Introducing Prefetching and Data Compression to Accelerate Checkpointing for Inverse Seismic Problems

THIAGO MALTEMPI[*], SANDRO RIGO[*], MARCIO PEREIRA, and HERVÉ YVIQUEL, Computer Systems Laboratory (LSC), University of Campinas (UNICAMP), Brazil
JESSÉ COSTA, Faculty of Geophysics, Pará Federal University, Brazil
ALAN SOUZA, CENPES Petrobras, Brazil
GUIDO ARAUJO, Computer Systems Laboratory (LSC), University of Campinas (UNICAMP), Brazil

CCS Concepts: • **Computing methodologies → Parallel computing methodologies**; • **Applied computing → Earth and atmospheric sciences**.

Additional Key Words and Phrases: seismic application, checkpointing, compression, prefetching

ACM Reference Format:
Thiago Maltempi, Sandro Rigo, Marcio Pereira, Hervé Yviquel, Jessé Costa, Alan Souza, and Guido Araujo. 2018. Introducing Prefetching and Data Compression to Accelerate Checkpointing for Inverse Seismic Problems. In . ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Reverse Time Migration (RTM) [1] is a Seismic Imaging technique that poses significant computational challenges, often involving hundreds of gigabytes of memory requirements and lengthy processing times lasting several days. RTM can be a critical component of a Full Waveform Inversion (FWI) solution. The RTM algorithm solves the wave equation by discretizing the pulse into a series of "timesteps". During the forward phase, the wave propagation is computed forward in time. Subsequently, the backward phase is initiated, where the same timesteps are processed in reverse, enabling the correlation of forward and backward data to generate the final image.

Our RTM implementation processes three-dimensional fields on NVIDIA GPUs incorporating domain decomposition across up to four GPU devices and uses the Revolve algorithm [3] for an optimized checkpointing strategy. The traditional checkpointing approach saves the checkpointing data into the host memory. It retrieves it from the host when demanded, introducing a notable performance bottleneck and making GPU idle while it waits for data transfer.

Our research introduces a prefetching mechanism to address this issue, anticipating memory transfers and applying compression to checkpoint data. Our experimental results demonstrate notable speedup for Prefetching (around 1.42x) and even greater improvement for Prefetching + Compression (ranging from 1.98 to 2.53x). Prefetching reduces data transferred from the host to the GPU by 4x due to the Checkpointing Buffer. When combined with compression,

[*]Both authors contributed equally to this research.

host-to-device transfers can be reduced by 16x compared to the traditional checkpointing implementation, thanks to the compression rate of 4.

There are attempts at using data compression to improve performance on scientific data and inverse problem solutions[2, 4, 6] but, to the best of our knowledge, this is the first work to propose a prefetching technique combined with checkpoint data compression inside the GPU.

2 METHODOLOGY AND EXPERIMENTAL RESULTS

The goal is to speed up checkpointing by introducing a double buffer combined with a prefetching mechanism and adding compression to the data stored in GPU. The Revolve algorithm [3] is specifically designed to address this type of problem. Revolve is a classic checkpoint implementation that optimally solves the problem of determining which checkpoints should be stored to get the best trade-off between memory consumption and re-computation. It is used as a controller for the application, commanding it to perform specific actions at each timestep.

Our main insight is that Revolve is deterministic, and its computational cost is negligible if compared to the main application. So, we designed a mechanism that runs Revolve before the main solver, recording its command actions and identifying situations when the desired data would not be in the local buffer. For those cases, if possible, we configure a prefetch operation, dispatching a load from host memory a few timesteps before the data is needed, so it will be in the local buffer when Revolve asks for restoring it during the main loop execution. This mechanism can anticipate, on average, 75 % of the host-to-device memory transfers in our test cases.

On top of prefetching, we added data compression to the GPU buffer. We integrated the cuZFP[5] compression library into our application, so every transfer between host and device uses only compressed data. This strategy reduced the amount of data transferred through PCI-e up to 16x and brought speedups from 1.9x to 2.5x. Table 1 summarizes quality metrics for images generated using compressed and uncompressed data. Two sample images from the Marmousi dataset are displayed in Figure 1.

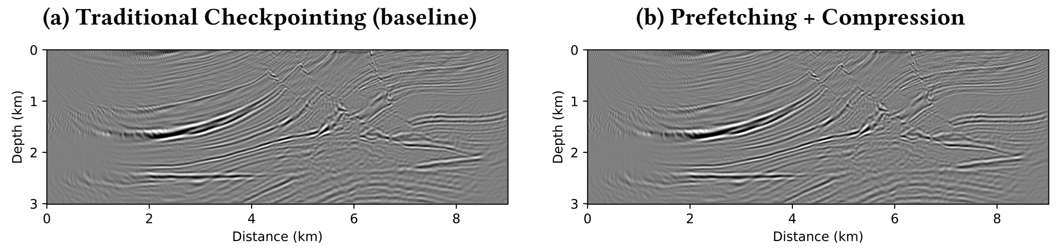

Fig. 1. A comparison of the migrated image for the Marmousi dataset with our baseline implementation (a), and prefetching + compression (b).

Dataset	ABS	PSNR
Salt	7.73	74.67
Marmousi	18.68	87.54
Large	3.99	55.97

Table 1. Compression Quality Evaluation for our Datasets.

REFERENCES

[1] E. Baysal, D. D. Kosloff, and J. W.C. Sherwood. 1983. Reverse time migration. *Geophysics* 48, 11 (1983), 1514–1524. https://doi.org/10.1190/1.1441434

[2] M Dmitriev, T Tonellot, HJ AlSalem, and S Di. 2022. Error-Bounded Lossy Compression in Reverse Time Migration. In *Sixth EAGE High Performance Computing Workshop*, Vol. 2022. EAGE Publications BV, 1–5.

[3] Andreas Griewank and Andrea Walther. 2000. Algorithm 799: Revolve: An Implementation of Checkpointing for the Reverse or Adjoint Mode of Computational Differentiation. *ACM Trans. Math. Softw.* 26, 1 (mar 2000), 19–45. https://doi.org/10.1145/347837.347846

[4] Navjot Kukreja, Jan Hückelheim, Mathias Louboutin, Paul Hovland, and Gerard Gorman. 2019. Combining Checkpointing and Data Compression to Accelerate Adjoint-Based Optimization Problems. In *Euro-Par 2019: Parallel Processing*, Ramin Yahyapour (Ed.). Springer International Publishing, Cham, 87–100.

[5] Peter Lindstrom. 2014. Fixed-Rate Compressed Floating-Point Arrays. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2674–2683. https://doi.org/10.1109/TVCG.2014.2346458

[6] Jingcheng Shen, Linbo Long, Xin Deng, Masao Okita, and Fumihiko Ino. 2023. A compression-based memory-efficient optimization for out-of-core GPU stencil computation. *The Journal of Supercomputing* (20 Feb 2023). https://doi.org/10.1007/s11227-023-05103-8

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009