Accelerating Quantum Computer Simulators using GPUs

Chaithanya Naik Mude, Vishnu Ramadas
{cmude, vramadas} @wisc.edu
University of Wisconsin-Madison

Introduction

Quantum computers is an ever growing area with new research changing the landscape regularly. Due to the dynamic nature of the field, and the fact that access to a quantum computer is not easy, researchers rely on quantum computer simulators (QCS). Simulators significantly reduce the barrier for researchers to prototype and investigate new algorithms. It also enables researchers to probe into deeper circuits that NISQ era computers struggle with [1]. Although QCS can run any quantum program that a real device can run, they quickly run into memory and computation time constraints [2].

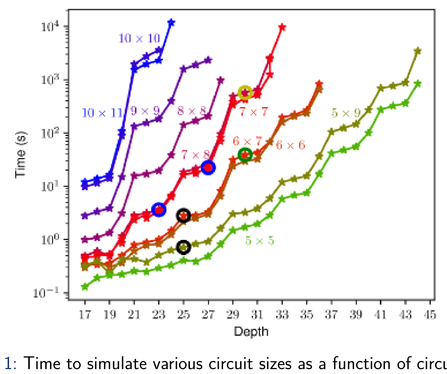


Figure 1: Time to simulate various circuit sizes as a function of circuit depth [3]

Modern CPUs offer various performance optimizations that software can make use of. For instance, state vector updates could use the vector processing unit, also called a SIMD unit, that allows an operation to work on multiple data elements at once [4]. Simulators can extract additional performance in multi-node CPUs by distributing computation across various nodes using libraries such as OpenMP or MPI [5] [6].
GPUs, on the other hand, have been a very popular option for large scale scientific computing since its invention by Nvidia. GPUs offer the ability to work with extremely large vectors/matrices by parallelly performing operations on its elements. For example, they can multiply matrices in $O(n)$ time since each element of a row of the resultant matrix can be computed independently of the others. GPUs are increasingly becoming a popular platform for QCS [7].

Quantum Simulations

Quantum simulations help explore different aspects of quantum systems without the need for complex instruments. They are important in evaluating the performance of algorithms like QAOA, VQE and also in understanding various quantum processes.
Quantum computer simulators also help in designing efficient error mitigation schemes by analyzing the designs using precise noise models through quantum simulations. They are used to tune quantum machine learning models and debug quantum algorithms. Qiskit Aer, Cirq, Qulacs, QuilSim, CuQuantum are a few popular and easily available quantum computer simulation frameworks that can be leveraged for research. Most simulators use a Schrodinger-style simulation that models the interaction between gates and state vectors and is the focus of our work.

Tensor Networks

Tensor network methods are most widely used methods for quantum simulations and . Tensors capture the idea of multi-linear maps and tensor networks are collection of tensors connected by contractions. They also utilize graphical representations to understand the workflow intuitively.
The main theme of these methods is approximate the quantum state using tensor networks. They employ lossy data compression that preserves the important properties of the quantum state. These methods enable simulation of large systems, use-cases include domains quantum chemistry, variational optimization, understanding quantum devices.

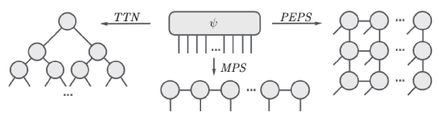


Figure 2: Tensor network methods

Some of the best known applications of tensor networks are 1D Matrix Product States (MPS), Tensor Trains (TT), Tree Tensor Networks (TTN), the Multi-scale Entanglement Renormalization Ansatz (MERA), Projected Entangled Pair States (PEPS) [8].

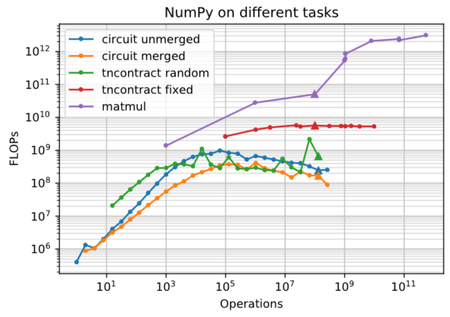


Figure 3: Flops vs operations for different tasks

Figure 3 shows the advantage of using tensor methods. Figure 4 highlights the speedup of using hybrid architectures that leverage both CPU and GPU vs those that only use either one of them. This is due to trade-off between cost of loading data into the GPUs and the parallelism offered by GPUs.

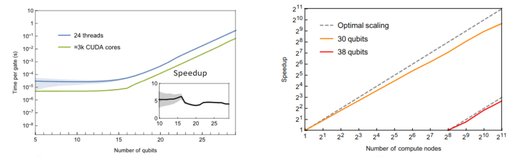
Backend Name	Device	Time (second)	Speedup
Torch_CPU	CPU	347	0.71×
NumPy (baseline)	CPU	246	1.00×
CuPy	GPU	6.7	36.7×
Torch_GPU	GPU	3.5	70.3×
Torch_CPU + Torch_GPU	Mixed	2.6	94.8×
NumPy + CuPy	Mixed	2.1	**117**×

Figure 4: Speedup comparison between Hybrid Devices vs CPU, GPU Devices

The recent work [9] employed tensor networks to analyze effects of leakage errors during quantum error correction in superconducting systems. The work [10] on hybrid tensor networks helps to simulate larger systems using relatively smaller processors.

QuEST

QuEST is first open-source quantum computer simulation toolkit that can work on multiple platforms such as a single-node CPU, multi-node CPU, or GPU [11]. It was released by The University of Oxford in 2019 and can simulate generic quantum circuits comprising of one and two-qubit and multi-qubit controlled gates quantum circuits. It can be used with pure states (using state vectors) and mixed states (using density matrices) under the presence of decoherence.
QuEST optimizes Schrodinger-style simulations by replacing certain gate operations with their effect. For instance, instead of multiplying an X gate with a state vector, QuEST simply switches the probability amplitudes of the states. It also speeds up simulation by distributing the workload across CPU cores using OpenMP, across different CPUs in a cluster using MPI, or on a GPU using CUDA.



(a) Performance on a Single Node (OpenMP vs CUDA)
(b) Normalized Performance on a Distributed System (MPI)

Figure 5: QuEST: Various parallelization approaches

QCLAB++

QCLAB++ is a new simulator developed at Lawrence Berkeley National Lab in early 2023 [1]. It is a simulator that tracks all the amplitudes of the wavefunction and, as a result, scales exponentially with an increase in the number of qubits and linearly with gate count. The scaling, however, can be made more efficient using tensor networks or sparse representation of wavefunctions at the cost of accuracy. QCLAB++ uses efficient quantum gate simulation algorithm from to calculate the effects of gates on the state vectors.

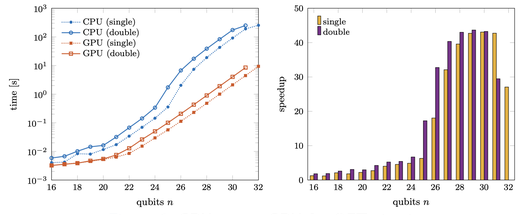


Figure 6: CPU versus GPU for QFT circuit

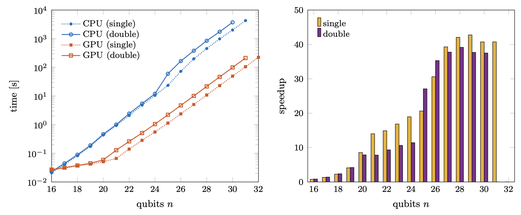


Figure 7: CPU versus GPU for Hamiltonian simulation circuit

Q-GPU

Q-GPU is a new simulator from the University of Pittsburg that aims to be a high-performance and scalable QCS that uses GPUs to accelerate compute [12]. Existing simulators such as Qiskit [13] statically assign all wavefunction amplitudes. This makes the GPU compute inefficient for large qubit system simulations since data transfer to the GPU takes up a significant time of the overall run. Q-GPU allocates state amplitudes dynamically and maximize the data transfer overlap with the gate operation compute. This results in a much better GPU utilization rate since the time spent idle waiting for data to arrive is lower than before. Q-GPU also uses dynamic zero state amplitude pruning and lossless compression of non-zero amplitudes to reduce the data transfer time. Additionally, it also performs dependency-aware quantum gate reordering to improve the chances of pruning zero state amplitudes.

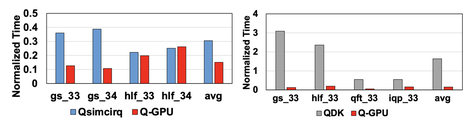


Figure 8: Comparison of Q-GPU with Google Qsim-Cirq v0.8.0 and Microsoft QDK v0.15

Conclusions and Future work

Leveraging efficient representations and parallelism through hardware can scale simulations to larger qubit systems. The hybrid approach of using smaller sized QPUs alongside to speed up certain tasks can improve performance of classical simulation which in-turn, along with efficient tensor network representations can scale for larger qubit systems.
We focus mainly on simulation speedups for our work, but various compression techniques that reduce the memory requirements are possible too. Work such as [14] and the more recent [15] propose techniques to compress the gate and state representations. The latter focuses on simulations that use tensor networks with representations that are highly parallelizable and provide both simulation speedups and reduced memory usage.

References

[1] R. V. Beeumen, D. Camps, and N. Mehta, "Qclab++: Simulating quantum circuits on gpus," 2023.
[2] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505–510, Oct 2019.
[3] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, and H. Neven, "Simulation of low-depth quantum circuits as complex undirected graphical models," 2018.
[4] C. Lomont, "Introduction to intel advanced vector extensions," 2011.
[5] OpenMP Architecture Review Board, "OpenMP application program interface version 3.0," May 2008.
[6] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard Version 4.0*, June 2021.
[7] E. Gutiérrez, S. Romero, M. A. Trenas, and E. L. Zapata, "Quantum computer simulation using the cuda programming model," *Computer Physics Communications*, vol. 181, no. 2, pp. 283–300, 2010.
[8] J. Biamonte and V. Bergholm, "Tensor networks in a nutshell," 2017.
[9] H. Manabe, Y. Suzuki, and A. S. Darmawan, "Efficient simulation of leakage errors in quantum error correcting codes using tensor network methods," 2023.
[10] X. Yuan, J. Sun, J. Liu, Q. Zhao, and Y. Zhou, "Quantum simulation with hybrid tensor networks," *Physical Review Letters*, vol. 127, July 2021.
[11] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, "Quest and high performance simulation of quantum computers," *Scientific Reports*, vol. 9, p. 10736, Jul 2019.
[12] Y. Zhao, Y. Guo, Y. Yao, A. Dumi, D. M. Mulvey, S. Upadhyay, Y. Zhang, K. D. Jordan, J. Yang, and X. Tang, "Q-gpu: A recipe of optimizations for quantum circuit simulation using gpus," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 726–740, 2022.
[13] Qiskit contributors, "Qiskit: An open-source framework for quantum computing," 2023.
[14] X.-C. Wu, S. Di, E. M. Dasgupta, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Full-state quantum circuit simulation by using data compression," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '19, (New York, NY, USA), Association for Computing Machinery, 2019.
[15] R.-Y. Sun, T. Shirakawa, and S. Yunoki, "Improved real-space parallelizable matrix-product state compression and its application to unitary quantum dynamics simulation," 2023.