Submission code: AP56R8LM7D

See this and other abstracts on our website: https://home.sbgf.org.br/Pages/resumos.php

Reverse Time Migration using a novel wavelet-based lossy compressor.

Oscar Mojica (Supercomputing Center for Industrial Innovation; SENAI CIMATEC), Leonildes Soares de Melo Filho (Repsol Sinopec Brasil)

SBGf Conference
18-20 NOV | Rio'25

Reverse Time Migration using a novel wavelet-based lossy compressor.

Abstract Summary

Reverse Time Migration (RTM) generates terabyte-scale datasets in large-scale runs, necessitating efficient data management. Error-bounded lossy compression techniques offer a promising solution by reducing storage needs while maintaining acceptable data accuracy. As lossy compression evolves, rigorous testing in applications like RTM is essential. This study evaluates SPERR, a new wavelet transform-based lossy compression framework, comparing it with established compressors SZ3 and zfp to assess its effectiveness in managing RTM's large data volumes.

Introduction

RTM, a key seismic imaging technique, generates large datasets requiring significant storage and I/O operations. Lossy compression frameworks offer substantial storage reduction without compromising results. Dmitriev et al. (2022) explored error-bounded lossy compression in acoustic RTM, evaluating zfp, SZ3, and Bitcomp for compression ratios, throughput, and image quality. Huang et al. (2023) introduced HyZ, a hybrid lossy compression method for RTM, improving performance and data quality. Barbosa and Coutinho (2023) propose a dual strategy involving lossy and lossless wavefield compression for parallel multi-core and GPU-based acoustic RTM, reducing data transfer and storage needs while maintaining image quality. Kukreja et al. (2022) integrated checkpointing with lossy compression for Full-Waveform Inversion, enhancing efficiency and scalability. Inspired by these, we propose to integrate a wavelet transform-based lossy compressor into RTM to handle memory limitations in large-scale simulations.

1 Error-bounded Lossy Compression

This overview compares SPERR, a transform-based open-source compressor, with two other open-source compressors—zfp (transform-based) and SZ3 (prediction-based)—selected for their CPU (de)compression support, aligning with our multi-core RTM software.

-***zfp*** (Lindstrom, 2014) operates as a block-transform-based compressor, akin to the method used in JPEG compression. Its approach involves employing a custom orthogonal transform on data blocks, followed by encoding the transformed coefficients using specialized bitplane encoders. This unique methodology enables zfp to achieve high compression ratios and speeds, thanks to its optimized transform and encoding techniques.

-***SPERR*** (Li et al., 2023) uses wavelet transforms to generate coefficients that compactly represent input data, with larger coefficients holding more information. Data reduction occurs during coefficient encoding (potentially leading to information loss) using SPECK (Tang and Pearlman, 2006). SPERR identifies outliers exceeding a specified Pointwise Error (PWE) tolerance, records their positions, and calculates corrections with a SPECK-inspired algorithm. It encodes wavelet coefficients

and outliers into separate bitstreams, which are merged and compressed losslessly with zstd (Collet, 2017) to produce the final output.

-***SZ3*** (Zhao et al., 2021) employs a prediction-based approach using multi-dimensional dynamic spline interpolation, eliminating the need to store linear regression coefficients as in SZ2.1 (Liang et al., 2018). Following prediction, SZ3 quantizes prediction errors and compresses the resulting integers with Huffman encoding and zstd lossless compressor, enhancing compression ratios, especially for higher error bounds.

2 Reverse-Time Migration

The core concept of RTM involves a three-part process: (a) simulating the advancement of a wave field forward in time through a suitable velocity model, (b) reversing the process by propagating the recorded data backward through the same model, and (c) combining the results of these two steps by applying an imaging condition. In our case, this simulation relies on the well known time-domain formulation of the wave equation:

$$\frac{1}{v^2(\boldsymbol{x})}\partial_{tt}p(\boldsymbol{x},t) - \nabla^2 p(\boldsymbol{x},t) = f(t), \tag{1}$$

where $p(\boldsymbol{x},t)$ is the pressure field, $v(\boldsymbol{x})$ the compressional wave velocity, $\boldsymbol{x}$ denotes the 2-D or 3-D spatial coordinates, t the time in $[0,T]$, and $f(t)$ is a given source term. The ∇^2 represents the laplacian and ∂_{tt} is the second-order temporal derivative. From the several available options for acoustic imaging conditions, we use the cross-correlation normalised by the square of the source illumination strength. The correlation images of all shots are summed up to get the image $I(\boldsymbol{x})$:

$$I(\boldsymbol{x}) = \sum_s \frac{\sum_t p_s(\boldsymbol{x},t)p_r(\boldsymbol{x},t)}{\sum_t p_s(\boldsymbol{x},t)p_s(\boldsymbol{x},t)}, \tag{2}$$

where $p_s(\boldsymbol{x},t)$ and $p_r(\boldsymbol{x},t)$ are the source and receiver wavefields, respectively.

3 Numerical experiments

We used a subsection of the SEAM acoustic isotropic velocity model, depicting a sedimentary basin bordered by salt. An 800-meter-deep water layer was added to mimic a moderately deep water setting. The modified model size is 200x350x200 with a 20-meter sampling interval in all directions. Synthetic data were generated using an 8 Hz Ricker wavelet source with a 4 s recording time, comprising 231 seismic shots sampled at 4 ms. All compressors were compiled with Intel oneAPI 2023.2.1 (icc/icpc). Tests ran on a compute node with two 20-core Intel Xeon Gold 6148 processors, 384 GB memory, and RHEL 8. We used a parallel Fortran RTM code, integrating SPERR and SZ3 via Fortran wrappers (available at `https://github.com/ofmla/fortran-sperr` and `https://github.com/ofmla/sz3_simple_example`). We stored forward-modeled wavefields on disk, reading them during backward propagation for imaging, even though nodes could store all snapshots uncompressed. With a 3 km and 1 km maximum offset, we cropped the v_p model and adjusted it for boundary layers. Forward propagation generated 2032 snapshots (252x198x148 single-precision floats each), requiring ≈60 GB per shot uncompressed. We evaluated lossy compression using point-wise absolute error as the error-bound (eb) type, which is supported by all tested compressors. We tested eb values of 1E-4, 1E-5, and 1E-6. Figure 1 illustrates compressibility across 231 shots, with color gradients (dark to light) indicating shot index. SPERR (left subplot) at eb=1E-4 starts with a high compression factor (>100000), drops within 500 time steps, stabilizes at 20000–30000,

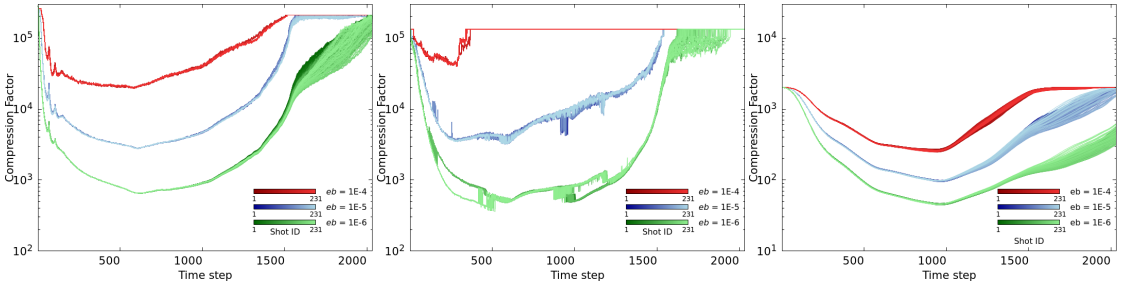

Figure 1: Compression factor achieved by SPERR (left), SZ3 (center), and zfp (right) for 231 shots across 2,032 time steps, with each snapshot compressed using different error bounds: eb=1E-4 (red), eb=1E-5 (blue), and eb=1E-6 (green).

and rises late, nearing initial levels; tighter bounds (eb=1E-5, eb=1E-6) follow a similar trend but with lower factors. SZ3 (center subplot) shows a three-phase pattern for eb=1E-5 and eb=1E-6 (decline, stabilization, late increase), but eb=1E-4 is unstable, suggesting failure. ZFP (right subplot) follows the same pattern but with lower compression ratios. Compressibility tracks wavefield evolution: high early and late due to zero-dominated data, lower mid-simulation as the wavefield spreads. At an error bound of 1E-6, SPERR, SZ3, and zfp were evaluated for memory savings by summing compressed byte streams across 231 shots (2032 snapshots each) and computing average compressed size per shot. SPERR excels, achieving a total of 9 GB (avg. 0.039 GB/shot), a 1539:1 compression ratio, and 99.94% memory savings (59.961 GB saved/shot). SZ3 yields 11.75 GB (avg. 0.051 GB/shot), a 1176:1 ratio, and 99.92% savings (59.949 GB saved/shot). ZFP, less efficient, totals 139.69 GB (avg. 0.605 GB/shot), with a 99:1 ratio and 98.99% savings (59.395 GB saved/shot). SPERR's superior compressibility reduces storage to 1.31 times smaller than SZ3 and over 15 times smaller than zfp. However, when maintaining a specified eb tolerance, SPERR takes longer to compress data com-

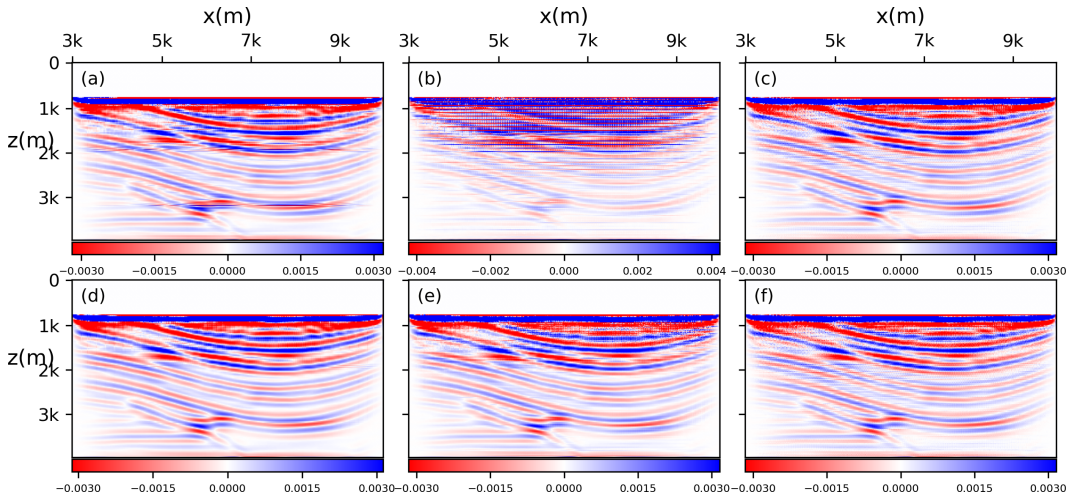

Figure 2: RTM images at $y = 5.99$ km. Top row (eb = 1E-5): (a) SPERR, (b) SZ3, (c) zfp. Bottom row (eb = 1E-6): (d) SPERR, (e) SZ3, (f) zfp. Colorbar represents normalized image values using TwoSlopeNorm[1], with vmin/vmax set to ± 1 standard deviation of the image data.

[1] https://matplotlib.org/stable/api/_as_gen/matplotlib.colors.TwoSlopeNorm.html

pared to zfp and SZ3. The figure 2 shows RTM images. At 1E-5 (top row), noise is more pronounced, especially in (b) SZ3, with visible artifacts. At 1E-6 (bottom row), images appear similar, with reduced noise overall but some residual noise at shallow depths, especially in (e) SZ3 and (f) zfp. SPERR (d) remains the least noisy.

4 Conclusions

In this paper, we evaluated SPERR, an error-bounded lossy compressor for scientific data, against SZ3 and ZFP for seismic imaging. At a fixed eb tolerance, SPERR typically achieves higher compression ratios and comparable image quality but it requires longer compression time than zfp and SZ3. When memory is a primary concern and performance is less critical, SPERR is the preferred choice.

5 Acknowledgements

The authors would like to thank Repsol Sinopec Brasil for supporting this research under the “Siren Project” agreement, in accordance with the regulations of the National Agency of Petroleum, Natural Gas, and Biofuels (ANP) RD&I levy fund. Oscar Mojica acknowledges the INCT-GP.

References

Barbosa, C. H., and A. L. Coutinho, 2023, Reverse time migration with lossy and lossless wavefield compression: 2023 IEEE 35th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), IEEE, 192–201.

Collet, Y., 2017, Zstandard-fast real-time compression algorithm: Github repository [online].

Dmitriev, M., T. Tonellot, H. AlSalem, and S. Di, 2022, Error-bounded lossy compression in reverse time migration: Sixth EAGE High Performance Computing Workshop, European Association of Geoscientists & Engineers, 1–5.

Huang, Y., K. Zhao, S. Di, G. Li, M. Dmitriev, T.-L. D. Tonellot, and F. Cappello, 2023, Towards improving reverse time migration performance by high-speed lossy compression: 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), IEEE, 651–661.

Kukreja, N., J. Hückelheim, M. Louboutin, J. Washbourne, P. H. Kelly, and G. J. Gorman, 2022, Lossy checkpoint compression in full waveform inversion: a case study with zfpv0. 5.5 and the overthrust model: Geoscientific Model Development, **15**, 3815–3829.

Li, S., P. Lindstrom, and J. Clyne, 2023, Lossy scientific data compression with sperr: 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 1007–1017.

Liang, X., S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, 2018, Error-controlled lossy compression optimized for high compression ratios of scientific datasets: 2018 IEEE International Conference on Big Data (Big Data), IEEE, 438–447.

Lindstrom, P., 2014, Fixed-rate compressed floating-point arrays: IEEE transactions on visualization and computer graphics, **20**, 2674–2683.

Tang, X., and W. A. Pearlman, 2006, Three-dimensional wavelet-based compression of hyperspectral images, *in* Hyperspectral data compression: Springer, 273–308.

Zhao, K., S. Di, M. Dmitriev, T.-L. D. Tonellot, Z. Chen, and F. Cappello, 2021, Optimizing error-bounded lossy compression for scientific data by dynamic spline interpolation: 2021 IEEE 37th International Conference on Data Engineering (ICDE), IEEE, 1643–1654.