Parallelizing Simulations of Large Quantum Circuits

Michael A. Perlin
JILA, NIST and University of Colorado at Boulder
Boulder, Colorado, USA
mika.perlin@gmail.com

Teague Tomesh
Princeton University
Princeton, New Jersey, USA
ttomesh@princeton.edu

Bradley Pearlman
JILA, NIST and University of Colorado at Boulder
Boulder, Colorado, USA
brpe4351@colorado.edu

Wei Tang
Princeton University
Princeton, New Jersey, USA
weit@princeton.edu

Yuri Alexeev
Argonne National Laboratory
Lemont, Illinois, USA
yuri@alcf.anl.gov

Martin Suchara
Argonne National Laboratory
Lemont, Illinois, USA
msuchara@anl.gov

ABSTRACT
We present a parallelization scheme for classical simulations of quantum circuits. Our scheme is based on a recent method to “cut” large quantum circuits into smaller sub-circuits that can be simulated independently, and whose simulation results can in turn be re-combined to infer the output of the original circuit. The exponentially smaller classical computing resources needed to simulate smaller circuits are counterbalanced by exponential overhead in terms of classical post-processing costs. We discuss how this overhead can be massively parallelized to reduce classical computing costs.

KEYWORDS
Quantum computing, quantum simulation, circuit cutting, distributed computing, parallel algorithms, HPC

1 INTRODUCTION
Quantum computers can solve certain computational tasks with exponential speedup over their classical counterparts [1]. It should therefore come as no surprise that classical simulations of quantum computations incur exponential cost: the best known techniques for evaluating general quantum circuits with \(N\) qubits have \(O(2^N)\) classical memory and runtime requirements. Nonetheless, such classical simulations play a major role in developing new quantum algorithms and understanding the behavior of quantum hardware.

Near-term quantum computing devices will be most suitable for executing variational quantum eigensolver (VQE) [10, 11] algorithms that solve approximate optimization problems, such as such as QAOA [4, 5]. The hardware-efficient ansatz (HWEA) is a particular family of VQE circuits that was designed for computability with near-term quantum hardware [6].

In this work, we implement a recently proposed method to “cut” quantum circuits that are too large to evaluate on available hardware. The process of cutting yields smaller, more tractable circuit fragments, but at the cost of exponentially large classical post-processing overhead [9]. We apply circuit cutting to classical simulations of HWEA circuits that are highly amenable to circuit cutting techniques, and discuss strategies to parallelize the associated post-processing overheads.

Figure 1: Overview of our Circuit Cutting software. Blue blocks show serial work and green blocks show highly parallel tasks.

2 METHODS
The main contribution of this work is a software tool that implements a variant of the circuit cutting algorithm first described in ref. [9]. Our tool consists four separate parts, which we refer to as the circuit generator, cutter, simulator, and uniter; see Figure 1.

Circuit Generator. In this work, we primarily focus on cutting and simulating HWEA circuits [6]. These circuits are characterized by layers of two-qubit gates sandwiched by single-qubit gates that are parameterized by angles; see Figure 1 for a sketch of a single-layer HWEA circuit. Although applications of HWEA circuits involve classical optimization over single-qubit gate parameters, in this work we are only concerned with application-agnostic features of HWEA circuits, and therefore assign random angles to these gates. The local connectivity of HWEA circuits makes them easy to partition into multiple fragments using only a few cuts, in contrast to e.g. the quantum fourier transform circuit with all-to-all qubit connectivity [8]. As a first step in our software pipeline, we generate a Qiskit QuantumCircuit object [2] representing a HWEA circuit with a chosen number of qubits and two-qubit gate layers.
The simulation of these variants is therefore trivially parallelizable (variant of Karger’s \textsc{Min-Cut} numerical techniques (level 2). In practice, we can be further parallelized using OpenMP and standard parallelized using MPI by assigning each term its own task (level 1). For individual circuit simulations, this can be done by assigning each variant a separate task. Individual variant (circuit) simulations are further natively parallelized in Qiskit’s \texttt{Aer} parallelization with MPI (level 1) and OpenMP (level 2), as sketched in Fig. 2. The sum of outer products is parallelizable with MPI, as each outer product can be computed independently before adding them all up. Furthermore, each outer product is an instance of the standard BLAS level 3 routine \texttt{DGEMM}, which is parallelized using OpenMP threads in standard numerical libraries such as Python’s \texttt{NumPy} and \texttt{SciPy}.

### 3 RESULTS & CONCLUSIONS

The runtime for naïve classical simulations of a quantum circuit scales exponentially with its number of qubits. Circuit cutting addresses some of this exponential cost by reducing the maximum number of qubits that need to be simulated in an individual quantum circuit. The exponential overhead of classical simulation is then offloaded from a circuit simulator to a fragment uniter, which must do an exponential amount of work to faithfully reproduce the outputs of the original, pre-cut circuit. By pushing exponential overhead to the uniter, circuit cutting enables data and thread level parallelization of this exponential cost through MPI and OpenMP (see Fig. 3).

In this way, our circuit cutting tool enables the evaluation of large quantum circuits by partitioning them into smaller, more manageable pieces. Efficient exploitation of the parallelism present in the uniter is central to the success of our circuit cutting tool.

### ACKNOWLEDGMENTS

We thank Jeffrey Larson, James Osborn, and Ruslan Shaydulin for technical contributions to this work. This material is based upon work supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under contract DE-AC02-06CH11357.
REFERENCES


