Enhancing Reverse Time Migration: Hybrid Parallelism plus Data Compression

Carlos Henrique S. Barbosa[1], Alvaro L. G. A. Coutinho[1]

[1]*High Performance Computing Center and Dept. of Civil Engineering, COPPE/Federal University of Rio de Janeiro*
Av. Horácio Macedo, 2030, Centro de Tecnologia - Bloco Rio de Janeiro, RJ 21941-598, Brasil
c.barbosa@nacad.ufrj.br, alvaro@nacad.ufrj.br

Abstract. Migration techniques like the Reverse Time Migration (RTM) are time-consuming and data-intensive. RTM is time-consuming due to the computational cost associated with the stability and dispersion conditions in the discrete two-way wave equation. On the other hand, RTM is data-intensive because of the number of input seismograms to be loaded and the need to store temporary files, such as the forward propagated wavefields, to build the migrated seismic images and partial migrated images before stacking. Thus, to overcome the time-consuming problem, we implement a Reverse Time Migration algorithm that explores the Message Passing Interface (MPI) library and the OpenMP+vectorization to explore parallelism in hybrid architectures. On the other hand, we implement compression techniques to reduce storage and to improve network data transfer. The data compression can be used to reduce the storage of seismograms, model parameters, migrated seismic images, and temporary files. Results show that an MPI/OpenMP+vectorization strategy is efficient in hybrid multi-core machines. Besides, high levels of data compression suggest that its use does not hamper the final migrated seismic images.

Keywords: Reverse Time Migration, Data compression, High-performance Computing

1 Introduction

In recent years, advances in seismic imaging technology and data processing have boosted the oil and gas industry capabilities to identify new oil and natural gas prospects, mainly offshore. Image generation for seismic interpretation relies on migration methods [1] that have as inputs seismograms and a velocity model. Given the seismograms, velocity estimates are obtained from techniques, such as Normal Move-Out [2, 3], tomography [4], or full-waveform inversion (FWI) [4–6]. However, critical aspects in imaging large domains are the computational costs and the huge amount of data required. For instance, 3D data acquisition is now commonplace, which means that we need to model full 3D volumes. Furthermore, if a seismic imaging technique considers solving partial differential equations (PDEs) associated with the wave models, the situation tends to be more complicated as the excitation signals bear high-frequency content, demanding fine space and time grids.

The geophysical imaging industry has relied for years on highly specialized computational implementations based on the Finite Difference (FD) method for their migration techniques. However, computer architectures have changed dramatically over the years, mostly driven by the commodity market. New computational architectures and parallel paradigms imply that codes require constant maintenance to beat obsolescence. Today, FD algorithms are still as popular as ever, and the applications where they are used have increased significantly. Nevertheless, independent of which numerical method used to discretize the PDEs, two main computational aspects need to be handled to reach a boost performance: CPU time and storage demands.

In this work, fast and effective computational optimization and porting strategies are shown to boost a Reverse Time Migration (RTM) numerical implementation. RTM is a high-resolution depth migration based on the two-way wave equation, and an appropriate imaging condition [1]. Because the wave propagation kernel concentrates most of the program execution time ($\approx$ 98% of total) and produces a solution called source wavefield used in the imaging condition, we focus the computational optimizations on the wave equation and strategies to store the forward-propagated source wavefield (or source wavefield). In the first place, to overcome the time-consuming problem, we implement a Reverse Time Migration algorithm that explores the Message Passing Interface (MPI) library and the OpenMP+vectorization to explore parallelism in hybrid architectures. On the other hand, we implement compression techniques to reduce the storage of temporary files, such as the source wavefield. Previous computational results were presented in [7] in an uncertainty quantification framework. In this work, we aim to detail the MPI/OpenMP+vectorization for hybrid multi-core machines. Besides, we deeply analyze how high levels of data compression influence the final migrated seismic images. The remainder of this work is organized as follows. Section 2 briefly reviews RTM. The computational details are given in Section 3. Section 4 shows the numerical experiments. The paper ends with a summary of our main findings.

2 Reverse Time Migration

RTM relies on the two-way wave equation and an appropriate imaging condition [1]. Here, we consider the isotropic acoustic wave equation, that is,

$$\nabla^2 p(\mathbf{r},t) - \frac{1}{v^2(\mathbf{r})}\frac{\partial^2 p(\mathbf{r},t)}{\partial t^2} = 0, \tag{1}$$

where $p(\mathbf{r},t)$ is the pressure defined at the position $\mathbf{r} = (r_x, r_y, r_z)$ for a given domain Ω and time $t \in [0,T]$. Equation eq. (1) is equipped with a free-surface boundary condition at the domain upper boundary, non-reflection boundary conditions at the lateral and domain bottom, and a proper initial condition. The image condition is the zero-lag cross-correlation of the forward-propagated source wavefield (S) and the backward-propagated receiver wavefield (R) normalized by the square of the source wavefield [1], given by,

$$I(\mathbf{r}) = \frac{\int_T S(\mathbf{r},t)R(\mathbf{r},t)dt}{\int_T S(\mathbf{r},t)^2 dt}. \tag{2}$$

The imaging condition amplitudes in eq. (2) provide an explicit physical relationship with the reflection coefficients [8]. Note that the source wavefield is the solution of eq. (1) with the seismic wavelet treated as a boundary condition (BC). Similarly, the receiver wavefield is the solution of the same eq. (1) with the observed data (seismograms) as a BC. The normalization term in eq. (2) produces a source-normalized cross-correlation image with the same unit, scaling, and sign as the reflection coefficient [1].

3 Computational Implementation

The second-order equation eq. (1) describes the acoustic pressure change in an arbitrary medium. The numerical discretization used to solve eq. (1) is based on the acoustic non staggered grid scheme (NSG) [9]. The NSG is the most simple and popular finite-difference numerical scheme used to discretize the acoustic wave equation where each derivative of eq. (1) can be approximated using numerical operators obtained by Taylor series expansions [9]. Here we apply a high-order discretization, an eighth-order accurate in space and second order in time, to reach better accuracy. Therefore, the discretized second-order wave equation in two dimensions is given by,

$$p_{i,k}^{n+1} = C_{i,k}\left[2b_0 + \sum_{m=1}^{4} b_m(p_{i+m,k}^n + p_{i-m,k}^n + p_{i,k+m}^n - p_{i,k-m}^n)\right] - 2p_{i,k}^n + p_{i,k}^{n-1} \tag{3}$$

where the indexes i, and k represent the directions (x, z), n is the temporal step, and p is the pressure. The matriz $C_{i,k} = (v_{i,k}\Delta t/h)^2$, where $v_{i,k}$ is the velocity, Δt is the temporal sampling rate, and h is the grid size. Lastly, b_m are finite difference coeficients. Recall that proper boundary conditions (free-surface and non-reflection) and initial conditions have to be applied to the discretized eq. (3). To satisfy the image condition eq. (2) we need to compute the wave equation twice, first the the forward-propagated source wavefield (S) and afterward the backward-propagated receiver wavefield (R).

Let $\mathbf{v}$ be the discrete version of the velocity field, and similarly, $\mathbf{p}$, $\mathbf{s}$, $\mathbf{f}$ at each time step. Note that the vectors $\mathbf{p}$, and $\mathbf{v}$ have the same dimension, that is $N = N_x * N_y$ (or $N = N_x * N_y * N_z$ in 3D), where N_x, N_y (and N_z) are the number of grid points in each Cartesian direction. The source and receiver wavefields are represented by the vectors $\mathbf{S}$ and $\mathbf{R}$. Each discrete seismogram is a vector of size $N_{rec} * (N_t + 1)$, where N_{rec} is the number of receivers, and $N_t = T/\Delta t$, with Δt is the time step. The seismic source $\mathbf{f}$ has dimension N_t. Algorithm 1 details the RTM, where a set of seismograms, $\{\mathbf{s}_1, \cdots, \mathbf{s}_{N_{shots}}\}$, a velocity field, $\mathbf{v}$, and a seismic source ($\mathbf{f}$) are given as inputs. The index N_{shots} represents the number of seismograms to be migrated. RTM iterates over the seismograms producing partial migrated images by calculating eq. (2). Note that each iteration over the number of seismograms solves the wave equation twice, one for the seismic source and other for the current seismogram. The computation of the imaging condition uses both solutions (source wavefield, and receiver wavefield), retrieving from persistent storage the source wavefield to build the migrated seismic section and stacking the partial results over time ($\mathbf{I}_{\sum t}$), and over the number of seismograms ($\mathbf{I}_{\sum shot\_id}$). At the end of Algorithm (1) we have the discrete image condition $\mathbf{I}$, which is a vector in $\mathbb{R}^N$.

Algorithm 1 Reverse Time Migration.

```
Require: $\mathbf{v}$, $\mathbf{f}$, and $\{\mathbf{s}_1, \cdots, \mathbf{s}_{N_{shots}}\}$
 1: function RTM(vector $\mathbf{v}$, vector $\mathbf{f}$, vectors $\{\mathbf{s}_1, \cdots, \mathbf{s}_{N_{shots}}\}$)
 2:     read $\mathbf{v}$, and $\mathbf{f}$
 3:     initialize $\mathbf{I}_{\sum shot\_id} = 0$
 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:             $n_t = n_t + i_t * \Delta t$
 9:             solve eq. (3) with the seismic wavelet treated as BC
10:             store $\mathbf{S}_{n_t}$                                          ▷ source wavefield
11:         end for
12:         initialize $n_\tau = 0$, and $\mathbf{I}_{\sum \tau} = 0$
13:         apply initial conditions for $i_\tau = 0$
14:         for $i_\tau = 1$ to $N_t$ do
15:             $n_\tau = N_t - (n_\tau + i_\tau * \Delta\tau)$                    ▷ reverse time
16:             read $\mathbf{S}_{n_\tau}$, and $\mathbf{s}_{shot\_id}$
17:             solve eq. (3) with the seismogram treated as BC                ▷ receiver wavefield
18:             calculate $\mathbf{I}_{\sum n_\tau} = \mathbf{I}_{\sum n_\tau} + (\mathbf{S}_{n_\tau}\mathbf{R}_{n_\tau})/(\mathbf{S}_{n_\tau}\mathbf{S}_{n_\tau})$    ▷ imaging condition
19:         end for
20:         stack $\mathbf{I}_{\sum shot\_id} = \mathbf{I}_{\sum shot\_id} + \mathbf{I}_{\sum n_\tau}$          ▷ stacking
21:     end for
22:     store $\mathbf{I}_{\sum shot\_id}$
23: end function
```

We use the MPI library to manage the execution of multiple shots, where batches of shots are assigned to different allocated MPI processes. We handle the set of shots per MPI process in line 4 of the Algorithm 1. As mentioned earlier, the second-order wave equation is discretized with a second-order finite difference scheme in time and an eighth-order scheme in space. Based on the discretized wave equation, its computational implementation takes advantage of OpenMP directives to explore multiple cores parallelism. Besides, the reverse problem implementation of the RTM explores the technique proposed by [10]. This technique guarantees the coercivity of the adjoint problem. Finally, the RTM code supports the Single-Instruction-Multiple-Data (SIMD) model and memory alignment allocation to ensure vectorization.

We also implement data compression to reduce persistent storage due to the need to register the forward-propagated source wavefield (see line 10 of the Algorithm 1). Because the source wavefield is a vector of size $N * (N_t + 1)$, the amount of data to be managed can increase drastically. To lead with this problem, we have chosen the open-source ZFP [11]. ZFP uses lossy but optionally error-bounded compression. Although bit-for-bit lossless compression of floating-point data is not always possible, ZFP also offers an accurate near-lossless compression [11]. Thus, the ZFP library is linked to RTM implementation to reduce the pressure on the network data transfer and reduce I/O to persistent storage.

4 Numerical Experiments

RTM is computationally demanding and data-intensive. We have been investing in high-performance computing (HPC) and data compression to reduce CPU time and storage demand. We designed computational experiments to assess RTM performance and measure the impact of data compression on the final seismic images. We present the weak and strong RTM scalability analysis in Section 4.1, and the influence of dealing with compressed information such as the source wavefield in Section 4.2. To illustrate these results, we choose the Marmousi benchmark, which is 3.0 km depth and 9.2 km in the horizontal direction. 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 [3], where the seismic source, 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.

4.1 Scalability analysis

We run all the tests related to the scalability analysis on a multi-core cluster called Lobo Carneiro (LB) at COPPE/UFRJ. The LB cluster has Intel Xeon E5-2670v3 (Haswell) CPUs with 24 cores per node. Figure 1 shows the RTM strong scalability considering a migration of one-shot for an increasing number of threads. The speedup reaches the maximum value of $6,24$ with 24 threads. However, the speedup values seem to stabilize (or not significantly increase its values) after we set 12 threads to run the test. Notice that the difference between the speedup values considering 12 threads and 1 thread is 5.73. On the other hand, the difference between the speedup values considering 24 threads and 12 threads is 0.51.

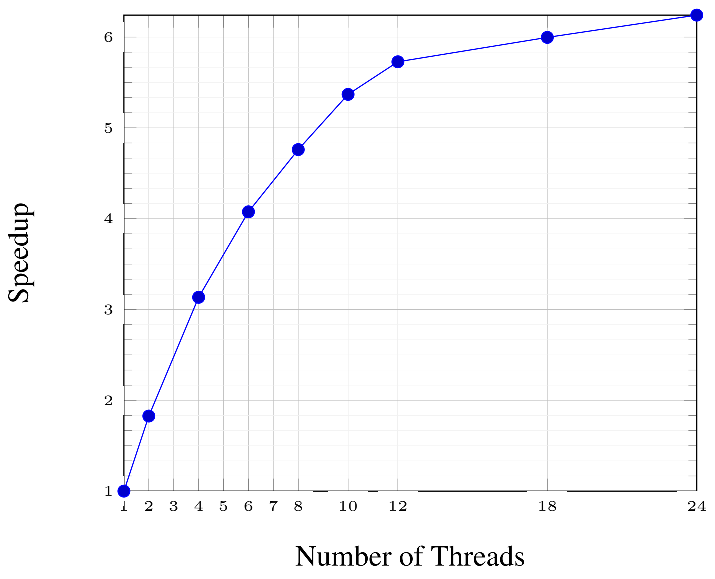

Figure 1. RTM scalability analysis on the Lobo Carneiro cluster at COPPE/UFRJ. Each node has a Haswell processor with 24 cores. The test consists of migration for one seismogram for different number of threads.

Because the acquisition survey has 82 seismograms (or shots) to be migrated, we use the MPI library to manage the execution of multiple shots. Thus, batches of shots can be assigned to different allocated MPI processes. The strong scalability test suggests that it is plausible to run two RTMs at the same node. Choosing 5 nodes and allocating 10 MPI processes, we run the RTM for 10 seismograms, and we record that each MPI process takes 7.24 seconds (with standard deviation value of 0.0655 seconds) to complete the execution for each seismogram. Finally, running the entire acquisition survey with the same 5 nodes, the migration time is ≈ 59.28 seconds. In this case, we observe that the CPU time for each MPI process has approximately the same value, which characterizes the weak scalability. Figure 2 shows the RTM outcome for the 82 seismograms after appropriated post-processing.

4.2 Data compression

Two critical stages exist when dealing with storage demands for RTM. The first is related to the number of seismograms, and the second relates to temporary information, such as the source wavefield, while RTM is running. Here, we choose to deal with the need for storage of the source wavefield, and we have chosen data compression for that. Thus, we begin the data compression analysis applying lossless and lossy compression to the source wavefield using the ZFP library linked to our RTM implementation. The assumed absolute errors for

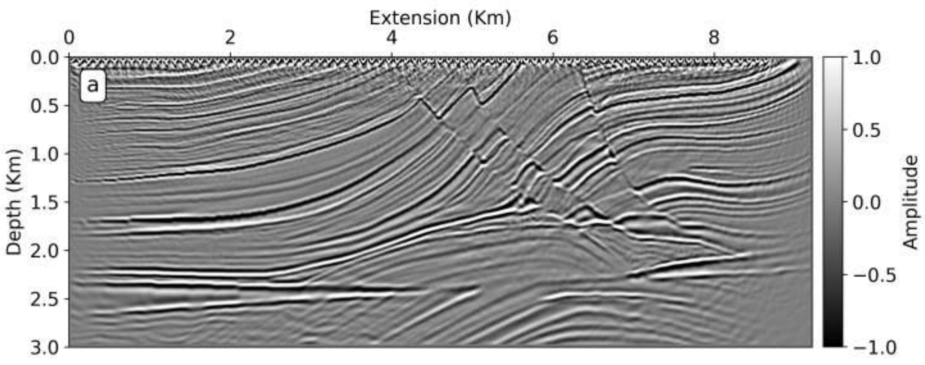

Figure 2. Migrated seismic image provided by the Reverse Time Migration technique.

the lossy compressions are 10^{-4}, and 10^{-6}. The demand for total storage without compression is 1.8 GB, 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 38.9%, 94.8%, and 99.2% less than the original data. Figure 3 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 of amplitude information changes during the wavefield propagation.

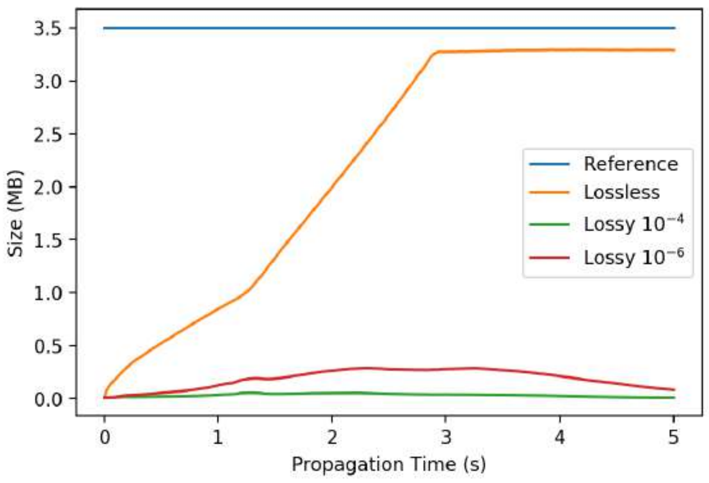

Figure 3. Size per time slice for the propagated-forward source wavefield without compression (blue line), with lossless compression (orange line), with a lossy tolerance compression of 10^{-4} (green line), and with a lossy tolerance compression of 10^{-6} (red line).

Although we observe high compression rates, we need to evaluate the error propagation due to source wavefield compression on the final migrated seismic images. Figure 4(a) shows the resulting seismic image without using compression that we define as the reference outcome. Figure 4(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 4(d)). However, we do not note any damage on the seismic images in Figures 4(b), and (c).

The errors produced by the difference among the reference image and seismic images built using compression can be seen in Figure 5. Figure 5(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}.

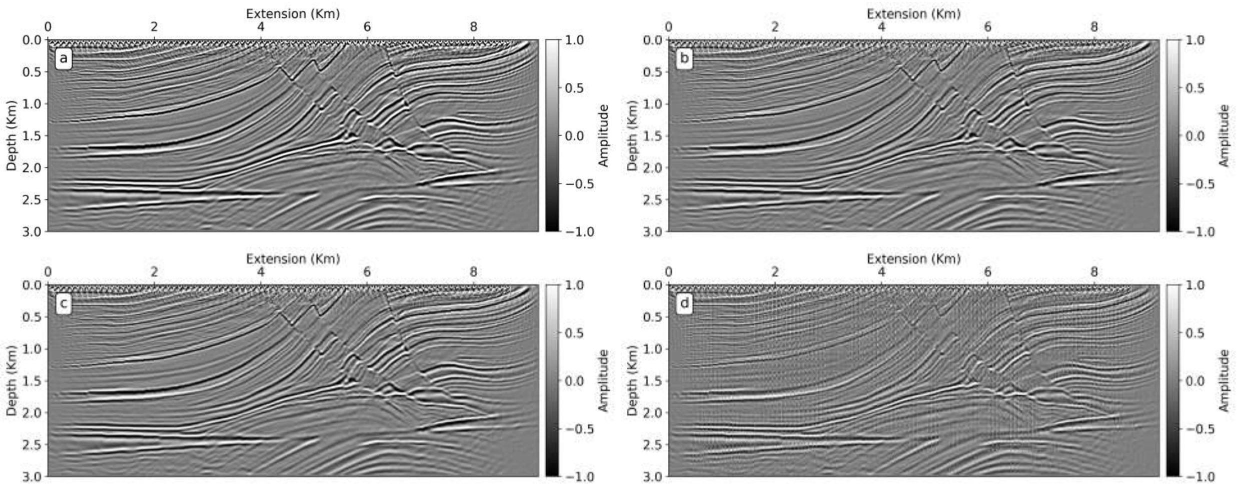

Figure 4. RTM outcomes after migration using the forward-propagated source wavefield (a) without compression, (b) with lossless compression, (c) with a lossy tolerance compression of 10^{-6}, and (c) with a lossy tolerance compression of 10^{-4}.

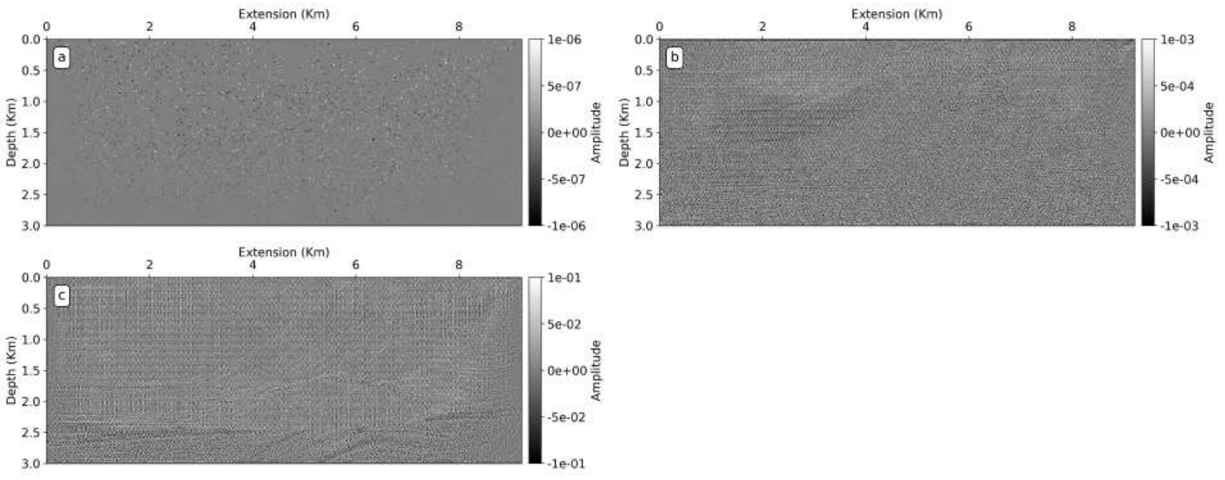

Figure 5. Difference among the reference migrated image (no compression) and (a) the migrated image after applying lossless compression on the source wavefield, (b) the migrated image after applying a lossy tolerance compression of 10^{-6} on the source wavefield, and the migrated image after applying a lossy tolerance compression of 10^{-4} on the source wavefield.

5 Conclusions

In this work, we report improvements made in hybrid parallel high-order finite-difference discretizations of the RTM in multi-core machines. RTM technique requires significant computing power and generates a massive amount of data. Thus, we use optimized hybrid computing solutions that explore OpenMP+vectorization and data compression, improving the RTM overall performance. These tools allow us to obtain a scalable solution for the Marmousi Velocity Model benchmarks. In our experiments, we use the ZFP library to compress information, such as the forward-propagated source wavefield. The compression strategy improves the network data transfer and reduces the amount of data up to 94%, assuming a tolerance error of 10^{-6}. The careful use of compression suggests that we can significantly reduce the storage demanding without hampering the final migrated seismic images.

Acknowledgements. This study was financed in part by CAPES, Brasil Finance Code 001. This work is also partially supported by FAPERJ, CNPq, and Petrobras. Computer time on Lobo Carneiro machine at COPPE/UFRJ is also acknowledged.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] Zhou, H.-W., Hu, H., Zou, Z., Wo, Y., & Youn, O., 2018. Reverse time migration: A prospect of seismic imaging methodology. Earth-Science Reviews, vol. 179, pp. 207–227.
[2] Yilmaz, Ö., 2001. Seismic data analysis: Processing, inversion, and interpretation of seismic data. Society of exploration geophysicists.
[3] Kearey, P., Brooks, M., & Hill, I., 2013. An introduction to geophysical exploration. John Wiley & Sons.
[4] Schuster, G. T., 2017. Seismic inversion. Society Exploration Geophysicists, 1 edition.
[5] Fichtner, A., 2010. Full seismic waveform modelling and inversion. Springer Science & Business Media.
[6] Parida, S. S., Sett, K., & Singla, P., 2018. An efficient pde-constrained stochastic inverse algorithm for probabilistic geotechnical site characterization using geophysical measurements. Soil Dynamics and Earthquake Engineering, vol. 109, pp. 132–149.
[7] Barbosa, C. H., Kunstmann, L. N., Silva, R. M., Alves, C. D., Silva, B. S., Mattoso, M., Rochinha, F. A., Coutinho, A. L., et al., 2020. A workflow for seismic imaging with quantified uncertainty. arXiv preprint arXiv:2001.06444.
[8] Chattopadhyay, S. & McMechan, G. A., 2008. Imaging conditions for prestack reverse-time migration. Geophysics, vol. 73, n. 3, pp. S81–S89.
[9] Di Bartolo, L., Dors, C., & Mansur, W. J., 2012. A new family of finite-difference schemes to solve the heterogeneous acoustic wave equationnew finite-difference schemes for acoustics. Geophysics, vol. 77, n. 5, pp. T187–T199.
[10] Givoli, D., 2014. Time reversal as a computational tool in acoustics and elastodynamics. Journal of Computational Acoustics, vol. 22, pp. 45–54.
[11] Lindstrom, P., Chen, P., , & Lee, E.-J., 2016. Reducing disk storage of full-3d seismic waveform tomography (f3dt) through lossy online compression. Computers & Geosciences, vol. 93, pp. 45–54.