Spline-Interpolation-Based Lossy Compression for Scientific Data

Jiannan Tian, Dingwen Tao
Indiana University
Bloomington, IN, United States
{jti1,ditao}@iu.edu

Sheng Di, Franck Cappello
Argonne National Laboratory
Lemont, IL, United States
sdi1@anl.gov,cappello@mcs.anl.gov

Abstract

Error-bounded lossy compression is a critical technique for significantly reducing scientific data volumes. With ever-emerging heterogeneous high-performance computing (HPC) architecture, GPU-accelerated error-bounded compressors such as cuSZ have been developed. In order to improve the data quality and the compression ratio while maintaining high throughput, an interpolation-based spline method is introduced, inspired by the existing CPU prototype. In this work, We present (1) an efficient GPU implementation of the 3D interpolative spline prediction method, (2) a finer-grained data chunking approach using anchor points to leverage the modern GPU architecture, and (3) an in-depth analysis of how such anchor point affects the error formation and the compression ratio, and (4) a preliminary result in performance on the state-of-the-art modern GPUs. Our solution can achieve 1) a higher compression ratio than the previous default prediction method in cuSZ, 2) the overall comparable data quality and compression ratio with the CPU prototype.

1 Introduction

Large-scale scientific applications for advanced instruments produce vast data for post hoc analysis. For instance, Hardware/Hybrid Accelerated Cosmology Code (HACC) [1, 2] may produce petabytes of data in hundreds of snapshots when simulating one trillion particles. It is easy to saturate the parallel file system (PFS) [3, 4], given their relatively low bandwidths, and eventually becomes the bottleneck of the application.

With data reduction becoming an effective method to alleviate I/O pressure, designing a scientific lossy compressor entails addressing three primary concerns: high fidelity preservation, high compression ratio, and fast processing capability. Most existing scientific lossy compressors (e.g., SZ [5, 6], FPZIP [7], ZFP [8]) tend to fulfill the demand of achieving the first two aspects. However, the ever-growing popularity of heterogeneous accelerators (e.g., modern GPUs) makes it a shift in focus to perform in situ data processing on GPU. To this end, porting the CPU-based compressors with the processing capability at multi-hundred (per core) megabytes to GPU with multi-hundred-gigabyte capability (supposedly matching the data generating rate [9]) is challenging, not to mention maintaining the data quality and sacrificing tolerable compression ratio.

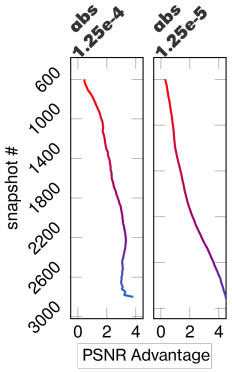

Figure 1: PSNR advantage of interpolative Spline 3D prediction on GPU over the default Lorenzo prediction.

The previous GPU-based error-bounded lossy compressors, cuSZ [10, 11], uses extrapolative Lorenzo predictor. As is pointed out in [12], for certain scientific applications such as climate and seismic imaging data, interpolation-based prediction can archive better data quality with an even higher compression ratio than the default SZ Lorenzo prediction. As shown in Fig. 1, by observing over 3,000 snapshots in a seismic reverse time migration (RTM) simulation, we found the interpolative spline method generally achieves better data quality in PSNR (peak signal-to-noise ratio) than the default Lorenzo prediction in SZ. However, how to fit the interpolation onto GPU by explicitly leveraging its architecture and how the potential modification would affect the compression ratio remain a question.

In this work, We hereby present (1) an efficient GPU implementation of the 3D interpolative spline prediction method, (2) a finer-grained data chunking approach using anchor points to leverage the modern GPU architecture, (3) an in-depth analysis of how such anchor point affects the error formation and the compression ratio, and (4) a preliminary result in performance on the state-of-the-art modern GPUs.

2 Design

Zhao et al. [12] proposed a Spline-3D design, which iteratively goes inward to interpolate from a global scale of the data field to one index unit. Since it spans the whole data field (the dimension is 449 × 449 × 235, *zyx*-order), it inevitably raises the question of how to map the data onto GPU considering the GPU architecture. More specifically, in order to saturate the memory bandwidth and hence lower the number of high-latency memory transactions, coalescing memory access of a multiple of 128 bytes is desired; for a thread block, non-batched access that spans the whole data field (e.g., $x = 0$ and $x = 128$ to predict $x = 64$) causes severe cache miss and hence degrades the performance. Formally, trilinear interpolation is a three-time linear interpolation corresponding to the three axes. The maximum number of interpolative iteration could be $\log_2\lceil\max(235, 445, 445)\rceil$ in our case when using the RTM dataset (from [12]). Suppose in one iteration, whose corresponding interpolation distance 2^k can make the data chunk

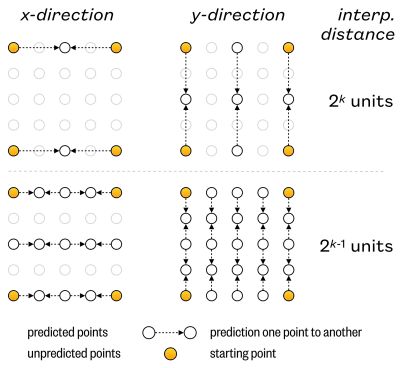

Figure 2: Mofified interpolation in concept; the boundary points are explicitly handled with finer-grained data chunking, called "anchor points".

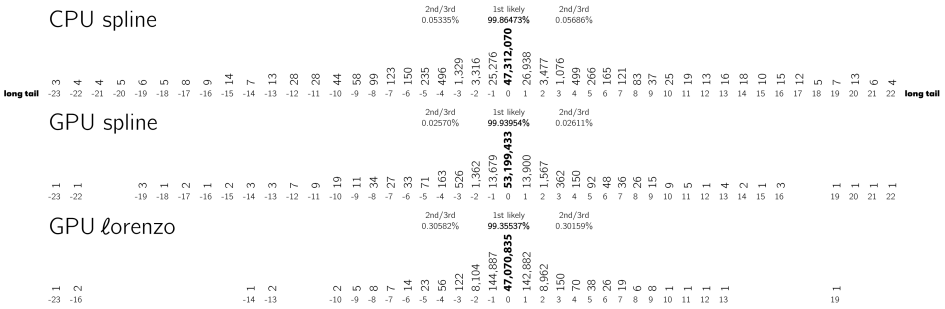

Figure 3: Error-quantization code distribution comparison across three prediction methods.

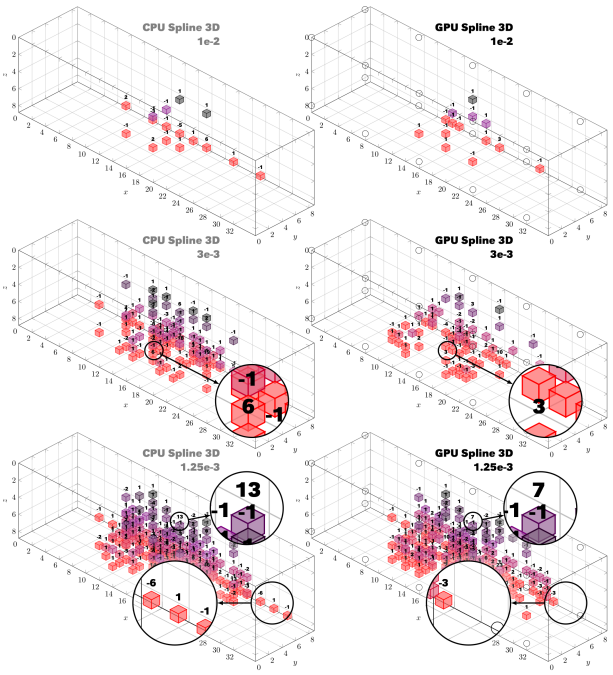

Figure 4: With anchor points (hollow circles) pinning the original values, GPU Spline 3D prediction results in fewer non-zero error-quantization codes and codes with smaller amplitude. This demonstration uses RTM simulation data No. 2815, with a quartet chunk at (0,0,0) selected. Note that (0,0,0) is more dramatic than other chunks.

fit into a thread block; multiple factors decide the proper k, which is unfeasible to exhaust at one time. Note that the interpolation of distance 2^k involves $2^{k+1}+1$ data elements along with one axis.

In practice, we first define an 8-by-8-by-8 data partition as a unit chunk. Then, a 9-by-9-by-9 chunk encloses 512 elements and becomes a unit workspace, with one unit padded on each axis. To immediately start the interpolation at a distance of 4, the points whose indices satisfy (idx $\mod 8 \equiv 0$) must be ready. To avoid global access that caused performance degradation, we propose to use anchor points that are losslessly saved along with the compressed quant-codes afterward. We select $(0,0,0)$ for each basic workspace as the anchor point. Therefore, 1 of 512 elements becomes an anchor point, making up roughly 1/512 overhead to save the lossless anchor point; therefore, the use of anchor points is well justified given the small overhead in compression ratio. Then, the interpolation performs a similar three-time linear prediction to the CPU design in [12]. The interpolation that starts with distance 2^k goes through k iterations down to distance 1, with each time the distance halved.

Figure 3 and 4 show that 1) in comparison with the default Lorenzo prediction, the error-quantization code is marginally more centralized and 2) in comparison with the CPU version, the anchor points results in can result in less non-zero error-quantizations and smaller in the error-quantization amplitude. In addition, by controlling over the input error bound, we can see the changing error-quantization in Figure 4; it is consistently good when introducing the anchor points.

3 Evaluation

Our evaluation setup is listed below,

platform A100 of ALCF-ThetaGPU, V100 of ANL-
dataset Seismic RTM data of over 3,000 snapshots.

Throughput The Spline 3D kernel can achieve over 200 GB/s in data processing, as shown below,

- Compression: 232.7 GB/s on V100, and 253.5 GB/s on A100
- Decompression: 242.3 GB/s on V100, and 335.8 GB/s on A100

Compression Ratio Figure 5 shows the variety of compression ratios across the whole dataset; in conclusion, the GPU implementation can achieve a comparable data reduction rate with the CPU version. Table 1 shows a detailed comparison at snapshot No. 2815; with anchor point and sparsity gather techniques, GPU spline can achieve a competitive compression ratio.

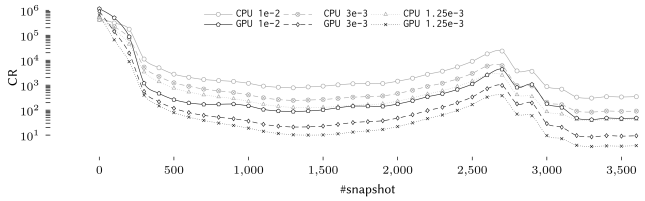

Figure 5: The variety in compression ratio across thousands of snapshots.

		GPU Spline Prediction			CPU Spline Prediction			GPU Lorenzo Prediction	
eb	quant.	freq.	CR	quality	freq.	CR	quality	CR	quality
1.25e-3	-1	156,221 0.2935%	G 67.0		243,936 0.5149%	G 35.4			
	0	52,834,426 99.2538%	GA 59.3	PSNR 68.7	46,707,654 98.5888%	G+VLE 70.9	PSNR 69.7	G+VLE (lower)	PSNR 63.6
	-1	152,777 0.2870%	GA+VLE 118.5	corr. 0.9979	235,588 0.4973%	(upper limit*) 397.5	corr. 0.9983		corr. 0.9937
3.00e-3	-1	61,234 0.1150%	G 177.0		112,857 0.2382%	G 82.4			
	0	53,081,227 99.7175%	GA 131.5	PSNR 63.9	47,088,719 99.3931%	G+VLE 164.8	PSNR 63.6	G+VLE (lower)	PSNR 58.0
	-1	61,518 0.1156%	GA+VLE 263.0	corr. 0.9937	112,969 0.2385%	(upper limit*) 938.8	corr. 0.9932		corr. 0.9771
1.00e-2	-1	13,679 0.0257%	G 827.0		25,276 0.0534%	G 369.2			
	0	53,199,433 99.9395%	GA 316.2	PSNR 59.1	47,312,070 99.8646%	G+VLE 738.4	PSNR 56.2	G+VLE 77.6	PSNR 53.9
	-1	13,900 0.0261%	GA+VLE 632.5	corr. 0.9809	26,938 0.0569%	(upper limit*) 3606.7	corr. 0.962		corr. 0.9390

Table 1: The comparison of the three methods. "G" stands for sparsity-gather, "A" stands for the overhead from recording the anchor points, and "+VLE" stands for the optional variable-length-encoding.

Conclusion

In this work, we propose a spline interpolation-based prediction to achieve higher data quality and a better compression ratio. By leveraging the fine-grained data chunking method and using anchor points, we can fit the originally large problem to GPU without hurting the compression ratio much. We also present the preliminary throughput evaluation on NVIDIA V100 and A100 GPUs.

Acknowledgments

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations—the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, to support the nation's exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357. This work was also supported by the National Science Foundation under Grants OAC-2042084, OAC-2034169, OAC-2003709, and CCF-1619253.

References

[1] S. Habib *et al.*, "HACC: Extreme scaling and performance across diverse architectures," *Communications of the ACM*, vol. 60, no. 1, pp. 97–104, 2016.

[2] S. C. V. Vishwanath and K. Harms, *Parallel i/o on mira*, https://www.alcf.anl.gov/files/Parallel_IO_on_Mira_0.pdf, Online, 2019.

[3] X. Liang *et al.*, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA: IEEE, 2018, pp. 438–447.

[4] X. Liang, S. Di, D. Tao, Z. Chen, and F. Cappello, "An efficient transformation scheme for lossy data compression with point-wise relative error bound," in *IEEE International Conference on Cluster Computing (CLUSTER)*, Belfast, UK: IEEE, 2018, pp. 179–189.

[5] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *2016 IEEE International Parallel and Distributed Processing Symposium*, Chicago, IL, USA: IEEE, 2016, pp. 730–739.

[6] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *2017 IEEE International Parallel and Distributed Processing Symposium*, Orlando, FL, USA: IEEE, 2017, pp. 1129–1139.

[7] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1245–1250, 2006.

[8] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[9] F. Cappello *et al.*, "Use cases of lossy compression for floating-point data in scientific data sets," *The International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1201–1220, 2019.

[10] J. Tian *et al.*, "Cusz: An efficient gpu-based error-bounded lossy compression framework for scientific data," in *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, 2020, pp. 3–15.

[11] J. Tian *et al.*, "Optimizing error-bounded lossy compression for scientific data on gpus," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, Los Alamitos, CA, USA: IEEE Computer Society, 2021, pp. 283–293. DOI: 10.1109/Cluster48925.2021.00047. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/Cluster48925.2021.00047.

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