Integrating TEZIP into LibPressio: A Case Study of Integrating a Dynamic Application into a Static C Environment

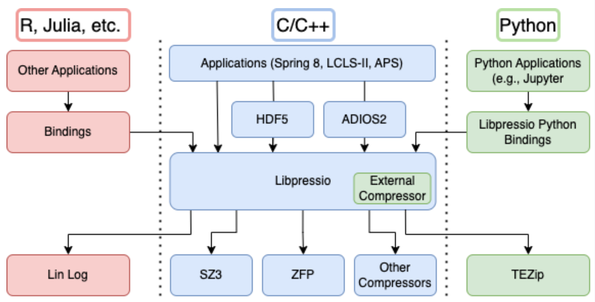
Isita Talukdar[1], Amarjit Singh[2] (advisor), Robert Underwood[3] (advisor), Kento Sato[2] (advisor), Weikuan Yu[4] (advisor)

University of California Berkeley[1], RIKEN Center for Computational Science[2], Argonne National Laboratory[3], Florida State University[4]

MOTIVATION

LCLS-II at SLAC, SNS at Oak Ridge Laboratory, and other instruments use software written in C and C++, producing huge volumes of time evolving data at high rate [6, 5]. Data compression can decrease the volume of data we need to move and store. TEZIP is a neural network (NN) based compressor designed for high-quality compression of time-evolving data, but TEZIP is written in Python and is not easily usable from or ported to C++ [4]. TEZIP isn't the only compressor with this challenge, such as the LinLogCompress.jl in Julia and other compressors using PyTorch/TensorFlow, e.g., Autoencoder Based Compressor [1, 3]. Bespoke C/C++ integrations for each compressor would be infeasible. **In this work we develop new components in LibPressio that allow us to integrate with TEZIP and other external compressors efficiently with a systematic approach [7]**

Figure 1: Flowchart Contextualizing the TEZIP-LibPressio Integration with other integration pathways across languages. Our contributions in green.

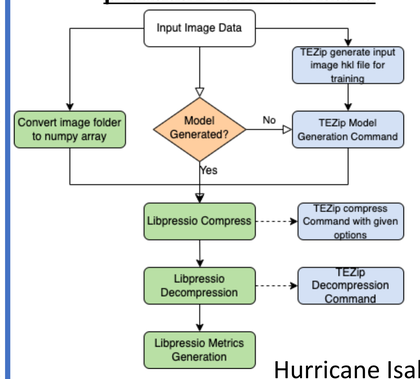


RESEARCH CHALLENGES

- TEZIP and LibPressio were designed for **different input formats** (colored images vs arrays) that need flexible and **efficient conversion** strategies
- TEZIP presents a **high startup overhead** from initialization that needs mitigation strategies.
- TEZIP was designed for **out-of-core** compression, and LibPressio for **in-core.**
- We need good strategies to **share memory** and **hide** file access **overheads** for fair comparisons.

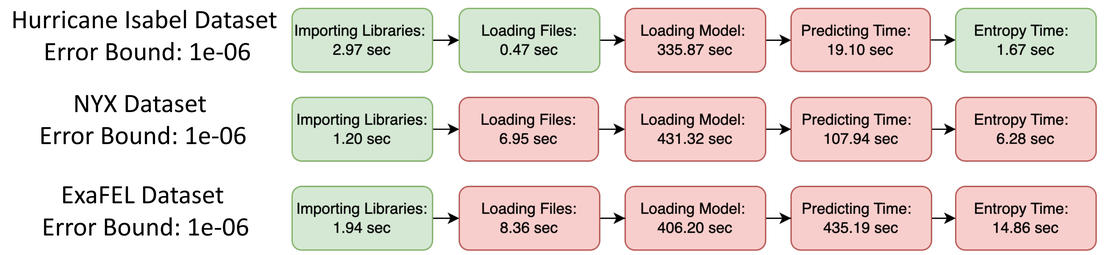
TEZIP COMPRESSION PRINCIPLES

Figure 2: Flowchart Depicting Basic Steps of TEZIP and calling protocol from LibPressio



- TEZIP is a neural network based compressor designed for time-evolving data
  - Image data is preprocessed into python objects for training and validation via hickle, which serializes to HDF5 [4]
  - Model architecture is saved in JSON format, along with the model weights in HDF5
- Similar to SZ3, TEZIP predicts the next data point based on the previous, and stores the difference (delta) between the prediction and the true next point [2]
- Unlike SZ3, TEZIP needs a **training phase** to train the Neural Network. This results in **overhead** in the form of initializing TensorFlow, the training of neural network
- Timing Profile of TEZIP compression and loading of the model shows opportunities for optimization during loading of the model, prediction, and entropy (Figure 3)

Figure 3: Flowchart Depicting Time Elapsed For Stages of TEZIP Compression



EVALUATION

Figure 4: Compressor Performance

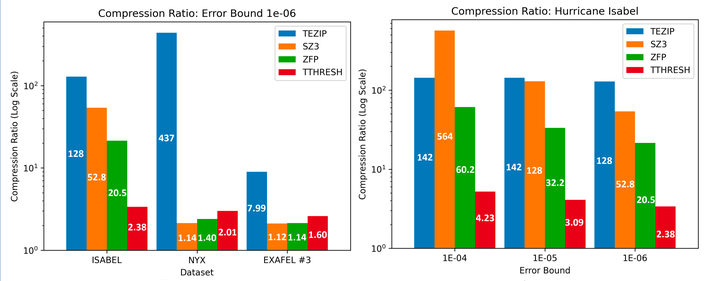


Figure 4A *Figure 4B*

Figure 5: Decompression Artifacts for SZ3 and TEZip

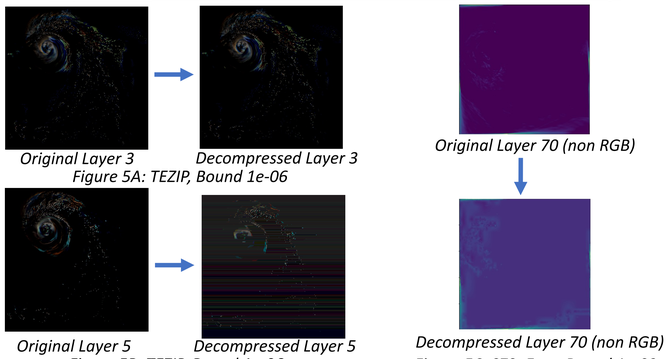


Figure 5A: TEZIP, Bound 1e-06

Figure 5B: TEZIP, Bound 1e-06

Figure 5C: SZ3, Error Bound 1e-02

Compressor Comparison Methodology

- Without this integration, comparison to other compressors would be difficult
- 4 Compressors Via LibPressio: TEZIP, SZ3, ZFP, TTHRESH
- 3 Error Bounds: 1e-04 to 1e-06, absolute error bound
- 3 Datasets from Scientific Data Reduction Benchmarks [8]
  - Hurricane ISABEL: Weather Simulation
  - NYX: Cosmological Simulation
  - ExaFEL: LCLS Serial Crystallography Image Data
- Compression Ratio And Artifact Analysis

Analysis

- Applications care about compression ratio and what artifacts are generated in the process. This integration makes an analysis possible.
- TEZIP's compression ratio for Hurricane Isabel is 128 which is **2.4** times greater than the SZ3's, 52.8 (Figure 4A).
- TEZIP: Since the neural network observes patterns and predict data, it can erroneously shadow images' most probable input for a given space. (Figure 5A, 5B)
- SZ3: Interpolation Error can result in one outlying data point creating a shadowing effect on the surrounding data (Figure 5C)

RESULTS

Basic Integration of TEZIP into LibPressio is completed

- Compression, decompression, and metric generation for TEZIP are enabled by LibPressio External Compressor Framework
- Artifacts generated during TEZIP processes shadow data using patterns found by NN

FUTURE WORK

Optimize integration to enable fair comparisons to other compressors

- Utilize Mochi to avoid start up overhead
- Utilize shared memory to avoid copies and FS overheads
- Explore the model training
  - Model training is in the "set_options" command, and is not included in the timing
- Research how to shorten the model loading and prediction time for TEZIP

CONCLUSION

- This work sets a precedent for the integration of non C/C++ compressors into LibPressio
- A similar framework can be used for other compressors in the future

REFERENCES

[1] Milan Klöwer, Miha Razinger, Juan J. Dominguez, Peter D. Düben, and Tim N. Palmer. 2021. Compressing atmospheric data into its real information content. Nature Computational Science 1, 11 (2021), 713–724. DOI:http://dx.doi.org/10.1038/s43588-021-00156-2

[2] Xin Liang et al. 2023. SZ3: A modular framework for composing prediction-based error-bounded lossy compressors. *IEEE Transactions on Big Data* 9, 2 (2023), 485–498. DOI:http://dx.doi.org/10.1109/tbdata.2022.3201176

[3] Jinyang Liu et al. 2021. Exploring autoencoder-based error-bounded compression for scientific data. 2021 IEEE International Conference on Cluster Computing (CLUSTER) (2021). DOI:http://dx.doi.org/10.1109/cluster48925.2021.00034

[4] Rupak Roy et al. 2021. Compression of time evolutionary image data through predictive deep neural networks. 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid) (2021). DOI:http://dx.doi.org/10.1109/ccgrid51090.2021.00014

[5] Galen Shipman et al. 2014. Accelerating data acquisition, reduction, and analysis at the Spallation Neutron Source. *2014 IEEE 10th International Conference on e-Science* (2014). DOI:http://dx.doi.org/10.1109/escience.2014.31

[6] J. Stohr. 2011. Linac Coherent Light Source II (LCLS-II) conceptual design report (2011). DOI:http://dx.doi.org/10.2172/1029479

[7] Robert Underwood, Victoriana Malvoso, Jon C. Calhoun, Sheng Di, and Franck Cappello. 2021. Productive and performant generic lossy data compression with Libpressio. 2021 7th International Workshop on Data Analysis and Reduction for Big Scientific Data (DRBSD-7) (2021). DOI:http://dx.doi.org/10.1109/drbsd754563.2021.00005

[8] Kai Zhao et al. 2020. SDRBench: Scientific Data Reduction Benchmark for lossy compressors. *2020 IEEE International Conference on Big Data (Big Data)* (2020). DOI:http://dx.doi.org/10.1109/bigdata50022.2020.9378449

ACKNOWLEDGMENTS

- *This work is supported in part by the National Science Foundation award 1952302. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.*
- *Argonne National Laboratory's contribution is based upon work supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-06CH1135.*
- *This work has been supported by the COE research grant in computational science from Hyogo Prefecture and Kobe City through Foundation for Computational Science.*

Please find Isita Talukdar's CV at this QR code

UC Berkeley Class of 2026