NC STATE UNIVERSITY

Compressing Quantum Circuit Simulation Tensor Data

Argonne NATIONAL LABORATORY

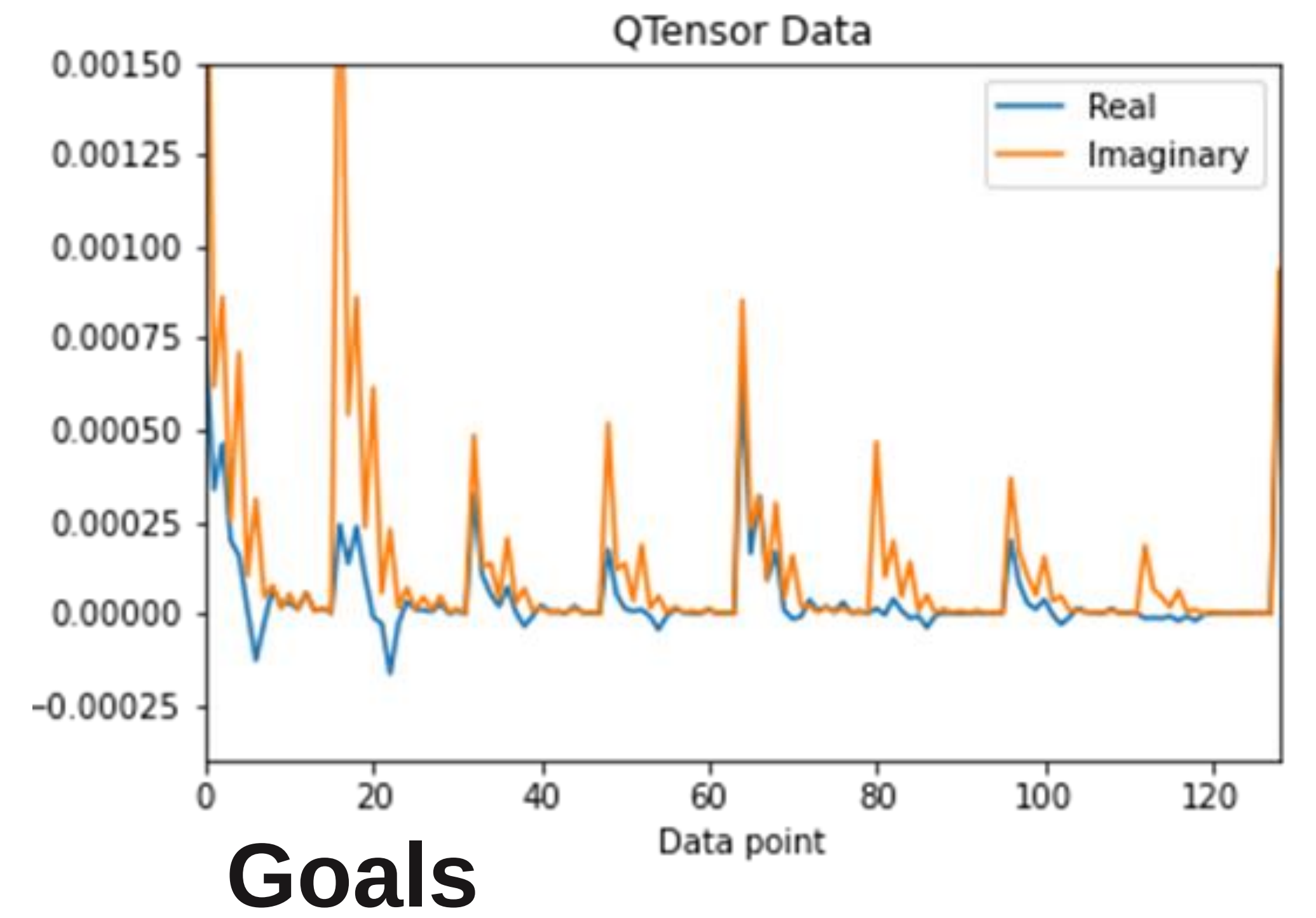
Department of Electrical and Computer Engineering

Milan Shah, Xiaodong Yu (Advisor), Sheng Di (Advisor), Franck Cappello (Advisor) and Michela Becchi (Advisor)

Introduction

Quantum circuit simulation can be carried out as a contraction over many quantum tensors. QTensor [1], a library built for quantum circuit simulation using a bucket elimination algorithm, contracts tensors to return a final energy value. Tensors represent quantum circuit gates that operate on quantum states, thus they are composed of floating-point complex numbers. A tensor with dimension d has 2^d data points. As bucket elimination advances, tensors can grow large, and memory becomes a bottleneck. To address memory limitations of circuit simulation while enabling more complex circuits to be simulated, we focus on implementing a lossy compressor that can compress the data stored in quantum circuit tensors while simultaneously preserving a final energy value within an error bound after decompression.

We study the effects of various lossy compression/decompression strategies on tensor data compressibility, throughput, and result error to ensure compression/decompression can be effective, fast, and does not heavily distort data. We target GPU as a compression platform due to its massively parallel architecture and optimize for GPU efficiency using CUDA 11.



QTensor Data

- Floating-point
- Periodic when laid out in 1-D
- Many values close to zero
- Complex numbers, thus split into real and imaginary components for storage

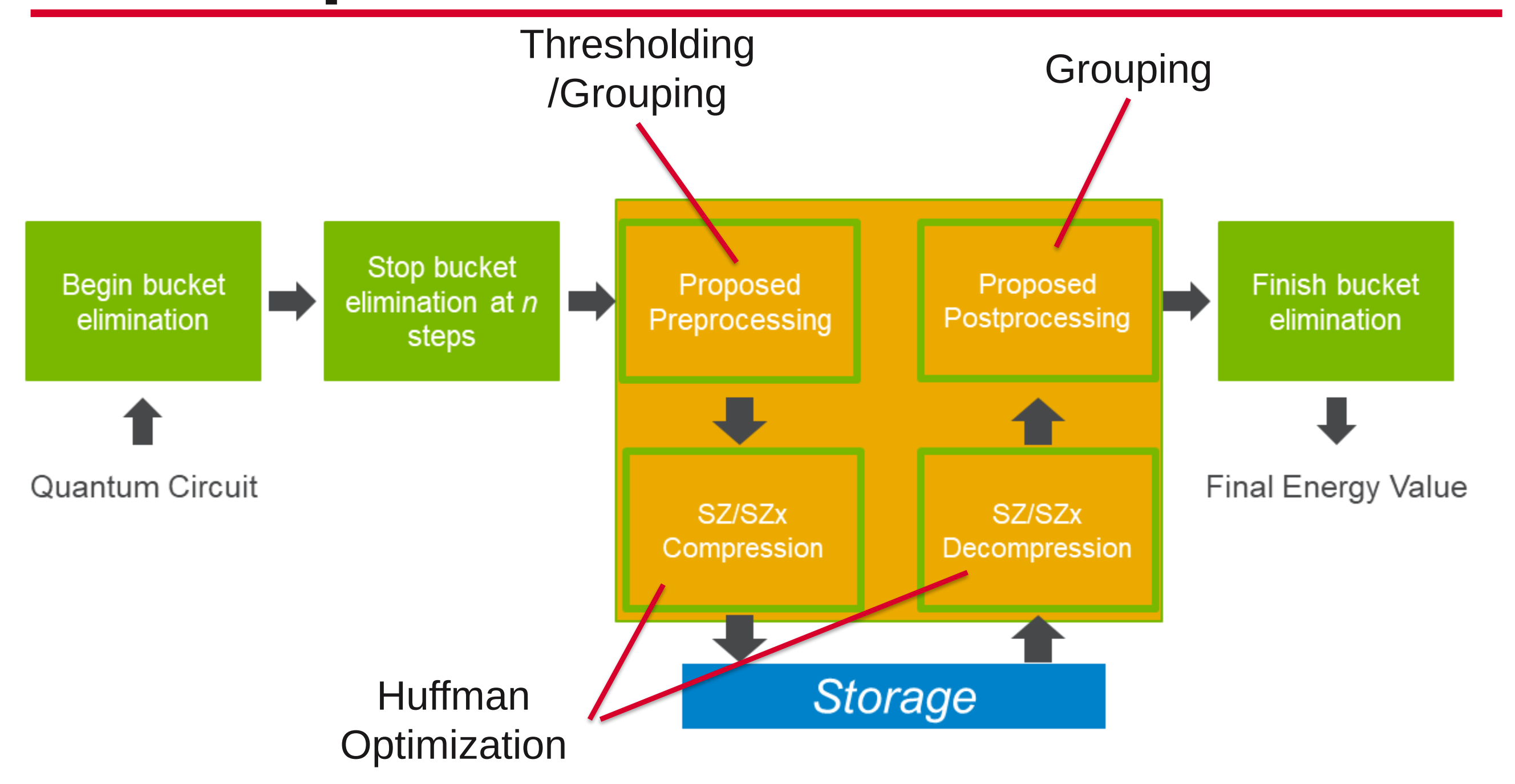
Challenges

- Need to bound final energy value
- FP data -> less compressible
- Data not smooth

Goals

- ➢ Extend existing lossy compressor frameworks, specifically SZ [2] and SZx [3] to compress/decompress quantum circuit simulation data
- ➢ Develop pre/postprocessing data transforms to boost compressibility and throughput
- ➢ Optimize underlying SZ and cuSZ (GPU implementation of SZ) [4] stages to increase throughput, specifically the Huffman encoding stage which can present a significant bottleneck when compressing metadata used in cuSZ
- ➢ Understand the effect of lossy compression/decompression on tensor contraction

Data Pipeline

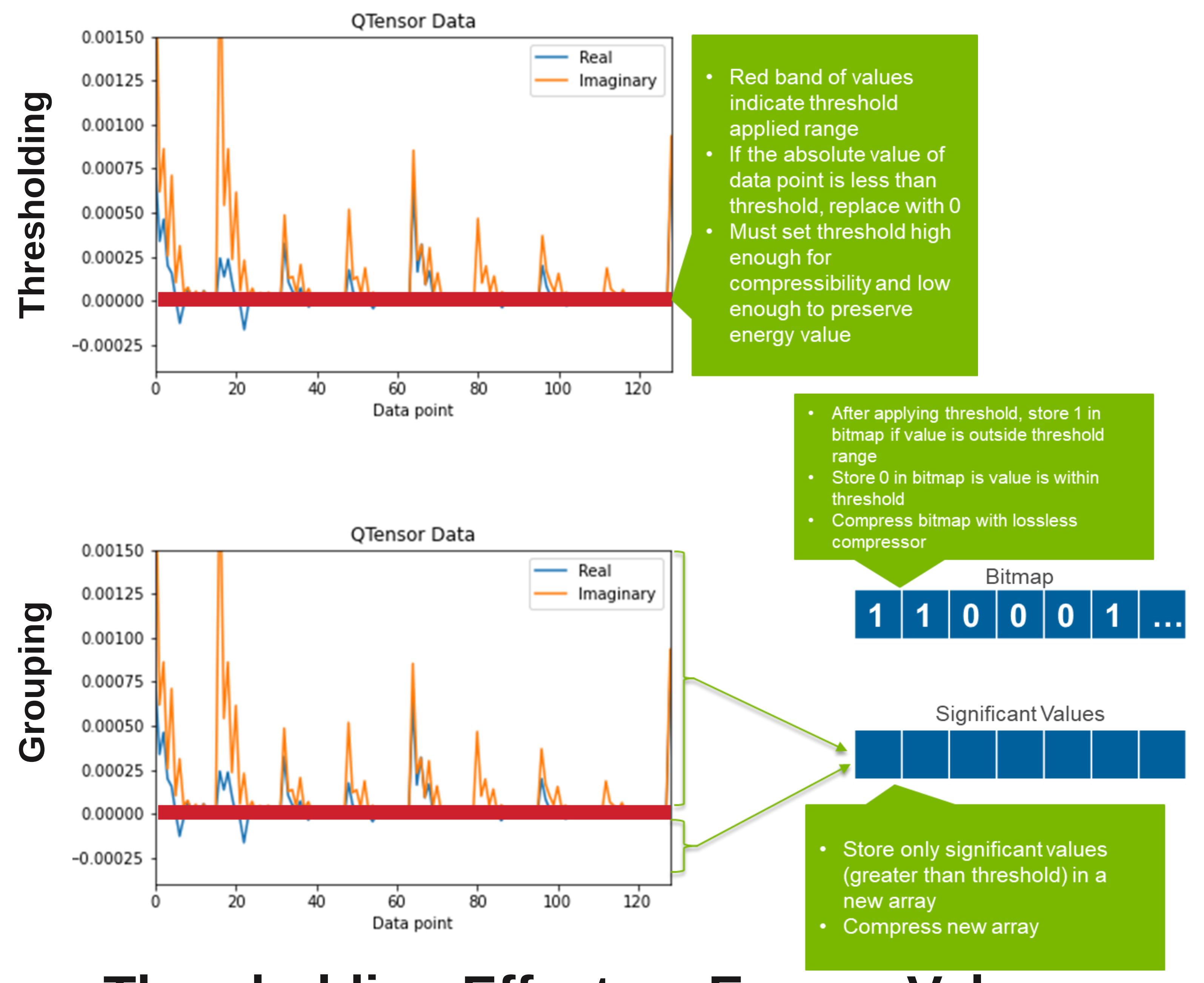


Pre/Postprocessing Steps

Thresholding: Since many tensor values are close to zero but not exactly zero when represented in floating-point representation, a threshold can be applied such that all values whose absolute value is less than the threshold are set to zero. Using this approach, the similarity across data points increases since many values become zero.

Grouping: Once tensors have a threshold step applied, the grouping method can rearrange data such that only nonzero values are compressed and zero values are ignored. A bitmap is used to store whether a location in the tensor corresponds to a zero or nonzero value. This can reduce the data size while simultaneously reducing the amount of data that must be sent through SZ/SZx.

Pre/Postprocessing Steps Visualized



- Red band of values indicate thresholded applied range
- If the absolute value of data point is less than threshold, replace with 0
- Must set threshold high enough for compressibility and low enough to preserve energy value

- After applying threshold, store 1 in bitmap if value is outside threshold range
- Store 0 in bitmap is value is within threshold
- Compress bitmap with lossless compressor

- Store only significant values (greater than threshold) in a new array
- Compress new array

Thresholding Effect on Energy Value

A relative-to-range (R2R) value is multiplied with the maximum of a dataset minus the minimum of the dataset to yield either a threshold or absolute error bound. We apply varying R2R thresholds and R2R absolute error bounds to QTensor data, compress then decompress the data, then finish the bucket elimination contraction to obtain the final energy value. We compare this value to the energy value of bucket elimination with no lossy compression as a middle stage.

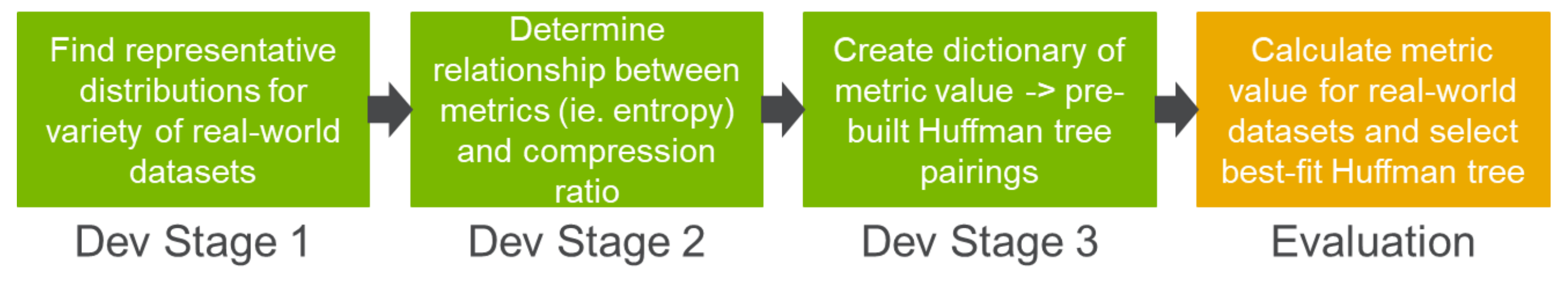
Percent Error of True Energy Value and R2R Threshold/Absolute Error Applied Energy Value

		Threshold (R2R)							
		0.25	0.1	0.05	0.01	0.005	0.001	0.0005	0.0001
R2R Error	0.25	0.743	0.605	0.268	0.232	0.211	1.206	0.338	0.666
	0.1	0.754	0.511	0.185	0.013	0.219	0.104	0.157	0.168
	0.05	0.751	0.440	0.279	0.073	0.158	0.069	0.094	0.066
	0.01	0.757	0.457	0.256	0.038	0.013	0.023	0.002	0.012
	0.005	0.757	0.457	0.242	0.031	0.015	0.010	0.002	0.011
	0.001	0.756	0.455	0.242	0.033	0.008	0.005	0.002	0.000
	0.0005	0.756	0.455	0.243	0.035	0.009	0.002	0.001	0.001
	0.0001	0.756	0.455	0.243	0.035	0.008	0.001	0.000	0.000

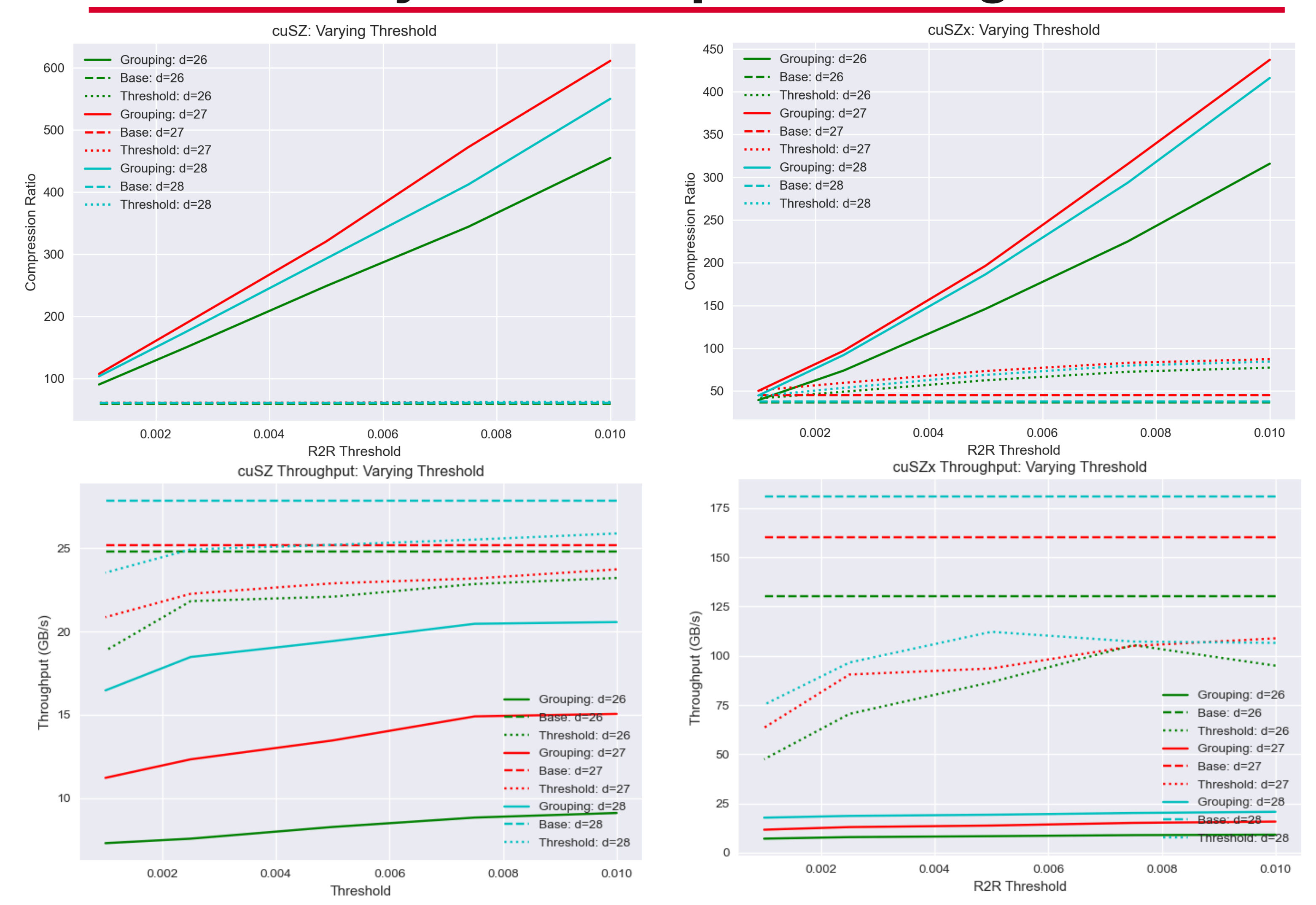
- SZ/SZx accept an absolute error bound that guarantees decompressed points are within specified value
- For above table, contraction is stopped with 16 steps left
- We target the red box of values (~1%-3% or less error) and explore different absolute error and threshold combinations in this range

Huffman Optimization

Huffman encoding is used in SZ and presents a performance bottleneck due to the serial nature of building a Huffman tree. In SZ (and cuSZ), data points are predicted based on previous data points and the distance from the prediction to the true value is expressed as a quantization of the error bound. These quantizations, expressed as integers, must be compressed and can be fed as a distribution into Huffman coding. We explore using prebuilt Huffman trees instead of custom building a Huffman tree for each dataset run through SZ/cuSZ. A tree is selected for a dataset based on the entropy of the tree and the dataset's quantization distribution.

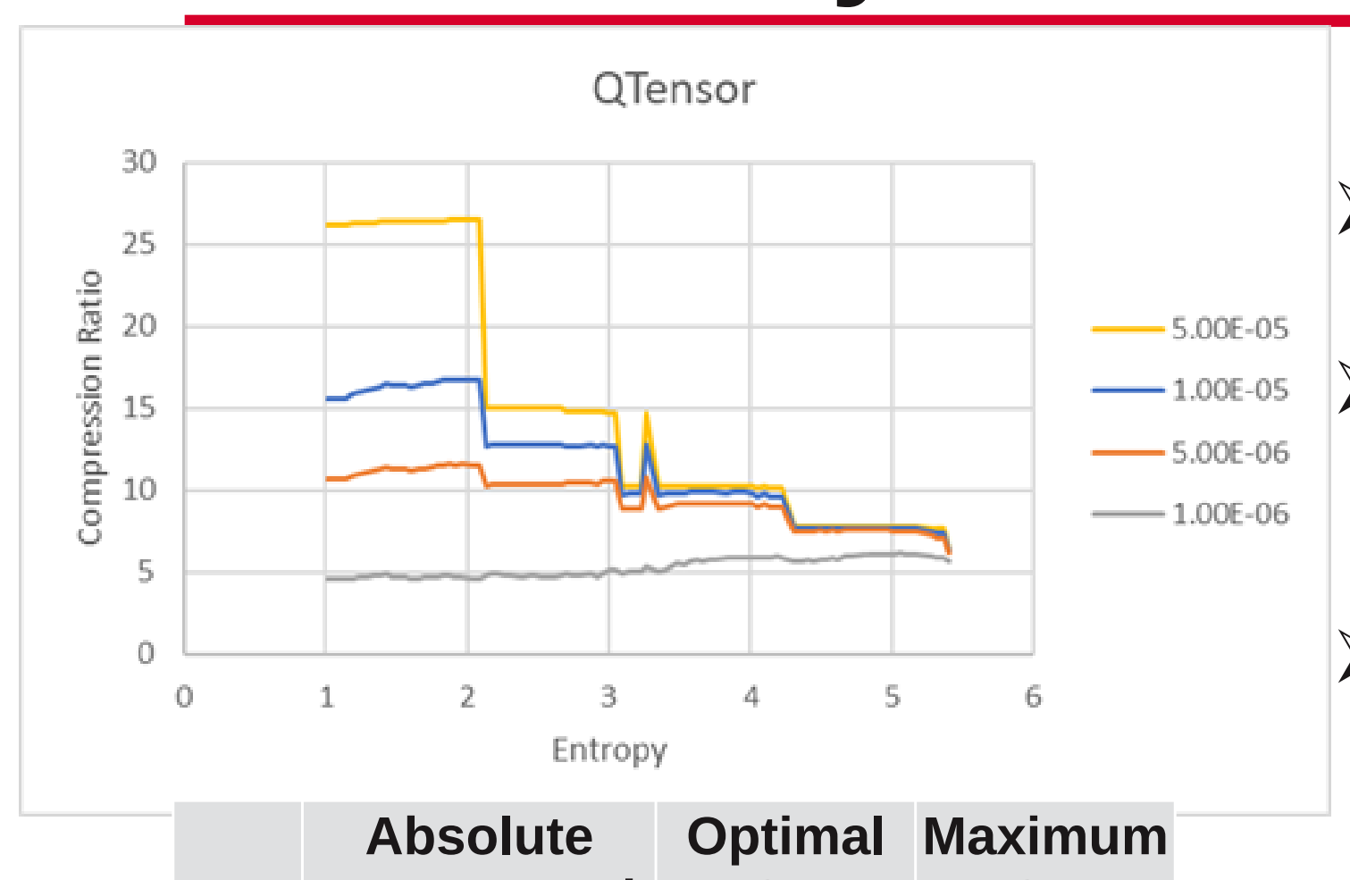

Find representative distributions for variety of real-world datasets
Determine relationship between metrics (ie. entropy) and compression ratio
Create dictionary of metric -> pre-built Huffman tree pairings
Calculate metric value for real-world datasets and select best-fit Huffman tree
Dev Stage 1
Dev Stage 2
Dev Stage 3
Evaluation

Preliminary Pre/Postprocessing Results



- Above plots show results for tensors varying in dimension from d = 26,27,28
- LZ4 is a CUDA-based lossless compressor added to compress the bitmap

Preliminary Huffman Results



	Absolute Error Bound	Optimal CR	Maximum CR
QTensor	5.00E-05	26.54	26.49
	1.00E-05	17.95	16.77
	5.00E-06	13.61	11.63
	1.00E-06	7.26	6.14

- ➢ Used normal distributions with varying entropies
- ➢ X-axis indicates entropy of normal distribution associated with a Huffman tree, Y-axis is compression ratio
- ➢ Lowest entropy normal distributions yielded Huffman trees that gave highest compression ratios for QTensor datasets with varying error bound

Conclusions and Future Work

For thresholding and grouping, compression ratios are promising, and these methods successfully boost compressibility while bounding energy result error to <3%. However, grouping can present a significant performance overhead that must be further minimized.

Pre-built Huffman trees can yield compression ratios that are close to optimal, but the normal distribution may not be the best fit for SZ quantization distributions. Additionally, only entropy has been studied as a selection criterion and other metrics can be explored.

Acknowledgements

This material was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant OAC-2003709, OAC-2104023. We acknowledge the computing resources provided on Bebop (operated by Laboratory Computing Resource Center at Argonne).

References

[1] Sheng Di and Franck Cappello. 2016. Fast Error-Bounded Lossy HPC Data Compression with SZ. In Proc. Of IPDPS '16 (IPDPS). 730–739. https://doi.org/10.1109/IPDPS.2016.11
[2] Roman Schutski, Danil Lykov, and Ivan Oseledets. 2020. Adaptive algorithm for quantum circuit simulation. Phys. Rev. A 101 (Apr 2020), 042335. Issue 4. https://doi.org/10.1103/PhysRevA.101.042335
[3] Jiannan Tian, Sheng Di, Kai Zhao, Cody Rivera, Megan Hickman Fulp, Robert Underwood, Sian Jin, Xin Liang, Jon Calhoun, Dingwen Tao, and Franck Cappello. 2020. cuSZ: An Efficient GPU-Based Error-Bounded Lossy Compression Framework for Scientific Data. In Proc. Of PACT '20 (Virtual Event, GA, USA) (PACT '20). Association for Computing Machinery, New York, NY, USA, 3–15. https://doi.org/10.1145/3410463.3414624
[4] Xiaodong Yu, Sheng Di, Kai Zhao, Jiannan Tian, Dingwen Tao, Xin Liang, and Franck Cappello. 2022. Ultrafast Error-Bounded Lossy Compression for Scientific Datasets. In Proc. Of HPDC '22(Minneapolis, MN, USA) (HPDC '22). Association for Computing Machinery, New York, NY, USA, 159–171. https://doi.org/10.1145/3502181.3531473