), stay in the same position (S), or one cell to the right (R). These states as described in Equation 2.1 can be read as "in state q_i, if the read symbol is a, go to state q_j, write symbol b, and move the tape head to H".

Everything that can be computed on models of computations we use these days can be computed on a Turing machine [18]. This hypothesis is known as the Church-Turing thesis. Related to the Church-Turing thesis is the extended Church-Turing thesis, which states that any physically realizable model of computation can be efficiently

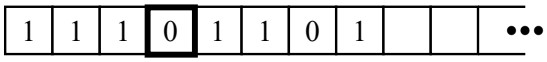

Figure 2.1: The tape of a single-tape Turing machine in an arbitrary state. Note that any multi-tape Turing machines can be efficiently simulated by a single-tape Turing machine [17], so complexity classes are not affected by changing between single-tape and multi-tape machines.

simulated on a Turing machine. That is, can a Turing machine simulate any model of computation in polynomial time? The quantum computational model brings doubt to this claim. It is known that quantum computers can efficiently simulate a Turing machine, so they are at least as powerful as classical computers [19]. However, there appears to be no efficient algorithm for simulating a quantum computer on a Turing machine [20]. Furthermore, Arute, Arya, Babbush, *et al.* [21] experimentally demonstrated a quantum computer sampling from a probability distribution intractable by a classical computer.

2.1.3 Complexity Classes

Complexity classes are sets of computational problems that share some common feature with regard to the computational resources they need to solve some problem [16]. They are defined in terms of a type of computational problem, computational model, and a bounded resource such as time or space. In general, most complexity classes describe decision problems solvable by deterministic Turing machines — though many complexity classes are defined in terms of other types of problems and computational models. This report mainly focuses on complexity classes involving Turing machines and quantum Turing machines.

The class **P** contains all decision problems solvable by a deterministic Turing machine in polynomial time. Problems that fall under this class are often referred to as tractable or easy problems [15]. The class **NP** (non-deterministic polynomial) contains all problems *verifiable* by a deterministic Turing machine in polynomial time. Equivalently, **NP** can be thought of as all problems solvable in polynomial time by a non-deterministic Turing machine. A non-deterministic Turing machine is a variant of the Turing machine which is not entirely determined by its input and transition function, but can choose from a set of possible transitions when transitioning. One could then define **NP** as consisting of two phases: first, a non-deterministic Turing machine makes a guess about the solution, and then a second, deterministic Turing machine verifies if the guess is correct. It is clear that $\mathbf{P} \subseteq \mathbf{NP}$, because if you can solve a problem in polynomial time, you can also verify it in polynomial time. A still unsolved and important question in computer science is whether $\mathbf{P} = \mathbf{NP}$? That is, can all problems that can be verified in polynomial time also be solved in polynomial time?

In computer science, it is sometimes possible to speed up computation using randomness. These kinds of algorithms are referred to as probabilistic algorithms and are defined in terms of a probabilistic Turing machine. A probabilistic Turing machine is a non-deterministic Turing machine that can choose from a set of possible transitions according to some probability distribution when transitioning. The probabilistic equivalent of **P** is **BPP** (bounded-error probabilistic polynomial time) and contains all decision problems solvable by a probabilistic Turing machine in polynomial time where a bounded error rate of 1/3 is allowed. Since a non-deterministic Turing machine can efficiently simulate a deterministic Turing machine, $\mathbf{P} \subseteq \mathbf{BPP}$. There are problems to

be known in **BPP** and not in **P**, but the number of such problems is decreasing, and Goldreich [22] and Nisan and Wigderson [23] even argue that **P** = **BPP**.

How do quantum computers relate to these complexity classes? Quantum computers are probabilistic computational devices, and its complexity class equivalent to **P** can be defined by replacing the probabilistic Turing machine from **BPP** with a quantum computer.[1] The class **BQP** (bounded-error quantum polynomial time) consists all decision problems solvable by a quantum computer in polynomial time where a bounded error rate of 1/3 is allowed. It is known that there are **NP** problems that can be efficiently solved on a quantum computer like integer factorization, discrete logarithms, and quantum many-body simulation. As mentioned in the previous section, quantum computers can also solve all problems in **P** efficiently, so **P** $\subseteq$ **BQP**. Furthermore, quantum computers are more powerful than classical probabilistic computers [24], giving **BPP** $\subseteq$ **BQP**. How **BQP** relates to **NP** exactly is still unknown, however, it seems unlikely that **BQP** = **NP** [25].

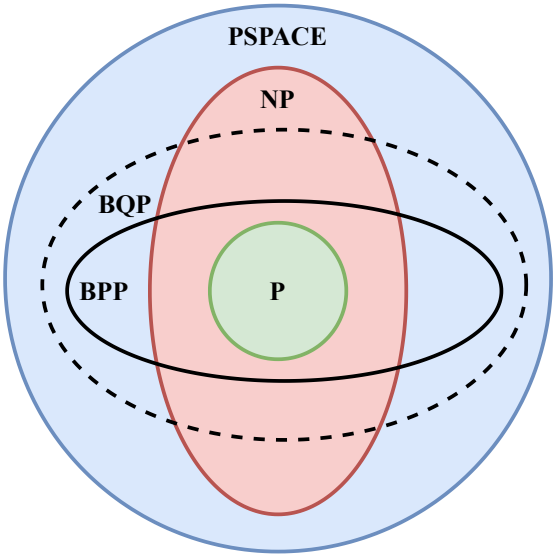

Figure 2.2: An overview of the hierarchy of the complexity classes discussed. **PSPACE** is the space equivalent of **P**, containing all problems that can be solved in polynomial space by a deterministic Turing machine.

2.2 Quantum Information

In classic information theory, the smallest unit of information is the bit. A bit can be in one of two states: 0 or 1.

2.3 Quantum Computation

[1] Note that quantum computers are not simply probabilistic Turing machines as will be shown in the following sections.

3

Hybrid Quantum-Classical Algorithms

3.1 Variational Quantum Eigensolver

3.2 Quantum Approximate Optimization Algorithm

4

Practical Hybrid Quantum-Classical Computing

This chapter is focused on the practical execution of HQCAs using the Quantum Inspire quantum computing platform and SURF's HPC center. First, an overview of the classical and quantum infrastructure is given. Then, HQCAs discussed in Chapter 3 are implemented and used for benchmarking and identifying the bottlenecks of the current infrastructure.

4.1 Quantum Inspire

4.2 Implementation

4.3 Benchmarks

5

Conclusion

Bibliography

[1] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines", *Journal of statistical physics*, vol. 22, no. 5, pp. 563–591, 1980.

[2] I. I. Manin, *Vychislimoe i nevychislimoe*. Sov. radio, 1980.

[3] R. P. Feynman, "Simulating physics with computers", *Int. J. Theor. Phys*, vol. 21, no. 6/7, 1982.

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[5] L. K. Grover, "A fast quantum mechanical algorithm for database search", in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[6] S. McArdle, S. Endo, A. Aspuru-Guzik, *et al.* (2018). Quantum computational chemistry. arXiv: 1808.10402 [quant-ph].

[7] C. H. Bennett and G. Brassard, "Quantum cryptography", *Theoretical Computer Science*, vol. 560, no. P1, pp. 7–11, 2014.

[8] J. Biamonte, P. Wittek, N. Pancotti, *et al.*, "Quantum machine learning", *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[9] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation", *Physical Review A*, vol. 86, no. 3, p. 032 324, 2012.

[10] J. Preskill, "Quantum computing in the NISQ era and beyond", *Quantum*, vol. 2, p. 79, 2018.

[11] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan, "Hybrid quantum-classical algorithms and quantum error mitigation", *Journal of the Physical Society of Japan*, vol. 90, no. 3, p. 032 001, 2021.

[12] T. Last, N. Samkharadze, P. Eendebak, *et al.*, "Quantum inspire: QuTech's platform for co-development and collaboration in quantum computing", in *Novel Patterning Technologies for Semiconductors, MEMS/NEMS and MOEMS 2020*, International Society for Optics and Photonics, vol. 11324, 2020, 113240J.

[13] A. Church *et al.*, "A note on the entscheidungsproblem", *J. Symb. Log.*, vol. 1, no. 1, pp. 40–41, 1936.

[14] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem", *Proceedings of the London mathematical society*, vol. 2, no. 1, pp. 230–265, 1937.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[16] S. Arora and B. Barak, *Computational complexity: A modern approach*. Cambridge University Press, 2009.

[17] J. Hartmanis and R. E. Stearns, "On the computational complexity of algorithms", *Transactions of the American Mathematical Society*, vol. 117, pp. 285–306, 1965.

[18] N. Dershowitz and Y. Gurevich, "A natural axiomatization of computability and proof of church's thesis", *Bulletin of symbolic logic*, vol. 14, no. 3, pp. 299–350, 2008.

[19] C. H. Bennett, "Logical reversibility of computation", *IBM journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[20] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer", *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985.

[21] F. Arute, K. Arya, R. Babbush, *et al.*, "Quantum supremacy using a programmable superconducting processor", *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[22] O. Goldreich, "In a world of P= BPP", in *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, Springer, 2011, pp. 191–232.

[23] N. Nisan and A. Wigderson, "Hardness vs randomness", *Journal of computer and System Sciences*, vol. 49, no. 2, pp. 149–167, 1994.

[24] E. Bernstein and U. Vazirani, "Quantum complexity theory", *SIAM Journal on computing*, vol. 26, no. 5, pp. 1411–1473, 1997.

[25] S. Aaronson, "BQP and the polynomial hierarchy", in *Proceedings of the forty-second ACM symposium on Theory of computing*, 2010, pp. 141–150.

Acronyms

HPC High Performance Computing

HQCA Hybrid Quantum-classical Algorithm

NISQ Noisy Intermediate-scale Quantum