CHRONOSCOPE: Circuit Heuristics for Runtime Optimization and NISQ Overhead Scaling via Critical-path Observation and Performance Estimation

Colin Maggard[1,2], Henry Wysong-Grass[1,2]
[1]Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI 53706 USA
[2]Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI 53706 USA

***Abstract*—Quantum computing in the Noisy Intermediate-Scale Quantum (NISQ) era presents unique challenges for resource estimation due to the absence of full error correction, variable gate fidelities, and limited qubit connectivity. We introduce** CHRONOSCOPE **(Circuit Heuristics for Runtime Optimization and NISQ Overhead Scaling via Critical-path Observation and Performance Estimation), an analytical framework for estimating quantum circuit execution time and assessing the impact of parallelism on runtime performance.** CHRONOSCOPE **evaluates circuit behavior across diverse topologies, technologies, and levels of parallel execution, drawing parallels to classical performance models such as Amdahl's Law and the Iron Law. Our results show that trapped-ion topologies offer superior scaling for problem sizes that align with trap sizes due to their intra-trap connectivity, whereas mesh topologies suffer from SWAP overhead and poor scalability. We demonstrate that performance depends strongly on the interplay between circuit structure and topology, underscoring the importance of topology-aware circuit design and compiler strategies in the NISQ era.**

I. Introduction

Quantum computing is a rapidly evolving computational paradigm that shows promise for solving certain classes of problems that are intractable for classical systems. However, unlike classical computing, which benefits from mature and well-established resource estimation methods based on program characteristics and cycle times, quantum computing presents new challenges. These include the impact of qubit topology, limited gate parallelization, variable gate latencies and fidelities, and the ever-present influence of noise. While existing frameworks like QuRE [1] provide valuable insights for error-corrected quantum computing systems, they are often geared toward fully fault-tolerant machines that remain out of reach with current technology. Instead, near-term quantum computing research must contend with the limitations of noisy intermediate-scale quantum (NISQ) systems, which operate without full error correction and are constrained by hardware capabilities. Unfortunately, there are very few resources for estimation of execution time on quantum circuits, which is detrimental since these circuits can be prohibitively expensive to run on web services [2] [3]. Classical systems benefit from performance models for both sequential and parallel systems, namely Iron Law and Amdahl's Law [4], but quantum systems do not have similar, easily defined models.

This manuscript introduces a fast, scalable, and customizable framework for assessing execution time for NISQ circuits, including the effect of parallelism on execution time. This framework, titled CHRONOSCOPE, is purely analytical and does not rely on simulation to calculate execution time.

The key contributions of this paper are summarized below:

- We propose an analytical model for estimating execution time of a quantum circuit with a given qubit technology, qubit network topology, algorithm and number of qubits.
- We calculate the scalability of speedup versus the number of circuit copies across a wide range of qubits on mesh and trapped-ion topologies for Quantum Approximate Optimization Algorithm (QAOA) and Quantum Fourier Transform (QFT).
- We analyze the results of scaling various problem sizes on different topologies and conclude on considerations that must be made to optimize quantum circuits.

This framework is a preliminary tool to encourage analytical performance modeling for NISQ-era quantum hardware since fully error-corrected hardware is still years to decades away.

II. Background

A. Noisy Intermediate Scale Quantum Computing

Noisy Intermediate-Scale Quantum (NISQ) computing refers to the current era of quantum technology, characterized by quantum processors with tens to a few hundred qubits that are powerful but still limited by

noise and imperfect control. The term was coined by John Preskill to describe a transitional phase between today's small-scale quantum devices and the future fault-tolerant quantum computers. These devices are "noisy" because each gate operation and measurement is subject to errors, and they are "intermediate-scale" because they are significantly larger than earlier prototypes but not yet large enough to realize full fault tolerance [5].

Despite their limitations, NISQ processors offer a promising platform to explore the power of quantum mechanics in computation. Potential applications include quantum simulation of chemical systems and materials, optimization problems, and machine learning tasks. However, their utility in practical applications remains uncertain and is a major topic of ongoing research.

In order to mitigate the noise caused by gate operations and measurements, as well as various other sources of noise, NISQ circuits must be run for multiple "shots" or circuit executions, which is analogous to running a classical program multiple times. Running a circuit for thousands or tens of thousands of shots provides a probability distribution from which the correct outcome of the quantum computation can be inferred. However, this increases total execution time, especially when shots are run sequentially. It may be possible to decrease overall execution time by running shots in parallel with multiple copies of the same circuit, which is one of the primary focuses of this paper.

B. Prior Work

Several prior efforts have tackled quantum performance and resource estimation, but most are either focused on fault-tolerant quantum computing or abstract away critical runtime and architectural details relevant to today's NISQ devices:

QuRE Toolbox [1]. QuRE provides analytical resource estimation tailored to fully fault-tolerant systems, estimating logical gate counts, qubit overhead, and T-depth under error correction assumptions. While valuable for long-term projections, it lacks applicability for near-term NISQ devices, where error correction is absent and physical noise plays a dominant role, requiring multiple shots per circuit to get accurate results.

Azure Quantum Resource Estimator [6]. Microsoft's estimator offers high-level insights into quantum algorithm resource needs under error correction. However, it is not designed to model noisy execution or circuit-level performance on current devices. Moreover, the underlying modeling techniques are opaque, limiting extensibility for runtime analysis.

SupermarQ [7]. This benchmarking suite compares quantum algorithms across hardware backends using technology-agnostic metrics such as parallelism and critical path length. While it facilitates cross-platform evaluation, SupermarQ does not directly model execution latency or simulate noise-aware performance impacts or the effect of multiple shots on execution time, making it less suitable for runtime analysis.

MQT Bench [8]. A recent tool targeting NISQ-era benchmarking, MQT Bench enables detailed comparisons of software and hardware performance using realistic circuit metrics such as gate counts, depth, and noise-aware simulation fidelity. It provides a more grounded view of near-term quantum system behavior but does not offer runtime modeling natively. This technology can be extended with analytical techniques to estimate circuit latency, which is what CHRONOSCOPE does.

Distributed Shot Execution [9]. Cut&Shoot splits up a single circuit copy into independent chunks and spreads the shots of each fragment across multiple quantum processing units (QPUs) and combines the results later to get the final result. This is a prime example of how quantum computing can be done in parallel to reduce overall execution time.

Early Fault-Tolerance Perspectives [10]. Chong et al. emphasize the hybrid nature of near-term quantum applications, proposing variational and adaptive methods to bridge the gap between NISQ and fault-tolerant regimes. However, their focus is algorithmic rather than architectural, and they do not provide detailed execution-time predictions.

These tools and concepts either target distant-future architectures or abstract away runtime details that are crucial for understanding real-world performance on current devices.

III. CHRONOSCOPE

Our work introduces CHRONOSCOPE, a toolkit for analytically estimating execution time of quantum circuits under varying qubit topologies and gate parameters. We develop a parallel execution model that emulates circuit cloning and determines speedup from critical-path execution analysis. Our tool can be customized for different connectivity graphs, technologies, and algorithms, producing plots of runtime and speedup as a function of parallelism.

A. Single-Copy Execution Time Estimation Features

Transpilation & Gate Counting. Each circuit is transpiled to specific topologies chosen for their popularity in research and industry. Any topology can be entered, but the predefined topologies that come with our framework are a 100-qubit mesh, and 20x5-qubit and 10x10-qubit trapped ion topologies. A given circuit is transpiled to a technology's native gate set and mapped to the provided topology by MQT Bench's transpiler functionality.

Critical Path Analyzer (CPA). We have developed a function that iterates over each qubit in a circuit and determines the maximal time-weighted qubit path. This critical path analyzer is run on the current circuit copy in its given topology; i.e. the size and layout of the circuit determines SWAP overhead which is taken into account with the CPA.

Execution Time Calculation. Our tool optimistically assumes that an optimal, or close to optimal, transpilation is performed before every run by performing multiple transpilations, applying CPA and then choosing the lowest cost circuit version for analysis. Gate latencies are each multiplied by the number of corresponding gates, and summed to find total circuit latency. Idling latencies are also calculated and added in.

B. Parallelism Analysis Features

Multiple Copies. Multiple copies of a quantum circuit of a given size can be mapped to the same topology and run in parallel, decreasing the number of shots that need to be run sequentially and reducing estimated execution time. This is analogous to the classical Amdahl's Law [4].

Execution Time Calculation. The number of shots necessary is divided by the number of copies, and the latency of each copy on the given topology is factored in to determine total execution time. As the topology nears capacity with an increasing number of circuit copies, the mapping becomes more constrained and more SWAP gates are necessary to execute all circuit copies in parallel, which increases circuit latency for each copy.

C. Other Features

We also implemented all of the features from SupermarQ [7] as well as a rudimentary shot estimation framework based on estimated fidelity and a complementary topology visualizer for network verification.

IV. Methodology

Qubit Technologies. We decide to use 1-qubit and 2-qubit gate latencies for IonQ's trapped ion qubits across all topologies and algorithms. Keeping this property constant will allow for more accurate conclusions to be drawn across topologies and number of copies. Additionally, by calculating speedup time, individual gate latencies are irrelevant and the ratio between 1-qubit and 2-qubit gate latencies becomes the dominating factor in determining speedup.

Qubit Topologies. We evaluate circuits on three 100-qubit topologies: 1) a 100-qubit mesh, 2) a 20x5 qubit trapped ion topology, and 3) a 10x10 qubit trapped ion topology. In the trapped ion topology, each subgroup (5, 10) is fully connected and one qubit from each subgroup is connected to a corresponding qubit from two other subgroups.

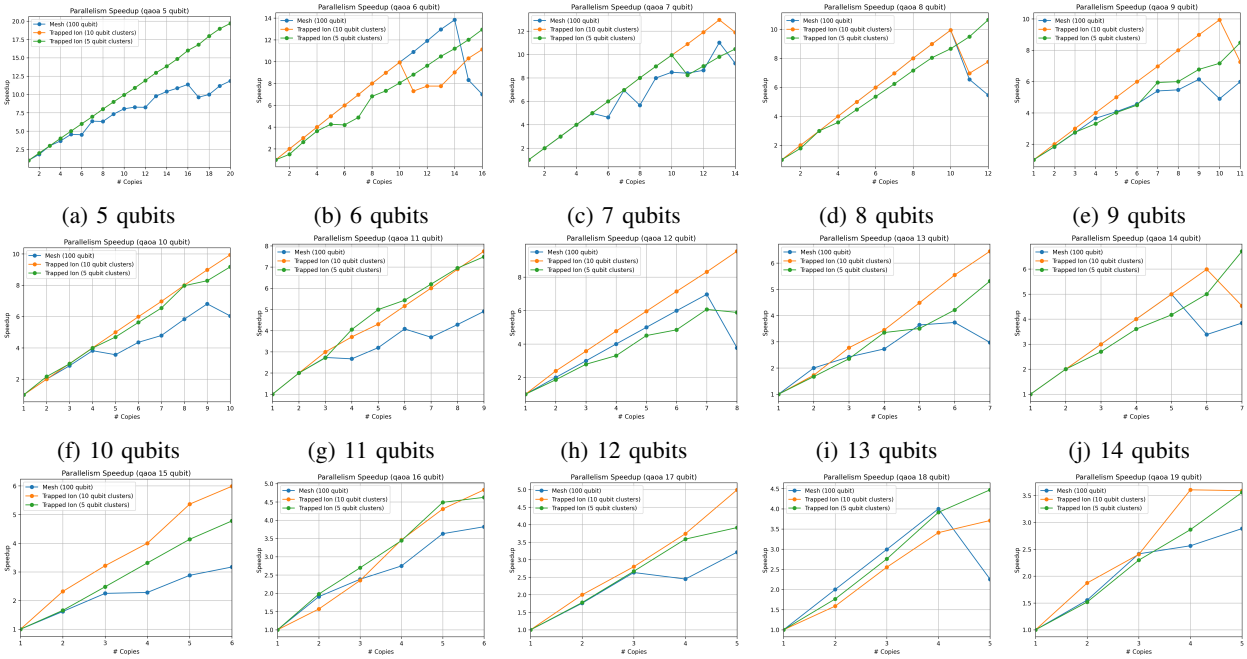
Quantum Algorithms. We evaluate Quantum Approximate Optimization Algorithm (QAOA) and Quantum Fourier Transform (QFT) as representative samples of relevant and common quantum algorithms.

Execution Time Analysis. Since a circuit can include many parallel copies of a circuit, it is important to distinguish the difference between a single shot of an algorithm compared to a shot of a whole circuit. A single algorithm execution will be referred to as a single shot, whereas a shot of an entire circuit, potentially containing many copies of the algorithm, is a circuit shot. A single shot's execution time will always be equal to or less than the execution time of a circuit. To evaluate this, we start by performing multiple transpilations of the benchmark circuit and performing CPA to determine the execution time of a circuit shot. In determining the execution time, we pick the transpilation that offers the lowest cost based on CPA, optimistically selecting the the best transpilation. The execution time of a single algorithm shot is calculated by dividing the circuit-level cost by the number of algorithm copies. This process is performed for algorithm sizes of 5-19 qubits for all chosen benchmarks on each topology.

Parallelism Analysis. We evaluate each circuit using 1 to $\lfloor \frac{100}{n_{\text{qubits}}} \rfloor$ parallel copies, where n_{qubits} is the number of qubits used in the circuit. It is necessary to floor this quantity to prevent attempting to analyze 'fractional' circuits. To assess the scalability and speedup associated with n_{copies}, we perform CPA and apply the formula

$$Cost_S = \lceil \frac{S}{\#copies} \cdot Cost_{circuit} \rceil \tag{1}$$

to calculate the execution time of the circuit, $Cost_S$, which is the execution time required to run S shots on a circuit with execution time $Cost_{\text{circuit}}$. The speedup is then calculated as the ratio between the cost of the parallel circuit and the cost of the circuit with no parallel copies.

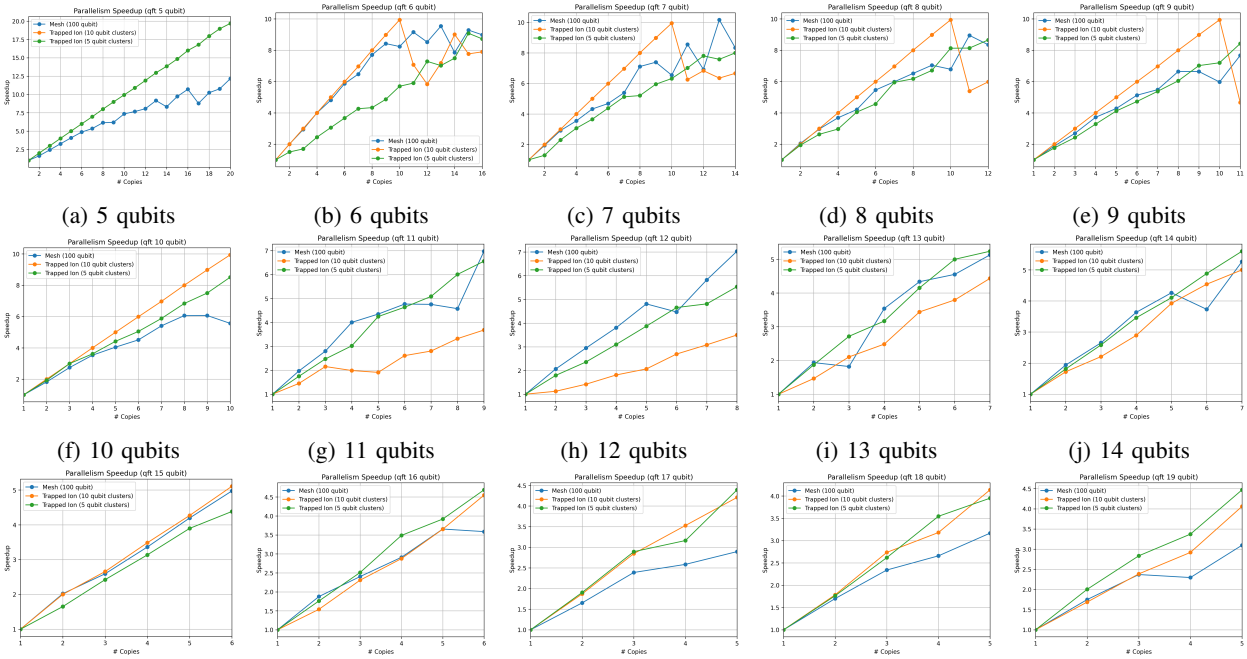


(a) 5 qubits (b) 6 qubits (c) 7 qubits (d) 8 qubits (e) 9 qubits

(f) 10 qubits (g) 11 qubits (h) 12 qubits (i) 13 qubits (j) 14 qubits

(k) 15 qubits (l) 16 qubits (m) 17 qubits (n) 18 qubits (o) 19 qubits

Fig. 1: QAOA speedups across qubit counts. Orange = 10x10 TI, Green = 20x5 TI, Blue = 100 Mesh



(a) 5 qubits (b) 6 qubits (c) 7 qubits (d) 8 qubits (e) 9 qubits

(f) 10 qubits (g) 11 qubits (h) 12 qubits (i) 13 qubits (j) 14 qubits

(k) 15 qubits (l) 16 qubits (m) 17 qubits (n) 18 qubits (o) 19 qubits

Fig. 2: QFT speedups across qubit counts. Orange = 10x10 TI, Green = 20x5 TI, Blue = 100 Mesh

V. Evaluation

A. Quantum Approximate Optimization Algorithm

The following conclusions can be drawn from the graphs provided for the relative speedups of QAOA with varying numbers of qubits with a number of copies proportional to $\frac{100}{num_{\text{qubits}}}$:

100-qubit Mesh Topology. The mesh topology never achieves perfectly linear speedup, and has a drastic decrease in proportional speedup as the number of copies increases due to limited connectivity and the excessive SWAP overhead necessary. As problem size scales up, the relative speedup gained via adding additional circuit copies decreases as well. Mesh only outperforms the other two categories for 6 and 18 qubit problem sizes, though this lead is removed around the maximum capacity of the network. In general, this topology performs the worst.

20x5-qubit Trapped Ion Topology. This trapped ion topology typically performs somewhere between the mesh and 10x10 trapped ion topologies in terms of linearity, and occasionally surpasses both, such as with problem sizes of 14 and 18. There are not typically sharp drop-offs from linearity of performance, but there are problem sizes that do not scale well due to limited connectivity between traps and the necessity of adding SWAP gates, such as 12 qubits.

10x10-qubit Trapped Ion Topology. This trapped ion topology scales the best on average, with nearly linear speedup for most problem sizes as the number of copies increases. It has sharp drop-offs from linearity of performance on the following combinations due to limited connectivity between traps and the necessity of adding SWAP gates:

- 6 qubits with 10+ copies
- 8 qubits with 8+ copies
- 9 qubits with 10+ copies
- 14 qubits with 6+ copies

To conclude, problem size is very important with respect to the topology being used; circuits whose copies can neatly "fit" within a well-connected subsection of the topology (e.g. a trap in the trapped ion topology) tend to scale better than circuits that are placed in areas with little connectivity. With that in mind it is important to use topologies which support the circuit being tested for the amount of copies required.

B. Quantum Fourier Transform

The following conclusions can be drawn from the graphs provided for the relative speedups of QFT with varying numbers of qubits with a number of copies proportional to $\frac{100}{num_{\text{qubits}}}$:

100-qubit Mesh Topology. The mesh topology has similar performance to the other two, with the exception of 5, 10, and 17+ qubits where it scales significantly worse. Mesh tends to outperform the trapped ion topologies on problem sizes that are not divisible by 5, which more cleanly fit into the traps used by the other topologies.

20x5-qubit Trapped Ion Topology. As the problem size increases, this topology tends to perform better, with suboptimal performance for problem sizes less than 11 (with the exception of 5) and superior performance for problem sizes greater than 15. This indicates that there are relatively few swaps required between traps, which prevents latency from skyrocketing due to SWAP overhead.

10x10-qubit Trapped Ion Topology. Surprisingly, this topology performs much worse than it does for QAOA on problem sizes greater than 10, indicating that there is much inter-trap traffic leading to high SWAP overhead and associated latency. This topology is still superior for problem sizes 10 or greater, however.

VI. Discussion

Our model reveals that speedup from circuit parallelism is nearly linear for topologies with good connectivity and limited SWAP requirements. Trapped-ion devices (with all-to-all connectivity) exhibited the best scaling when problem size correlated well with the size of the traps, while mesh showed diminishing returns due to SWAP overhead. However, when there was significant inter-trap traffic, trapped-ion often performed worse due to SWAP bottlenecks at the points of connection to other traps. Therefore, it is important to understand the size and SWAP overhead of a given circuit to effectively select a topology which does not throttle performance.

We observed that compiler-based layout optimization is often superior to manual placement strategies. We also noted difficulty with implementing poorly documented topologies like heavy hex, which can be manually integrated into our framework if desired.

VII. Conclusion

The CHRONOSCOPE framework is a step in a promising direction in the domain of quantum performance modeling. By offering an analytical, simulation-free method for estimating circuit execution time—while accounting for hardware topology, gate latencies, and critical-path dependencies, CHRONOSCOPE fills a gap left by current tools that either focus narrowly on theoretical error-corrected systems or ignore runtime behavior altogether.

The framework's use of established tools like MQT Bench and SupermarQ further grounds it in the broader ecosystem while extending their utility with critical-path-based runtime estimation. This dual emphasis on modularity and topology awareness makes CHRONOSCOPE

especially valuable for developers, researchers, and hardware architects seeking fast, comparative insights into circuit performance across different NISQ platforms.

However, the tool's optimistic assumptions about transpilation efficiency, uniform gate latencies, and limited consideration of realistic noise effects must be addressed in future iterations to enhance predictive fidelity. Incorporating empirical validation through simulation or execution on actual hardware and expanding support for diverse quantum technologies and topologies will be critical to transitioning CHRONOSCOPE from a conceptual model into a fully-fledged resource for quantum algorithm deployment and cost estimation in the NISQ era. Additionally, more effort is required to create functionality that can estimate the number of shots required to accurately infer results from a given quantum circuit on a given setup.

REFERENCES

[1] M. Suchara, A. J. McCaskey, and T. S. Humble, "Estimating the resources for quantum computation with the qure toolbox," *arXiv preprint arXiv:1304.2333*, 2013.

[2] Amazon Web Services, "AWS Braket Pricing." https://aws.amazon.com/braket/pricing/, 2024. Accessed: 2025-05-02.

[3] IBM Quantum, "IBM Quantum Pricing." https://www.ibm.com/quantum/pricing, 2024. Accessed: 2025-05-02.

[4] L. Eeckhout, *Computer Architecture Performance Evaluation Methods*. Morgan & Claypool Publishers, 2010.

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

[6] S. Lopez Bravo, "Introduction to the resource estimator - azure quantum." https://learn.microsoft.com/en-us/azure/quantum/intro-to-resource-estimation, jul 2024. Microsoft Learn, [Online].

[7] T. Tomesh, A. Hein, V. Gheorghiu, V. Kliuchnikov, N. Wiebe, *et al.*, "Supermarq: A scalable quantum benchmark suite," *arXiv preprint arXiv:2202.11045*, apr 2022.

[8] N. Quetschlich, L. Burgholzer, and R. Wille, "Mqt bench: Benchmarking software and design automation tools for quantum computing," *Quantum*, vol. 7, pp. 1062–1062, jul 2023.

[9] S. Lee, C. Lee, K. G. Mun, and D. Kim, "Decision tree algorithm considering distances between classes," *IEEE Access*, vol. 11, pp. 12345–12356, 2023.

[10] F. T. Chong *et al.*, "Variational quantum algorithms in the era of early fault tolerance," *arXiv preprint arXiv:2503.20963*, 2025.