Poster Title

A PEPS Plugin for TNQVM

Poster Authors

Srikar Chundury | schundu3@ncsu.edu
Justin Lietz | lietzjg@ornl.gov
Eduardo Antonio Coello Pérez | coellopereea@ornl.gov
Amir Shehata | shehataa@ornl.gov
In-Saeng Suh | suhi@ornl.gov
Frank Mueller | fmuelle@ncsu.edu

Poster Abstract

This work introduces an extension to the Tensor Network Quantum Virtual Machine (TNQVM) tool, enhancing the existing stack of ExaScale Tensor Network (ExaTN), ExaScale Accelerator (XACC), and TNQVM. It features a new plugin that enables efficient simulation of a Projected Entangled Pair State (PEPS), a 2D tensor network. To improve simulation efficiency for PEPS, we have implemented the snake boundary contraction algorithm. By integrating this capability into the existing stack, we enhance the overall functionality and versatility of the framework. We tested this new PEPS topology for a simple GHZ bell-pair generation quantum circuit and saw that its runtime is very close to that of the MPS topology. We estimate that the real potential of the PEPS topology becomes discernible when quantum circuits with multidimensional entanglement are simulated using tensor networks. In such cases, 1D tensor networks fail to represent or contract them efficiently.

Poster Relevance

- Quantum Software Engineering
- Quantum Computing

A PEPS Plugin for TNQVM

Srikar Chundury[1,2], Justin Lietz[2], Eduardo Antonio Coello Pérez[2], Amir Shehata[2], In-Saeng Suh[2], Frank Mueller[1]

[1]*Department of Computer Science*
North Carolina State University
Raleigh, NC 27695, USA
{schundu3,fmuelle}@ncsu.edu

[2] *National Center for Computational Sciences*
Oak Ridge National Laboratory
Oak Ridge, TN 38830, USA
{lietzjg,coellopereea,shehataa,suhi}@ornl.gov

***Abstract*—This work introduces an extension to the Tensor Network Quantum Virtual Machine (TNQVM) tool, enhancing the existing stack of ExaScale Tensor Network (ExaTN), ExaScale Accelerator (XACC), and TNQVM. It features a new plugin that enables efficient simulation of a Projected Entangled Pair State (PEPS), a 2D tensor network. To improve simulation efficiency for PEPS, we have implemented the snake boundary contraction algorithm. By integrating this capability into the existing stack, we enhance the overall functionality and versatility of the framework. We tested this new PEPS topology for a simple GHZ bell-pair generation quantum circuit and saw that its runtime is very close to that of the MPS topology. We estimate that the real potential of the PEPS topology becomes discernible when quantum circuits with multidimensional entanglement are simulated using tensor networks. In such cases, 1D tensor networks fail to represent or contract them efficiently.**

***Index Terms*—Quantum Computing, Quantum Software Engineering, Tensor network, Quantum Circuit Simulation, Projected Entangled Pair State.**

A. Introduction: Efficient quantum simulation is crucial to the design and development of quantum algorithms since quantum hardware today has multiple disadvantages, including decoherence, costly initial state preparation, gate errors (single and two-qubit). Statevector simulations of quantum circuits are known to have memory bottlenecks because of the exponential growth of the Hilbert space. Tensor networks alleviate this problem by representing and storing very large tensors as factorizations, e.g., MPS, TTN, MERA, and PEPS, among others.

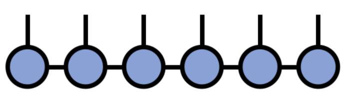

Fig. 1. Matrix Product State / Tensor Train [1]

Matrix Product State (MPS) tensor networks, schematically represented in Fig. 1, have gained significant prominence in quantum physics applications due to their efficient evolution in both real and imaginary time. This characteristic proves invaluable for investigating quantum dynamics, thermalization phenomena, and directly simulating systems at finite temperatures.

In Natural Language Processing (NLP) applications, Tree Tensor Networks (TTN) as depicted in Fig. 2, have been used in the context of DisCoCat models, which are used in sentiment analysis [2]. DisCoCat models combine distributional semantics and compositional distributional semantics, and TTNs have been employed to represent and manipulate the tensor structures that arise from these models.

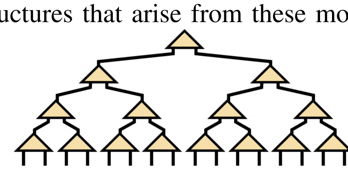

Fig. 2. Tree Tensor Network [1]

Projected Entangled Pair State (PEPS) tensor networks (Fig. 3) have mainly been utilized as a framework for approximating quantum wave functions, specifically focusing on ground states associated with two-dimensional Hamiltonians. Additionally, PEPS neatly fits into 2D lattice-like quantum qubit hardware structures.

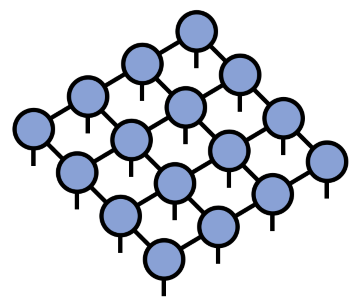

Fig. 3. Projected Entangled Pair State [1]

Tensor networks can only delay the need for higher dimensional tensors up to later contraction stages. If PEPS contains n tensors, its contraction still takes O(n^7) time when done exactly. But when coupled with approximate contraction techniques, we can lower the memory and compute costs significantly. Since these techniques then become approximate, they suit variational quantum algorithms well.

ExaTN [3] is a tensor network library that provides tensor methods at scale, leveraging another numerical library called TAL-SH. It entails logic that builds various topologies of tensor network via the builder component. TNQVM [3] is a plugin to XACC [4] that simulates a quantum circuit at scale. It provides a visitor interface for which multiple topologies can be implemented, each of which convert a given quantum circuit to its tensor network equivalent.

In this work, we integrate the PEPS tensor network topology with TNQVM and implement the snake boundary contraction algorithm leveraging pre-implemented topologies like MPS and MPO.

B. Approach: Quantum Circuit simulation broadly involves 2 steps: 1) circuit to tensor network conversion, and 2) manipulating/contracting this tensor network for various observables and more. We reuse the truncated SVD functionality that already exists within the TNQVM library.

Algorithm 1 Conversion of a Quantum Circuit to PEPS

```
pepsNetwork ← exatn::builder("PEPS").initialize(X,Y);
for gate_i in circuit.gates() do
    gateTensor ← tnqvm::getGateTensor(gate_i);
    if gate_i.rank() == 2 then
        mergedTensor ← pepsNetwork.getQubitTensor(i) ⊗ gateTensor;
        pepsNetwork.setQubitTensor(a, mergedTensor);
    end if
    if gate_i.rank() == 4 then
        mergedTensor ← pepsNetwork.getQubitTensor(a) ⊗ pepsNetwork.getQubitTensor(b);
        contractedTensor ← mergedTensor ⊗ gateTensor;
        U, V ← TruncatedSVD(contractedTensor);
        pepsNetwork.setQubitTensor(a, U);
        pepsNetwork.setQubitTensor(b, V);
    end if
end for
```

We first convert a given quantum circuit to a PEPS tensor network as described in Algorithm 1. TNQVM already provides implementations of multiple "visitors", e.g., MPS, MPO, and TTN. We leverage the MPS tensor network to implement the snake (zig-zag) boundary contraction algorithm inspired by [5]. Our contraction algorithm first forms a boundary sequence starting from top-left corner and follows a zig-zag path to cover all the nodes in the PEPS network. We then construct an MPS tensor network using this node traversal sequence. TNQVM's existing MPS-visitor is used for observables or density matrices.

C. Tests: We tested PEPS tensor network contraction using the algorithm described above for the Greenberger-Horne-Zeilinger (GHZ) state generation circuit. PEPS can be used with TNQVM just like other visitors, but with extra parameters (row and column sizes) for the 2D tensor network initialization. We make an attempt to choose them automatically if these parameters are not provided by the user. An example of GHZ(9) with TNQVM is as follows:

```
xacc::Initialize();
auto xasmCompiler = xacc::getCompiler("xasm");
auto ir = xasmCompiler->compile(R"(__qpu__
void test2(qbit q) {
   H(q[0]);
   for(int i = 0; i < 8; i++) {
      CNOT(q[i], q[i+1]);
   }
   for (int i = 0; i < 9; i++) {
      Measure(q[i]);
   }}
)");
std::vector<int> bitstring(9, 0);
auto program = ir->getComposite("test2");
auto accelerator = xacc::getAccelerator(
"tnqvm", {
   std::make_pair(
      "tnqvm-visitor", "exatn-peps"
   ),
   std::make_pair("shots", 10),
   std::make_pair("lx", 3),
   std::make_pair("ly", 3)
});
auto qreg = xacc::qalloc(9);
accelerator->execute(qreg, program);
qreg->print();
```

D. Preliminary Results: Fig. 4 illustrates that PEPS has competitive runtime to MPS by testing the PEPS plugin for the above described GHZ quantum circuit. These sample tests are run on an Intel(R) Core(TM) i7-4820K at 3.70GHz with 8 cores, L1 data and instruction caches of 128 KiB each, an L2 cache of 1 MiB, an L3 cache of 10 MiB and 8GB of DRAM.

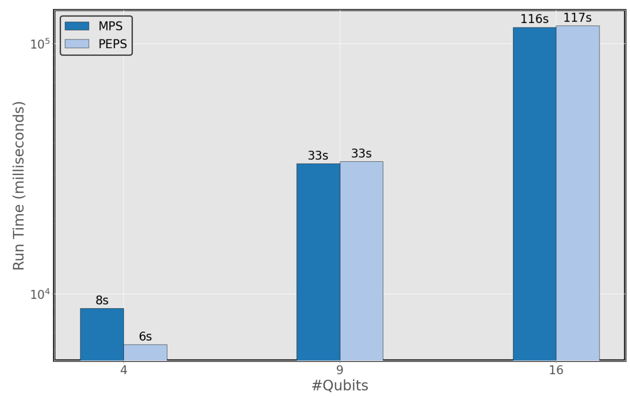

Fig. 4. GHZ Tensor Network Simulation with TNQVM

E. Conclusion / Future Work: We developed a new "visitor" for TNQVM that enables PEPS simulation capturing 2D entanglement in Hamiltonian problems, among others. We leveraged pre-existing tensor network topologies, such as MPS and MPO, to facilitate the contraction of PEPS. It is experimental and requires extensive tests and benchmarking at scale before which comparisons can be drawn against other quantum simulation libraries. Other approximate PEPS contraction can should be explored, leading to efficient simulation of more complicated quantum circuits.

Acknowledgment: 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.

References

[1] https://tensornetwork.org, accessed: July 20, 2023.
[2] V. Martinez and G. Leroy-Meline, "A multiclass qnlp sentiment analysis experiment using discocat," 2022.
[3] T. Nguyen, D. Lyakh, E. Dumitrescu, D. Clark, J. Larkin, and A. McCaskey, "Tensor network quantum virtual machine for simulating quantum circuits at exascale," 2021.
[4] A. J. McCaskey, D. I. Lyakh, E. F. Dumitrescu, S. S. Powers, and T. S. Humble, "Xacc: A system-level software infrastructure for heterogeneous quantum-classical computing," 2019.
[5] C. Guo, Y. Liu, M. Xiong *et al.*, "General-purpose quantum circuit simulator with projected entangled-pair states and the quantum supremacy frontier," *Physical Review Letters*, vol. 123, no. 19, p. 190501, 2019.

A PEPS plugin for TNQVM

S. Chundury[1,2], J. Lietz[1], E. A. C. Pérez[2], A. Shehata[1], I.-S. Suh[1], and F. Mueller[2]
[1] National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37830
[2] Department of Computer Science, North Carolina State University, Raleigh, NC 27606

Summary

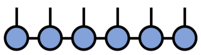

Fig (1) MPS [3]

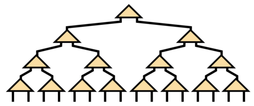

Fig (2) TTN [3]

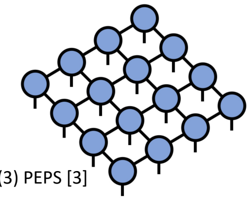

Fig (3) PEPS [3]

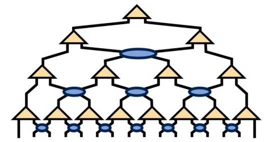

Fig (4) MERA [3]

Here we integrate the Projected Entangled Pair State (PEPS) tensor network topology with Tensor Network Quantum Virtual Machine (TNQVM) and implement the snake boundary contraction algorithm leveraging pre-implemented topologies like MPS and MPO

2 Step Process
- ➔ **Represent a Quantum Circuit as a PEPS tensor network**
- ➔ **Contract this tensor network using approximate algorithms for observables**

Motivation

2D Tensor Network (PEPS)
- **Efficiently represent quantum many-body systems in two or higher dimensions**
- **2D Hamiltonians**

Background

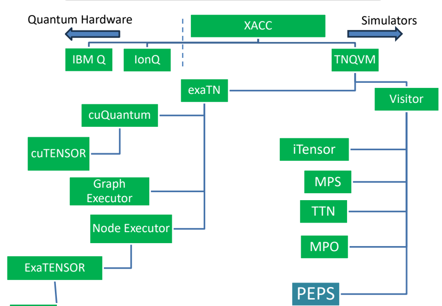

Fig (5) HPC Quantum Software Stack

Approach

- **Snake Boundary Algorithm**
  - **Zig-Zag Pattern** [2]

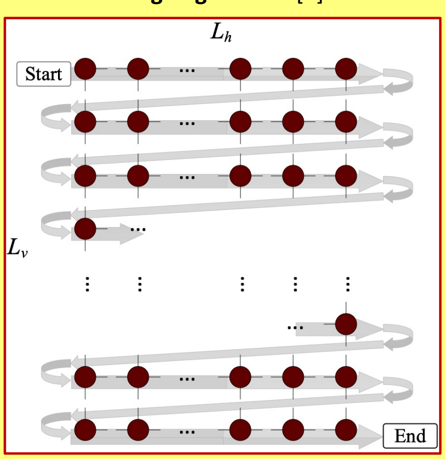

Fig (6) Snake PEPS contraction

- **Re-use existing visitor features**
- **Contraction sequence from above step → MPS Visitor**
- **Build a new MPS Tensor Network (ExaTN)**
- **Contract this MPS using pre-implemented methods**

Quantum Circuit to TN

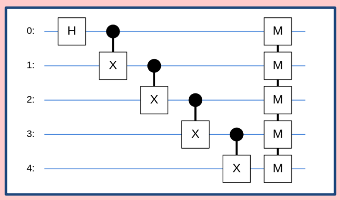

Fig (7) GHZ Quantum Circuit

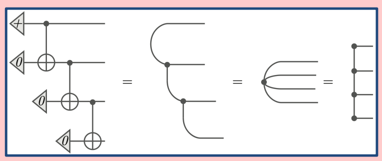

Fig (8) GHZ Quantum Circuit to MPS [4]

- **Conversion to PEPS can be tricky to visualize!**

Algorithm 1 Conversion of Quantum Circuit to PEPS

```
pepsNetwork ← exatn::builder(”PEPS”).initialize(X,Y);
for gate_i in circuit.gates() do
    gateTensor ← tnqvm::getGateTensor(gate_i);
    if gate_i.rank() == 2 then
        mergedTensor ← pepsNetwork.getQubitTensor(i) ⊗ gateTensor;
        pepsNetwork.setQubitTensor(a, mergedTensor);
    end if
    if gate_i.rank() == 4 then
        mergedTensor ← pepsNetwork.getQubitTensor(a) ⊗ pepsNetwork.getQubitTensor(b);
        contractedTensor ← mergedTensor ⊗ gateTensor;
        U, V ← TruncatedSVD(contractedTensor);
        pepsNetwork.setQubitTensor(a, U);
        pepsNetwork.setQubitTensor(b, V);
    end if
end for
```

- **All methods used in the above algorithm are pre-implemented in TNQVM already**

Plugin Usage (GHZ-9)

```
xacc::Initialize();
auto xasmCompiler = xacc::getCompiler("xasm");
auto ir = xasmCompiler->compile(R"(__qpu__
void test2(qbit q) {
   H(q[0]);
   for(int i = 0; i < 8; i++) {
      CNOT(q[i], q[i+1]);
   }
   for (int i = 0; i < 9; i++) {
      Measure(q[i]);
   }
}
)");
std::vector<int> bitstring(9, 0);
auto program = ir->getComposite("test2");
auto accelerator = xacc::getAccelerator(
"tnqvm", {
   std::make_pair(
      "tnqvm-visitor", "exatn-peps"
   ),
   std::make_pair("shots", 10),
   std::make_pair("lx", 3),
   std::make_pair("ly", 3)
});
auto qreg = xacc::qalloc(9);
accelerator->execute(qreg, program);
qreg->print();
```

GHZ Circuit

PEPS Visitor Meta

Execute

Preliminary Results

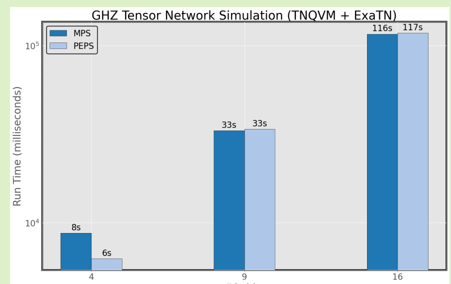

Fig (9) MPS vs PEPS for GHZ

- **Runtime(PEPS) ≈ Runtime(MPS)**
- **Test of correctness using the PEPS plugin for the GHZ quantum circuit**

Conclusion

- **This new "visitor", PEPS, enables 2D tensor network simulation capturing multidimensional entanglement**
- **Merit: Utilizes pre-developed tensor network topologies, such as MPS and MPO → contraction of PEPS**

Future Work

- **Extensive tests and benchmarking at scale, for problems asking for 2D entanglement representation**
- **Explore other PEPS approximate contraction algorithms**
- **Explore sparsity induced in PEPS Tensor Networks**

References

[1] McCaskey, Alexander J. Tensor Network Quantum Virtual Machine (TNQVM). Computer software. https://www.osti.gov//servlets/purl/1340180. Vers. 00. USDOE. 18 Nov. 2016. Web.
[2] Guo C, Liu Y, Xiong M, et al. General-Purpose Quantum Circuit Simulator with Projected Entangled-Pair States and the Quantum Supremacy Frontier. Phys Rev Lett. 2019;123(19):190501. doi:10.1103/PhysRevLett.123.190501
[3] https://tensornetwork.org
[4] Jacob Biamonte and Ville Bergholm. "Tensor Networks in a Nutshell." 2017. arXiv:1708.00006 [quant-ph].

This work was supported in part by NSF grants 2217020, 2120757 and 1818914. 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.