Reverse Time Migration with Lossy and Lossless Wavefield Compression

Carlos HS Barbosa
Department of Civil Engineering
Federal University of Rio de Janeiro (UFRJ)
Rio de Janeiro, Brazil
c.barbosa@nacad.ufrj.br

Alvaro LGA Coutinho
Department of Civil Engineering
Federal University of Rio de Janeiro (UFRJ)
Rio de Janeiro, Brazil
alvaro@nacad.ufrj.br

***Abstract*—Seismic imaging techniques like Reverse Time Migration (RTM) are time-consuming and data-intensive activities in the field of geophysical exploration. The computational cost associated with the stability and dispersion conditions in the discrete two-way wave equation makes RTM time-consuming. Additionally, RTM is data-intensive due to the need to manage a considerable amount of information, such as the forward propagated wavefields (source wavefield), to build the final migrated seismic image according to an imaging condition. In this context, we introduce lossy and lossless wavefield compression for parallel multi-core and GPU-based RTM to alleviate the data transfer between processor and disk. We use OpenACC for enabling GPU parallelism and the ZFP library aligned to decimation based on the Nyquist sampling theorem to reduce storage. We study experimentally the effects of wavefield compression for both GPU-based and optimized OpenMP+vectorization RTM versions. Multi-core and GPU-based RTM have been linked to the ZFP library to compress the source wavefield on-the-fly once it has been decimated according to the Nyquist sampling theorem to calculate the imaging condition. This approach can reduce drastically the persistent storage required by the technique. However, it is essential to understand the impact of using compressed wavefields on the migration process that builds the seismic image. In this context, we show how much storage can be reduced without compromising the seismic image's accuracy and quality.**

***Index Terms*—Seismic imaging, High-performance Computing, Reverse Time Migration, OpenMP/OpenACC, Compression**

I. Introduction

Reverse Time Migration (RTM) is a depth seismic migration technique that provides a reliable high-resolution representation of the Earth's subsurface useful for exploring hydrocarbons [10]. The classical RTM is based on the two-way wave equation and an imaging condition [21]. Two major problems related to its algorithm developments are defining strategies to deal with the huge amount of information demanded by the forward-propagated (source) wavefield solution and providing efficient and portable codes for the main available platforms in the academy and industry. The source wavefield must be accessed backward in time during the backward-propagated (receiver) wavefield calculation to build the imaging condition. Such requirement forces the source wavefield storage, which is constantly exchanged between CPU/GPU and memory, to update the solution over space and time and calculate the imaging condition. Moreover, solving the governing wave equations of RTM by numerical methods is computationally intensive due to the number of operations to update the grid points. For instance, using the FDM to discretize the two-way isotropic acoustic wave equation demands a different number of operations per grid point, depending on the discretization order.

RTM is classified as a time-consuming and data-intensive technique [24], [26], [30], [31]. From a time-consuming viewpoint, RTM algorithms need to be developed to take full advantage of all computer hardware technologies, such as Central Processing Units (CPUs) equipped with multi-cores [31]–[33], graphic processing units (GPUs) [24], [26], [30], [32], and vector processors (VPs) [34]–[37] machines. On the other hand, strategies to diminish input/output (I/O) related to the source wavefield revolve mainly around full/partial reconstruction techniques [16]–[19], [22], [31], [38]. Methods for storing the source wavefield can take advantage of Nyquist sampling theorem to decimate the temporal evolution of the wavefield propagation [16], [18], [22]. This process can be considered the simplest way to compress the source wavefield [22], [39], [53]. Lossless/lossy compressors linked or not to decimation techniques can also reduce even more the storage demand. On top of that, careful use of lossy compressors does not hamper the final seismic images [22], [31], [52].

Most scientific applications that use compression methods, such as climate, cosmology, molecular dynamics, and fluid dynamics simulations, apply them for further analyses like data visualization [40]–[42]. Moreover, it is common to compress scientific data to accelerate inter-GPU communications and transmission between the CPU and GPU for collaborative workflows [41], [52]. In the context of geophysical applications like RTM, compressing wavefields imposes significant challenges due to the nature of the problem. The wavefield simulation solution must be compressed and decompressed on-the-fly to support the imaging condition calculation. Additionally, the compressed wavefield must be stored and retrieved from disk as the information volume can reach tens of petabytes for large-scale scenarios. This inherent characteristic of the RTM problem increases the computational complexity and presents coding challenges in managing wavefield storage.

In this context, we introduce RTM algorithms that consider the lossy and lossless compression of the source wavefield solution, which is also beneficial for the emerging heterogeneous machines based on multi-CPU/GPUs. We describe and analyze RTM algorithms linked to decimation and compression methods. In this sense, before storing the source wavefield, RTM compresses it on-the-fly after its decimation for further calculation of the imaging condition. The strategy reduces the persistent storage required by the method. Besides, we analyze how much is possible to reduce the source wavefield storage and the quality of the seismic images in terms of the difference to a reference solution. We also evaluate the computational performance of the RTM algorithms linked to compression methods for heterogeneous CPU/GPU machines in terms of execution time, storage demand, and overhead. We noted that in situations with high overheads due to the time spent compressing the source wavefield if we choose an adequate platform and a proper compression rate, it is possible to reduce execution time and storage demand.

II. Related Work

A. GPU implementations for RTM

The literature for RTM-based GPU implementation is vast [24], [26], [30], [32], [33], [43]–[46], [50]. Most of the work is based on CUDA implementations. Some of them focus on the OpenACC library, looking for portability. For instance, [24] implemented the seismic modeling and RTM on single and multi-GPUs using a hybrid MPI+OpenACC approach which aims to develop portable high-level directive-based codes across heterogeneous platforms for seismic imaging applications. Their main contributions show that seismic applications can benefit from improvements in the OpenACC programming models by controlling the GPU's shared memory and overlapping MPI communications with GPU computations to improve computational performance.

Serpa et al. [26] evaluated three different computational optimizations based on multicore and GPU architectures and investigated their codes' performance, energy efficiency, and portability. The results show that computational implementations for GPUs architectures are more efficient than multicore ones even the energy consumption been higher. However, it exists a trade-off between performance, portability and energy efficiency that should be consider on CUDA and OpenACC based implementations. OpenACC programming is easier than CUDA, however the authors argument that the user has to define how much loss of performance is acceptable for portable codes.

B. Data compression and bufferization to mitigate I/O in geophysical applications

Storage demand is also crucial for GPU-based RTM. In this context, Liu et al. [43] propose RTM with RBC to diminish migration algorithms' storage demand, showing that such a strategy benefits GPU implementations. The GPU computational implementation was coded in CUDA and tested for 2-D RTM applications. Sun and Fu [22] took advantage of the Nyquist sampling theory and a compression technique to store the wavefield at the Nyquist time step to reduce the amount of information on their application. Kukreja et al. [47] proposed a new approach that combines checkpointing methods with error-bounded lossy compression for the Full Waveform Inversion (FWI). The checkpointing methodology decimates wavefield propagation for further compression using the ZFP library. The authors concluded that their combined checkpointing+compression methodology reduces data movement, execution time, and peak memory. Huang et al. [54] developed a hybrid lossy compression method (called HyZ) and evaluated the compression ratio as well as the compression/decompression speed for several lossy compression algorithms (including HyZ, BR, SZx, SZ, SZ-Interp, ZFP, etc.) within an OpenMP-based RTM framework. The authors only compared the image qualities of a single RTM snapshot and the stacked image for the BR and HyZ compressors. Furthermore, they also concluded that HyZ improves the overall RTM performance from $6.29\times$ to $6.60\times$ over their original workflow.

Finally, Alturkestani et al. [50] present a non-intrusive Multilayer Buffer System (MLBS) framework that aims to maximize RTM I/O bandwidth in the presence of GPU hardware accelerators. According to the authors, the MLBS hides the RTM's I/O overhead by asynchronously creating opportunities for overlapping data motion from GPU to disk and vice-versa (wavefield I/O) with computations of the wave equation stencil [51].

III. Contributions

Compressing the wavefield has been effectively employed in geophysical applications, such as RTM and FWI [22], [43], [47], [53], [54]. Specifically, the ZFP methodology remains insufficiently explored in terms of image quality after compression/decompression of the wavefield and computational performance of the on-the-fly wavefield compression across different platforms, including CPUs and GPUs. In this context, this paper makes the following contributions:

- Introduction of on-the-fly lossy and lossless compression methods to reduce the source wavefield storage of RTM algorithms. The lossy and lossless compressors are based on the decimation method and the ZFP methodology[1].
- Providing a detailed algorithm description that can be implemented on heterogeneous machines with CPU and GPU hardware.
- Experimental studies of the effects of compressing the source wavefield solution on the migration process that builds the seismic image by applying the imaging condition.
- Evaluation of the computational performance of the RTM algorithms linked to the compression methods in terms of execution time, storage demand, and overhead. These numerical experiments consider implementations in supercomputers equipped with CPU and GPU hardware.

[1]https://computing.llnl.gov/projects/zfp

IV. Reverse Time Migration

Reverse Time Migration (RTM) is a depth migration technique based on the two-way wave equation and an imaging condition [10], [20], [21]. Building the imaging condition requires solving the wave equation twice. The first solution is described (for an acoustic medium) by the second-order partial differential equation:

$$\nabla^2 p(\mathbf{r}, t) - \frac{1}{v^2(\mathbf{r})} \frac{\partial^2 p(\mathbf{r}, t)}{\partial t^2} = f(t)\, \delta(\mathbf{r} - \mathbf{r}_s), \tag{1}$$

where p is the pressure (also referred as source wavefield), v the compressional wave velocity, $\mathbf{r}$ the spatial coordinates, t the time in $[0, T]$, $f(t)$ is the seismic source at the position $\mathbf{r} = \mathbf{r}_s$, and δ is the Dirac delta function. The pressure p is defined in a domain $\Omega \subset \mathbb{R}^{n_{sd}}$, $n_{sd} = 2, 3$. The second-order differential equation (1) needs initial and boundary conditions. A natural initial condition is to define $p(\mathbf{r}, 0) = \partial p(\mathbf{r}, 0)/\partial t = 0$ for $\mathbf{r} \in \Omega$. Lastly, we set $p(\mathbf{r}, t) = 0$ on $\partial\Omega \in \mathbb{R}^{n_{sd}}$, where $\partial\Omega$ is the domain boundary.

The second solution is obtained by solving the following equation:

$$\nabla^2 \bar{p}(\mathbf{r}, \tau) - \frac{1}{v^2(\mathbf{r})} \frac{\partial^2 \bar{p}(\mathbf{r}, \tau)}{\partial \tau^2} = s(\mathbf{r}, \tau)\, \delta(\mathbf{r} - \mathbf{r}_r), \tag{2}$$

where $\bar{p}$ is the receiver wavefield, $s(\mathbf{r}, \tau)$ is the seismogram recorded at the receivers positions $\mathbf{r} = \mathbf{r}_r$, and $\tau = T - t$ is the reversal time evolution [11], where $\tau \in [0, T]$. $\bar{p}$ is also defined in $\Omega \subset \mathbb{R}^{n_{sd}}$, and corresponding initial, and boundary conditions should be set.

Solving (1) and (2) in finite domains generates artificial reflections coming from the boundaries. Normally, it is common to use additional equations to eliminate such artificial events. Among the several options in the literature, the Convolutional Perfectly Matched Layer (CPML) [13], [14] and the damping factors for plane waves introduced by [15] are the most used. Although unusual in wave propagation simulation studies, the RBCs, first introduced by [16], can also be employed in seismic imaging methods based on the two-way wave equation, such as RTM and FWI [16]–[19].

Once we have the source and receiver wavefields, the imaging condition can be calculated as:

$$I(\mathbf{r}) = \frac{\int_0^T p(\mathbf{r}, t)\ \bar{p}(\mathbf{r}, \tau)\ dt}{\int_0^T [p(\mathbf{r}, t)]^2\, dt}, \tag{3}$$

where $I(\mathbf{r})$ is called source-normalized convolutional imaging condition [12]. The seismic imaging from (3) has the same unit, scaling, and sign of the reflection coefficient [10].

V. Data Compression

Data compression is an efficient strategy to reduce persistent storage in seismic migration algorithms like RTM. Compressing the data consists of converting an input data stream into another one that has a smaller size, where the stream can be a file on disk or a buffer on memory. There are two classes of methods for data compression, which are the lossy and lossless algorithms. If the numerical values are only approximately represented after they have been decompressed, the compression is called lossy. Otherwise, if the values are represented exactly as the original data, it is called lossless compression [7], [8]. There are several lossy and lossless compressors such as Gzip [1], DEFLATE algorithm [3], ZLIB [4], SZ library [5], MGARD library [6], TuckerMPI [2], ZFP library [8] among others.

We have chosen the ZFP compressor, which is an open-source C/C++ library for compressing integer and floating-point data of multidimensional numerical arrays. The ZFP compressor [7] operates on $d-$dimensional arrays ($d \in [1, 4]$) by partitioning them into blocks of 4^d values. The blocks are independently compressed/decompressed from each other. Because of that, ZPF is recognized as one of the most effective high-speed lossy floating-point compressors. To achieve high compression rates, ZFP uses lossy but optionally error-bounded compression controlled by four different parameters known as expert, fixed-rate, fixed-precision, and fixed-accuracy modes. On the other hand, although bit-for-bit lossless compression of floating-point data is not always possible, ZFP also offers an accurate near-lossless compression. For that, we can use the reversible mode [8], [9].

VI. Computational Implementation

The numerical discretization used to solve the acoustic wave equations (1) and (2) are based on the non-staggered grid scheme (NSG) [23]. The NSG is the most simple and popular finite-difference numerical scheme used to discretize Partial Differential Equations (PDEs) where each derivative can be approximated using numerical operators obtained by Taylor series expansions [23]. Applying a finite-difference scheme to (1), that is 8th-order in space and 2nd-order in time, produces the following discretization:

$$\begin{aligned} p_{i,j,k}^{n_t+1} = C_{i,j,k}[2b_0 + & \\ \sum_{m=1}^{4} b_m (p_{i+m,j,k}^{n_t} + p_{i-m,j,k}^{n_t} + p_{i,j+m,k}^{n_t} & \\ + p_{i,j-m,k}^{n_t} + p_{i,j,k+m}^{n_t} + p_{i,j,k-m}^{n_t})] & \\ - 2p_{i,j,k}^{n_t} + p_{i,j,k}^{n_t-1} - f_{i,j,k}^{n_t}. & \end{aligned} \tag{4}$$

where the indexes i, j and k represent the directions (x, y, z), n_t is the temporal step, and p is the pressure. The matrix $C_{i,j,k} = (v_{i,j,k} \Delta t/h)^2$, where $v_{i,j,k}$ is the velocity, and $h = \Delta x = \Delta y = \Delta z$ is the grid size. Lastly, b_m are the finite difference coefficients. Similar discretization can be obtained for (2).

Discretizing the imaging condition (3) yields:

$$I_{i,j,k} = \frac{\sum_{n_t=0}^{N_t-1} \bar{p}_{i,j,k}^{n_\tau} \cdot p_{i,j,k}^{n_t}}{\sum_{n_t=0}^{N_t-1} p_{i,j,k}^{n_t} \cdot p_{i,j,k}^{n_t}}, \tag{5}$$

where $\cdot$ and $\div$ represent the component-wise multiplication (also known as Hadamard product) and division operations for the (i, j, k)-th component of a vector, and $n_\tau = N_t - n_t$ represents the discrete reversal time.

Therefore, the FDM discretization of the acoustic wave equation leads to the discrete versions of the velocity field, source wavefield, receiver wavefield, imaging condition, and seismic source represented by the vectors $\mathbf{v}$, $\mathbf{p}$, $\bar{\mathbf{p}}$, $\mathbf{I}$, and $\mathbf{f}$, respectively. For the 3D case, the vectors $\mathbf{v}$, $\mathbf{I}$, $\mathbf{p}$, and $\bar{\mathbf{p}}$ have the dimension $N = N_x \times N_y \times N_z$, where N_x, N_y and N_z are the number of grid points in each Cartesian direction. Lastly, the seismic source $\mathbf{f}$ has dimension N_t for each shot, where $N_t = T/\Delta t$.

Algorithm 1 shows computational implementation of RTM which compress the source wavefield to deal with persistent storage. RTM algorithm needs as inputs a velocity field, a seismic source, and a set of seismograms, $\{\mathbf{s}_1, \cdots, \mathbf{s}_{N_{shots}}\}$ that contains information about the medium reflectivity. The computation of the imaging condition uses, among others, the source, and receiver wavefield solutions to build the migrated seismic section that stacks the partial results over time $\left(\mathbf{I}^k_{\sum n_\tau}\right)$, and over the number of seismograms $\left(\mathbf{I}^k_{\sum shot\_id}\right)$. The sum in line 21 approximates the time integral of the seismic imaging (3).

Algorithm 1 Reverse Time Migration with Wavefield Compression

```
Require: $\mathbf{v}$, $\{\mathbf{s}_1, \cdots, \mathbf{s}_{N_{shots}}\}$, and $\mathbf{f}$
 1: function RTM( vector $\mathbf{v}$, vectors $\{\mathbf{s}_1, \cdots, \mathbf{s}_{N_{shots}}\}$, vector $\mathbf{f}$ )
 2:     read $\mathbf{v}$, $\mathbf{f}$, and $\{\mathbf{s}_1, \cdots, \mathbf{s}_{N_{shots}}\}$
 3:     initialize image condition $\mathbf{I}_{\sum shot\_id} = 0$
 4:     for $shot\_id = 1$ to $N_{shots}$ do
 5:         initialize $n_t = 0$
 6:         apply initial conditions for $i_t = 0$
 7:         for $i_t = 1$ to $N_t$ do
 8:             $n_t = i_t * \Delta t$
 9:             solve equation (1)                    ▷ source wavefield
10:             compress $\mathbf{p}_{n_t}$ for $n_t \mid mod(n_t, \Delta t_{nyq}) = 0$
11:             store the compressed $\mathbf{p}_{n_t}$
12:         end for
13:         initialize $n_\tau = 0$, and $\mathbf{I}_{\sum \tau} = 0$
14:         apply initial conditions for $i_\tau = 0$
15:         for $i_\tau = 1$ to $N_t$ do
16:             $n_\tau = (N_t - i_\tau) * \Delta\tau$           ▷ reversal time
17:             read $\mathbf{s}_{shot\_id}$
18:             read compressed $\mathbf{p}_{n_\tau}$ for $n_\tau$
19:             decompress $\mathbf{p}_{n_\tau}$
20:             solve equation (2)                  ▷ receiver wavefield
21:             $\mathbf{I}_{\sum n_\tau} = \mathbf{I}_{\sum n_\tau} + (\mathbf{p}_{n_\tau} \cdot \bar{\mathbf{p}}_{n_\tau}) \,/\, (\mathbf{p}_{n_\tau} \cdot \mathbf{p}_{n_\tau})$
22:         end for
23:         stack $\mathbf{I}_{\sum shot\_id} = \mathbf{I}_{\sum shot\_id} + \mathbf{I}_{\sum n_\tau}$
24:     end for
25:     $\mathbf{I} \leftarrow \mathbf{I}_{\sum shot\_id}$
26:     store $\mathbf{I}$
27: end function
```

We employ the ZFP fixed-accuracy mode to compress the 4-D numerical array that stores the source wavefield $\mathbf{p}_{n_t}$. According to ZFP documentation[2], the fixed-accuracy mode guarantees that each reconstructed value falls within a user-specified absolute error tolerance, which usually results in the smallest RMS error and therefore highest peak signal to noise ratio for the same rate [29]. The ZFP library can be linked to the algorithm as a static/dynamic link for C/C++ language implementation, for instance. Thus, we modify the RTM algorithm aiming to compress/decompress the source wavefield (lines 10, 11, 18, and 19). Besides, instead of storing the source wavefield for every time step (Δt) based on the FDM, we took advantage of the Nyquist theory as explored by [22] to store the wavefield at the Nyquist time step to reduce the amount of information. The Nyquist time step Δt_{nyq} is defined as,

$$\Delta t_{nyq} = \frac{1}{2(f_{max} - f_{min})}, \tag{6}$$

where f_{max} and f_{min} are the highest and lowest frequency of the seismic source.

Therefore, lines 10 and 11 in Algorithm 1 show the compression of the multidimensional array $\mathbf{p}$ for the Nyquist time step Δt_{nyq}, and its storage on disk. Lines 18 and 19 describe the reading of $\mathbf{p}$ also for Δt_{nyq} and its decompression.

A. OpenMP Multi-core Parallelism

We follow [12], [31] who presented an RTM algorithm that explores the Message Passing Interface (MPI) library and the OpenMP+vectorization to deal with parallelism in hybrid multi-core machines. In this sense, our multi-core implementation is written in C programming language, and it takes advantage of OpenMP directives to explore multiple cores/threads parallelism and the single-instruction-multiple-data (SIMD) model to deal with the data-level parallelism. Thus, the SSE or AVX instructions (for Intel compilers) activate vectorization depending on where we compile the applications.

As we can see in Algorithm 1, the RTM is composed of two parts. The first one calculates the source wavefield, and the second part calculates the receiver wavefield. The RTM assesses the source and receiver wavefields by solving two different wave equations. Thus, the multi-core/thread parallelism and vectorization can be applied to the second-order wave equations that are discretized with a second-order finite difference scheme in time and an eighth-order scheme in space (equation (4)). We use the OpenMP directives `pragma omp parallel for` and `pragma omp simd` to parallelize the nested loops of the forward wavefield propagation, backward wavefield propagation, and seismogram insertion. Reversal time implementation of the RTM explores the technique proposed by [11]. This technique guarantees the coercivity of the adjoint problem. The reversal time wavefield propagation is calculated after the forward wavefield propagation by inserting the seismic signals of the seismogram. The insertion of the seismogram uses the variable changing $\tau = T - t$.

[2]https://zfp.readthedocs.io/en/release0.5.5/modes.html#fixed-accuracy-mode

B. OpenACC GPU parallelism

The GPU programming model, based on OpenACC[3] directives, aims to provide an easier way for scientific applications coding [24], [25]. Besides, compared to CUDA, OpenACC programming demands less coding efforts in heterogeneous environments with CPU+GPU [24], [26]. The OpenACC implementation needs to deal with three main issues: CPU (host) calculations, GPU calculations, and communications to and from the GPU. Thus, any computational implementation must maximize the GPU computations and prevent communications between the host and GPU.

Algorithm 2 details the OpenACC implementation for the RTM which implements the compression of wavefield (Algorithm 1). We follow the OpenACC-based implementation strategies presented in [30] to accommodate the compression based on decimation and the ZFP library. Thus, the three different colors in Algorithm 2 stand for host computations (green), data transfer (blue), and GPU calculations (red). The first part moves the compressed source wavefield during its calculation from the GPU to the host and stores it in disk (lines 6 to 9). In general, storing the source wavefield in a disk is needed because the GPU memory or RAM is insufficient to store it, even after its compression. The second part of the RTM algorithm (lines 14 to 19) moves back the source wavefield from the host to GPU and decompressed it, calculates the receiver wavefield, and builds the imaging condition. To calculate the receiver wavefield, line 12 moves the seismograms from the disk to the GPU.

Algorithm 2 requires two data transfers for the velocity field and seismic source, N_{shots} data transfers for the seismograms, $2 \times N_t$ data transfers for the source wavefield, and N_{shots} data transfers for the seismic images. Thus, we use the `copyin` directive for copying the velocity field, seismic source, and seismograms from host to GPU, `copyout` directive for copying the seismic images from GPU to host, and `create` directive to allocate on GPU the vectors for the wavefield solution. The `kernels loop` directive creates the parallel region and parallelizes the nested loops. Lastly, we use the `update self` directive to move the source wavefield from the GPU and store it in the disk, and `update device` to read the source wavefield from the disk and move it to the GPU.

Algorithm 2 RTM GPU Implementation with Wavefield Compression 1

```
Require: v, f, and $\{\mathbf{s}_1, \cdots, \mathbf{s}_{N_{shots}}\}$
 1: function RTM_GPU( vector v, vectors $\{\mathbf{s}_1, \cdots, \mathbf{s}_{N_{shots}}\}$, vector f )
 2:     allocate data variables
 3:     read v, and f
 4:     Move data to GPU
 5:     for time = first to last do
 6:         solve wave equation (1)
 7:         compress source wavefield
 8:         move source wavefield to host
 9:         store source wavefield for each time
10:     end for
11:     read seismograms
12:     Move seismogram to GPU
13:     for time = first to last do
14:         set reversal time evolution
15:         read source wavefield
16:         Move source wavefield to GPU
17:         decompress source wavefield
18:         solve wave equation (2)
19:         calculate imaging condition
20:     end for
21:     update host with seismic image
22:     store seismic image
23:     deallocate all the data variables
24: end function
```

VII. Numerical Experiments

In this subsection, we present the performance analysis of the RTM using two different computational platforms. We analyze the execution times, overhead, storage demand, and impact of compressing the wavefields on the RTM solutions. The CPU and GPU clusters are machines from the Santos Dumont system at the National Scientific Computing Laboratory at Petrópolis/Brazil [4]. The CPU cluster has $2\times$ Intel Xeon E5-2695v2 Ivy Bridge processors with 2.4GHZ and 24 cores per node. On the other hand, the CPU-GPU cluster has a $2\times$ Intel Xeon Skylake 6252, 2.4GHZ with 48 cores and $4 \times$ NVIDIA V100 per node.

[3]https://www.openacc.org/

[4]https://sdumont.lncc.br/support_manual.php?pg=support

A. Effects of compressing the wavefield on the seismic image

To illustrate RTM results considering the source wavefield compression, we choose the 2D Marmousi benchmark, which is 3.0 km in depth and 9.2 km in the horizontal direction (Figure 1). We choose to synthesize the observed seismograms solving the two-way wave equation with the reference velocity field provided by the benchmark. Therefore, we simulate a fixed split-spread acquisition [27], where the seismic source is a Ricker-type with a cutoff frequency of 45 Hz, is placed on the surface and moved 100 meters for each shot. Near-surface hydrophones record the seismic signals that compose the seismograms, and the receivers are equally spaced of 12.5 meters. Thus, the dataset for seismic migration is composed of 82 seismograms.

Figure 2(a) shows the resulting seismic image without using compression that we define as the reference outcome. Figure 2(b), (c), and (d) show the migrated seismic images built by RTM linked to the compression library. These results apply the lossless compression, the lossy tolerance compression of 10^{-6}, and the lossy tolerance compression of 10^{-4} to source wavefields. Observe that the aggressive lossy tolerance compression of 10^{-4} damages the final seismic image (Figure

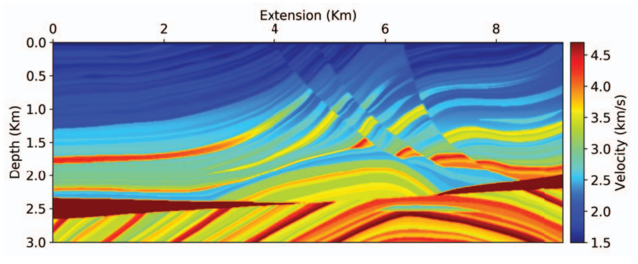

Fig. 1. 2D velocity field from the Marmousi benchmark.

2(d)). However, we do not note any damage on the seismic images in Figures 2(b), and (c).

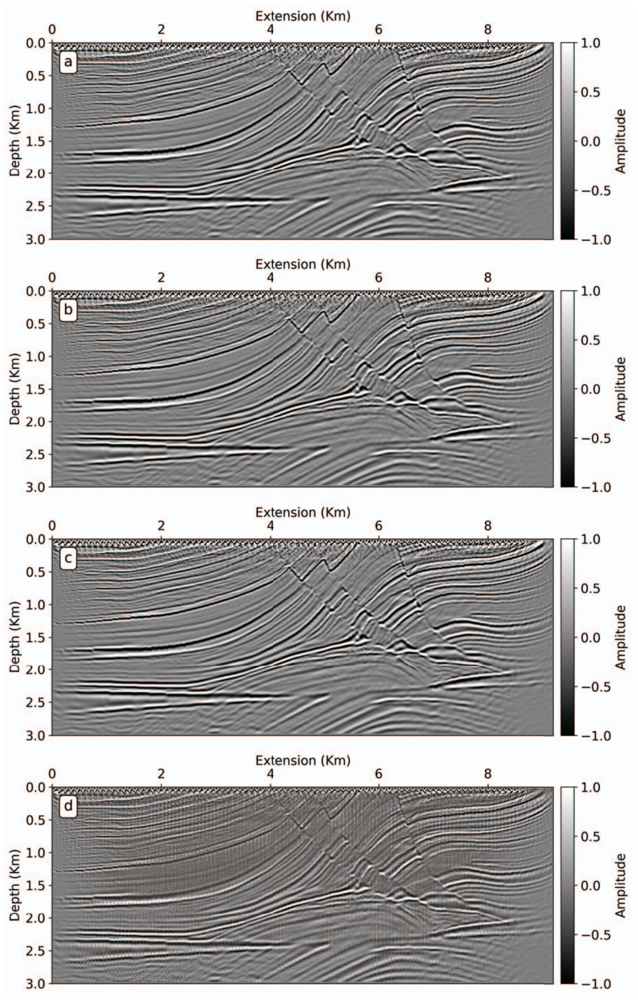

Fig. 2. RTM outcomes after migration using the forward-propagated source wavefield (a) without compression, (b) with lossless compression, (c) with a lossy tolerance compression of 10^{-6}, and (c) with a lossy tolerance compression of 10^{-4}.

The errors produced by the difference between the reference image and seismic images built using compression can be seen in Figure 3. We use the following component-wise difference $I^{diff}_{i,j,k}$ given by:

$$I^{diff}_{i,j,k} = I^{ref}_{i,j,k} - I^{c}_{i,j,k} \tag{7}$$

where $I^{ref}_{i,j,k}$ is the reference seismic image without compressing the source wavefield and $I^{c}_{i,j,k}$ the seismic image that takes into account the wavefield compression (decimation, lossless, and lossy accuracy). Notice that $I^{ref}_{i,j,k}$ and $I^{c}_{i,j,k}$ are related to the imaging condition from (5).

Figure 3(a) shows that lossless compression of the source wavefield provides the best result. In this case, the error is of order 10^{-6}. On the other hand, a lossy tolerance compression of 10^{-4} of the source wavefield produces the most inferior outcome. Lastly, a lossy tolerance compression of 10^{-6} suggests that lossy compression can also be an option to deal with the storage of source wavefields. In this case, the error is of order 10^{-3}.

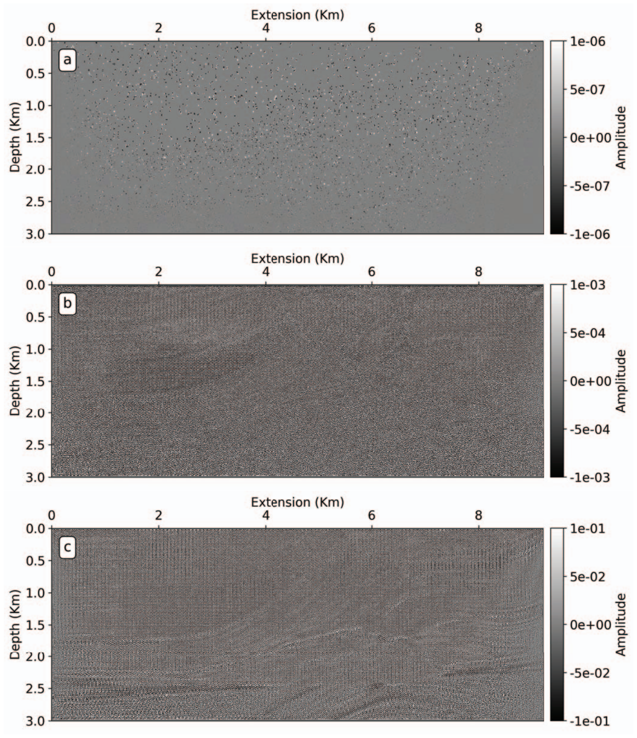

Fig. 3. Component-wise difference among the reference migrated image (no compression) and (a) the migrated image after applying lossless compression on the source wavefield, (b) the migrated image after applying a lossy tolerance compression of 10^{-6} on the source wavefield, and the migrated image after applying a lossy tolerance compression of 10^{-4} on the source wavefield.

a) Wavefield storage demand for the 2D Marmousi benchmark: The total demand to store the source wavefield in disk without compression is 1.8 GB per shot, and after applying compression is 1.1 GB for lossless compression, 92.9 MB for the lossy tolerance compression of 10^{-6}, and 15.2 MB for the lossy tolerance compression of 10^{-4}. These values represent compression ratios of 1.64, 18.84, and 121.26, respectively. Figure 4 shows the storage demand per time slice for the source wavefield without compression (blue line), with lossless compression (orange line), with a lossy tolerance compression of 10^{-6} (red line), and with a lossy tolerance compression of 10^{-4} (green line). We see that the size varies for each time step when we use data compression. This behavior occurs because amplitude information changes during wavefield propagation.

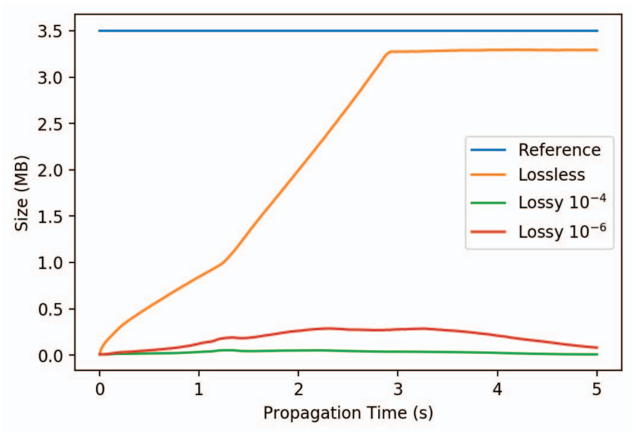

Fig. 4. Size per time slice for the 2D source wavefield without compression (blue line), with lossless compression (orange line), with a lossy tolerance compression of 10^{-4} (green line), and with a lossy tolerance compression of 10^{-6} (red line).

B. Performance analysis for 3D domains: HPC4E benchmark

The 2D RTM results shown in Figures 2 and 3 suggest that compression can be a good strategy to reduce the storage demand concerning the source wavefield. Thus, we discuss in this section the storage demand and overhead of storing the source wavefield for a 3D application. For this, we have chosen the MODEL AF provided by the High Performance Computing for Energy (HPC4E) Seismic Test Suite [5] [28]. The MODEL AF is a model designed as a set of 15 layers with constant velocity values and flat topography (Figure 5). Besides, the velocity parameter model (velocity field) covers an area of $10 \times 10 \times 4.5$ km. Again, we synthesized the observed seismograms by solving the two-way wave equation with the reference velocity field. The seismic source is a Ricker-type with a cutoff frequency of 20 Hz placed near the surface at location $[5000.0, 5000.0, 25.0]$ meters. Near-surface hydrophones record the seismic signals that compose the seismograms, and the receiver geometry follows the expressions:

$$r_x = 25.0(i-1) + 1012.5 \text{ with } i = 1, \cdots, 320, \quad (8)$$
$$r_y = 25.0(j-1) + 1012.5 \text{ with } j = 1, \cdots, 320, \quad (9)$$

where the pair $[r_x, r_y]$ meters represents the receiver locations near the surface.

Figure 6 shows the storage demand in disk per time slice for the source wavefield without compression (blue line), with

[5]https://hpc4e.bsc.es/downloads/hpc-geophysical-simulation-test-suite

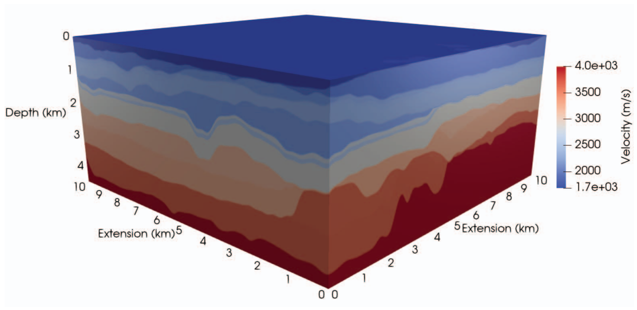

Fig. 5. 3D velocity field provided by the HPC4E Seismic Test Suite.

lossless compression (orange line), with a lossy tolerance compression of 10^{-6} (green line), and with a lossy tolerance compression of 10^{-4} (red line). Note that size changes for each time step when we apply the data compression similar to the 2D experiment from Figure 4. Again, this behavior occurs because amplitude information changes during the wavefield propagation. On the other hand, the total demand to store the source wavefield in disk for all time steps without compression is 65.16 GB per shot, and after applying compression is 49.82 GB for lossless compression, 31.34 GB for the lossy tolerance compression of 10^{-6}, and 21.68 GB for the lossy tolerance compression of 10^{-4}. These values represent compression ratios of 1.31, 2.08, and 3.06, respectively. But, remember that an aggressive compression damages the final RTM result, as we illustrate in Figures 2(d), and 3(c) for the 2D Marmousi benchmark application.

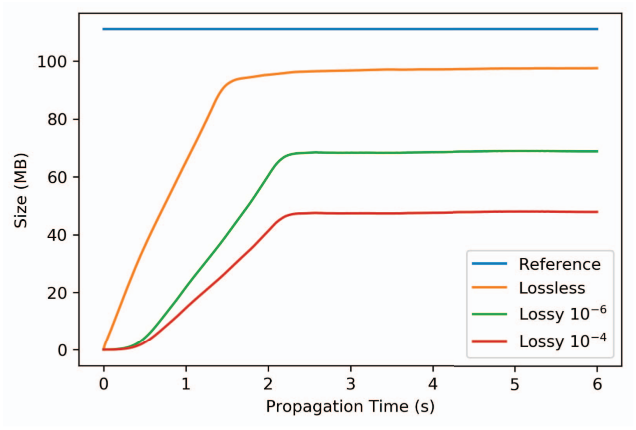

Fig. 6. Size per time slice for the 3D source wavefield without compression (blue line), with lossless compression (orange line), with a lossy tolerance compression of 10^{-4} (green line), and with a lossy tolerance compression of 10^{-6} (red line).

Table I shows the execution time and overhead measurements for the 3D RTM considering the source wavefield compression. Notice that we apply two error tolerances for the lossy compression, which are 10^{-6} and 10^{-4}. Thus, the tests in this section consist of running the the 3D RTM implementations based on Algorithm 1 on the CPU (Intel

Xeon Ivy Bridge) and GPU (NVIDIA V100) platforms from Santos Dumont cluster. The CPU-based implementations for the RTM algorithms consider the 24 cores on the running tests. The execution time and overhead values shown in Table I suggest that the overhead for the source wavefield compression and decompression is high. For instance, the execution time increased 203%, 147%, and 122% when we consider the wavefield compression with the lossless tolerance, lossy error tolerance of 10^{-6}, and lossy error tolerance of 10^{-4} on the CPU platform. Considering the GPU platform, the execution time increased 381%, 245%, and 179% when the wavefield compression is applied with the lossless tolerance, lossy error tolerance of 10^{-6}, and lossy error tolerance of 10^{-4}.

TABLE I
TIME REQUIREMENTS AND OVERHEAD FOR THE 3D RTM IMPLEMENTATION CONSIDERING DIFFERENT COMPRESSION RATES FOR THE SOURCE WAVEFIELD. THE MEASUREMENTS TAKE INTO ACCOUNT TEN EXECUTIONS OF EACH IMPLEMENTATION.

Platform	Method	Time (s)[Var. (s)]	Overhead (s)[Var. (s)]
CPU	None	1543.670 [4.108]	-
CPU	Lossless	3130.821 [387.605]	1587.151 [389.555]
CPU	Lossy 10^{-6}	2261.357 [556.679]	717.687 [561.601]
CPU	Lossy 10^{-4}	1878.574 [523.458]	334.904 [520.352]
V100	None	256.166 [46.810]	-
V100	Lossless	1232.587 [2.910]	976.421 [3.191]
V100	Lossy 10^{-6}	882.737 [0.992]	626.571 [0.841]
V100	Lossy 10^{-4}	713.542 [10.135]	457.376 [8.472]

a) Migrated seismic image for the 3D HPC4E benchmark: Figure 7 shows the stacked seismic image for the 3D HPC4E Seismic Test Suite benchmark. As mentioned earlier, we generated the observed seismograms by simulating the wave propagation and recording the seismic signals at the locations following (8) near the surface at 25.0 meters in depth. The survey acquisition also follows the source geometry expressed by:

$$s_x = 200.0(i-1) + 1000.0 \text{ with } i = 1, \cdots, 41, \quad (10)$$

$$s_y = 200.0(j-1) + 1000.0 \text{ with } j = 1, \cdots, 41, \quad (11)$$

where the pair $[s_x, s_y]$ meters represents the seismic source locations near the surface at 25.0 meters in depth. Thus, survey acquisition has 1681 shots that can be used in seismic migration.

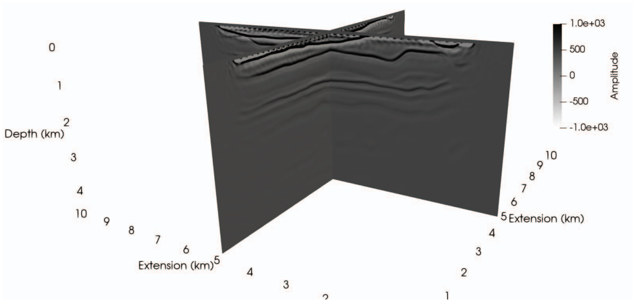

Fig. 7. Seismic image for the 3D HPC4E Seismic Test Suite.

VIII. DISCUSSIONS

Incorporating the ZFP library to compress/decompress the source wavefield raises the RTM execution time in 203%, 147%, and 122% for the lossless tolerance, lossy error tolerance 10^{-6}, and lossy error tolerance 10^{-4} on the CPU cluster, and 381%, 245%, and 179% on the GPU cluster, respectively. However, the storage requirement decreased 23.5%, 51.9%, and 66.7% on both CPU and GPU platforms, considering the same error tolerances. Choosing the compression strategy to reduce the wavefield storage requires better platforms to compensate for the overhead, such as multi-GPU clusters. For instance, assuming the lossy compression of order 10^{-6} the execution time on the GPU platform is $1.75\times$ better than the best RTM execution time on the CPU platform and demands 51.9% less information to be stored on disk.

Furthermore, the ZFP compression added around 600.789s to compress the source wavefield per shot, considering the 3D numerical experiments. This value seems huge, but considering the compression per time step, where $N_t = 600$, the ZFP compressor takes 1.445s to compress a vector of $501 \times 501 \times 235$ grid points. The time evolution nature of the RTM formulation adds an extra complexity layer to compression techniques.

Finally, we can see in Figures 2 and 3 that the final results can reach excellent visual quality. However, the tolerance defined by the lossless and lossy compression can impact the image quality. The seismic image results present errors of order 10^{-3} for the most aggressive compression case and errors of order 10^{-6} for the lossless compression. Assuming the lossy compression of order 10^{-6}, the execution time on the GPU platform is 57.2% better than the best RTM execution time on the CPU platform and demands 51.9% less information to be stored on disk. Figures 2(c) and 3(b) support that choice is also a good option for RTM applications.

IX. CONCLUSIONS AND FUTURE WORK

This work explores strategies based on decimation and compression methodologies to reduce the storage in disk required by the conventional RTM algorithm. The RTM algorithms are implemented in C language, using vectorization, OpenMP, and OpenACC to take into account the best features of supercomputers equipped with CPU/GPU hardware. In this context, the key findings are summarized as follows:

- Analysis of the seismic images results show that compressing the source wavefield did not damage the seismic image quality for controllable compression ratios, such as the decimated lossless and 10^{-6} lossy accuracy.
- Significant overhead values due to the wavefield compression were observed. However, choosing the GPU implementation compensates for the execution time spent in compression for the same tolerance.
- The RTM solutions with wavefield compression significantly reduce the storage demand for both 2D and 3D experiments. The wavefield compression ratios reach the value of 18.84 for the 2D experiment and 2.08 for the 3D experiment without hampering the seismic image quality.

As future research, tuning the ZFP compressor and exploring other compression techniques deserve better attention. For instance, compression methods, such as the Gzip, ZLIB, SZ, HyZ, and MGARD libraries [1], [4]–[6], [42], [49], [54], or the emerging Artificial Intelligence-based approaches, such as auto-encoder compression [48], can be explored to improve compression efficiency and accelerate geophysical applications. Besides, better speedups and less overhead could be achieved by overlapping the wave equation and compression calculations with the source wavefield data movement from GPU to disk.

ACKNOWLEDGMENT

This study was financed in part by CAPES, Brazil Finance Code 001. This work is also partially supported by FAPERJ, CNPq, and Petrobras. Computer time on Santos Dumont machine at the National Scientific Computing Laboratory (LNCC - Petrópolis)

REFERENCES

[1] P. Eutsch, “GZIP File Format Specification Version 4.3”. https://www.rfc-editor.org/info/rfc1952, 1996, Accessed: 2022-06-29.

[2] W. Wang, W. Zhang, and T. Lei. ”Compression of seismic forward modeling wavefield using TuckerMPI.” Computers & Geosciences, pp. 105298, 2023.

[3] A. Feldspar. “An explanation of the DEFLATE Algorithm”. https://zlib.net/feldspar.html, 1997, Accessed: 2022-06-29.

[4] J.-L. Gailly, M Adler, “Zlib compression library”, 2004, Available in: http://www.dspace.cam.ac.uk/handle/1810/3486.

[5] S. Di, and F. Cappello. ”Fast error-bounded lossy HPC data compression with SZ.” 2016 ieee international parallel and distributed processing symposium (ipdps). IEEE, 2016.

[6] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky. ”Multilevel techniques for compression and reduction of scientific data-The multivariate case.” SIAM Journal on Scientific Computing, 41.2, pp. A1278-A1303, 2019.

[7] P. Lindstrom, P. Chen, and E.-J. Lee. ”Reducing disk storage of full-3D seismic waveform tomography (F3DT) through lossy online compression.” Computers & Geosciences, 93, pp. 45-54, 2016.

[8] P. Lindstrom. ”Fixed-rate compressed floating-point arrays.” IEEE transactions on visualization and computer graphics, 20, 12, 2674-2683, 2014.

[9] J. Diffenderfer, and A. L. Fox, J. A. Hittinger, G. Sanders, P. G. Lindstrom. ”Error analysis of zfp compression for floating-point data.” SIAM Journal on Scientific Computing, 41, 3, A1867-A1898, 2019.

[10] H.-W. Zhou, H. Hu, Z. Zou, Y. Wo, O. Youn. ”Reverse time migration: A prospect of seismic imaging methodology.” Earth-science reviews, 179, 2018, pp. 207-227.

[11] D. Givoli. ”Time reversal as a computational tool in acoustics and elastodynamics.” Journal of Computational Acoustics, 22, 03, 2014, pp. 1430001.

[12] C. H. S. Barbosa. ”Advanced Computational Strategies for Reverse Time Migration”. PhD Thesis, Federal University of Rio de Janeiro, Brazil, 2023.

[13] D. Komatitsch and R. Martin. ”An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation.” Geophysics, 72, 5, 2007, pp. SM155-SM167.

[14] D. Pasalic and R. McGarry. ”Convolutional perfectly matched layer for isotropic and anisotropic acoustic wave equations: 80th Annual International Meeting, SEG, Expanded Abstracts, pp. 2925–2929, doi: 10.1190/1.3513453.” Abstract, 2010.

[15] C. Cerjan, D. Kosloff, R. Kosloff, and M. Reshef. ”A nonreflecting boundary condition for discrete acoustic and elastic wave equations.” Geophysics, 50, 4, 1985, pp. 705-708.

[16] R. G. Clapp. ”Reverse time migration with random boundaries: 79th Annual International Meeting, SEG, Expanded Abstracts, pp. 2809–2813.” Abstract, 2009.

[17] R. G. Clapp. ”Reverse time migration: Saving the boundaries: Technical Report SEP-136.” Stan-6 ford Exploration Project 7, 2008.

[18] B. D. Nguyen and G. A. McMechan. ”Five ways to avoid storing source wavefield snapshots in 2D elastic prestack reverse time migration.” Geophysics, 80, 1, 2015, pp. S1-S18.

[19] Q. Li, L.-Y. Fu, R.-W. Wu, and Q. Du, ”Efficient acoustic reverse time migration with an attenuated and reversible random boundary.” IEEE Access, 8, 2020, pp. 34598-34610.

[20] G. T. Schuster. ”Seismic inversion”. Tulsa, OK U.S.A. 74137-3575, Society of Exploration Geophysicists, 2017.

[21] J. F. Claerbout. ”Imaging the earth’s interior”. Vol. 1. Stanford, California, Stanford University, 1985.

[22] W. Sun, and L.-Y. Fu. ”Two effective approaches to reduce data storage in reverse time migration.” Computers & Geosciences, 56, 69-75.

[23] L. Di Bartolo, C. Dors, and W. J. Mansur. ”A new family of finite-difference schemes to solve the heterogeneous acoustic wave equation.” Geophysics, 77, 5, 2012, pp. T187-T199.

[24] A. Qawasmeh, R. Maxime, H. Calandra, B. M. Chapman. ”Performance portability in reverse time migration and seismic modelling via OpenACC.” The International Journal of High Performance Computing Applications, 31, 5, 2017, pp. 422-440.

[25] N. Kushida, Y-T Lin, P. Nielsen, R. Le Bras. ”Acceleration in acoustic wave propagation modelling using OpenACC/OpenMP and its hybrid for the global monitoring system.” Accelerator Programming Using Directives: 6th International Workshop, WACCPD 2019, Denver, CO, USA, November 18, 2019, Revised Selected Papers 6. Springer International Publishing, 2020.

[26] M. Serpa, Pavan, J. Pablo E. H. M. Cruz, R. L. and Machado, J. Panetta, A. Azambuja, A. S. Carissimi, P. O. A. Navaux. ”Energy efficiency and portability of oil and gas simulations on multicore and graphics processing unit architectures.” Concurrency and Computation: Practice and Experience, 33, 18, 2021, pp. e6212.

[27] P. Kearey, B. Michael, I. Hill. An introduction to geophysical exploration. Vol. 4. John Wiley & Sons, 2002.

[28] J. De la Puente. D6.3 Website deploying a suite of geophysical tests for wave propagation problems on extreme scale machines. HPC4E - High Performance Computing for Energy 689772, HPC4E Consortium Partners, 2015.

[29] K. Zhao, Kai, S. Di, X. Lian, S. Li, D. Tao, J. Bessac, Z. Chen, F. Cappello. ”SDRBench: Scientific data reduction benchmark for lossy compressors.” 2020 IEEE International Conference on Big Data (Big Data). IEEE, 2020.

[30] C. H. S. Barbosa, A. L. G. A. Coutinho. “Multi-GPU 3-D Reverse Time Migration with Minimum I/O”. In: Navaux, P., Barrios H., C. J., Osthoff, C., et al. (Eds.), High Performance Computing, pp. 160–173, Cham, September. Springer International Publishing. ISBN: 978-3-031-23821-5.

[31] C. H. S. Barbosa, A. L. G. A. Coutinho. ”Enhancing Reverse Time Migration: Hybrid Parallelism plus Data Compression”. In: Proceedings of the XLI Ibero-Latin-American Congress on Computational Methods in Engineering. ABMEC, November, 2020.

[32] M. S. Serpa, E. H. Cruz, M. Diener, A. M. Krause, P. O. Navaux, J. Panetta, A. Farrés, C. Rosas, M. Hanzich. Optimization strategies for geophysics models on manycore systems. The International Journal of High Performance Computing Applications, 33(3), 473-486, 2019.

[33] L. Qu, R. Abdelkhalak, H. Ltaief, I. Said, D. Keyes. Exploiting temporal data reuse and asynchrony in the reverse time migration. The International Journal of High Performance Computing Applications, 2022.

[34] C. H S Barbosa, L. N. Kunstmann, R. M. Silva, C. D. Alves, B. S. Silva, D. M. S. Filho, M. Mattoso, F. A. Rochinha, A. L. G. A. Coutinho. A workflow for seismic imaging with quantified uncertainty. Computers & Geosciences, 145, pp. 104615, 2020.

[35] R. Mathur, R. Atcheson, Y. Kubo. Optimizations for seismic applications on the NEC SX-Aurora TSUBASA. In SEG International Exposition and Annual Meeting. OnePetro, October, 2020.

[36] V. Etienne, A. Momin, L. Gatineau, S. Momose. Performance Characterization of a Vector Architecture for Seismic Applications. In Fifth EAGE Workshop on High Performance Computing for Upstream. EAGE Publications BV, September, 2021.

[37] S. Momose, Y. Kubo, M. Ikuta. Performance Evaluation of Stencil Calculation in RTM Code. In Fifth EAGE Workshop on High Performance Computing for Upstream. EAGE Publications BV, September, 2021.

[38] W. W. Symes. Reverse time migration with optimal checkpointing. Geophysics, 72, 5, pp. SM213-SM221, 2007.

[39] M. Ainsworth, S. Klasky, B. Whitney. Compression using lossless decimation: analysis and application. SIAM Journal on Scientific Computing, 39, 4, pp. B732-B757, 2017.

[40] G. F. Barros, M. Grave, J. J. Camata, A. L. G. A. Coutinho. Enhancing dynamic mode decomposition workflow with in situ visualization and data compression. Engineering with Computers, pp. 1-22, 2023.

[41] S. Jin, P. Grosset, C. M. Biwer, J. Pulido, J. Tian, D. Tao, J. Ahrens. Understanding GPU-based lossy compression for extreme-scale cosmological simulations. In 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (pp. 105-115). IEEE, May, 2020.

[42] K. Zhao, S. Di, X. Lian, S. Li, D. Tao, J. Bessac, Z. Chen, F. Cappello. SDRBench: Scientific data reduction benchmark for lossy compressors. In 2020 IEEE International Conference on Big Data, pp. 2716-2724. IEEE, December, 2020.

[43] H. Liu, B. Li, H. Liu, X. Tong, Q. Liu, X. Wang, W. Liu. The issues of prestack reverse time migration and solutions with graphic processing unit implementation. Geophysical Prospecting, 60, 5, pp. 906-918, 2012.

[44] G. F. Liu, X. H. Meng, Z. J. Yu, D. J. Liu. An efficient scheme for multi-GPU TTI reverse time migration. Applied Geophysics, 16, 1, , pp. 56-63, 2019.

[45] Y. Wang, H. Zhou, X. Zhao, Q. Zhang, P. Zhao, X. Yu, Y. Chen, Y. CuQ-RTM: A CUDA-based code package for stable and efficient Q-compensated reverse time migrationCUDA-based Q-RTM. Geophysics, 84, 1, pp. F1-F15, 2019.

[46] J. Fang, H. Chen, H. Zhou, Y. Rao, P. Sun, J. Zhang. Elastic full-waveform inversion based on GPU accelerated temporal fourth-order finite-difference approximation. Computers & Geosciences, 135, pp. 104381, 2020.

[47] N. Kukreja, J. Hückelheim, M. Louboutin, J. Washbourne, P. H. Kelly, G. J. Gorman. Lossy checkpoint compression in full waveform inversion: a case study with ZFPv0. 5.5 and the overthrust model. Geoscientific Model Development, 15, 9, pp. 3815-3829, 2022.

[48] J. Wittmer, J. Badger, H. Sundar, T. Bui-Thanh. An autoencoder compression approach for accelerating large-scale inverse problems. arXiv preprint arXiv:2304.04781, 2023.

[49] X. Liang, B. Whitney, J. Chen, L. Wan, Q. Liu, D. Tao, J. Kress, D. Pugmire, M. Wolf, N. Podhorszki, S. Klasky and others. Mgard+: Optimizing multilevel methods for error-bounded scientific data reduction. IEEE Transactions on Computers, 71, 7, pp. 1522-1536, 2021.

[50] T. Alturkestani, H. Ltaief, and D. Keyes. Maximizing I/O bandwidth for reverse time migration on heterogeneous large-scale systems. In Euro-Par 2020: Parallel Processing: 26th International Conference on Parallel and Distributed Computing, Warsaw, Poland, August 24–28, 2020, Proceedings 26 (pp. 263-278). Springer International Publishing, 2020.

[51] T. Alturkestani, T. Tonellot, H. Ltaief, R. Abdelkhalak, V. Etienne, and D. Keyes. Mlbs: Transparent data caching in hierarchical storage for out-of-core hpc applications. In 2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC) (pp. 312-322). IEEE, 2019.

[52] R. Kriemann, H. Ltaief, M. B. Luong, F. E. H. Pérez, H. G. Im, and D. Keyes. High-Performance Spatial Data Compression for Scientific Applications. In European Conference on Parallel Processing (pp. 403-418). Cham: Springer International Publishing, August 2022.

[53] 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) (pp. 1643-1654). IEEE, April, 2021.

[54] Y. Huang, K. Zhao, S. Di, G. Li, M. Dmitriev, T. L. D. Tonellot, & F. Cappello. Towards Improving Reverse Time Migration Performance by High-speed Lossy Compression. In 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid) (pp. 651-661). IEEE, May, 2023.