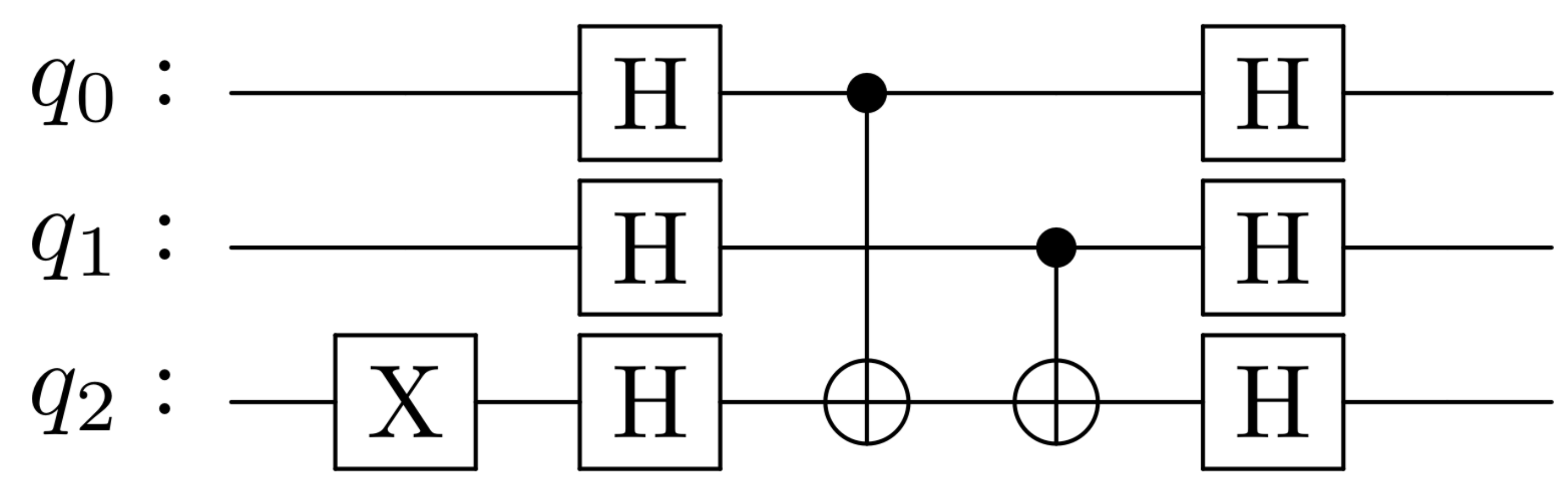
Quantum Simulation on Trainium

Quan Do, Richard Yin, Runzhao Guo, Jens Palsberg

Introduction

Because access to quantum computers is limited, researchers often do quantum simulation on classical computers. Large simulations are infeasible to perform on typical desktop computers because of exponential memory requirements and execution time. We use AWS Trainium to simulate quantum circuits, taking advantage of its tensor cores and high memory bandwidth.

A quantum circuit consists of n qubits, as well as a list of operations called gates. Gates can apply to one or more qubits, changing or entangling their states.



The circuit's state ψ is a vector of 2^n complex numbers, and each gate is a unitary matrix U. The state of the circuit evolves by multiplying each gate with the state vector (i.e. $\psi_{i+1} = U\psi_i$). In general, each gate involves multiplying a $2^n \times 2^n$ matrix with a $2^n \times 1$ vector.

Implementation

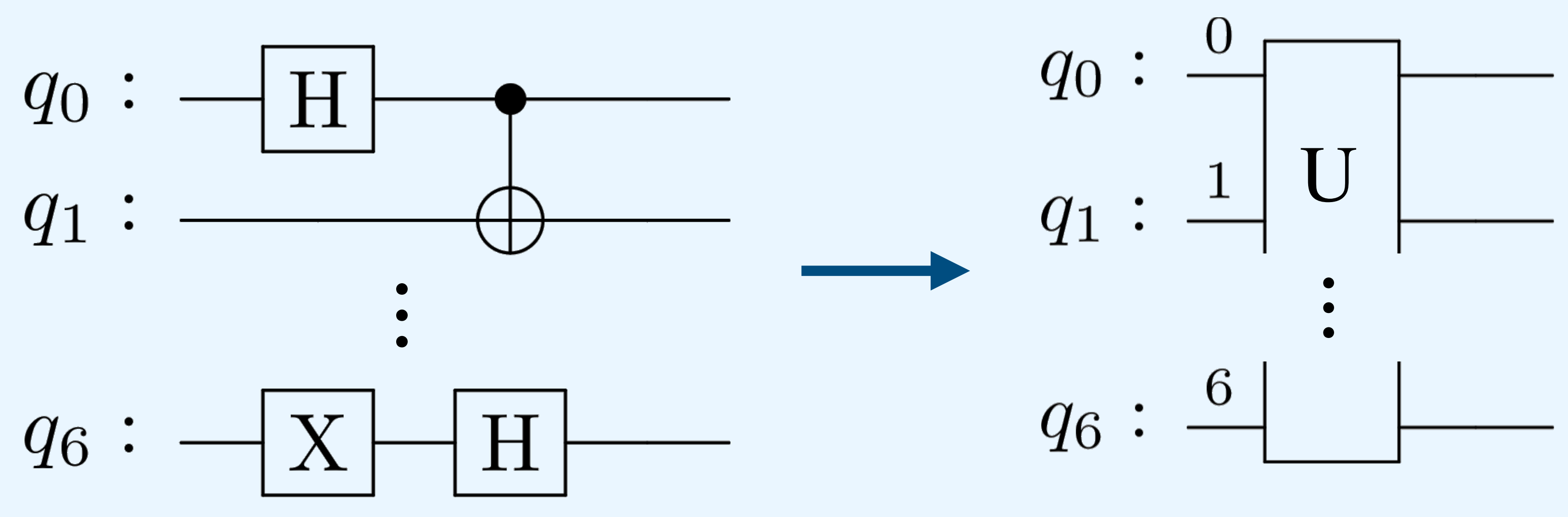
Trainium supports multiplying up to 128×128 unitaries, or seven-qubit gates. In general, applying each gate is $O(2^{2n})$, but we can get $O(2^n)$ by *shuffling the qubits around* (i.e. permuting the state vector). The resulting gate operation is a block-diagonal matrix.

$$\psi_{i+1} = \begin{pmatrix} U & 0 & 0 & 0 \\ 0 & U & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \dots & U \end{pmatrix} \psi_i$$

One method is simply moving the entire state vector from Trainium to the host, and performing the permutation on CPU. This is easy to implement, but most of the execution time is spent transferring the state vector data between devices.

Another method uses Trainium's dynamic indexed load/store to access 128 arbitrary elements of the state vector at a time from HBM into SBUF for each matrix multiplication. This requires complicated index calculation on the Trainium device, but it reduces data transfer to the host.

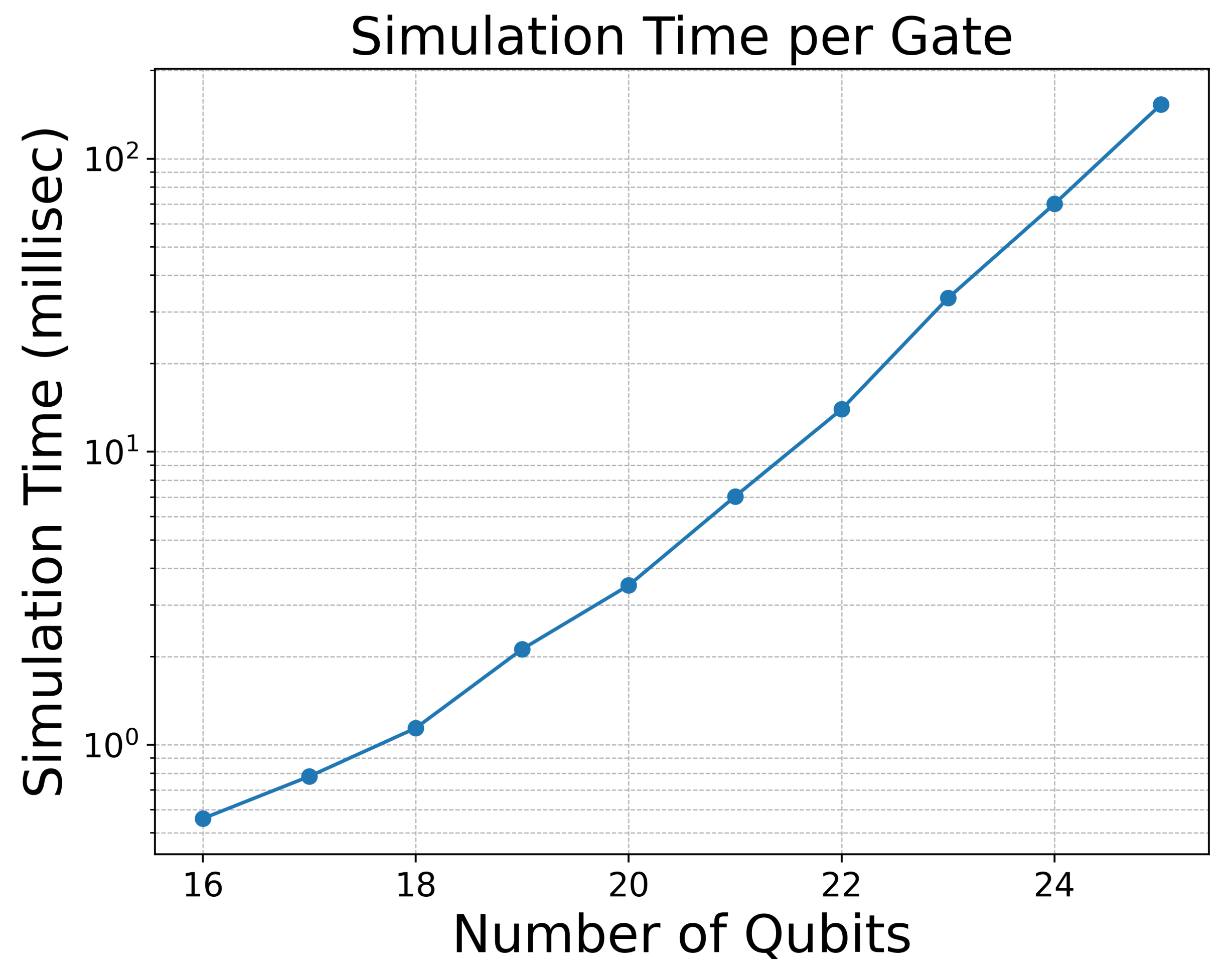
To reduce the number of matrix multiplications, we can preprocess the circuit by combining gates. For a set of seven qubits, any consecutive gates that act on only those qubits can be combined into a single seven-qubit gate. This can be easily done on CPU, as we only need to do 128×128 matrix multiplications.



To multiply complex floats, we use the 3M method (Van Zee et al. 2016), which requires three real-valued matrix multiplication operations.

$$\begin{aligned} W^r &:= A^r B^r, \quad W^i := A^i B^i \\ S_A &:= A^r + A^i, \quad S_B := B^r + B^i \\ C^r &:= C^r + W^r - W^i \\ C^i &:= C^i + S_A S_B - W^r - W^i \end{aligned}$$

Results



Future Work

One limitation of our current code is the compilation time. For large circuits, compiling the NKI code takes significantly more time than actually running the simulation. This can be somewhat mitigated with work in adjusting the kernel and the PyTorch model to make compilation simpler, as well as improvements in the compiler itself.

Currently, our quantum simulator only runs on a single Trainium 1 chip. We hope to parallelize our framework to run on multiple chips, offering up to a 16x performance increase with the larger trn1 instances.