





# Multi-GPU 3-D Reverse Time Migration with Minimum I/O

Carlos H. S. Barbosa<sup>(✉)</sup>  and Alvaro L. G. A. Coutinho 

High-Performance Computing Center, COPPE Federal University of Rio de Janeiro,  
Rio de Janeiro, Brazil  
{c.barbosa,alvaro}@nacad.ufrj.br

**Abstract.** Seismic imaging techniques based on two-way wave equations are computationally and data-intensive activities in the oil and gas industry. For instance, Reverse Time Migration (RTM) migrates a set of SEG-Y format data from the disk called a seismogram. Besides, during execution, the RTM application needs to store the forward-propagated wavefield (or source wavefield) on disk to build the final seismic image. Storing the source wavefield for multiple RTMs executing in parallel is even more challenging because the storage capacity can reach tens of Terabytes of information. Aiming to mitigate the storage demand, we develop a 3-D RTM with source wavefield reconstruction. The source wavefield is reconstructed by introducing a new wave equation to the problem and adjusting the initial and boundary conditions to take advantage of random boundary conditions' (RBC) properties. The RBC does not suppress unwanted waves coming from the artificial boundary enabling full wavefield recovery. We also develop a hybrid OpenACC/MPI implementation for the 3-D RTM on a multi-GPU machine. We test the RTM implementation on the 3-D HPC4E Seismic Test Suite. The numerical experiments show that the OpenACC/MPI 3-D RTM, which implements the wavefield reconstruction, presents the best execution times and hard disk demands.

**Keywords:** High performance computing · Reverse time migration · Wavefield reconstruction and OpenACC/MPI implementation

## 1 Introduction

Reverse Time Migration (RTM) is a depth migration technique that provides a reliable high-resolution representation of the Earth subsurface, useful for seismic interpretation and reservoir characterization [26]. RTM is based on the two-way wave equation and an appropriate imaging condition. Generally, the two-way wave equation is solved by numerical methods such as the Finite Difference Method (FDM) and the Finite Element Method (FEM). Besides, imaging conditions need the computational implementation of the forward-propagated wavefield (or source wavefield) for further access in reversal order to build the seismic image.

Advances in wave propagation algorithms and wavefield storage developments, and hardware acceleration implementations are some of the main challenges concerning RTM [26]. For instance, the most effective non-reflecting boundary condition, Perfectly Matched Layer (PML), demands additional partial differential equations (PDEs) to be solved on artificial layers around the domain [9, 17] to deal with unwanted reflections due to truncated domains. A way to overcome this issue is to use the random boundary conditions (RBC) proposed by [6]. Thus, instead of suppressing unwanted waves by inserting new equations into the problem, the methodology proposed by [6] is based on exploring low correlations with non-coherent signals coming from an artificial boundary with random velocities. On the other hand, the source wavefield in the RTM technique is a bottleneck due to the amount of information that has to be stored on a disk to build the imaging condition [26]. Strategies to diminish input/output (I/O) related to the source wavefield storage or its reconstruction are presented by [1, 5–7, 13, 16, 22, 23]. Among them, [22] presented two strategies to reduce data storage, where one is based on the Nyquist sampling theorem, and the second one uses a lossless compression algorithm. In this sense, [1] studied the numerical impact of applying lossless and lossy compression to the RTM source wavefield. They show that the careful use of high levels of data compression can significantly reduce the storage demand without hampering the final seismic images. However, instead of storing the wavefield, its reconstruction is a viable possibility. This can be done by checkpoint methods [7, 23], using wavefield recording around the boundary [5, 16], or by initial value reconstruction (IVR) [6, 13, 16].

Independent of the RTM implementation strategy, all can use HPC techniques to boost their performance. [18] implemented the seismic modeling and RTM on single and multi-GPUs using a hybrid MPI+OpenACC approach aiming to develop portable high-level directive-based codes across heterogeneous platforms for seismic imaging applications. [20] evaluated three different computational optimizations based on multicore and GPU architectures and investigated their codes' performance, energy efficiency, and portability. Nevertheless, the storage demand issue remained in the RTM-based GPU implementations presented by the earlier research. For this, [15] implemented the RTM with RBC to diminish the storage demand in migration algorithms showing that such a strategy is beneficial for GPU implementations. The GPU computational implementation with the RBC technique was coded in CUDA and tested only for 2-D RTM applications. For 3-D environments, [2] developed a wave propagation modeler and a RTM algorithm exploring the main RBC characteristics. It was shown that RTM with RBCs performs better on vector processors and GPU machines than CPU platforms.

In this context, we developed an RTM approach for 3-D environments that explore the main characteristics of the RBC to mitigate calculations on the artificial boundaries and enable the source wavefield reconstruction. The RTM with wavefield reconstruction takes advantage of the RBC's non-dissipative energy property and implements the IVR technique to build the imaging condition with minimum storage. Our implementation is particularly suited for GPUs because

we eliminate the need for storage for the whole source wavefield. We present the computational times and disk storage results for two algorithmic choices (with and without IO) in two different computational platforms: a CPU cluster and a CPU-GPU cluster. We show that our computational implementations are efficient, scalable, and portable with minimum interference on the optimized baseline code.

The remainder of the work is organized by introducing the RTM mathematical background in Sect. 2. Section 3 details the computational implementation along with optimizations for the NVIDIA V100 platform. In Sect. 4, we present numerical experiments, where we expose the execution time requirements, speedups, and hard disk demand for each computational implementation, as well as the RTM outcomes (seismic images). The paper ends with a summary of our main findings in Sect. 5.

## 2 Reverse Time Migration

Reverse Time Migration (RTM) is a depth migration technique based on the two-way wave equation, and an imaging condition [26]. Solving the wave equation twice to build the imaging condition is necessary. The first solution, called forward-propagated wavefield (source wavefield), can be obtained by solving the equation:

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

where,  $p$  is the pressure,  $v$  the velocity for the compressional wave,  $\mathbf{r}$  the spatial coordinates,  $t$  the time in  $[0, T]$ , and  $f(\mathbf{r}_s, t)$  the seismic source at the position  $\mathbf{r}_s$ . The pressure  $p$  is defined in a domain  $\Omega \subset \mathbb{R}^3$ . 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  $\partial\Omega \in \mathbb{R}^2$ , where  $\partial\Omega$  is the domain boundary.

The second solution is obtained by solving the following equation:

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

where,  $\bar{p}$  is the backward-propagated wavefield (receiver wavefield),  $s(\mathbf{r}_r, \tau)$  is the seismogram recorded at the receivers positions  $\mathbf{r}_r$ , and  $\tau = T - t$  is the reversal time evolution defined as in [8], where  $\tau \in [0, T]$ .  $\bar{p}$  is also defined in  $\Omega \subset \mathbb{R}^3$ , and corresponding initial, and boundary conditions should be set. Once we have the source and receiver wavefields, the imaging condition can be calculated as:

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

where  $I(\mathbf{r})$  is called source-normalized cross-correlated imaging condition. The source-normalized cross-correlation image in Eq. (3) has the same unit, scaling, and sign of the reflection coefficient [26].

### 3 Computational Implementation and Optimizations

Our RTM implementation employs the explicit Finite Difference Method (FDM) to solve the acoustic wave equation. The finite difference stencil for Eqs. (1) and (2) are 8th-order in space and 2nd-order in time. Thus, the numerical discretization leads to the discrete version of the velocity field, source wavefield, receiver wavefield, seismic source, and seismograms represented by the vectors  $\mathbf{v}$ ,  $\mathbf{p}$ ,  $\bar{\mathbf{p}}$ ,  $\mathbf{f}$ , and  $\mathbf{s}$ , respectively. For the 3-D case, the vectors  $\mathbf{v}$ ,  $\mathbf{p}$ ,  $\bar{\mathbf{p}}$  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. On the other hand, the seismogram is a vector of size  $N_{rec} \times (N_t + 1)$ , where  $N_{rec}$  is the number of receivers, and  $N_t = T/\Delta t$ , with  $\Delta t$  the time step. Lastly, the seismic source  $\mathbf{f}$  has dimension  $N_t$  for each shot.

#### 3.1 Classical Reverse Time Migration

Algorithm 1 presents the RTM implementation, which is one of the simplest ways to build the cross-correlated imaging condition. The colors in Algorithm 1 stand for host computations (black), data transfer (blue), and GPU calculations (red), which will be better explained in Sect. 3.3. The RTM needs as inputs a velocity field, a seismic source, and a set of seismograms,  $\{\mathbf{s}_1, \dots, \mathbf{s}_{N_{shots}}\}$  that contains information about the medium reflectivity. The computation of the imaging condition uses the source and receiver wavefield solutions to build the migrated seismic section that stacks the partial results over time ( $\mathbf{I}_{\Sigma_{n_\tau}}$ ), and over the number of seismograms ( $\mathbf{I}_{\Sigma_{shot\_id}}$ ). We compute the source wavefield by solving the wave equation with the independent term being the seismic source and storing it in disk for further access (step 10 in red). On the other hand, the recorded seismograms induce the computation of the receiver wavefield. At the end of Algorithm 1, we obtain the discrete seismic image  $I \in \mathbb{R}^N$ , where the amplitude variations represent physical properties changes. Both source and receiver wavefields can be obtained by solving the wave equation propagation over a temporal loop (the inner loops of Algorithm 1 - lines 7 and 14) for each *shot\_id* (loop in line 4). The shot refers to the seismic source that starts the wave propagation, and each one is localized in the domain represented by the finite-difference grid.

A computational implementation of absorbing boundary conditions (ABCs) leads to non-spurious reflections on the truncated domain. Among the several options in the literature, the Convolutional Perfectly Matched Layer (CPML) [9, 17] and the damping factors for plane waves introduced by [4] are the most common. Although unusual in wave propagation simulation studies, the RBCs, first introduced by [6], can also be employed in seismic imaging methods based on the two-way wave equation, such as the RTM and FWI [5, 6, 13, 16].

#### 3.2 Reverse Time Migration with Wavefield Reconstruction

Storing and accessing the source wavefield into and from the hard disk is computationally demanding. For instance, the disk requirements to store the source

**Algorithm 1.** Reverse Time Migration

---

**Require:**  $\mathbf{v}$ ,  $\{\mathbf{s}_1, \dots, \mathbf{s}_{N_{shots}}\}$ , and  $\mathbf{f}$

- 1: **function** RTM( vector  $\mathbf{v}$ , vectors  $\{\mathbf{s}_1, \dots, \mathbf{s}_{N_{shots}}\}$ , vector  $\mathbf{f}$  )
- 2:   read  $\mathbf{v}$ ,  $\mathbf{f}$ , and  $\{\mathbf{s}_1, \dots, \mathbf{s}_{N_{shots}}\}$
- 3:   initialize image condition  $\mathbf{I}_{\Sigma shot\_id} = 0$
- 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 equation (1) ▷ source wavefield
- 10:       store  $\mathbf{p}_{n_t}$  for all  $n_t$
- 11:     **end for**
- 12:     read  $\mathbf{s}_{shot\_id}$
- 13:     initialize  $n_\tau = 0$ , and  $\mathbf{I}_{\Sigma \tau} = 0$
- 14:     apply initial conditions for  $i_\tau = 0$
- 15:     **for**  $i_\tau = 1$  to  $N_t$  **do**
- 16:        $n_\tau = N_t - (n_\tau + i_\tau * \Delta \tau)$  ▷ reverse time
- 17:       read  $\mathbf{p}_{n_\tau}$
- 18:       solve equation (2) ▷ receiver wavefield
- 19:       calculate  $\mathbf{I}_{\Sigma n_\tau} = \mathbf{I}_{\Sigma n_\tau} + (\mathbf{p}_{n_\tau} \bar{\mathbf{p}}_{n_\tau}) / (\mathbf{p}_{n_\tau} \mathbf{p}_{n_\tau})$  ▷ imaging condition
- 20:     **end for**
- 21:     stack  $\mathbf{I}_{\Sigma shot\_id} = \mathbf{I}_{\Sigma shot\_id} + \mathbf{I}_{\Sigma n_\tau}$  ▷ stacking
- 22:   **end for**
- 23:    $\mathbf{I} \leftarrow \mathbf{I}_{\Sigma shot\_id}$
- 24:   store  $\mathbf{I}$
- 25: **end function**

---

wavefield propagation for the 3-D case is  $4 \times N_{shots} \times N_t \times N$  Bytes, where the value 4 stands for single precision representation of a real number. Considering a hypothetical scenario of the wavefield propagation in a grid of  $200 \times 200 \times 200$  grid points during 6.0 s step-wised of 0.5 ms leads to a disk storage demand of  $\approx 178.81$  GB per shot. Executing the same example in parallel considering 20 shots elevates the disk requirements to 3.5 TB.

Although compression techniques [1, 10, 11, 14], decimation strategies based on the Nyquist theory [22, 25] and checkpoint methods [23] reduces persistent storage, applications of RTM for large scale for frequencies up to 20.0 Hz is still challenging [19]. Another way to overcome this issue, explored in this work, is to reconstruct the wavefield from information generated during the first RTM part, that is, forward wave propagation [16]. To reconstruct the source wavefield, we implement the IVR methodology first explored by [7] and [23]. The IVR proposed by [23] stores temporary states of the wavefield known as checkpoints. Such states are after used for recursive re-computations of the source wavefield. The complete reconstruction of the wavefield can be achieved by keeping all energy in the system. However, unwanted signals come from the boundary due to the absence of attenuated layers in truncated domains. This issue can be

circumvented by generating incoherent signals coming from the boundary, as explored in [6], by introducing boundaries with randomized velocities.

The RBC proposed by [6] is based on the idea that what matters for the calculation of the RTM imaging condition is the coherent reflections coming from the boundaries. Thus, [6] proposed to introduce a random component to the velocity field at the boundaries. Notice that the random velocity field has to respect the numerical stability constraint of the FDM. It is expected that the random source wavefield coming from the boundaries does not coherently correlate with the receiver wavefield. Besides, a smoother transition from the inner domain to the boundaries is ideal. The smooth transition will avoid unwanted immediate reflections of the randomized area. One way to build a smooth transition area is by multiplying coefficients  $c_i$  to the random vector velocity  $\mathbf{v}$  in the normal direction to the boundaries, where the index  $i \in [1, \dots, N_a]$  with  $N_a$  been the boundary thickness size. The coefficients are responsible for decreasing down the velocities values, and [21] suggested the values computed by linear and Gaussian functions.

Further, this strategy does not impose an extra cost on the wave equation calculation. An alternative way to avoid coherent signals coming from the boundaries is presented by [13], where they used an extra viscoacoustic wave equation in the boundaries to attenuate the wavefield. In this work, we employ the strategy presented in [21]. Details of the RBC algorithm can be observed in [6]. Here, we will describe the modifications for Algorithm 1 aiming to eliminate the storage requirements of the forward-propagated wavefield.

First, we need a third second-order wave equation as follows:

$$\nabla^2 p^R(\mathbf{r}, \tau) - \frac{1}{v^2(\mathbf{r})} \frac{\partial^2 p^R(\mathbf{r}, \tau)}{\partial \tau^2} = 0, \quad (4)$$

where  $p^R$  is the reconstructed source wavefield defined in  $\Omega \subset \mathbb{R}^3$ . Boundary conditions can be set as Eqs. (1), and (2), that is  $p^R(\mathbf{r}, t) = 0$  on  $\partial\Omega$ . Lastly, the initial conditions are set as  $p^R(\mathbf{r}, 0) = p(\mathbf{r}, T)$ , and  $\partial p^R(\mathbf{r}, 0) / \partial t = \partial p(\mathbf{r}, T) / \partial t$  after solving Eq. 1, and  $\tau = T - t$  is the reversal time.

Algorithm 2 details the RTM that implements the source wavefield reconstruction. Again, the color pattern represents the host computations (black), data transfer (blue), and GPU calculations (red), which will be better explained in Sect. 3.3. We use the vector  $\mathbf{p}^R$  to represent the finite difference discretization of Eq. (4). The first part of the RTM with wavefield reconstruction calculates the source wavefield, and the last two instants of the wavefield are stored (line 11). After reading the stored wavefield instants, the second part of the algorithm, that calculates the receiver wavefield, also computes the reconstruction of the source wavefield  $\mathbf{p}^R$  by solving Eq. (4). Thus, the modified algorithm stores only two source wavefield panels instead of all panels for each  $n_t$ . This strategy comes with the additional cost of solving one extra wave equation.

**Algorithm 2.** Reverse Time Migration with Wavefield Reconstruction

---

**Require:**  $\mathbf{v}$ ,  $\{\mathbf{s}_1, \dots, \mathbf{s}_{N_{shots}}\}$ , and  $\mathbf{f}$

- 1: **function** RTM( vector  $\mathbf{v}$ , vectors  $\{\mathbf{s}_1, \dots, \mathbf{s}_{N_{shots}}\}$ , vector  $\mathbf{f}$  )
- 2:   read  $\mathbf{v}$ ,  $\mathbf{f}$ , and  $\{\mathbf{s}_1, \dots, \mathbf{s}_{N_{shots}}\}$
- 3:   create a RBC as Algorithm 2 from [6]
- 4:   initialize image condition  $\mathbf{I}_{\Sigma_{shot\_id}} = 0$
- 5:   **for**  $shot\_id = 1$  to  $N_{shots}$  **do**
- 6:     initialize  $n_t = 0$
- 7:     apply initial conditions for  $i_t = 0$
- 8:     **for**  $i_t = 1$  to  $N_t$  **do**
- 9:        $n_t = n_t + i_t * \Delta t$
- 10:       solve equation (1) ▷ source wavefield
- 11:       store  $\mathbf{p}_{n_t}$  for  $N_{t-1}$ , and  $N_t$
- 12:     **end for**
- 13:     initialize  $n_\tau = 0$ , and  $\mathbf{I}_{\Sigma_\tau} = 0$
- 14:     read  $\mathbf{s}_{shot\_id}$
- 15:     read  $\mathbf{p}_{N_t}$ , and  $\mathbf{p}_{N_{t-1}}$
- 16:     apply initial conditions for  $i_\tau = 0$
- 17:     **for**  $i_\tau = 1$  to  $N_t$  **do**
- 18:        $n_\tau = N_t - (n_\tau + i_\tau * \Delta \tau)$  ▷ reverse time
- 19:       solve equation (2) ▷ receiver wavefield
- 20:       solve equation (4) ▷ wavefield reconstruction
- 21:       calculate  $\mathbf{I}_{\Sigma_{n_\tau}} = \mathbf{I}_{\Sigma_{n_\tau}} + (\mathbf{p}_{n_\tau}^R \bar{\mathbf{p}}_{n_\tau}) / (\mathbf{p}_{n_\tau}^R \mathbf{p}_{n_\tau}^R)$  ▷ imaging condition
- 22:     **end for**
- 23:     stack  $\mathbf{I}_{\Sigma_{shot\_id}} = \mathbf{I}_{\Sigma_{shot\_id}} + \mathbf{I}_{\Sigma_{n_\tau}}$  ▷ stacking
- 24:   **end for**
- 25:    $\mathbf{I} \leftarrow \mathbf{I}_{\Sigma_{shot\_id}}$
- 26:   store  $\mathbf{I}$
- 27: **end function**

---

### 3.3 Hybrid OpenACC/MPI Implementation

The GPU programming model based on OpenACC directives aims to provide an easier way for scientific applications coding [12, 18]. Besides, compared to CUDA and OpenCL, OpenACC programming demands less coding efforts in heterogeneous environments with CPU+GPU [18, 20]. The OpenACC implementation deals 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 1 also details the host and GPU calculations and the communication between them. Notice that we use three different colors to represent the host computations (black), data transfer (blue), and GPU calculations (red). The first operations made by the host are data allocation followed by disk reading and storage of the velocity field and seismic source information in the vectors  $\mathbf{v}$ , and  $\mathbf{f}$ . These steps are represented in line 2 of Algorithm 1. Lines 8 and 9 show the GPU operations for the wave equation calculation once the necessary

information is transferred and allocated. Line 10 shows that the source wavefield needs to be moved during its calculation from the GPU to the host to be stored on the disk. In general, storing the source wavefield in a disk is needed because the GPU memory (or RAM) is insufficient to store it. The second part of the RTM algorithm (lines 12 to 21) moves back the source wavefield from host to GPU, calculates the receiver wavefield, and builds the imaging condition. The calculation of the receiver wavefield needs the seismograms stored on the disk. Thus, the host reads the seismogram from the disk and transfers it to the GPU. Algorithm 1 requires two data transfers for the velocity field and seismic source,  $N_{shots}$  data transfers for the seismograms, and  $2 \times N_t$  data transfers for the source wavefield.

The OpenACC implementation based on Algorithm 2 follows the same strategy presented in Algorithm 1. Remember that Algorithm 2 implements the wavefield reconstruction, and one extra wave equation is required for that. Because of that, its computational implementation does not fully store the source wavefield, only the last two time-frames. The data transfer based on the OpenACC implementation occurs between the two main stages of the RTM technique and not during the temporal loops as the Algorithm 1. Thus, Algorithm 2 requires only four data transfers between the GPU and host for the source wavefield. We use for both Algorithms 1 and 2 the ACC DATA COPYIN directive for transferring the data from the host to GPU. ACC DATA COPYOUT directive transfers the data from GPU to host. ACC DATA CREATE allocates necessary vectors in the GPU. For parallelization, we use the ACC LOOP directive.

Message Passing Interface (MPI) library manages 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 lines 4 and 5 of the Algorithms 1 and 2, respectively. Each MPI process can be assigned to a GPU or CPU node.

## 4 Numerical Experiments

In this section, we present the performance analysis of the 3-D RTM using two different computational platforms: a CPU cluster and a CPU-GPU machine. Both belongs to the Santos Dumont system at the National Scientific Computing Laboratory at Petrópolis/Brazil<sup>1</sup>. The CPU cluster has Intel Xeon E5-2695v2 Ivy Bridge processors with 2.4 GHZ and 24 cores per node, where the nodes are connected by an FDR (Forteen Data Rate) infiniband network (56 Gb/s) All the 24 cores have been used for the CPU experiments. On the other hand, the CPU-GPU cluster has a CPU Intel Skylake GOLD 6148, 2.4 GHZ with 24 cores and  $4 \times$  NVIDIA V100 per node. In this case, the nodes are connected by an EDR (Enhanced Data Rate) infiniband network (100 Gb/s). Both CPU and CPU-GPU nodes are supported by a Lustre filesystem v2.12.

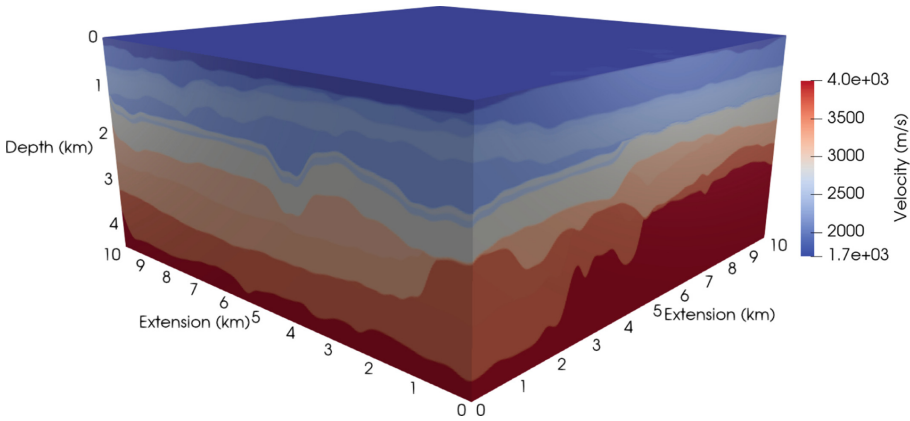
We have chosen the MODEL AF provided by the HPC4E Seismic Test Suite<sup>2</sup> for the 3-D RTM experiments. Figure 1 shows the velocity field provided by the

<sup>1</sup> [https://sdumont.lncc.br/support\\_manual.php?pg=support](https://sdumont.lncc.br/support_manual.php?pg=support).

<sup>2</sup> <https://hpc4e.bsc.es/downloads/hpc-geophysical-simulation-test-suite>.



HPC4E benchmark defined as MODEL AF. The MODEL AF is a model designed as a set of 15 layers with constant velocity values and flat topography. Besides, the velocity parameter model (velocity field) covers an area of  $10 \times 10 \times 4.5$  km. We have used a  $501 \times 501 \times 235$  grid size with 25.0 m of grid space to represent the velocity field. Notice that the grid size for the 3-D cases includes  $N_a = 50$  and half of the finite difference stencil length at the top of the velocity model to simulate the free surface. We used the Ricker seismic source [24] with 20 Hz cutoff frequency placed near the surface. The total acquisition time is 6.0 seconds step-wised of 1.0 ms. The experiments consist of running the RTM implementations for a single shot and a seismic acquisition and presenting the execution times, disk requirements, and speed-ups.



**Fig. 1.** 3-D velocity field provided by the HPC4E Seismic Test Suite.

*Single Shot Experiment:* The first experiment consists of executing the RTM applications for a single seismic source (single shot) located at [5000, 5000] m. The RTM follows the implementations presented in Algorithms 1 and 2. The seismograms for the RTM are represented by the seismic signals recorded in a seismic survey. The receiver geometry of the seismogram follows the expressions:

$$r_x = 25.0(i - 1) + 1012.5 \text{ with } i = 1, \dots, 320, \quad (5)$$

$$r_y = 25.0(j - 1) + 1012.5 \text{ with } j = 1, \dots, 320, \quad (6)$$

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

Table 1 shows the average time execution and the hard disk requirements for the 3-D RTM implementations. The RTM based on Algorithm 1 requires the full storage of the source wavefield. Nevertheless, instead of storing the source wavefield for every time step ( $\Delta t$ ) 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  $\Delta t_{nyq}$  is defined as,

$$\Delta t_{nyq} = \frac{1}{2(f_{max} - f_{min})}, \quad (7)$$

where  $f_{max}$  and  $f_{min}$  are the highest and lowest frequency of the seismic source. For the Ricker wavelet that we use for the RTM test case,  $f_{max} = 100.0$  Hz and  $f_{min} = 0.0$  Hz. Thus, the Nyquist time step based on Eq. 7 is  $\Delta t_{nyq} = 10.0$  ms against to  $\Delta t = 1.0$  ms, 10 times bigger than the FDM time step.

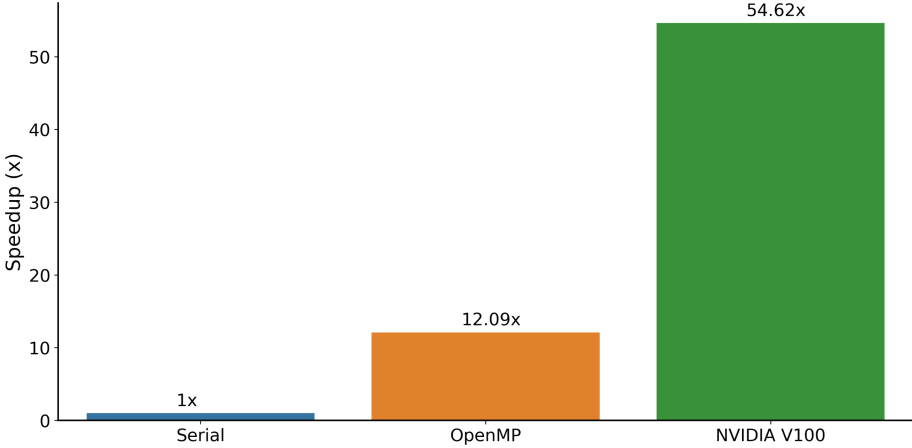
Hence, Algorithm 1 which implements the wavefield storage requires 132.062 GB of hard disk for NVIDIA V100 against 0.439 GB of the wavefield reconstruction implementation (Algorithm 2). The best average execution time refers to the RTM that implements the wavefield reconstruction. The RTM with wavefield reconstruction is  $\approx 2.11\times$  faster than the wavefield storage implementation on NVIDIA V100. Although the storage demand decreases drastically from 132.062 GB to 0.439 GB, the same does not occur in the total execution time. We observe that in both Algorithms 1 and 2 the forward and backward solutions of the wave equation take 80.0 s. In Algorithm 1, all the remaining time is spent in I/O. However, in Algorithm 2, the wavefield reconstruction takes most of the additional execution time.

**Table 1.** Comparison of hard disk and time requirements for the 3-D RTM implementation with wavefield storage, and wavefield reconstruction.

Method	Platform	Hard disk (GB)	Av. time (s) [variance (s)]
Wavefield Storage	NVIDIA V100	132.062	256.166 [46.810]
Wavefield Reconstruction	NVIDIA V100	0.439	121.431 [1.371]

We also compare the RTM speedups across the platforms CPU Cluster and NVIDIA V100. For the OpenMP RTM implementation, we follow [1, 3], where we implement an MPI/OpenMP+vectorization strategy on multi-core machines. The implementation takes advantage of OpenMP directives to explore multiple cores parallelism, it supports Single-Instruction-Multiple-Data (SIMD) model and memory alignment to ensure vectorization. We can see in Fig. 2 that the OpenMP implementation speedup is 12.09, and the OpenACC implementation speedup is 54.62. All the implementations for the platform comparisons are based on Algorithm 2 which describes the RTM with wavefield reconstruction and requires minimum I/O. Thus, the performance of the RTM implementation with OpenACC is  $4.52\times$  OpenMP implementation.

*Seismic Survey Experiment:* The final experiments consider running the RTM for a survey geometry with 1681 seismograms. The geometry acquisition for the seismic survey follows the expressions:



**Fig. 2.** Reverse Time Migration speedup across the platforms Santos Dumont CPU Cluster and NVIDIA V100 for the  $501 \times 501 \times 235$  grid.

$$s_x = 200.0(i - 1) + 1000.0 \text{ with } i = 1, \dots, 41, \quad (8)$$

$$s_y = 200.0(j - 1) + 1000.0 \text{ with } j = 1, \dots, 41, \quad (9)$$

where, the pair  $[s_x, s_y]$  meters represents the seismic source locations near the surface.

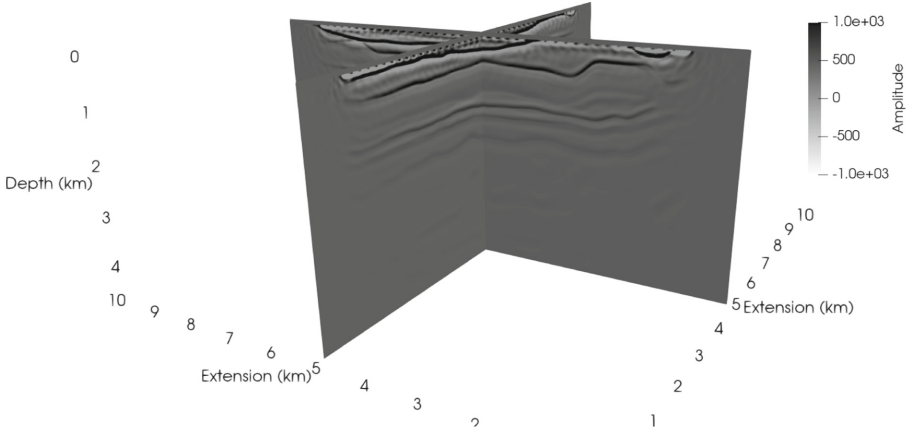
Firstly, we compare the total time of executing the RTM on CPU Cluster and NVIDIA V100. For the CPU cluster, we consider 16 nodes, where each node has 24 physical cores. Considering the GPU machine, we set 4 nodes because each one has 4 NVIDIA V100 adding up to 16 GPUs. Table 2 shows the total time of executing the RTM with wavefield reconstruction on CPU cluster and NVIDIA V100. We can see that the OpenACC implementation performs better than the OpenMP implementation for the same strategy which eliminates I/O related to the source wavefield. In this experiment, the RTM considering 16 NVIDIA GPUs is 7.62 times faster than the RTM running on 16 CPUs nodes.

**Table 2.** Comparison of the total execution time of the 3-D RTM implementation with wavefield reconstruction on CPU cluster and NVIDIA V100.

Method	Platform	Total time
Wavefield reconstruction	CPU cluster	1259 min 22.560 s
Wavefield reconstruction	NVIDIA V100	170 min 56.420 s

Figure 3 shows the stacked seismic image for one migrated shot of the 3-D HPC4E Seismic Test Suite benchmark. We generated the observed seismogram

by simulating the wave propagation and recording the seismic signals at the locations following the Eqs. 5 and 6 near the surface at 25.0 m in depth. The survey acquisition follows the geometry expressed in Eqs. 8 and 9 and takes into account 1681 shots.



**Fig. 3.** Seismic image for the 3-D HPC4E Seismic Test Suite.

## 5 Conclusions

This work studies RTM algorithms for 3-D environments that mitigate the source wavefield’s storage and explores hybrid architectures for speeding-up seismic applications. We eliminate the need of storing the source wavefield by reconstructing it through IVR based on the RBC. The RBC mitigates calculations on the artificial boundaries simplifying coding compared to versions with damping layers. Our algorithmic choices benefit computational architectures like GPUs. For instance, our numerical experiments show that the RTM based on the wavefield reconstruction performed better on the NVIDIA V100 than on Intel Xeon multi-CPU platforms for 3-D applications. Besides, the 3-D RTM algorithms based on the wavefield reconstruction demand less storage and are faster than the classical RTM storing the source wavefield. We also compare the RTM execution time with wavefield reconstruction for a seismic survey with 1681 shots. In this case, the RTM takes advantage of multi-GPUs and multi-CPUs to run the entire application. The RTM for multi-GPUs is 7.62 times faster than the RTM for multi-CPU platforms. We use high-level programming models such as OpenACC for the NVIDIA GPU and OpenMP for the Multi-CPU for all computational implementations. The high-level programming models allow code portability and little code interference on the optimized baseline version. We point out that the computational implementation based on the OpenACC library is

one of the simplest ways to produce fast and portable codes maintaining high-performance rates. Nevertheless, further performance gains can be obtained by using tailored optimizations, sacrificing portability.

**Acknowledgements.** 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. Barbosa, C.H., Coutinho, A.L.: Enhancing reverse time migration: hybrid parallelism plus data compression. In: Proceedings of the XLI Ibero-Latin-American Congress on Computational Methods in Engineering. ABMEC (2020)
2. Barbosa, C.H., Coutinho, A.L.: Seismic modeling and migration with random boundaries on the NEC SX-Aurora TSUBASA. arXiv preprint [arXiv:2204.03380](https://arxiv.org/abs/2204.03380) (2022)
3. Barbosa, C.H., et al.: A workflow for seismic imaging with quantified uncertainty. *Comput. Geosci.* **145**, 104615 (2020)
4. Cerjan, C., Kosloff, D., Kosloff, R., Reshef, M.: A nonreflecting boundary condition for discrete acoustic and elastic wave equations. *Geophysics* **50**(4), 705–708 (1985)
5. Clapp, R.G.: Reverse time migration: saving the boundaries. Stanford Exploration Project 137 (2008)
6. Clapp, R.G.: Reverse time migration with random boundaries. In: SEG Technical Program Expanded Abstracts 2009, pp. 2809–2813. Society of Exploration Geophysicists (2009)
7. Faria, E.: Migração antes do empilhamento utilizando propagação reversa no tempo, January 1986. <http://www.cpgg.ufba.br/pgeof/resumos/gfm/gfm0056a.html>
8. Givoli, D.: Time reversal as a computational tool in acoustics and elastodynamics. *J. Comput. Acoust.* **22**(03), 1430001 (2014)
9. Komatitsch, D., Martin, R.: An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. *Geophysics* **72**(5), SM155–SM167 (2007)
10. Kukreja, N., Hükelheim, J., Louboutin, M., Hovland, P., Gorman, G.: Combining checkpointing and data compression to accelerate adjoint-based optimization problems. In: Yahyapour, R. (ed.) Euro-Par 2019. LNCS, vol. 11725, pp. 87–100. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-29400-7\\_7](https://doi.org/10.1007/978-3-030-29400-7_7)
11. Kukreja, N., Hükelheim, J., Louboutin, M., Washbourne, J., Kelly, P.H., Gorman, G.J.: Lossy checkpoint compression in full waveform inversion: a case study with ZFPv0. 5.5 and the overthrust model. *Geosci. Model Dev.* **15**(9), 3815–3829 (2022)
12. Kushida, N., Lin, Y.-T., Nielsen, P., Le Bras, R.: Acceleration in acoustic wave propagation modelling using OpenACC/OpenMP and its hybrid for the global monitoring system. In: Wienke, S., Bhalachandra, S. (eds.) WACCPD 2019. LNCS, vol. 12017, pp. 25–46. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-49943-3\\_2](https://doi.org/10.1007/978-3-030-49943-3_2)
13. Li, Q., Fu, L.Y., Wu, R.S., Du, Q.: Efficient acoustic reverse time migration with an attenuated and reversible random boundary. *IEEE Access* **8**, 34598–34610 (2020)

14. Lindstrom, P., Chen, P., Lee, E.J.: Reducing disk storage of full-3D seismic waveform tomography (F3DT) through lossy online compression. *Comput. Geosci.* **93**, 45–54 (2016)
15. Liu, H., et al.: The issues of prestack reverse time migration and solutions with graphic processing unit implementation. *Geophys. Prospect.* **60**(5), 906–918 (2012)
16. Nguyen, B.D., McMechan, G.A.: Five ways to avoid storing source wavefield snapshots in 2D elastic prestack reverse time migration. *Geophysics* **80**(1), S1–S18 (2015)
17. Pasalic, D., McGarry, R.: Convolutional perfectly matched layer for isotropic and anisotropic acoustic wave equations. In: SEG Technical Program Expanded Abstracts 2010, pp. 2925–2929. Society of Exploration Geophysicists (2010)
18. Qawasmeh, A., Hugues, M.R., Calandra, H., Chapman, B.M.: Performance portability in reverse time migration and seismic modelling via OpenACC. *Int. J. High Perform. Comput. Appl.* **31**(5), 422–440 (2017)
19. Schuster, G.T.: *Seismic Inversion*. Society of Exploration Geophysicists, Tulsa (2017)
20. Serpa, M.S., et al.: Energy efficiency and portability of oil and gas simulations on multicore and graphics processing unit architectures. *Concurr. Comput. Pract. Exp.* **33**(18), e6212 (2021)
21. Silva, K.C.: Modelagem, mrt e estudos de iluminação empregando o conceito de dados sísmicos blended, January 2012. <http://www.coc.ufrj.br/pt/dissertacoes-de-mestrado/112-msc-pt-2012/2265-karen-carrilho-da-silva>
22. Sun, W., Fu, L.Y.: Two effective approaches to reduce data storage in reverse time migration. *Comput. Geosci.* **56**, 69–75 (2013)
23. Symes, W.W.: Reverse time migration with optimal checkpointing. *Geophysics* **72**(5), SM213–SM221 (2007)
24. Wang, Y.: Frequencies of the Ricker wavelet. *Geophysics* **80**(2), A31–A37 (2015)
25. Zand, T., Malcolm, A., Gholami, A., Richardson, A.: Compressed imaging to reduce storage in adjoint-state calculations. *IEEE Trans. Geosci. Remote Sens.* **57**(11), 9236–9241 (2019)
26. Zhou, H.W., Hu, H., Zou, Z., Wo, Y., Youn, O.: Reverse time migration: a prospect of seismic imaging methodology. *Earth Sci. Rev.* **179**, 207–227 (2018)