Accelerating Quantum Subcircuit Reconstruction Utilizing Multi-Node Computation

Chuck Garcia
University of Texas at Austin
chuckgarcian@utexas.edu

Ellie Vogel
Duke University
ellievogel@duke.edu

Wei Tang
Princeton University
wtang@princeton.edu

Margaret Martonosi
Princeton University
mrm@princeton.edu

***Abstract*—Today's Quantum Computers (QCs) face significant engineering challenges that limit their size and fidelity. To execute realistic applications, researchers must develop ways to move past these constraints. CutQC enables small QCs to run larger quantum circuits by cutting circuits into smaller subcircuits and then utilizing classical resources and Kronecker products to reproduce the desired output. In this work, we enhance the reconstruction process. In particular, we develop a distributed PyTorch implementation of CutQC's classical post-processing step, which can run across multiple nodes on either GPU or CPU devices. Our results show that our PyTorch implementation executes circuits of up to 35 qubits on 10- and 15-qubit QCs with efficient reconstruction. We use single-node computation as a baseline and then compare the results to the runtimes of our PyTorch workflows. By utilizing parallelism and specified devices, the reconstruction step of hybrid quantum-classical circuit execution can be improved. The original CutQC paper demonstrated a 60X to 8600X runtime speedup over classical simulation, and our work increases this speedup. These results show that large quantum circuits can be run on smaller QCs by leveraging circuit cutting with data parallelism techniques to maintain a reasonable runtime. This allows researchers to maintain the high fidelity of smaller machines while executing larger circuits.**

***Index Terms*—Quantum Computing, Multi-Node Processing, GPU Processing, Quantum Circuit Cutting, Hybrid Computing**

I. Introduction

Quantum computing represents a transformative opportunity in computational technology, with the potential to solve classically intractable problems [1]. However, QC is currently restricted to NISQ (Noisy Intermediate-Scale Quantum) devices, where limited qubit counts constrain the size of runnable applications. CutQC [2] is an open-source code repository that offers a scalable hybrid computing framework. Using classical computing resources for reconstruction enables the evaluation of quantum circuits that are too large to be run with high fidelity on today's quantum computers (QCs).

It is relevant to this study to note both classical and quantum computing trends, in particular the proliferation of Graphics Processing Units (GPUs) and a diverse array of Single Instruction, Multiple Data (SIMD) architectures. These developments have led to the use of multi-node, multi-threaded computation [3], ushering in unprecedented opportunities for the intersection of classical and quantum computing. Specifically, these devices can quickly compute large numbers of Kronecker Products (KPs), which we utilize to reconstruct outputs from the produced subcircuits. Using these devices in hybrid quantum computation increases the size of potential applications that can complete within a reasonable runtime.

The very characteristics of QC that make its hardware implementations promising for currently-intractable problems are also ones that limit our ability to classically-simulate QC systems [4]. Quantum computing's advances have recently included high-quality qubits [5]; recent advancements have led to Quantinuum's trapped-ion QCs with 56-qubits, arbitrary connectivity, and 99.843(5)% two-gate fidelity [4]. For superconducting QCs, IBM has released technology with both high qubit counts and high quality qubits. Increased emphasis has also been placed on quantum error correction [6] [7].

The ability to cut a quantum circuit is founded in the fact that an arbitrary quantum operation can be decomposed into any set of orthonormal matrix bases, but with an exponentially higher overhead from reconstructing the partitioned subcircuits. [8] The number of KPs required for reconstruction is 4^K, with K being the number of edges cut.

This study advances circuit cutting, a promising method for increasing the scale of executable circuits. Our enhanced reconstruction process demonstrates the integration of high-performance computing techniques with quantum algorithms.

II. Methodology

We first discuss our primary research goals, followed by research design methodology and an overview of our implementation.

This work's primary goal is to update CutQC's implementation from a TensorFlow predecessor to the new distributed parallel PyTorch implementation discussed here. This new implementation offers ease of use (PyTorch is more widely supported) and ease of mapping onto a rich variety of platforms. By scaling out mappings to multiple GPUs, the PyTorch implementation also offers the promise for runtime speedup.

The previous implementation employs single-layer vectorization over a given problem instance to execute individual KPs in parallel. However, since vectorization occurs along a single dimension, the maximum number of simultaneous KPs is upper-bounded by the capabilities of a single node. Our implementation introduces "multi-layered vectorization"

This material is based upon work supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Co-design Center for Quantum Advantage (C2QA) under contract number DE-SC0012704.

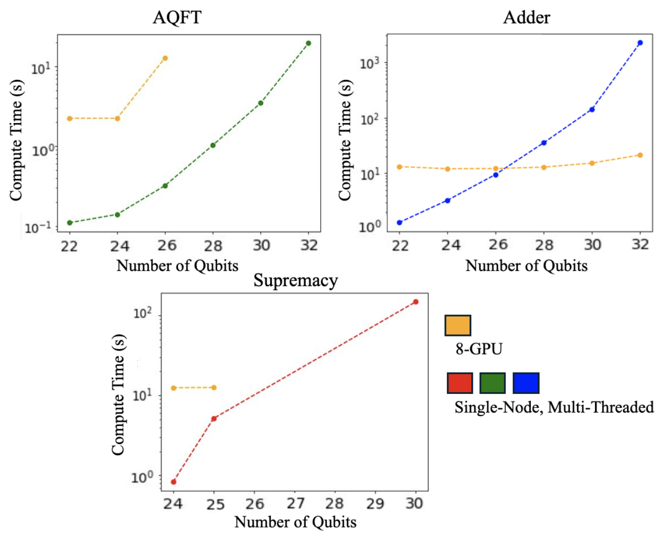

Fig. 1. Runtimes for circuits with increasing qubit counts in both the single-node, multi-threaded implementation and the 8-GPU implementation. Missing data points are attributed to the memory constraints of only using 8 GPUs. Initially, the size of the circuit is not large enough to counteract the various costs of using multiple nodes. As the circuit size grows, the improvement in runtime becomes more clear.

by separating the vectorized KPs into batches. This approach allows us to have N arbitrary points of vectorization occurring simultaneously, where N represents the number of available nodes.

The circuits we execute in this paper are as follows. The Approximate Quantum Fourier Transform (AQFT) is a common subroutine in promising quantum algorithms [9]. We also utilize the Bernstein-Vazirani algorithm, which solves the hidden string problem and requires $n + 1$ qubits for a binary string of length n. [10]. Finally, the adder circuit is a ripple-carry adder with one ancilla and linear depth [11].

We run the classical reconstruction post-processing step on circuits with an increasing number of qubits for a varying number of nodes. We first run the circuits on one node, with 64 allocated CPU cores and 7 GB allocated memory per core. We then run the circuits on 2 machines, with 4 GPUs each totalling 8 GPUs.

For a given reconstruction problem instance, our implementation follows a coordinated batch processing scheme [12]. Coordination and creation of batches is orchestrated by the host node, using message passing and collective directives as the primary mode of inter-process communication. All nodes participate in a collective reduction operation to send the results back to the host, and synchronization is achieved through barriers and blocked message passing. All low level communication backend is supported via Nvidia's *NCCL* framework.

III. Results

Figure 1 shows the results of running AQFT, Adder, and Supremacy circuits with both the single-node, multi-threaded implementation and the multi-GPU implementation. We find that running circuits with a multi-node implementation incurs the expected communication and synchronization costs compared to utilizing a single node. These costs limit the size of circuits available to be run on a specified number of GPUs and significantly impact runtime when the circuits are small. As a result, the 8-GPU implementation demonstrates an advantage only beyond a certain threshold size. We expect that there is an optimal level of parallelism for each application and circuit size, where below the optimization point is slow due to the ability to further distribute the workload and above the optimization point is inefficient due to incurred costs. One future path of research is to determine the optimal level of parallelism.

For the Adder circuit, the 8-GPU implementation shows relatively constant performance regardless of circuit size. The single-node, multi-threaded implementation has an exponential increase in runtime as the circuit size increases. As a result, the 8-GPU implementation becomes advantageous for larger Adder circuits.

The results presented here show opportunities for these techniques. Different benchmark characteristics will influence the types of performance improvement offered by these methods. For the AQFT and Supremacy circuits, we find that memory constraints limit the size of circuit that can be ran on a specified number of GPUs: in this case 8 GPUs. We expect that, if this memory were to be increased, the 8-GPU implementation would also reduce runtime after a certain threshold qubit count in the Supremacy and AQFT tests.

References

[1] A. K. Fedorov, N. Gisin, S. M. Beloussov, and A. I. Lvovsky. Quantum computing at the quantum advantage threshold: a down-to-business review, 2022.

[2] Wei Tang, Teague Tomesh, Martin Suchara, Jeffrey Larson, and Margaret Martonosi. CutQC: Using small quantum computers for large quantum circuit evaluations. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 473–486, 2021.

[3] Syed Nauyan Rashid. Getting started with pytorch distributed. *Medium*, May 2023.

[4] Matthew DeCross, Reza Haghshenas, Minzhao Liu, Enrico Rinaldi, and Johnnie Gray et al. The computational power of random quantum circuits in arbitrary geometries, 2024.

[5] MIT Technology Review. What's next for quantum computing, 2023. Accessed: 2024-07-16.

[6] Rajeev Acharya, Igor Aleiner, Richard Allen, Trond I. Andersen, Markus Ansmann, and et al. Suppressing quantum errors by scaling a surface code logical qubit, 2022.

[7] Joschka Roffe. Quantum error correction: an introductory guide. *Contemporary Physics*, 60(3):226–245, July 2019.

[8] Tianyi Peng, Aram W. Harrow, Maris Ozols, and Xiaodi Wu. Simulating large quantum circuits on a small quantum computer. *Physical Review Letters*, 125(15), October 2020.

[9] Adriano Barenco, Artur Ekert, Kalle-Antti Suominen, and Päivi Törmä. Approximate quantum fourier transform and decoherence. *Physical Review A*, 54(1):139–146, July 1996.

[10] Peter Young. The bernstein-vazirani algorithm, October 2019. Accessed: 2024-07-17.

[11] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit, 2004.

[12] Brendan Burns. *Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services*. O'Reilly Media, 2018.