Seismic imaging is a computationally demanding and data-intensive activity in the oil and gas industry. Reverse Time Migration (RTM) used in seismic applications needs to store the forward-propagated wavefield (or source wavefield) on disk. Aiming to mitigate the storage demand, we develop an RTM that implements the source wavefield reconstruction by introducing a new wave equation to the problem. We adjust the initial and boundary conditions to take advantage of the properties of random boundary conditions (RBC). The RBC does not suppress unwanted waves coming from the artificial boundary enabling the full wavefield recovery. Besides, it explores low correlations with non-coherent signals due to the random velocities in the boundary. We also develop compiler-guided implementations on a vector processor for seismic modeling and RTM, essential for Least-square Migration, Full Waveform Inversion, and Uncertainty Quantification applications. We test the seismic modeling and RTM on the 2-D Marmousi benchmark and 3-D HPC4E Seismic Test Suite. The numerical experiments show that the RTM which implements the wavefield reconstruction presents the best results in terms of execution time and hard disk demand. Lastly, the vector processor implementation is the one that requires fewer code modifications compared to the optimized baseline versions of the seismic modeling and RTM and GPU implementations, particularly for large 3D grids.

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 [Zhou et al., 2018]. The 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, some 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, wavefield storage, and hardware acceleration are some of the main challenges concerning RTM [Zhou et al., 2018]. 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 [Komatitsch and Martin, 2007; Pasalic and McGarry, 2010] to deal with unwanted reflections due to truncated domains. On the other hand, the forward-propagated wavefield concerning 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 [Zhou et al., 2018]. Besides, because RTM is time-consuming and data-intensive [Barbosa and Coutinho, 2020; Qawasmeh et al., 2017; Serpa et al., 2021], RTM needs to be developed to take advantage of computer hardware technologies such as CPUs.
Efficient non-reflecting boundary conditions such as the PML can demand excessive numerical calculations beyond the wave equations [Li et al., 2020]. A way to overcome this issue is to use the random boundary conditions (RBC) proposed by [Clapp 2009]. Thus, instead of suppressing unwanted waves by inserting new equations into the problem, the methodology proposed by [Clapp 2009] is based on exploring low correlations with non-coherent signals coming from an artificial boundary with random velocities. Strategies to diminish input/output (I/O) related to the forward-propagated wavefield storage or its reconstruction are presented by [Faria, 1986, Symes, 2007, Clapp, 2008, 2009, Sun and Fu, 2013, Nguyen and McMechan, 2015, Barbosa and Coutinho, 2020, Li et al., 2020]. Among them, Sun and Fu [2013] 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, Barbosa and Coutinho [2020] studied the numerical impact of applying lossless and lossy compression to the forward-propagated wavefield of the RTM. They show that the careful use of high levels of data compression can significantly reduce the storage demand without hampering the final seismic images. Instead of storing the wavefield, its reconstruction is a viable possibility. This can be done by checkpoint methods [Faria, 1986, Symes, 2007], using wavefield recording around the boundary [Clapp, 2008, Nguyen and McMechan, 2015], or by initial value reconstruction (IVR) [Clapp, 2009, Nguyen and McMechan, 2015, Li et al., 2020, Nogueira and Porsani, 2021], on the other hand, developed a dynamic approach for the RTM that delimits the computational domain by the seismic wavefront. According to the authors, the proposed strategy leads to memory savings and reduced processing times compared to the conventional implementation of the RTM.

Independent of the RTM implementation strategy, all can use HPC techniques to boost their performance. Aiming to develop portable high-level directive-based codes across heterogeneous platforms for seismic imaging applications, [Qawasmeh et al., 2017] implemented the seismic modeling and RTM on a single as well multiple GPUs using a hybrid MPI+OpenACC approach. [Serpa et al., 2021] evaluated three different computational optimizations based on multicore and GPU architectures and investigated the performance, energy efficiency, and portability of the codes. Nevertheless, the storage demand issue remained in the RTM-based GPU implementations presented by the earlier research. For this, [Liu et al., 2012] implemented the RTM with RBC to diminish the storage demand need 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.

In this context, we developed a wave propagation modeler and an RTM approach for 2-D and 3-D environments that explore the main characteristics of the RBC to mitigate calculations on the artificial boundaries. Besides, our RTM implementation takes advantage of the RBC’s non-dissipative energy in the system to reconstruct the full forward-propagated wavefield with minimum storage by the IVR technique. Our implementation is particularly suited for the new generation of vector processors, the NEC SX-Aurora TSUBASA. Computational times and disk storage results for two algorithmic choices (with and without IO) are compared in different computational platforms: a CPU cluster, a CPU-GPU cluster, and the vector processor. 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 mathematical background concerning the seismic modeling and RTM technique in sections 2 and 3. Section 4 details the computational implementation for the seismic modeling and RTM along with optimizations on NEC SX-Aurora TSUBASA and NVIDIA Volta V100 platforms to highlight the differences between the two. In section 5, we present numerical experiments where we expose the execution time requirements, speedups, and hard disk demand for each computational implementation, as well as the seismic modeling and RTM outcomes. The paper ends with a summary of our main findings in section 6.

# 2 Seismic Modeling

In geophysical applications, seismic modeling is referred to as simulating the wave propagation in the Earth subsurface [Tago et al., 2012]. Understanding the propagation of seismic waves is one of the cornerstones of geophysical data processing, such as RTM and FWI [Tago et al., 2012, Igel, 2017]. For an acoustic medium, the wave equation is described by the second-order partial differential equation as follows,

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

(1)

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

Reverse Time Migration (RTM) is a depth migration technique based on the two-way wave equation, and an imaging condition \cite{zhou2018}. Solving the wave equation twice to build the imaging condition is necessary. The first solution, called forward-propagated wavefield, can be obtained by solving the equation (1). The second solution is obtained by solving the following equation:

\[
\nabla^2 \tilde{p}(\mathbf{r}, \tau) - \frac{1}{v^2(\mathbf{r})} \frac{\partial^2 \tilde{p}(\mathbf{r}, \tau)}{\partial \tau^2} = s(\mathbf{r}, \tau),
\]

where \( \tilde{p} \) is the backward-propagated wavefield, \( s(\mathbf{r}, \tau) \) is the seismogram recorded at the receivers positions \( \mathbf{r}_r \), and \( \tau = T - t \) is the reversal time evolution defined as in \cite{givoli2014}, where \( \tau \in [0, T] \). \( \tilde{p} \) is also defined in \( \Omega \subset \mathbb{R}^{n_d} \), and corresponding initial, and boundary conditions should be set.

Once we have the forward- and backward-propagated wavefields, the imaging condition can be calculated as:

\[
I(\mathbf{r}) = \frac{\int_0^T \tilde{p}(\mathbf{r}, \tau) \tilde{p}(\mathbf{r}, \tau) \, dt}{\int_0^T |\tilde{p}(\mathbf{r}, \tau)|^2 \, dt},
\]

where \( I(\mathbf{r}) \) is called source-normalized cross-correlated imaging condition. The source-normalized cross-correlation image in equation (3) has the same unit, scaling, and sign of the reflection coefficient \cite{zhou2018}.

4 Computational Implementation and Optimizations

Our numerical implementation of seismic modeling and RTM employs the explicit Finite Difference Method (FDM) to solve the acoustic wave equation. The finite difference stencil for equations (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, forward-propagated wavefield, backward-propagated wavefield, seismic source, and seismograms represented by the vectors \( \mathbf{v}, \mathbf{p}, \tilde{\mathbf{p}}, \mathbf{f}, \) and \( \mathbf{s} \), respectively. For the 3-D case, the vectors \( \mathbf{v}, \mathbf{p}, \tilde{\mathbf{p}} \) 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. The details of seismic modeling and RTM algorithms are presented in sections 4.1, 4.2, and 4.3. Section 4.4 presents the computational optimization and parallelization developed for the SX-Aurora TSUBASA vector engine and, for comparison purposes, GPUs systems.

4.1 Seismic Modeling

The description of the seismic modeling algorithm is quite simple once we have the discretized version of the wave equation. Algorithm [1] shows that for the acoustic wave equation, only two inputs are needed. The first one is a velocity field \( \mathbf{v} \) representing the spatial velocity distribution for the compressional wave (P-wave), and the second is a seismic source \( \mathbf{f} \) containing information about the seismic signature, called seismic source, responsible for initiating the wave propagation through the medium.

The wave equation propagation is solved over a temporal loop (the inner loop of Algorithm [1]) for each \( \text{shot}_\text{id} \) (loop in line 3). 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. The algorithm finishes recording a seismogram \( s_{\text{shot}_\text{id}} \) associated with each shot. A computational implementation of absorbing boundary conditions (ABCs) leads to spurious reflections on the truncated domain. Among the several options in the literature, the Convolutional Perfectly Matched Layer (CPML) \cite{komatitsch2007, pasalic2010} and the damping factors for plane waves introduced by \cite{cerjan1985} are the most common. Although unusual in wave propagation simulation studies, the RBCs, first introduced by \cite{clapp2009}, can also be employed in seismic imaging methods based on the two-way wave equation, such as the RTM and FWI \cite{clapp2008, clapp2009, nguyen2015, li2020}. Further discussions about the use of the RBC with the RTM are made in the wavefield reconstruction section 4.3.

4.2 Reverse Time Migration

Algorithm [1], detailed in section 4.1, is the kernel for the RTM algorithm presented in Algorithm [2]. Again, the two inputs are the velocity field and the seismic source. Besides, the RTM needs a set of seismograms, \( \{s_1, \ldots, s_{\text{numrec}}\} \) that contains information about the medium reflectivity. The computation of the imaging condition uses the forward-propagated, and
Algorithm 1 Seismic modeling

Require: \( v \) and \( f \)

1: function SEISMIC MODELING( vector \( v \), vector \( f \) )
2: read \( v \) and \( f \)
3: for shot\_id = 1 to \( N_{shots} \) do
4: initialize \( n_t = 0 \)
5: apply initial conditions for \( i_t = 0 \)
6: for \( i_t = 1 \) to \( N_t \) do
7: \( n_t = n_t + i_t \times \Delta t \)
8: solve equation (1)
9: record seismogram signals near surface
10: end for
11: store \( s_{shot\_id} \)
12: end for
13: end function

backward-propagated wavefield solutions to build the migrated seismic section that stacks the partial results over time \((I_{\sum \tau})\), and over the number of seismograms \((I_{\sum \text{shot}_{id}})\). We compute the forward-propagated 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 backward-propagated wavefield. At the end of Algorithm 1 we obtain the discrete seismic image \( I \in \mathbb{R}^{N_x \times N_y \times N_z} \), where the amplitude variations represent physical properties changes.

Algorithm 2 Reverse Time Migration

Require: \( v \), \( \{s_1, \ldots, s_{N_{shots}}\} \), and \( f \)

1: function RTM( vector \( v \), vectors \( \{s_1, \ldots, s_{N_{shots}}\} \), vector \( f \) )
2: read \( v \), \( f \), and \( \{s_1, \ldots, s_{N_{shots}}\} \)
3: initialize image condition \( I_{\sum \text{shot}_{id}} = 0 \)
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: \( n_t = n_t + i_t \times \Delta t \)
9: solve equation (1) \( \triangleright \) source wavefield
10: store \( p_{n_t} \) for all \( n_t \)
11: end for
12: initialize \( n_{\tau} = 0 \), and \( I_{\sum \tau} = 0 \)
13: apply initial conditions for \( i_{\tau} = 0 \)
14: for \( i_{\tau} = 1 \) to \( N_{\tau} \) do
15: \( n_{\tau} = N_{\tau} - (n_{\tau} + i_{\tau} \times \Delta \tau) \) \( \triangleright \) reverse time
16: read \( p_{n_{\tau}} \), and \( s_{shot\_id} \)
17: solve equation (2) \( \triangleright \) receiver wavefield
18: calculate \( I_{\sum \tau} = I_{\sum \tau} + (p_{n_{\tau}} \bar{p}_{n_{\tau}}) / (p_{n_{\tau}} p_{n_{\tau}}) \) \( \triangleright \) imaging condition
19: end for
20: stack \( I_{\sum \text{shot}_{id}} = I_{\sum \text{shot}_{id}} + I_{\sum \tau} \) \( \triangleright \) stacking
21: end for
22: \( I \leftarrow I_{\sum \text{shot}_{id}} \)
23: store \( I \)
24: end function

The RTM implementation presented in Algorithm 2 is one of the simplest ways to build the cross-correlated imaging condition. The algorithm involves calculating the wave equation twice and storing the forward-propagated wavefield to access it in the reverse way to correlate with the backward-propagated wavefield. However, storing and accessing the forward-propagated wavefield is computationally demanding. In this work, we implement the wavefield reconstruction based on the IVR technique with pseudorandomized wavefield proposed by [Clapp 2009] to deal with persistent storage of the forward-propagated wavefield. The next section 4.3 details the IVR strategy and modifications for Algorithm 2.
4.3 Wavefield Reconstruction

The basic RTM implementation as presented in Algorithm 2 suffers from persistent I/O due to the need to store the forward-propagated wavefield in disk for further access to calculate the imaging condition. One way to overcome this issue, explored in this work, is to reconstruct the forward-propagated wavefield from information generated during the first part of the RTM, that is, forward wave propagation [Nguyen and McMechan 2015]. To reconstruct the forward-propagated wavefield, we implement the IVR methodology first explored by Faria [1986] and Symes [2007]. The IVR proposed by Symes [2007] stores temporary states of the wavefield known as checkpoints. Such states are after used for recursive recomputations of the forward-propagated wavefield. On the other hand, Faria [1986] uses a single checkpoint to initiate the backpropagation of the wavefield. However, using this concept with non-reflective boundary conditions can result in an inefficient reconstruction of the forward-propagated wavefield due to signal attenuation in the boundary [Silva 2012]. 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 on the boundaries used to simulate truncated domains. One way to overcome this issue is generating incoherent signals coming from the boundary as explored in Clapp [2009] by introducing boundaries with randomized velocities.

The Random Boundary Condition (RBC) proposed by Clapp [2009] 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, Clapp [2009] 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 forward-propagated wavefield coming from the boundaries does not coherently correlate with the backward-propagated 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 \( i \in [1, \ldots, N_a] \) with \( N_a \) being the size thickness of the boundaries. The coefficients are responsible for slowing down the velocities values, and Silva [2012] showed that values between the linear and Gaussian functions, represented by equations (4) and (5), build the best coefficients, that is,

\[
g(x) = (N_a - x) \left( \frac{1}{N_a - 1} \right),
\]

\[
h(x) = \exp \left( -120(x - 1)^2 \left( \frac{1}{N_a - 1} \right)^2 \right),
\]

where, \( x \in [1, N_a] \). Let \( g \) and \( h \) be the discrete version of the functions \( g(x) \), and \( h(x) \) after numerical discretization, thus the damping coefficients assume values in \( h \leq c_i \leq g \) for \( i \in [1, \ldots, N_a] \).

It is worth mentioning that this strategy is effective and 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 Li et al. [2020], where they used an extra viscoacoustic wave equation in the boundaries to attenuate the wavefield. In this work, we employ the strategy presented by Silva [2012]. Details of the RBC algorithm can be observed in Clapp [2009]. Here, we will describe the modifications for Algorithm 2 aiming to eliminate the storage requirements of the forward-propagated wavefield.

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

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

where \( p^R \) is the reconstructed forward-propagated wavefield defined in \( \Omega \subset \mathbb{R}^{n_d} \). Boundary conditions can be set as equations (1), and (2), that is \( p^R (\mathbf{r}, 0) = 0 \) 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 equation (1) and \( \tau = T - t \) is the reversal time.

Algorithm 3 highlights in blue the main modifications in the basic RTM algorithm. We use the vector \( p^R \) to represent the finite difference discretization of equation (6). The first part of the RTM with wavefield reconstruction calculates the forward-propagated wavefield, and the last two moments of the wavefield are stored (line 11). After reading the stored wavefield moments, the second part of the algorithm that calculates the backward-propagated wavefield also calculates the reconstruction of the forward-propagated wavefield \( p^R \) by solving equation (6). Thus, the modified algorithm stores only two panels of forward-propagated wavefield instead of all panels for each \( n_t \). This strategy comes with the additional cost of solving one extra wave equation.

Figure 1 shows the representation of the propagation of the forward wavefield (Figures 1(A), 1(B), 1(C), and 1(D)) and its reconstruction (Figures 1(E), 1(F), 1(G), and 1(H)) in a constant velocity field of 2000 m/s based on Algorithm 5.
The experiment from Figure 1 implements the RBC, and, thus, it is possible to see the incoherent signals coming from the boundaries. Besides, the computational implementation for the Algorithm 3 maintains all energy inside the domain, allowing the complete reconstruction of the forward-propagated wavefield.

### Algorithm 3 Reverse Time Migration with Wavefield Reconstruction

**Require:** $v$, $\{s_1, \cdots, s_{N_{\text{shot}}}\}$, and $f$

1. function $\text{RTM}(\text{vector } v, \text{vectors } \{s_1, \cdots, s_{N_{\text{shot}}}\}, \text{vector } f)$
   2. read $v$, $f$, and $\{s_1, \cdots, s_{N_{\text{shot}}}\}$
   3. create a RBC as Algorithm 2 from Clapp [2009]
   4. initialize image condition $I_{\Sigma_{\text{shot}_i}} = 0$
   5. for $\text{shot}_id = 1$ to $N_{\text{shot}}$
      6. initialize $n_t = 0$
      7. for $i_t = 0$ to $N_t$
         8. $n_t = n_t + i_t * \Delta t$
         9. solve equation (1) $\triangleright$ source wavefield
         10. store $p_{n_t}$ for $N_t - 1$, and $N_t$
      11. end for
   12. initialize $n_\tau = 0$, and $I_{\Sigma_\tau} = 0$
   13. read $p_{N_t}$, and $p_{N_t - 1}$
   14. apply initial conditions for $i_\tau = 0$
   15. for $i_\tau = 1$ to $N_\tau$
      16. $n_\tau = N_\tau - (n_\tau + i_\tau * \Delta \tau)$ $\triangleright$ reverse time
      17. read $s_{\text{shot}_id}$
      18. solve equation (2) $\triangleright$ receiver wavefield
      19. solve equation (6) $\triangleright$ wavefield reconstruction
      20. calculate $I_{\Sigma_{n_\tau}} = I_{\Sigma_{n_\tau}} + (p_{n_\tau}^R \bar{p}_{n_\tau}) / (p_{n_\tau}^R p_{n_\tau}^R)$
      21. imaging condition
   22. end for
   23. stack $I_{\Sigma_{\text{shot}_id}} = I_{\Sigma_{\text{shot}_id}} + I_{\Sigma_{n_\tau}}$ $\triangleright$ stacking
   24. end for
   25. $I \leftarrow I_{\Sigma_{\text{shot}_id}}$
   26. store $I$
27. end function

### 4.4 Vector Processor and OpenACC Implementations

We develop versions for the seismic modeling and RTM on a traditional scalar CPU to get an optimized baseline implementation. Our optimized baseline implementations take advantage of the Single-Instruction-Multiple-Data (SIMD) model and memory alignment allocation to ensure vectorization [Barbosa et al., 2020]. Both versions were implemented in C language, and they were used as the start point for the vector processor and GPU implementations, based on OpenACC directives. The sections 4.4.1 and 4.4.2 present the main vector processor and GPU characteristics and details the computational optimizations and parallelization for the seismic applications of this work.

#### 4.4.1 Vector Processor Implementation

Our computational vector processor implementation is directed toward the NEC SX-Aurora TSUBASA vector processor. The SX-Aurora TSUBASA architecture consists of a vector engine (VE) equipped with a vector processor and a vector host (VH). In this architecture, the VE runs the entire application, while the VH is responsible for processing system calls invoked by the application [Komatsu et al., 2018]. Besides, the architecture avoids frequent data transfers between the VE and its VH. Developing scientific applications for the SX-Aurora TSUBASA is straightforward because no special coding is required, and the developers do not need to take care of the system calls.

In this sense, the implementations for the Algorithms 1, 2, and 3 presented in sections 4.1, 4.2, and 4.3 do not have any special code modification with respect to the optimized serial version that is our baseline implementation. Although the NEC SX-Aurora TSUBASA provides support for OpenMP implementation, we have been using automatic parallelization and vectorization activated by the NEC compilations flags -O4, mparallel, -fivdep, and -mparallel-innerloop.
4.4.2 OpenACC Implementation

The GPU programming model, based on OpenACC directives, aims to provide an easier way for scientific applications coding [Qawasmeh et al., 2017, Kushida et al., 2019]. Besides, compared to CUDA and OpenCL, OpenACC programming demands less coding efforts in heterogeneous environments with CPU+GPU [Qawasmeh et al., 2017, Serpa et al., 2021]. 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 4 details the host and GPU calculations and the communication between them for the seismic modeling. 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 $v$ and $f$. These steps are shown in lines 2 and 3 in Algorithm 4. Following, the main data, such as the vectors $v$ and $f$, are moved to the GPU (line 4). Lines 6 and 7 show the GPU operations for the wave equation calculation once the necessary information is transferred and allocated. The seismogram is the outcome of the wave equation simulation, and it is transferred from GPU to the host in line 9. The final operations of the host are seismogram storage and data deallocation in lines 10 and 11. Notice that for seismic modeling, only three communications are necessary. Because the velocity field and seismic source are information provided for the seismic
modeling, two transfers are made from the host to GPU. In the end, the host stores the seismogram after its transfer from GPU to host. We use 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.

Algorithm 4 Seismic Modeling GPU Implementation

Require: v, and f
1: function SEISMIC_MODELING_GPU( vector v, vector f )
2: allocate data variables ▷ Host computations
3: read v, and f
4: Move data to GPU ▷ Data transfer and allocation
5: for time = first to last do
6: solve wave equation (1) ▷ GPU computations
7: record seismogram
8: end for
9: update host with seismogram ▷ Data transfer and deallocation
10: store seismogram ▷ Host computations
11: deallocate data variables
12: end function

Most of the OpenACC implementations presented in Algorithm 4 for the seismic modeling are the same for the RTM algorithm. The differences between the seismic modeling and RTM algorithms are mainly related to data transfer. For instance, Algorithm 5 details the OpenACC implementation for the RTM which implements the wavefield storage (Algorithm 2). Again, we use three different colors to represent the host computations (green), data transfer (blue), and GPU calculations (red). The first part of Algorithm 5 moves the 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. The second part of the RTM algorithm (lines 13 to 19) moves back the source wavefield from host to GPU, calculates the receiver wavefield, and builds the imaging condition (lines 15 to 18). Algorithm 5 requires two data transfers for the velocity field and seismic source, \( N_{\text{shots}} \) data transferring for the seismograms, and \( 2 \times N_t \) data transferring for the source wavefield.

Algorithm 5 RTM GPU Implementation based on Algorithm 2

Require: v, f, and \{s_1, \cdots, s_{N_{\text{shots}}}\}
1: function RTM_GPU( vector v, vectors \{s_1, \cdots, s_{N_{\text{shots}}}\}, vector f )
2: allocate data variables ▷ Host computations
3: read v, and f
4: Move data to GPU ▷ Data transfer and allocation
5: for time = first to last do
6: solve wave equation (1) ▷ GPU computations
7: record source wavefield for every time
8: move source wavefield to host ▷ Data transfer
9: store source wavefield for each time ▷ Host computations
10: end for
11: read seismograms
12: Move seismogram to GPU ▷ Data transfer and allocation
13: for time = first to last do
14: set reversal time evolution
15: read source wavefield for each reversal time ▷ Host computations
16: Move source wavefield to GPU ▷ Data transfer and allocation
17: solve wave equation (2) ▷ GPU computations
18: calculate imaging condition
19: end for
20: update host with seismic image ▷ Data transfer and deallocation
21: store seismic image ▷ Host computations
22: deallocate all the data variables
23: end function
The OpenACC implementation based on Algorithm 3 is shown in Algorithm 6. Remember that Algorithm 3 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 5. Thus, Algorithm 6 requires only four data transfers between the GPU and host for the source wavefield. We use for both Algorithms 5 and 6 the same pragma directives that we use in seismic modeling. The ACC DATA COPYIN, ACC DATA COPYOUT for data transfer, ACC DATA CREATE for data allocation, and ACC LOOP for parallelization.

Algorithm 6 RTM GPU Implementation based on Algorithm 3

Require: v, f, and \{s_1, \ldots, s_{N_{shots}}\}

1: function RTM_GPU( vector v, vectors \{s_1, \ldots, s_{N_{shots}}\}, vector f )
2: allocate data variables ▶ Host computations
3: read v, and f
4: Move data to GPU ▶ Data transfer and allocation
5: for time = first to last do
6: solve wave equation (1) ▶ GPU computations
7: record source wavefield for every time
8: end for
9: update host with the last two wavefield timeframes ▶ Data transfer and deallocation
10: read seismograms
11: Move seismogram and wavefield timeframes to GPU ▶ Data transfer and allocation
12: for time = first to last do
13: set reversal time evolution
14: solve wave equation (2) ▶ GPU computations
15: solve wave equation (6)
16: calculate imaging condition
17: end for
18: update host with seismic image ▶ Data transfer and deallocation
19: store seismic image ▶ Host computations
20: deallocate all the data variables
21: end function

5 Numerical Results

In this section, we present the performance analysis of the seismic modeling and RTM using three different computational platforms: a CPU cluster, a CPU-GPU cluster, and a vector processor. The CPU cluster and CPU-GPU cluster are multicore machines from the Santos Dumont system at the National Scientific Computing Laboratory at Petrópolis/Brazil. The CPU cluster has 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 CPU Intel Skylake GOLD 6148, 2.4GHZ with 24 cores and 4 × NVIDIA Volta V100 per node. Finally, the vector processor is the NEC SX-Aurora TSUBASA Type 10B with 8 vector cores, and 48GB memory of 48GB. To show the results for the specified platforms, sections 5.1.1 and 5.2.1 present the performance analysis of the seismic modeling for different grid sizes and sections 5.1.2 and 5.2.2 show the analysis for the RTM for only one chosen grid size. As test cases, we have chosen the 2-D Marmousi velocity model [Versteeg, 1994] shown in Figure 2 for the 2-D experiments and the velocity field provided by the HPC4E Seismic Test Suite [3] for the 3-D experiments (Figure 3).

5.1 2-D Experiments

5.1.1 Seismic Modeling:

the 2-D Marmousi benchmark provides a velocity field that has a depth of 3.0 km and 9.3 km in the horizontal direction. Its original grid has 737 × 240 grid points, where the grid space has 12.5 meters. Considering the size thickness

\[1https://sdumont.lncc.br/support_manual.php?pg=supp\]
\[2https://www.hpc.nec/documents/guide/pdfs/Aurora_ISA_guide.pdf\]
\[3https://hpc4e.bsc.es/downloads/hpc-geophysical-simulation-test-suite\]
$N_a = 50$ except at the surface, where we set half of the finite difference stencil length to simulate the free-surface, the new grid size has $837 \times 294$ grid points. We create more three grids for the velocity field starting from the original one. The grids have $1574 \times 534$, $3048 \times 1014$, and $5996 \times 1974$ grid points. We use them to simulate seismic wave propagation (seismic modeling) and measure computational performance for different architectures. Each seismic modeling simulates a fixed-spread acquisition providing a single shot (seismogram) outcome. Thus, the simulation propagates the wavefield for 4 seconds with a time step of 0.5 milliseconds. We use the Ricker seismic source [Wang, 2015] of the cutoff frequency of 40 Hz that is placed near the surface.

We measure the absolute time for each run changing only the grid size. The OpenACC implementation of the seismic modeling for the NVIDIA Volta V100 is based on Algorithm 4. The vector processor implementation of the seismic modeling for the SX-Aurora TSUBASA is based on Algorithm 1 supported by the compilation flags presented in subsection 4.4.1. We ran each application ten times, and we took the time measurements for the NVIDIA Volta V100 and SX-Aurora TSUBASA vector engine platforms that are shown in Table 1. We can observe in Table 1 that SX-Aurora
Table 1: Seismic Modeling performance measurements for different grid sizes on the NVIDIA Volta V100 and SX-Aurora TSUBASA. The average time is calculated for ten execution time measurements.

| Grid Size    | NVIDIA Volta V100 | SX-Aurora TSUBASA |
|--------------|-------------------|-------------------|
| Average Time (s) | Variance (s)     | Average Time (s) | Variance (s) |
| Grid 1: 837 × 294 | 1.132             | 0.488             |
| Grid 2: 1574 × 534 | 1.875             | 0.912             |
| Grid 3: 3048 × 1014 | 2.615             | 2.416             |
| Grid 4: 5996 × 1974 | 6.962             | 7.885             |

TSUBASA performs better for all grids except for Grid 4. Besides, we observe fewer system fluctuations for the vector processor runs. On the other hand, the seismic modeling performed better on NVIDIA Volta V100 than SX-Aurora TSUBASA for the 5996 × 1974 grid. However, execution times can be considered of the same order because of the system fluctuations.

Figures 4 and 5 show the seismic modeling speedup across the platforms Santos Dumont CPU Cluster, NVIDIA Volta V100 and SX-Aurora TSUBASA vector processor for the 837 × 294 and 5996 × 1974 grids. We ran the optimized serial and OpenMP implementations of seismic modeling on the Santos Dumont CPU Cluster. The optimized serial implementation ran on a single core, and the OpenMP version ran on 24 cores on a single node. We have chosen the execution time of the optimized serial version as the reference time to calculate the speedup. Thus, the seismic modeling speedup for the optimized serial implementation is set as 1.0. Figure 4 shows that the OpenACC implementation for the NVIDIA Volta V100 platform had the worst speedup for the 837 × 294 grid, that is 6.7. The seismic modeling performed better on SX-Aurora TSUBASA for the same grid size, reaching the speedup value of 15.6. Notice that the OpenMP implementation is 1.6× better than the OpenACC implementation, and the vector processor implementation 1.44× better than the OpenMP implementation. On the other hand, the speedup conclusions drastically change for the 5996 × 1974 grid. The best speedup is from the OpenACC implementation for the NVIDIA Volta V100, which is 52.8. The speedup of the vector processor implementation on SX-Aurora TSUBASA is 46.8, and the speedup of the OpenMP implementation is 20.8. Thus, the OpenACC and vector processor implementation are 2.54×, and 2.24× better than the OpenMP implementation. Remember that SX-Aurora TSUBASA has 8 vector cores against 24 CPU cores of Santos Dumont CPU cluster.

Figure 4: Seismic modeling speedup across the platforms Santos Dumont CPU Cluster, NVIDIA Volta V100 and SX-Aurora TSUBASA Vector Engine for the 837 × 294 grid.
Lastly, Figure 5 shows the wavefield propagation for a single shot located at \( [x, z] = [4600.9, 12.5] \) meters in the Marmousi velocity field for the instants 0.25 s, 1.0 s, 1.5 s, 2.5 s, 3.0 s, and 3.5 s. Remembering, we use the Ricker seismic source \([\text{Wang, 2015}]\) of the cutoff frequency of 40 Hz. Besides, we consider the RBC in the boundaries for the wave propagation numerical simulation, aiming to eliminate the coherent reflections. In this experiment, we use the computational implementation of Algorithm 1 to generate the outcomes shown in Figure 6.

### 5.1.2 Reverse Time Migration

We also use the 2-D Marmousi benchmark for the RTM experiments. However, we have been using only one grid size, which is \( 1574 \times 534 \). The grid size considers \( N_a = 50 \) except at the top of the velocity model, where we set half of the finite difference stencil length to simulate the free surface. RTM experiment simulates a fixed-spread acquisition for one single shot. The seismic source is the Ricker wavelet \([\text{Wang, 2015}]\) of a cutoff frequency of 40 Hz placed near the surface. We generated the observed seismogram by modeling the wave propagation and recording the signal near the surface. The tests run the RTM application and calculate the average execution time for ten measurements on the SX-Aurora TSUBASA and NVIDIA Volta V100 for further comparison. Each algorithm is supported by the OpenACC and vector processor implementations presented in subsections 4.4.1 and 4.4.2.

Table 2 shows the average execution time and the hard disk requirements for the 2-D RTM implementations based on Algorithms 2, 3, 5, and 6. Algorithms 2 and 5 require 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 \([\text{Sun and Fu, 2013}]\) to store the wavefield at the Nyquist time step to reduce the amount of information. The Nyquist time step \( \Delta t_{\text{nyq}} \) is defined as,

\[
\Delta t_{\text{nyq}} = \frac{1}{2(f_{\text{max}} - f_{\text{min}})},
\]

where \( f_{\text{max}} \) and \( f_{\text{min}} \) are the highest and lowest frequency of the seismic source. For the Ricker wavelet that we use for the RTM test case, \( f_{\text{max}} = 100.0 \) Hz and \( f_{\text{min}} = 0.0 \) Hz. Thus, the Nyquist time step is \( \Delta t_{\text{nyq}} = 5.0 \) ms against to \( \Delta t = 0.5 \) ms for the finite difference time step.

Therefore, Algorithms 2 and 5 which implements the wavefield storage requires 2.638 GB of hard disk on the SX-Aurora TSUBASA and NVIDIA Volta V100 against 0.005 GB for the wavefield reconstruction implementations (Algorithms 3 and 6). This represents 527.6\( \times \) less information to be stored. Even using high efficient data compressors, such as the ZFP library \([\text{Lindstrom et al., 2016}]\), that level of storage savings can not be achieved. For instance, \([\text{Barbosa}\)
Figure 6: Propagation of the wavefield in the Marmousi velocity field with RBCs for the instants 0.25 s (A), 1.0 s (B), 1.5 s (C), 2.5 s (D), 3.0 s (E), and 3.5 s (F). The results were provided by Algorithm 1.

Table 2: Comparison of hard disk and time requirements for the 2-D RTM implementation with the wavefield storage, and the wavefield reconstruction.

| Method                  | Platform       | Hard disk (GB) | Av. Time (s) | Variance (s) |
|-------------------------|----------------|----------------|--------------|--------------|
| Wavefield Storage       | NVIDIA V100   | 2.638          | 7.969        | 0.149        |
| Wavefield Reconstruction| NVIDIA V100   | 0.005          | 3.500        | 1.415        |
| Wavefield Storage       | SX-Aurora TSUBASA | 2.638      | 9.016        | 0.068        |
| Wavefield Reconstruction| SX-Aurora TSUBASA | 0.005      | 4.489        | 0.002        |

and Coutinho [2020] showed that using the ZFP lossy compression with the tolerance of $10^{-6}$ demands $18.64 \times$ less information than the original data to be stored, which is far away from the hard disk requirements of the Algorithms 3 and 6 implementations.

According to the measurements shown in Table 2, the OpenACC implementation with wavefield reconstruction presents the best time execution on average, that is 3.5 s against 4.489 s of the vector processor implementation for Algorithm 3, 7.969 s of the OpenACC implementation for Algorithm 5, and 9.016 s of the vector processor implementation for Algorithm 2. Consequently, the RTM with wavefield reconstruction is $2.28 \times$ faster than the wavefield storage implementation for the NVIDIA Volta V100 and $2.01 \times$ faster than wavefield storage implementation for the SX-Aurora TSUBASA. Remember that Algorithms 3 and 6 implement one extra wave equation to reconstruct the source wavefield.

Concerning the speedup calculations, we calculate them only for the RTM based on Algorithms 3 and 6 that implement the wavefield reconstruction. The OpenACC implementation presents the best result as we can see in Figure 7. Again, our reference time is the optimized serial RTM code that was executed on the Santos Dumont CPU Cluster. We also ran the OpenMP implementation on the Santos Dumont CPU cluster using 24 CPU cores. Thus, the speedup of OpenMP implementation is 44.0 against 81.3 and 63.4 for the OpenACC and vector processor implementations. Therefore, the OpenACC speedup is $1.85 \times$ better than the OpenMP speedup and $1.28 \times$ better than the vector processor speedup. On the other hand, the vector processor speedup is $1.44 \times$ better than OpenMP speedup. Again, remember that SX-Aurora TSUBASA has 8 vector cores.
Figure 7: Reverse Time Migration speedup across the platforms Santos Dumont CPU Cluster, NVIDIA Volta V100 and SX-Aurora TSUBASA Vector Engine for the $1574 \times 534$ grid.

Figure 8 shows the partial seismic image for the 2-D Marmousi benchmark with one shot located at $[x, z] = [4600.0, 12.5]$ meters. For this experiment, we use the Ricker wavelet [Wang, 2015] of a cutoff frequency of 40 Hz and an observed seismogram generated by simulating a fixed spread acquisition recording the signals near the surface at 12.5 meters in depth. We generated the seismic image result shown in Figure 8 based on Algorithm 3 which implements the wavefield reconstruction strategy based on the IVR technique with the RBC.

Figure 8: Seismic image for one migrated shot of 2-D Marmousi benchmark.
We simulate the wavefield propagation based on Algorithm 1 for the vector processor optimizations and Algorithm
We have chosen the MODEL AF provided by the HPC4E Seismic Test Suite for the 3-D experiments. The MODEL
Volta V100 and SX-Aurora TSUBASA vector processor for the 501
AF is a 3-D model designed as a set of 15 layers with constant velocity values and flat topography. Besides, the
velocity parameter model covers an area of 10 \times 10 \times 4.5 \text{ km}. We have used two grid sizes for the seismic modeling
experiments, where which one has 501 \times 501 \times 235 and 901 \times 901 \times 416 grid points with 25.0, and 12.5 meters of
grid space, respectively. Notice that the grid sizes for the 3-D cases include \( N_a = 50 \) and half of the finite difference
stencil length at the top of the velocity model to simulate the free-surface. The experiments consist of the wavefield
propagation for a single shot located at \([5000,5000]\) meters for a maximum time of 6.0 seconds. We used the Ricker
seismic source \cite{Wang2015} of the cutoff frequency of 20 Hz placed near the surface. Concerning the acquisition
geometry, the seismic signals (seismograms) are recorded following the expressions:

\[
\begin{align*}
    r_x &= 25.0(i - 1) + 1012.5 \text{ with } i = 1, \ldots , 320, \\
    r_y &= 25.0(j - 1) + 1012.5 \text{ with } j = 1, \ldots , 320,
\end{align*}
\]

where, the pair \([r_x, r_y]\) meters represents the receiver locations on the surface.

We simulate the wavefield propagation based on Algorithm 1 for the vector processor optimizations and Algorithm
4 for the OpenACC implementation, both exposed in subsections 4.4.1 and 4.4.2. Table 3 details the average time
measurements of the seismic modeling on the NVIDIA Volta V100 and SX-Aurora TSUBASA vector engine platforms.
The second and fourth columns show the time measurements for the 501 \times 501 \times 235 grid size. The execution time
is almost the same on both platforms, and it differs by approximately 1.0 second. On the other hand, the wavefield
simulation on the 901 \times 901 \times 416 grid size took 330.363 s for the NVIDIA VOLTA V100 and 203.940 s for the
SX-Aurora TSUBASA, representing an execution time difference of 126.423 seconds. The results show that both
platforms have similar performances for the smallest grid. However, the SX-Aurora TSUBASA performs better for the
larger grid. Oppositely to the 2-D experiments, the 3-D implementations do not show relevant system fluctuations for
both platforms and grid sizes in the execution time measurements.

Table 3: Seismic Modeling performance measurements for different grid sizes on the NVIDIA Volta V100 and
SX-Aurora TSUBASA. The average time is calculated for ten execution time measurements.

| Grid 1: 501 \times 501 \times 235 | Grid 2: 901 \times 901 \times 416 |
|-------------------------------|-------------------------------|
| Average Time (s) | Variance (s) | Average Time (s) | Variance (s) |
|------------------|-------------|------------------|-------------|
| 42.1505          | 0.0464      | 43.132           | 0.018       |
| 330.363          | 0.0464      | 203.940          | 0.177       |

5.2 3-D Experiments

5.2.1 Seismic Modeling

We have chosen the MODEL AF provided by the HPC4E Seismic Test Suite for the 3-D experiments. The MODEL
AF is a 3-D model designed as a set of 15 layers with constant velocity values and flat topography. Besides, the
velocity parameter model covers an area of 10 \times 10 \times 4.5 \text{ km}. We have used two grid sizes for the seismic modeling
experiments, where which one has 501 \times 501 \times 235 and 901 \times 901 \times 416 grid points with 25.0, and 12.5 meters of
grid space, respectively. Notice that the grid sizes for the 3-D cases include \( N_a = 50 \) and half of the finite difference
stencil length at the top of the velocity model to simulate the free-surface. The experiments consist of the wavefield
propagation for a single shot located at \([5000,5000]\) meters for a maximum time of 6.0 seconds. We used the Ricker
seismic source \cite{Wang2015} of the cutoff frequency of 20 Hz placed near the surface. Concerning the acquisition
geometry, the seismic signals (seismograms) are recorded following the expressions:

\[
\begin{align*}
    r_x &= 25.0(i - 1) + 1012.5 \text{ with } i = 1, \ldots , 320, \\
    r_y &= 25.0(j - 1) + 1012.5 \text{ with } j = 1, \ldots , 320,
\end{align*}
\]

where, the pair \([r_x, r_y]\) meters represents the receiver locations on the surface.

We simulate the wavefield propagation based on Algorithm 1 for the vector processor optimizations and Algorithm
4 for the OpenACC implementation, both exposed in subsections 4.4.1 and 4.4.2. Table 3 details the average time
measurements of the seismic modeling on the NVIDIA Volta V100 and SX-Aurora TSUBASA vector engine platforms.
The second and fourth columns show the time measurements for the 501 \times 501 \times 235 grid size. The execution time
is almost the same on both platforms, and it differs by approximately 1.0 second. On the other hand, the wavefield
simulation on the 901 \times 901 \times 416 grid size took 330.363 s for the NVIDIA VOLTA V100 and 203.940 s for the
SX-Aurora TSUBASA, representing an execution time difference of 126.423 seconds. The results show that both
platforms have similar performances for the smallest grid. However, the SX-Aurora TSUBASA performs better for the
larger grid. Oppositely to the 2-D experiments, the 3-D implementations do not show relevant system fluctuations for
both platforms and grid sizes in the execution time measurements.

Figures 9 and 10 show the seismic modeling speedup across the platforms Santos Dumont CPU Cluster, NVIDIA
Volta V100 and SX-Aurora TSUBASA vector processor for the 501 \times 501 \times 235 and 901 \times 901 \times 416 grids. Again, the
execution time measurement of the optimized serial version is the reference time to calculate the speedups. Thus, the
seismic modeling speedup for the optimized serial implementation is set as 1.0. We ran the optimized serial, OpenMP,
and OpenACC implementations of seismic modeling on the Santos Dumont CPU-GPU Cluster. The optimized serial
implementation ran on a single core, and the OpenMP version ran on 24 cores on a single node. Figure 9 shows that the
OpenACC, and vector processor implementations have practically the same performance for the 501 \times 501 \times 235 grid.
The speedup of the seismic modeling for the OpenACC implementation on NVIDIA V100 is 36.22, and the speedup
for the vector processor implementation on SX-Aurora TSUBASA is 35.4. Both speedups are 4.27 \times faster than the
OpenMP seismic modeling implementation. Remember that the SX-Aurora TSUBASA vector processor has 8 vector
cores.

The seismic modeling implementation on SX-Aurora TSUBASA have the best performance for the 901 \times 901 \times 461
grid size as shown in Figure 10. The speedup of the OpenMP implementation is 9.3, the OpenACC implementation
speedup is 27.31, and the vector processor implementation speedup is 44.25. Thus, the OpenACC implementation
on NVIDIA V100 performed 1.62 \times worse than the vector processor implementation on SX-Aurora TSUBASA and
2.94 \times better than the OpenMP implementation. The speedup of the vector processor implementation for the SX-Aurora
TSUBASA is 4.75 \times better than the OpenMP implementation speedup.

Figure 11 shows four timeframes of the wavefield propagation in the 3-D velocity field provided by the HPC4E Seismic
Test Suite. The timeframes refer to the instants 0.6 s (upper left), 1.0 s (upper right), 1.5 s (lower left), 2.0 s (lower
Figure 9: Seismic modeling speedup across the platforms Santos Dumont CPU Cluster, NVIDIA Volta V100 and SX-Aurora TSUBASA Vector Engine for the $501 \times 501 \times 235$ grid.

right) for a single shot located at $[x, y, z] = [5000.0, 5000.0, 25.0]$ meters. In this experiment, we ran the computational implementation of Algorithm 1 for the 3-D case to generate the outputs shown in Figure 11.

5.2.2 Reverse Time Migration

We also use the MODEL AF for the 3-D RTM experiments, but only for one grid size, which is $501 \times 501 \times 235$. The RTM simulation follows the same parameter configurations presented for the seismic modeling. Grid space of 25.0 meters, shot location at $[5000, 5000]$ meters, maximum propagation time of 6.0 seconds, Ricker seismic source [Wang, 2015] of cutoff frequency of 20 Hz, and geometry acquisition following equations (8) and (9). We simulate the 3-D wave propagation to generate the observed seismogram by recording the signals near the surface. Again, the tests for the RTM consist of running the application based on Algorithms 2, 3, 5, and 6 and measuring the execution time on the SX-Aurora TSUBASA and NVIDIA Volta V100 platforms.

Table 4 shows the average time execution and the hard disk requirements for the 3-D RTM implementations for the $501 \times 501 \times 235$ grid. The RTM based on Algorithms 2 and 5 implement the wavefield storage for the Nyquist time step described by equation (7), and Algorithms 3 and 6 describe the wavefield reconstruction. We set the Nyquist time step value as $\Delta t_{nyq} = 10.0$ ms against $\Delta t = 1.0$ ms for the finite difference time step. Hence, Algorithms 2 and 5 which implements the wavefield storage requires 132.062 GB of hard disk for NVIDIA V100 and SX-Aurora TSUBASA against 0.439 GB of the wavefield reconstruction implementations (Algorithms 3 and 6). This represents 300.82× less information to be stored.

The best average executions times refer to the RTM that implements the wavefield reconstruction on NVIDIA Volta V100 and SX-Aurora TSUBASA. Nevertheless, the vector processor implementation of the RTM performed better than the OpenACC implementation, which is 108.582 s for the vector processor implementation against 139.786 s for the OpenACC implementation. Considering the wavefield storage implementations in Algorithms 2 and 5 the best average execution time is from NVIDIA Volta V100, where the OpenACC implementation took 256.166 s to run, and the vector processor 430.944 s. The RTM with wavefield reconstruction is 1.83× faster than the wavefield storage implementation for the NVIDIA Volta V100 and 3.08× faster than the wavefield storage implementation for the SX-Aurora TSUBASA. The average execution time for the RTM implementation with wavefield reconstruction on SX-Aurora TSUBASA is 3.98× better than the wavefield storage on the same platform, 1.29× and 2.36 better than the wavefield reconstruction and wavefield storage implementations for the NVIDIA Volta V100.

Comparing the RTM speedups across the platforms Santos Dumont CPU Cluster, NVIDIA Volta V100, and SX-Aurora TSUBASA, we can see in Figure 12 that the OpenMP implementation speedup is 12.09, the OpenACC implementation
6 Conclusions

This work studies RTM algorithms for 2-D and 3-D environments that mitigate the source wavefield’s storage 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 the NEC SX-Aurora TSUBASA vector processor. For instance, our numerical experiments show that the RTM based on the wavefield reconstruction performed better on the SX-Aurora TSUBASA than on Intel Xeon multi-CPPUs and NVIDIA GPU platforms for 3-D large applications. Besides, the 2-D and 3-D RTM algorithms based on the wavefield reconstruction demand less storage and are faster than the classical RTM storing the source wavefield. In this sense, the RTM based on the wavefield reconstruction demands $527.6\times$ less information to be stored than the RTM based on wavefield storage for the 2-D test case and $300.82\times$ less information to be stored for the 3-D test case. We also developed improved 2-D and 3-D seismic modeling algorithms with the RBC for the multi-CPU, GPU, and vector
Seismic Modeling and Migration on the SX-Aurora TSUBASA

A P REPRINT

Figure 11: Propagation of the wavefield in the 3-D velocity field provided by the HPC4E Seismic Test Suite for the instants 0.6 s (upper left), 1.0 s (upper right), 1.5 s (lower left), 2.0 s (lower right). The results were provided by Algorithm II.

processor platforms due to their importance to LS-RTM, FWI, and UQ applications. Again, the NEC SX-Aurora TSUBASA is better for large 3D cases. We use high-level programming models such as compilation directives for the NEC SX-Aurora TSUBASA, 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 compilation flags is the simplest way to produce fast and portable codes maintaining high-performance rates. Nevertheless, further performance gains can be obtained by using tailored optimizations, sacrificing portability.

Acknowledgments 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), and on the NEC SX-Aurora TSUBASA at NEC LATIN AMERICA S.A. is also acknowledged.

References

Hua-Wei Zhou, Hao Hu, Zhihui Zou, Yukai Wo, and Oong Youn. Reverse time migration: A prospect of seismic imaging methodology. Earth-science reviews, 179:207–227, 2018.

Dimitri Komatitsch and Roland Martin. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. Geophysics, 72(5):SM155–SM167, 2007.

Damir Pasalic and Ray McGarry. Convolutional perfectly matched layer for isotropic and anisotropic acoustic wave equations. In SEG Technical Program Expanded Abstracts 2010, pages 2925–2929. Society of Exploration Geophysicists, 2010.

Carlos HS Barbosa and Alvaro LGA 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, 2020.

Ahmad Qawasmeh, Maxime R Hugues, Henri Calandra, and Barbara M Chapman. Performance portability in reverse time migration and seismic modelling via openacc. The International Journal of High Performance Computing Applications, 31(5):422–440, 2017.

Matheus S Serpa, Pablo J Pavan, Eduardo HM Cruz, Rodrigo L Machado, Jairo Panetta, Antônio Azambuja, Alexandre S Carissimi, and Philippe OA Navaux. Energy efficiency and portability of oil and gas simulations on multicore and graphics processing unit architectures. Concurrency and Computation: Practice and Experience, page e6212, 2021.
Figure 12: Reverse Time Migration speedup across the platforms Santos Dumont CPU Cluster, NVIDIA Volta V100 and SX-Aurora TSUBASA Vector Engine for the $501 \times 501 \times 235$ grid.

Figure 13: Seismic image for one migrated shot of 3-D HPC4E Seismic Test Suite.

Qingqing Li, Li-Yun Fu, Ru-Shan Wu, and Qizhen Du. Efficient acoustic reverse time migration with an attenuated and reversible random boundary. IEEE Access, 8:34598–34610, 2020.

Robert G Clapp. Reverse time migration with random boundaries. In Seg technical program expanded abstracts 2009, pages 2809–2813. Society of Exploration Geophysicists, 2009.

EL Faria. Migração antes do empilhamento utilizando propagação reversa no tempo, January 1986. URL http://www.cpgg.ufba.br/pgeof/resumos/gfm/gfm0056a.html.

William W Symes. Reverse time migration with optimal checkpointing. Geophysics, 72(5):SM213–SM221, 2007.
Robert G Clapp. Reverse time migration: Saving the boundaries. Stanford Exploration Project, 137, 2008.

Weijia Sun and Li-Yun Fu. Two effective approaches to reduce data storage in reverse time migration. Computers & Geosciences, 56:69–75, 2013.

Bao D Nguyen and George A McMechan. Five ways to avoid storing source wavefield snapshots in 2d elastic prestack reverse time migration. Geophysics, 80(1):S1–S18, 2015.

Peterson Nogueira and Milton J Porsani. 3d reverse time migration using a wavefield domain dynamic approach. Journal of Applied Geophysics, 190:104345, 2021.

Hongwei Liu, Bo Li, Hong Liu, Xiaolong Tong, Qin Liu, Xiwen Wang, and Wenqing Liu. The issues of prestack reverse time migration and solutions with graphic processing unit implementation. Geophysical Prospecting, 60(5):906–918, 2012.

J Tago, V Cruz-Atienza, E Chaljub, Romain Brossier, O Coutant, S Garambois, V Prieux, S Operto, D Mercerat, J Virieux, et al. Modelling seismic wave propagation for geophysical imaging. In Seismic waves-research and analysis. IntechOpen, 2012.

Heiner Igel. Computational seismology: a practical introduction. Oxford University Press, 2017.

Dan Givoli. Time reversal as a computational tool in acoustics and elastodynamics. Journal of Computational Acoustics, 22(03):1430001, 2014.

Charles Cerjan, Dan Kosloff, Ronnie Kosloff, and Moshe Reshef. A nonreflecting boundary condition for discrete acoustic and elastic wave equations. Geophysics, 50(4):705–708, 1985.

Karen C Silva. Modelagem, mrt e estudos de iluminação empregando o conceito de dados sísmicos blended, January 2012. URL http://www.coc.ufrj.br/pt/dissertacoes-de-mestrado/112-msc-pt-2012/2265-karen-carrilho-da-silva.

Carlos HS Barbosa, Liliane NO Kunstmann, Rômulo M Silva, Charlan DS Alves, Bruno S Silva, MS Djalma Filho, Marta Mattoso, Fernando A Rochinha, and Alvaro LGA Coutinho. A workflow for seismic imaging with quantified uncertainty. Computers & Geosciences, 145:104615, 2020.

Kazuhiro Komatsu, Shintaro Momose, Yoko Isebe, Osamu Watanabe, Akihiro Musa, Mitsuo Yokokawa, Toshikazu Aoyama, Masayuki Sato, and Hiroaki Kobayashi. Performance evaluation of a vector supercomputer sx-aurora tsubasa. In SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, pages 685–696. IEEE, 2018.

Noriyuki Kushida, Ying-Tsong Lin, Peter Nielsen, and Ronan Le Bras. Acceleration in acoustic wave propagation modelling using openacc/openmp and its hybrid for the global monitoring system. In International Workshop on Accelerator Programming Using Directives, pages 25–46. Springer, 2019.

Roelof Versteeg. The marmousi experience: Velocity model determination on a synthetic complex data set. The Leading Edge, 13(9):927–936, 1994.

Yanghua Wang. Frequencies of the ricker wavelet. Geophysics, 80(2):A31–A37, 2015.

Peter Lindstrom, Po Chen, and En-Jui Lee. Reducing disk storage of full-3d seismic waveform tomography (f3dt) through lossy online compression. Computers & Geosciences, 93:45–54, 2016.