A Novel Approach to Sparsity in Quantum Simulations

Srikar Chundury[1,2], In-Saeng Suh[2], and Frank Mueller[1]

[1] Department of Computer Science, North Carolina State University

[2] National Center for Computational Sciences, Oak Ridge National Laboratory

Problem

- Efficient simulation methods → crucial for quantum algorithms
  - **High cost**
  - **Error-prone**
  - **Decoherence**

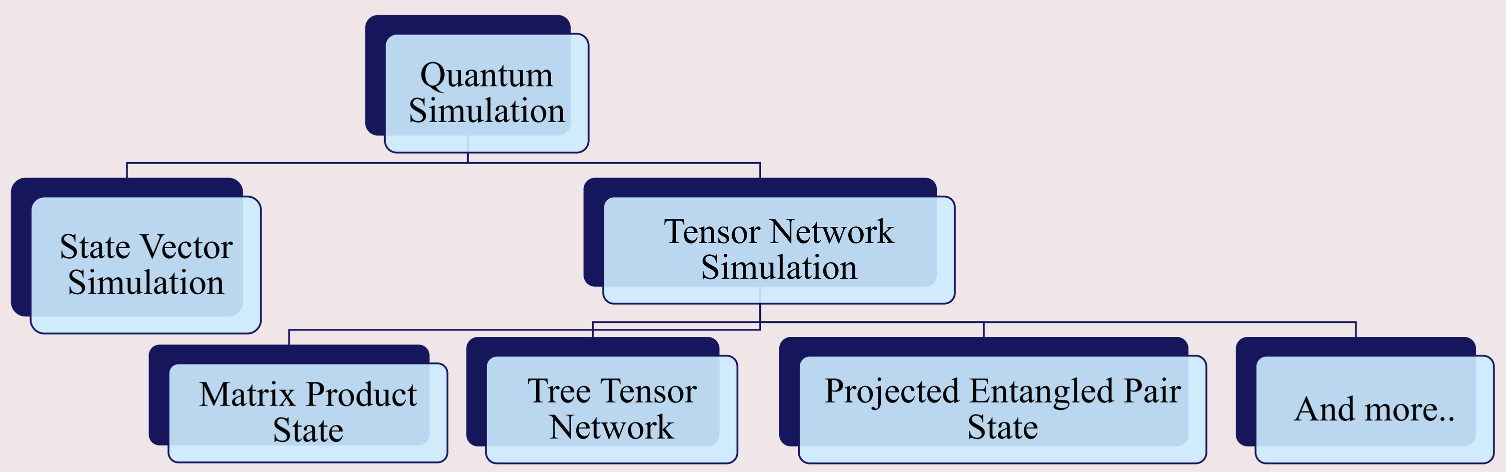

Fig 1: Quantum Simulation Methods

- **State Vector** method does not scale well
  - Memory requirements grow **exponentially**
- **Tensor Networks**
  - Break large tensors into a **network** of multiple tensors **interconnected** by bond interactions
  - This approach **delays** the need for greater memory requirements until later contraction stages
- **Conserving Memory** → crucial to enable the scalability of quantum simulations to large number of qubits

Approach

- **Explore Sparsity Patterns** → unique to quantum
  - Seen in intermediate gates, sub-circuits, or circuit unitaries

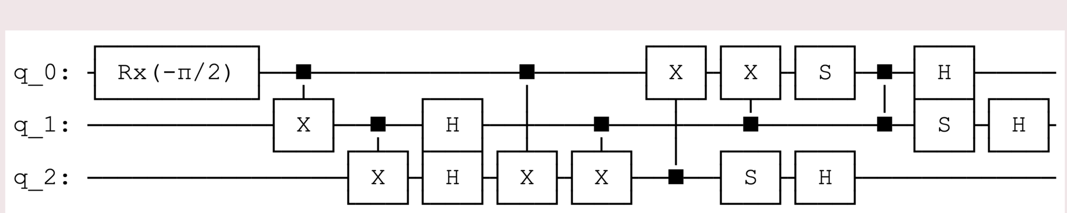

Fig 2(a): Mermin Bell Pair Generation

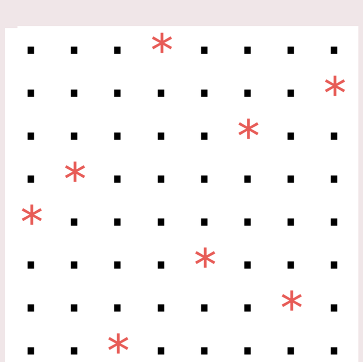

Fig 2(b): Unitary

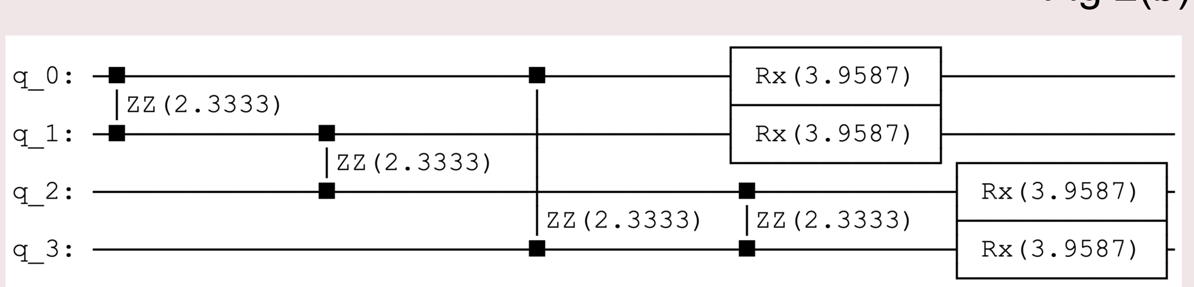

Fig 3(a): Max-Cut (square) QAOA sub-circuit

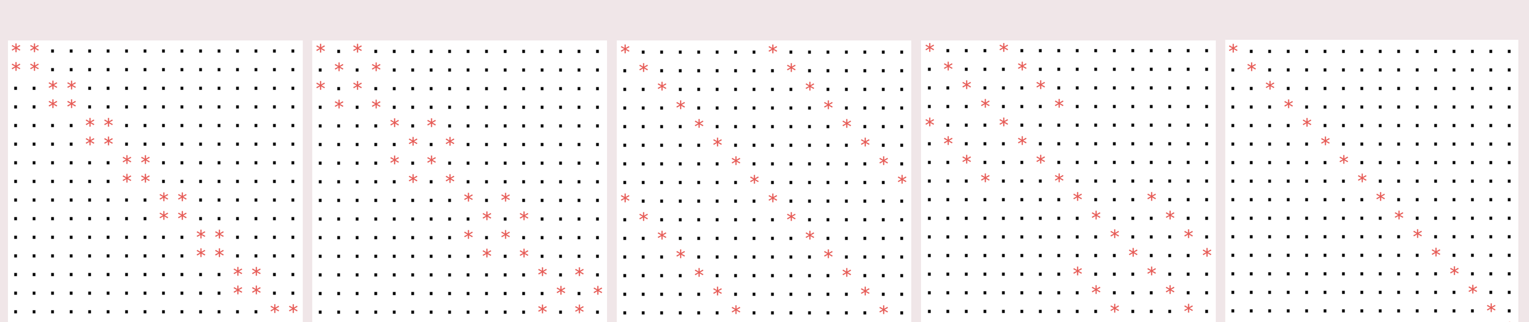

Fig 3(b): Some of the intermediate unitary matrices

Solution

- **Exploit these patterns** → new sparse data format

data[diagonal index] ⟵ diagonal elements

  - Like SciPy[4] **DIA** but for arbitrary number of diagonals
  - Offers significant memory savings

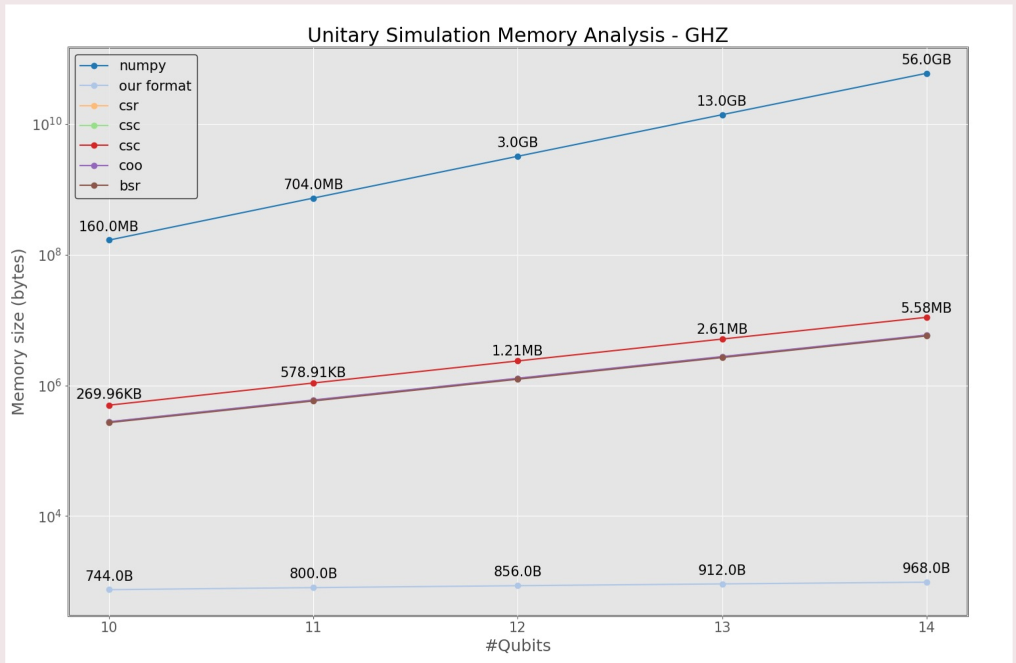

Fig 4: Memory requirements across formats for GHZ unitary simulation

  - Enables linear (or a factor of) spM-spM kernel: $O(d \times d \times n)$ where d → number of non-empty diagonals, n → matrix size
- **Extend** this sparse **matrix** format → matricized **tensor** format
  - For tensor network simulations

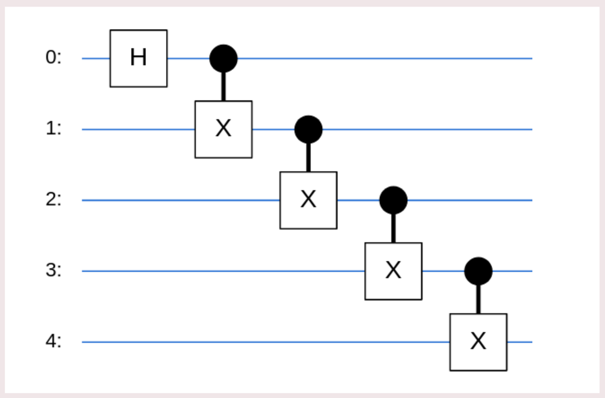

Fig 5(a): GHZ Circuit

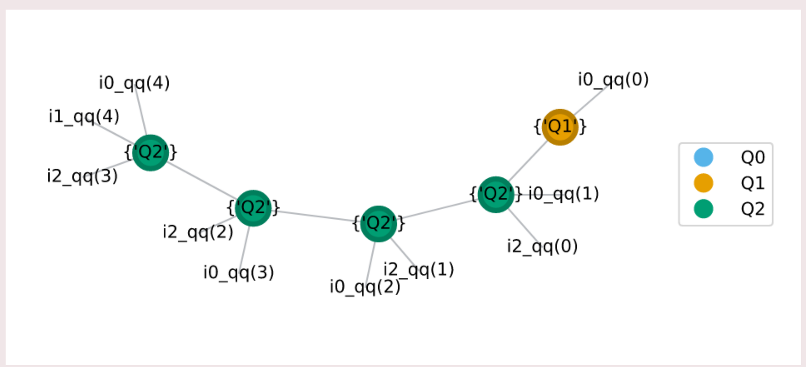

Fig 5(b): GHZ Tensor Network i_{Time Step}_qq{Qubit Number}

  - Sparsity patterns can be seen in tensors (stored as matrices) too

Experiments

- **SupermarQ[3]** benchmark suite: (GHZ, Hamiltonian, Mermin-Bell Pair)
- Empirical Analysis: Using **Qiskit[1]** unitaries at each time-step
  - Naïve chain matrix multiplication → mimick state-vector simulation
- Integrated with **Quimb[2]** for tensor-network based simulation
- All tests have been run on a single node: 192GB DDR4, 16 physical cores @ 2.50GHz, Cache: L1: 32 KB, L2 cache: 1 MB, L3 cache: 10 MB

References

1. Qiskit: Framework for Quantum Computing; doi: 10.5281/zenodo.2573505
2. Quimb: Tensor network library; doi: 10.21105/joss.00819
3. SupermarQ: Quantum benchmark suite; doi: 10.1109/HPCA53966.2022.00050
4. Scipy: Scientific computing numerical library; doi: 10.1038/s41592-019-0686-2
5. TNQVM: Tensor Network Quantum Virtual Machine; doi: 10.1145/3547334

COO →Co-Ordinate format, CSR →Compressed Sparse Row format, DIA →Diagonal

Results

- Our new format: ~10 times faster than the dense format for state vector simulation → see Fig 6

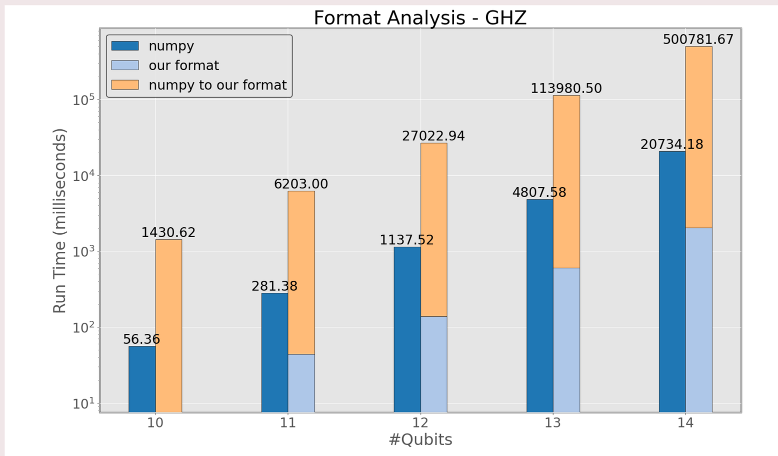

Fig 6: Dense vs our format runtime analysis

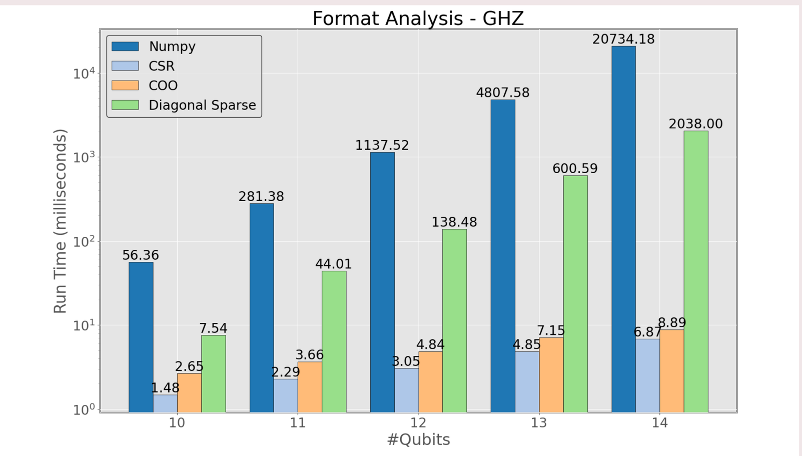

Fig 7: State vector simulation for different formats

- Other sparse formats run faster for fewer qubits → see Fig 7
  - State vector: very inefficient for >14 qubits → use Tensor Networks

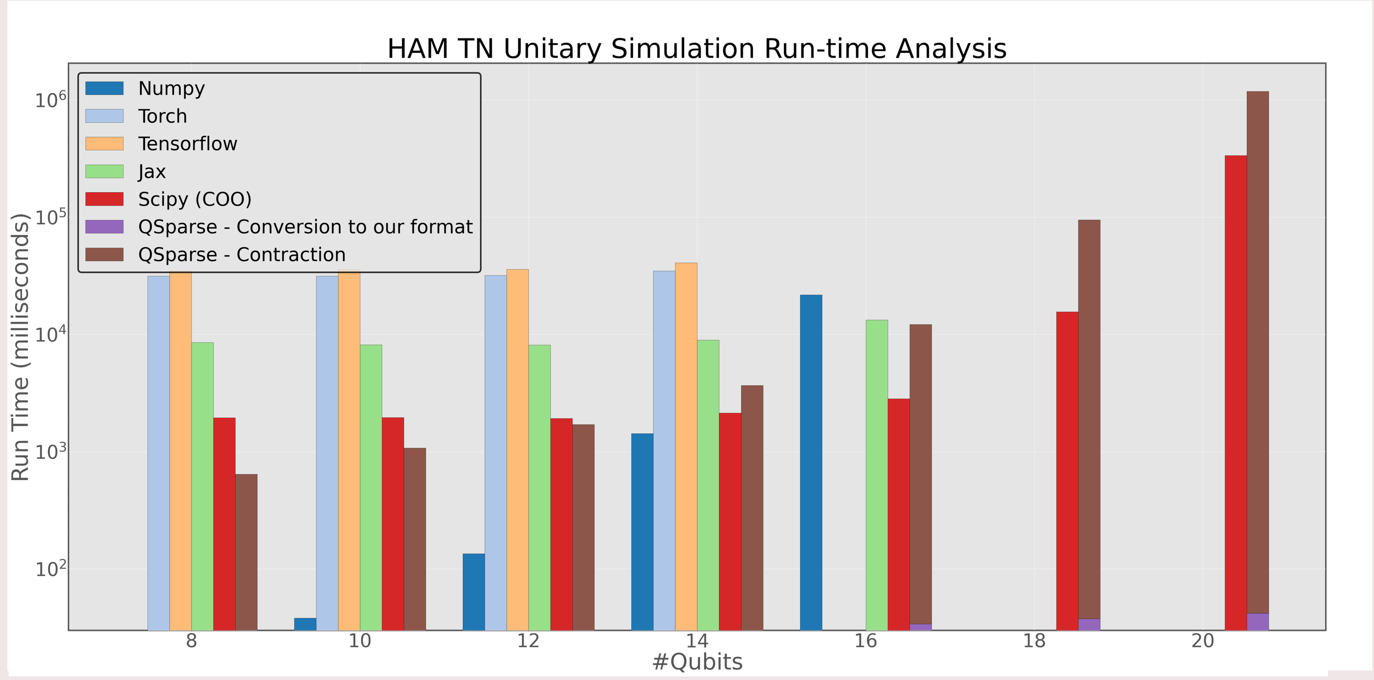

Fig 8: Tensor Network contraction comparison across numerical libraries

- But all the dense formats fail to contract the HAM TN for >=18 qubits
  - Numpy cannot go beyond 16 qubits → see Fig 8
- For our format, the last transpose operation takes a very long time
  - Omitted this step for the 20-qubit run
- Opportunity: to simulate larger number of qubits with smaller memory
- Challenges:
  - Format Conversion operation: dense to our format
  - Reshape operation: changing shape without moving data
  - Transpose operation: moving internal data around

Conclusion and Future Work

- This serves as a proof-of-concept for the advantages of our approach
  - Can simulate for larger number of qubits with the same memory
- Transpose operation is the bottleneck for tensor network contraction
  - Demands a better specialized kernel and further optimization
- Integrate with simulation libraries like TNQVM[5]
  - To explore the possibility of bypassing the format conversion step

This work was supported in part by NSF awards DMR-1747426, PHY-1818914, MPS-2120757, and CCF-2217020. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.