RosneT: A Block Tensor Algebra Library for Out-of-Core Quantum Computing Simulation

Sergio Sanchez-Ramirez* , Javier Conejero† , Artur Garcia-Saez*‡
*QUANTIC, Barcelona Supercomputing Center, Barcelona, Spain
†Workflows and Distributed Computing, Barcelona Supercomputing Center, Barcelona, Spain
‡Qilimanjaro, Barcelona, Spain
E-mail: {sergio.sanchez.ramirez, javier.conejero, artur.garcia}@bsc.es

***Keywords*—tensor network, quantum computing, simulation, out-of-core, task-based programming, COMPSs, distributed computing, HPC.**

I. Introduction

***Abstract*—With recent Quantum Devices showing increasing capabilities to perform controlled operations, further development on Quantum Algorithms may benefit from Quantum Simulations on classical hardware. Among important applications one finds verification and debugging of Quantum Algorithms, and elucidating the frontier for real Quantum Advantage of new devices [1]. Tensor Networks are regarded as an efficient numerical representation of a Quantum Circuit, but exponential growth forces tensors to be distributed among computing nodes. A number of methods and libraries have appeared recently to implement Quantum Simulators with Tensor Networks [2], [3] intended for HPC clusters. In this work we develop a Python library called `RosneT` using a task-based programming model able to extend all tensor operations into distributed systems, and prepared for existing and upcoming Exascale supercomputers. It is compatible with the Python ecosystem, and offers a simple programming interface for developers.**

Tensors as Quantum States. The Quantum State $\Psi_{\mathbf{i}}$ describing a Quantum computation is represented by a n-order unit tensor on $\bigotimes_{i=1}^{n}\mathbb{C}^2$ where n is the number of qubits and $\mathbf{i} = i_1 \dots i_n$ are the indices of the qubits. This tensor can be further decomposed in a Tensor Network of tensors A_k. Using a general tensor contraction operation Φ we define the state as $|\Psi_{\mathbf{i}}\rangle = \sum_{i_1 \dots i_n} \Phi(A_k)|i_1 \dots i_n\rangle$.

Quantum Gates acting over Quantum States are also represented as tensors of the network, such that any physical observable is obtained solely from operations over the tensors. When contracting the tensor network, intermediate tensors will be created whose order is greater. Generally, as the depth of the networks increases, so does the order of the greatest intermediate tensor. This imposes limits on the size of the tensor networks that we can contract, as the size of tensor grows exponentially fast with its order.

A common method for handling large tensors is tensor cutting or slicing [4], [5], [6]. Slicing is based on the idea of Feynman's Path Integral formulation of quantum mechanics, that all the paths in a tensor network equally contribute to the final quantum state. An index (or edge) in the tensor network represents a vector space of dimensionality D between two tensors. If we select one and project it into one of its basis, then we are filtering $\frac{1}{D}$ of the paths and the intermediate tensors that contain the index will shrink to occupy a $\frac{1}{D}$ of its original space at the cost of $\frac{1}{D}$ fidelity in the final result. Furthermore,

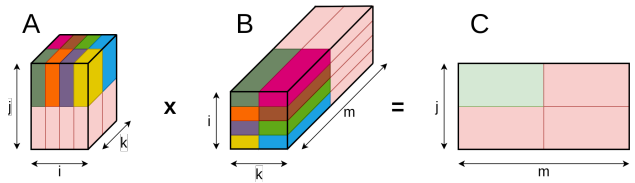

Fig. 1. Example of a block tensor contraction between two 3-order tensors into a 2-order tensor. To compute a block of C (light green), the needed pairwise A, B blocks are colored in a range of colors by matching couples.

if we make copies of the tensor network, each projected to a different basis of the selected index, and repeat this process, we end up with a massively parallel problem and can calculate up to the desired fidelity. However, contraction of non-sliced tensors become redundant and may add a significant overhead to computation. We notice that by viewing tensor projections as block divisions of the original tensors, this compute overhead disappears and becomes an indicator of block reuse while still can fit in a distributed system.

Block Tensor Contraction. Based on the Dimension Extents layout [2], [3], we slice tensors into uniform grids of sub-tensors or blocks. Employing the example in Figure 1, block tensor contraction between tensors A_{ijk} and B_{ikm} is performed following these steps:

1) For each output block in tensor C, we select the blocks from A and B needed to compute the resulting block. These blocks are chosen by matching coordinates from non-contracting indexes j, m.
2) Once the input blocks of A and B have been selected, we group them in pairs by matching the coordinates of contracting indexes i, k.
3) Each of the pairwise coupled blocks, when contracted locally, will compute a partial contribution to the output block.
4) Partial result blocks are summed into the block of C.

II. Implementation

COMP Superscalar (COMPSs) [7] is a task-based programming model for distributed computing. It is supported by a runtime that manages several aspects of the applications' execution, such as task dependency analysis, data synchronization and resource management. In the COMPSs programming model, the developer has to explicitly mark a function as task. On call, COMPSs will execute it in a free distributed

worker selected by the scheduler. The type and direction of the task dependencies can be annotated. COMPSs has bindings for C/C++, Java and Python languages [8]. We opted for Python due to its popularity on the scientific community having already a rich ecosystem above which we can leverage our work.

Alternative distributed runtimes available in Python are Daskand Ray. The main features that differentiate COMPSs are:

- Data is located out-of-core. Our benefits here are two-fold: (1) Fault-tolerance is straightforward and ensured for each task and (2) with a higher secondary storage, the size of the quantum states we can represent grows.
- Detailed tracing and profiling support with Extrae and Paraver.
- Dynamic Heterogeneous Computing. The user may provide several implementations for the same routine and COMPSs will select the implementation on runtime based on the available resources.

III. Evaluation

Block size has a non-trivial effect on the execution time of simulations. On Figure 2, the execution time of a Random Quantum Circuit simulation for different maximum block sizes is shown. Note that if the maximum block size is too large, the grain of parallelism is too coarse and it does not fully exploit the resources. Moreover, if the maximum block size is too thin, the (de)serialization overhead surpasses the effective execution time.

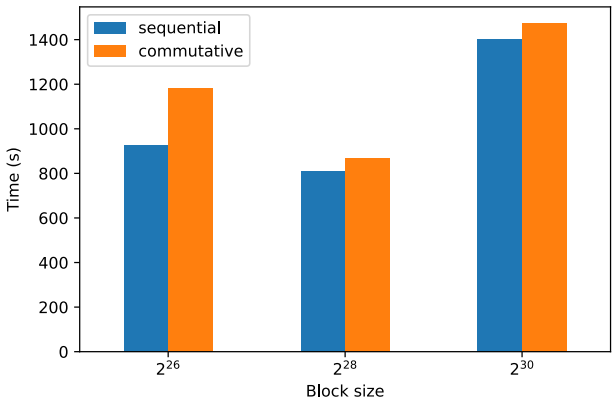

Fig. 2. Execution times for the contraction of a Random Quantum Circuit with 53 qubits and depth 12, varying the maximum block size.

IV. Conclusions

Using COMPSs runtime our library features out-of-core data location, tracing and profiling support, and potential for heterogeneous computing. As an example thanks to *numpy*'s dispatching mechanism, we can leverage over *numpy* derivatives like `cupy` adding GPU acceleration to `RosneT`. Our library is a novel contribution bringing flexible tools to perform large-scale simulations of Quantum systems to a community of Quantum developers.

V. Acknowledgment

This work bas been published in proceedings of the Second International Workshop on Quantum Computing Software (QCS), 2021 [9]. We acknowledge support from project QuantumCAT (ref. 001- P-001644), co-funded by the Generalitat de Catalunya and the European Union Regional Development Fund within the ERDF Operational Program of Catalunya, and European Union's Horizon 2020 research and innovation programme under grant agreement No 951911 (AI4Media). This work has also been partially supported by the Spanish Government (PID2019-107255GB) and by Generalitat de Catalunya (contract 2014-SGR-1051). This work is co-funded by the European Regional Development Fund under the framework of the ERFD Operative Programme for Catalunya 2014-2020, with 1.527.637,88C. Anna Queralt is a Serra Húnter Fellow.

References

[1] B. Villalonga *et al.*, "Establishing the quantum supremacy frontier with a 281 pflop/s simulation," *Quantum Science and Technology*, vol. 5, no. 3, p. 034003, apr 2020. [Online]. Available: https://doi.org/10.1088/2058-9565/ab7eeb

[2] E. Solomonik *et al.*, "Cyclops Tensor Framework: reducing communication and eliminating load imbalance in massively parallel contractions," p. 13.

[3] D. I. Lyakh, "Domain-specific virtual processors as a portable programming and execution model for parallel computational workloads on modern heterogeneous high-performance computing architectures," *International Journal of Quantum Chemistry*, vol. 119, no. 12, p. e25926, Jun. 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/qua.25926

[4] B. Villalonga *et al.*, "A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware," *npj Quantum Information*, vol. 5, no. 1, p. 86, Dec. 2019, arXiv: 1811.09599. [Online]. Available: http://arxiv.org/abs/1811.09599

[5] I. L. Markov *et al.*, "Quantum Supremacy Is Both Closer and Farther than It Appears," *arXiv:1807.10749 [quant-ph]*, Sep. 2018, arXiv: 1807.10749. [Online]. Available: http://arxiv.org/abs/1807.10749

[6] J. Chen *et al.*, "Classical Simulation of Intermediate-Size Quantum Circuits," *arXiv:1805.01450 [quant-ph]*, May 2018, arXiv: 1805.01450. [Online]. Available: http://arxiv.org/abs/1805.01450

[7] R. M. Badia *et al.*, "COMP Superscalar, an interoperable programming framework," *SoftwareX*, vol. 3-4, pp. 32–36, Dec. 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2352711015000151

[8] E. Tejedor *et al.*, "PyCOMPSs: Parallel computational workflows in Python," *The International Journal of High Performance Computing Applications*, vol. 31, no. 1, pp. 66–82, Jan. 2017. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1094342015594678

[9] S. Sánchez-Ramírez *et al.*, "Rosnet: A block tensor algebra library for out-of-core quantum computing simulation," in *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*, 2021, pp. 1–8.

Sergio Sánchez-Ramírez received his BSc degree in Sound and Image Engineering from Universidad Politécnica de Madrid (UPM), Spain in 2019. The following year, he worked as a Researcher at the CITSEM-UPM research center on the NEMESIS-3D-CM project. He completed his MSc degree in Innovation and Research in Informatics, specialization in High-Performance Computing in 2021. Currently, he is a first year PhD candidate at QUANTIC - BSC, working in large-scale quantum computing simulation using tensor networks.