ARTICLE INFO:
Received : July 19, 2017
Revised : July 04, 2018
Accepted : September 06, 2018
CT&F - Ciencia, Tecnologia y Futuro Vol 8, Num 2 Dec. 2018. pages 75 - 82
DOI : https://doi.org/10.29047/01225383.83

ctyf@ecopetrol.com.co

COMPARATIVE ANALYSIS OF 3D RTM IMPLEMENTATION STRATEGIES FOR AN EFFICIENT USE OF MEMORY IN A SINGLE GPU

ANÁLISIS COMPARATIVO DE LAS ESTRATEGIAS DE IMPLEMENTACIÓN RTM 3D PARA EL USO EFICIENTE DE LA MEMORIA EN UNA GPU.

Salamanca, William-A[a]; Ramirez, Ana-B.[a]; Vivas, Flor-A.[b]*

ABSTRACT
Reverse-Time Migration (RTM) is a two-way wave-equation based method used to generate images of the Earth's subsurface. RTM has been successfully used in seismic imaging as it allows defining complex structural areas. However, RTM is a highly computational expensive algorithm that requires the computation of both the source and the receiver wavefields for each shot. Fortunately, numerical methods that compute the wave propagation using the wave equation are highly parallelizable, so they can take leverage on GPU features. However, the main problem of a GPU-RTM implementation is memory management. To take advantage of the GPU computing capabilities, the transfers to host RAM memory storage, or more expensive hard disk storage must be avoided. We present the analysis of three different strategies to implement RTM using only the memory available on a single GPU: (1) Stored wavefield checkpointing (2) Backpropagation of source wavefield using stored boundaries, and (3) Backpropagation of source wavefield using the two last snapshots and random boundaries, showing that the large amount of memory required in the first two strategies becomes a restriction over the model size. The last method (using random boundary conditions) is shown as a suggested solution to the memory problem of using a single GPU.

RESUMEN
La migración reversa en tiempo (RTM) es un método basado en la ecuación de onda bidireccional usado para generar imágenes del subsuelo. RTM ha sido empleado exitosamente en la exploración sísmica debido a que resuelve áreas de alta complejidad estructural. Sin embargo, RTM es un algoritmo con un alto costo computacional que requiere el cálculo del campo de la fuente y el campo de los receptores en cada disparo. Afortunadamente, los métodos numéricos que permiten la extrapolación del campo de onda son altamente paralelizables y se sacan provecho de la capacidad de cómputo de la GPU. Sin embargo, el principal problema de una implementación GPU-RTM es el manejo de memoria. Para sacar provecho de la capacidad de cómputo de la GPU, se evitaron las transferencias de memoria hacia la RAM del host u otras más costosas como las transferencias al disco duro. Se presenta el análisis de tres diferentes estrategias para implementar RTM usando únicamente la memoria disponible en una GPU: (1) Almacenar el campo en puntos de control, (2) Retropropagación del campo de la fuente almacenando las fronteras, y (3) Retropropagación del campo de la fuente usando los dos últimos *snapshots* y fronteras aleatorias, mostrando que la gran cantidad de memoria requerida por las dos primeras estrategias se convierte en una restricción sobre el tamaño del modelo. El último método (usando condiciones de frontera aleatoria) se presenta como la solución sugerida para el problema de memoria usando únicamente una GPU.

KEYWORDS / PALABRAS CLAVE

Graphical Processing Units | Reverse-Time Migration 3D | Wavefield Computation Strategies.
Unidades de Procesamiento Gráfico (GPU)| Migración Reversa en Tiempo 3D| Estrategias de cálculo del campo de onda.

AFFILIATION

[a]Universidad Industrial de Santander, carrera 27 calle 9, C.P 680002, Bucaramanga, Colombia,
[b] Ecopetrol - Instituto Colombiano del Petróleo, km 7 vía Bucaramanga- Piedecuesta, C.P 681011, Piedecuesta Colombia.
*email: william.salamanca@correo.uis.edu.co

1. INTRODUCTION

Reverse-Time Migration (RTM) is a seismic imaging method that uses the solution of the full wave equation to obtain imaging of complex areas, particularly those with abrupt changes of lateral and vertical velocities. It was initially proposed by Baysal [1], but only the last decade progress in computing technologies has allowed the implementation of the algorithm and its practical use in 3D depth seismic processing projects. In recent years, there has been growing interest in high-performance computing (HPC) architectures based on graphic processing units (GPUs) that assemble a considerable amount of computing devices with single instruction multiple thread (SIMT) cores [2]. HPC based on GPUs are suitable architectures for the implementation of RTM, because of the inherent parallelism on each time step in the algorithm [3]. In addition to the computing resources, RTM also demands large memory resources, as the source and receiver wavefields must be available for computing imaging condition. Thus, depending on the volume of the wavefields, the implementation can exceed the limit of a single device (GPU) memory. This implies the use of higher memory hierarchy levels, which degrade the code performance due to the data transfer overhead.

A different approach to overcome the GPU memory constraint is to use implementation strategies where memory requirements are reduced in exchange for increasing the number of computing operations because the source wavefield should be recomputed. Some strategies have been already implemented for CPU architectures, such as optimal checkpointing proposed by Dussaud et al. [4], which can be adapted to GPU architectures.

The aim of this work is to evaluate three different GPU implementation strategies of the RTM method, focused on memory management such that a 3D shot can be migrated in a single GPU, thus avoiding expensive transfers from (or to) host RAM or hard disk memory. The main contribution of our work is to demonstrate that the robust oil industry RTM algorithm can be effectively mapped to GPU based architectures. Our RTM implementation is based on a 3D Time Domain Finite Difference numerical scheme to solve the acoustic wave equation, with absorbing boundary conditions. The conclusions of this work about the GPU memory management in the analyzed strategies can be extended to simulations based on the same numerical scheme for other forms of wave equations, such as visco-acoustic, elastic or electro-magnetic.

This document is organized as follows: the RTM method is introduced in Section 2 and the implementation strategies of the RTM method in Section 3. Section 4 summarizes the main characteristics of the GPU architecture used for this work. Section 5 presents the performance results from the different RTM implementations and, finally, the conclusions and discussion are presented in Section 6. All the implementation strategies are assessed in terms of the resulting image quality, memory requirements, and execution times.

2. THEORETICAL FRAME

RTM is a mainstream method used to generate subsurface images in complex areas, which favors the identification of possible reservoir locations. RTM generates a reflectivity map R(x,y,z) by computing an imaging condition from the correlation between the source and the receiver wavefields. This condition is given by

$$R(x,y,z)=\sum_{\forall shots}\sum_{t=0}^{tmax} p_s(x,y,z,t)\times p_r(x,y,z,t) \quad (1)$$

where p_s *(x,y,z,t)* is the source wavefield, which is obtained by propagating the source through the medium; and p_r *(x,y,z,t)* is the receivers wavefield, which is obtained by backpropagating the seismic traces information. Both wavefields require a numerical solution of the wave equation that models the propagation of a seismic source through a medium. In this work, we use the constant density acoustic wave equation to model the propagation of a seismic source. The isotropic acoustic wave equation is given by

$$\frac{1}{v^2(x,y,z)}\frac{\partial^2 p(x,y,z,t)}{\partial t^2}=\nabla^2 p(x,y,z,t) = \frac{\partial^2 p(x,y,z,t)}{\partial x^2}+\frac{\partial^2 p(x,y,z,t)}{\partial y^2}+\frac{\partial^2 p(x,y,z,t)}{\partial z^2} \quad (2)$$

where ∇^2 is the Laplacian operator, and v is the p—wave velocity model of the subsurface. One of the most common methods to find the numerical solution of the wave equation is applying finite differences to the temporal derivative and compute the Laplacian operator with finite differences or pseudo-spectral methods. The 2nd order finite difference approximation in time for the temporal part of the equation is given by

$$\frac{p^{k-1}-2p^k+p^{k+1}}{\Delta t^2}=v^2\nabla^2 p^k \quad (3)$$

In Equation 3, p^k is the p wavefield in the time $t= k\Delta t$. The source wavefield can be computed from *t=0* to *t=tmax*, assuming a non-perturbed media in the two first time steps and knowing the source position and its wavelet signature. In the same way, the receiver wavefield can be back propagated from *t=tmax* to *t=0* assuming zero values in the last two time-snapshots and injecting the seismic traces as sources.

ABSORBING BOUNDARY CONDITIONS

Representing the wavefield on a memory-constrained computing system implies that only a reduced time-space part of the wavefield can be stored. This means that artificial boundaries are created in the border of the model. Depending on the numerical method, the effect of the border can be: periodicity, if pseudo-spectral methods are used; or reflections, if finite differences are used. In any case, the energy reaching the artificial boundaries must be vanished, so that it does not affect the migrated image. Several algorithms have been proposed to manage the artificial boundaries. Perfectly Matched Layer (PML) was used in this work for the first two strategies, as it is an efficient and simple method. The third strategy uses a random boundary method as proposed by Clapp [5]. In particular, for the PML

absorbing boundary conditions, we opted for the convolutional PML (CPML), proposed by Pasalic [6], which applies PML directly on the second-derivative wave equation in (2). The main idea of CMPL is to introduce a complex stretching parameter given by

$$s^i = 1 + \frac{\sigma_i}{\alpha_i + j\omega} \tag{4}$$

in the differential operators of the acoustic wave equation on each direction $i \in \{x,y,z\}$. σ_i and α_i are parameters that adjust the behavior of vanishing energy. The new differential operators in the wave equation are

$$\begin{aligned} \frac{\partial}{\partial i} &\rightarrow \frac{\partial}{\partial i} + \psi_i \\ \frac{\partial^2}{\partial i^2} &\rightarrow \frac{\partial^2}{\partial i^2} + \frac{\partial \psi_i}{\partial i} + \zeta_i \end{aligned} \tag{5}$$

where ψ_i and ζ_i are two additional wavefields that are computed recursively on each time step as follows

$$\begin{aligned} \psi_i^n &= a_i \psi_i^{n-1} + b_i \left(\frac{\partial}{\partial i}\right)^n \\ \zeta_i^n &= a_i \zeta_i^{n-1} + b_i \left[\left(\frac{\partial^2}{\partial i^2}\right)^n + \left(\frac{\partial \psi_i}{\partial i}\right)^n\right] \end{aligned} \tag{6}$$

In previous equations, a_i and b_i are given by.

$$\begin{aligned} a_i &= e^{(\sigma_i + \alpha_i)\Delta t} \\ b_i &= \frac{\sigma_i}{\sigma_i + \alpha_i}(a_i - 1) \end{aligned} \tag{7}$$

Now, substituting the second derivatives given by Equation 5 in the acoustic wave equation in (2), we have

$$\begin{aligned} &\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} + \frac{\partial \psi_x}{\partial x} + \frac{\partial \psi_y}{\partial y} + \frac{\partial \psi_z}{\partial z} + \\ &\zeta_x + \zeta_y + \zeta_z = \frac{1}{v^2}\frac{\partial^2 p}{\partial t^2} \end{aligned} \tag{8}$$

Equation 8 is applied near the model borders. To use this equation, the computation of a time snapshot follows the next three steps:

1. Update ψ_i^n using ψ_i^{n-1} and the first derivative of p^n.
2. Update ζ_i^n using ζ_i^{n-1}, the second derivative of p^n and the first derivative of ψ_i^n, computed in the previous step.
3. Use Equation 8 to compute p^{n+1}.

The fields ψ_i and ζ_i require extra memory for each PML border.

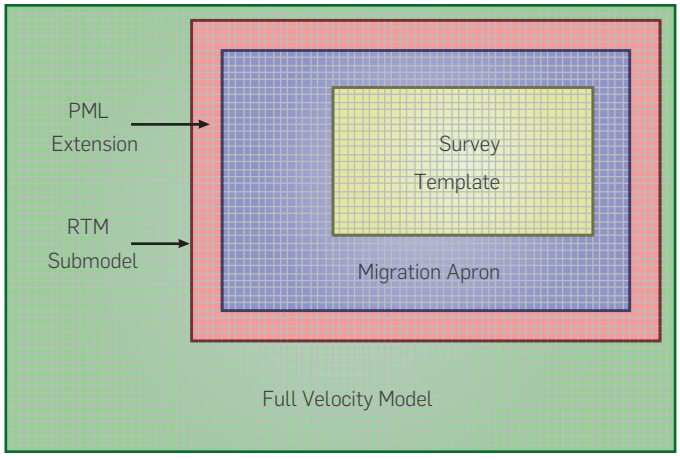

Figure 1. Top view for RTM shot migration sub-model

3. EXPERIMENTAL DEVELOPMENT

MEMORY MANAGEMENT STRATEGIES

The implementation of the RTM algorithm requires large amounts of memory in real seismic surveys. Naive implementations of this algorithm can easily exceed the GPU memory limits, as it is shown in **Table 1**; therefore, it is necessary to consider strategies to reduce the memory usage. These strategies recompute the wavefields when they are required, instead of storing the entire wavefields. Thus, memory-constrained architectures, such as GPUs, can be used at the cost of increasing the execution times of the algorithm.

In the RTM algorithm, each shot requires a sub-model size according to the patch size, the migration apron, and the absorbing boundary layer width, as shown in **Figure 1**. We estimate the memory capacity required by the RTM algorithm for a FDTD approximation of 8th order in space and 2nd order in time, an extended absorbing boundary layer *L=8* grid points, *tmax*=5s, single precision floating point representation, and a full migration apron. The velocity model size and the time step can be reduced depending on the central frequency *fq* of the source wavelet. This reduces significantly the memory requirements, but the numerical method must satisfy the stability and a minimum number of points per wavelength. **Table 1** shows the amount of required memory, and maximum spatial and time sampling steps for different central frequencies of the source wavelet, assuring a minimum number of points per wavelength of *ppwl = 3*. The information given in **Table 1** is for the SEG synthetic velocity model.

Table 1. Variation of the memory requirement to store the full wavefield for different frequencies fq of the source wavelet.

fq[Hz]	dh[m]	Nx	Ny	Nz	dt[ms]	nt	Wavefield Snapshot [MB	Full Wavefield [GB]	Wavefield Boundaries [GB]
3	160	85	85	26	16.17	309	0.75	0.23	0.06
6	80	169	169	50	8.08	618	5.71	3.53	0.67
12	40	337	337	99	4.04	1237	44.97	55.63	6.13
18	27	505	505	148	2.69	1855	150.97	280.06	21.72
24	20	673	673	197	2.02	2474	356.91	882.99	52.74
31	16	841	841	246	1.61	3092	695.96	2151.92	104.5

Note in Table 1 that for source wavelet of central frequency of 31 Hz, the computing system may require up to 2 TB of memory to store the source wavefield for all the 3092 time steps. This amount of memory could be available at low memory hierarchy levels, such as hard disk drive, although its use may cause very slow and inefficient execution of the RTM algorithm. To overcome this problem, various strategies have been reported [4]. In this work, we focus in evaluating such strategies to select those that can be used in industry applications using a single GPU to migrate a single RTM shot.

STRATEGY 1: STORING WAVEFIELD CHECKPOINTING

The first strategy uses the available memory in the GPU device to store checkpoints in the source modeling process. When the imaging condition requires a non-stored source wavefield snapshot, it is recomputed from the nearest checkpoint. Additional to uniform checkpoint distribution along the modeling time steps, optimal checkpointing strategies have also been reported in the literature [7],[8]. These strategies increase the computational complexity, while reducing the memory complexity to a logarithmic function of the total number of steps.

STRATEGY 2: BACKPROPAGATE THE SOURCE WAVEFIELD USING STORED BOUNDARIES

In the previous strategy, the source wavefield is recomputed in the forward direction starting from checkpointing snapshots. However, the source propagation is a reversible process itself, taking into account that we can solve for p^{i-1} in Equation 3 instead of solving for p^{i+1}. This allows us to propagate the source field up to tmax and then recompute it backwards, saving only the two last time steps with two memory buffers.

The main problem with this strategy is that the energy at the boundaries cannot be reversed, as the PML vanishes the energy reaching the model's border. This means that on a single time step, the field in the inner part of the model can be recovered, but the PML must be stored [9],[4]. Strategy 2 is an efficient implementation strategy, in terms of memory management, but the amount of memory used to store the wavefield at the boundaries can depend largely on the size of the model. For the velocity models shown in Table 1, and assuming FDTD approximation of 8th order in the spatial derivatives, the wavefield at the boundaries ranges from 0.06 GB up to 104.5 GB of memory. Currently, the most advanced GPU cannot handle such amount of memory, making this strategy impractical for industry applications using a single GPU.

This strategy can be extended to the RTM with a visco-acoustic modeling but the wavefield should be reconstructed including attenuation phenomena [10]

STRATEGY 3: BACKPROPAGATE THE SOURCE WAVEFIELD USING ONLY TWO LAST SNAPSHOTS AND RANDOM BOUNDARIES

The random boundaries technique proposed by Clapp [5] extends the velocity model with carefully generated random values. The goal of this technique is to generate a random reflection pattern of the energy reaching the model's border. Its two main characteristics are: first, the energy reflected by the boundaries at the source wavefield is not coherent with the receiver wavefield and, therefore, the final image will be affected only by a random noise; second, the energy is not vanished and the modeling process is now reversible with a stable scheme. The latter reduces significantly the memory requirements as the wavefield at the boundaries does not need to be stored, and the source wavefield only requires the last two snapshots to be recomputed. These features allow for successful application of strategy 3 to in industry applications where a 3D shot can be migrated with RTM in a single GPU without any problem with the memory available in the device..

GPU ARCHITECTURE

The RTM strategies under study in this work will be implemented on the Nvidia K40 GPU. Therefore, in this Section we describe only the GPU architecture, and the specifications of the CPU, host RAM and I/O bandwidth are irrelevant for the paper's main objective.

GPUs are parallel computing devices designed for video operations such as image rendering. These devices contain hundreds of cores based on a Single Instruction Multiple Thread (SIMT) architecture and have been adapted for scientific computation algorithms. Nowadays, general purpose applications can be developed using these devices with programming models and APIs like CUDA and OpenCL.

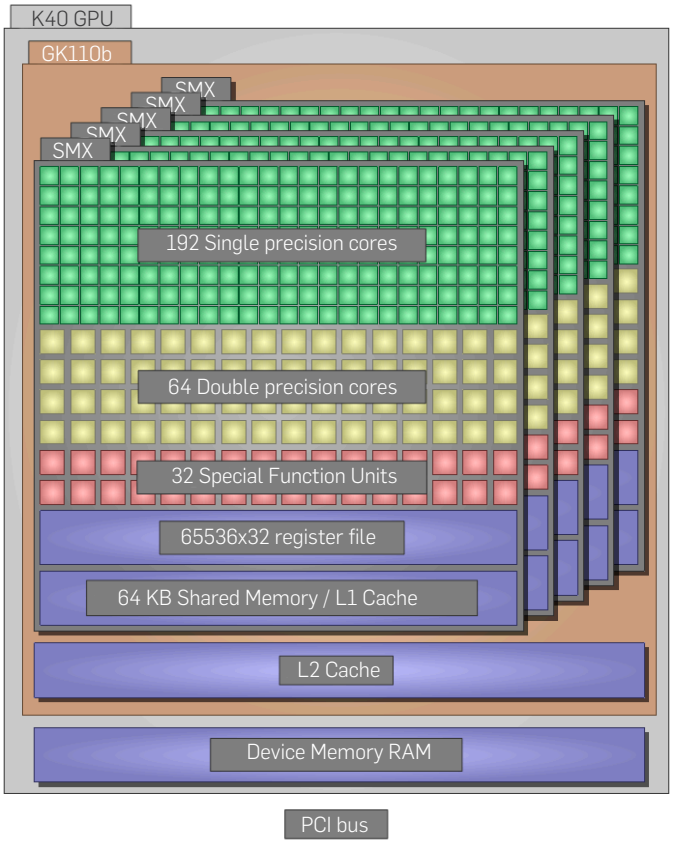

Figure 2. GPU architecture

The GPU gathers thousands of computing elements in a single chip adding new memory hierarchy levels as shown in Figure 2. These computing elements represent more computational power than conventional CPUs, but they require optimal data transferences to achieve the best performance. Data reaches GPU from CPU through

the PCI bus, and it is stored on the device memory. Then, the data can be accessed from the computing cores and can be cached in shared, L1 and L2 memories.

Programmers employ cores launching a large amount of threads on the GPU. This hides memory latencies creating a pipeline between computing and memory access operations. The threads have their own hierarchy, which constrains the hardware resources they can access. This is shown in **Figure 3**. A single thread can have its own general purpose registers; a group of threads, called a block, may have access to the resources of a single Streaming Multiprocessor (SMX) including shared memory, and the grid conformed by the whole set of thread blocks can access all device resources including the device RAM memory.

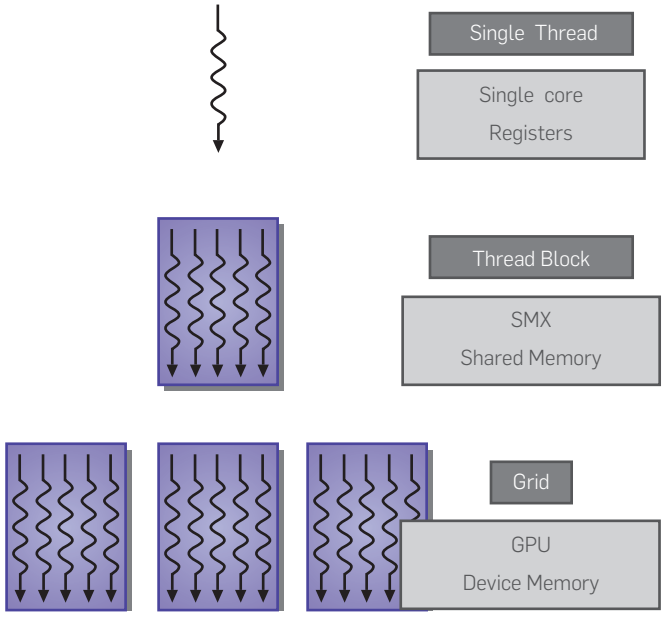

Figure 3. GPU thread architecture. Adapted from [2]

Table 2. Model Parameters and acquisition used in all experiments

Parameter	Value
Nx	169
Ny	169
Nz	50
dx	80 [m]
dy	80 [m]
dz	80 [m]
Shots	196 [shots]
dt(data)	4 [ms]
fq(data)	5 [Hz]

To understand the GPU components and to make efficient applications, programmers use the hardware abstraction model of the GPU given by the available APIs. Also, GPU vendors provide recommendations to improve application's performance, especially on memory access. These recommendations include the reduction of memory transfers between CPU and GPU, but due to the large memory required by the RTM algorithm, GPU RAM could be not enough to store all the wavefields, and CPU RAM may be used. This increases the data transfers between GPU and CPU and decreases performance significantly; hence, computing strategies become the solution to make all calculations using only the GPU memory.

4. RESULTS

The three strategies of the RTM method were implemented on a Nvidia K40 GPU. The constant density acoustic wave equation was solved using FDTD approximation, and the CPML condition was implemented to avoid artificial reflections on the model's border for the first two strategies. The last strategy uses random boundaries to manage artificial boundaries.

The essence of all strategies is the routine to generate one time-step in the modeling process. At each time-step, the wave equation is solved with an explicit scheme, such that each point in the model can be computed independently. In the proposed parallel implementation, one time-step is generated in two GPU kernels: one for the wave equation, and one for the CPML absorbing boundary conditions, such that all spatial points of a single snapshot are computed in parallel. Before starting a GPU process, the data must be transferred to the GPU memory. The transfers between GPU and CPU must be minimized as they are slow compared with the computing time. The capacity of the GPU memory available to process each shot for the K40 GPU device memory is 12 GB.

Table 3. L2-error norm of the migrated image using all strategies compared with the migrated image storing the entire wavefields. GPU memory and execution times required for all strategies.

Strategy	kc	L2-error	GPU Memory per shot [GB]	Execution Time [s]
1	4	0	8.16	17.55
1	8	0	3.17	33.05
1	14	0	1.88	52.78
1	20	0	1.36	72.70
2	N.A.	2.09×10^{-6}	1.47	20.05
3	N.A.	4.48×10^{-4}	0.16	23.27

The proposed parallel implementation for the three strategies was tested using the SEG velocity model shown in **Figure 4**. The model parameters and data are shown in **Table 2**. The average execution time to migrate a single shot and the GPU memory required by all strategies are shown in **Table 3**.

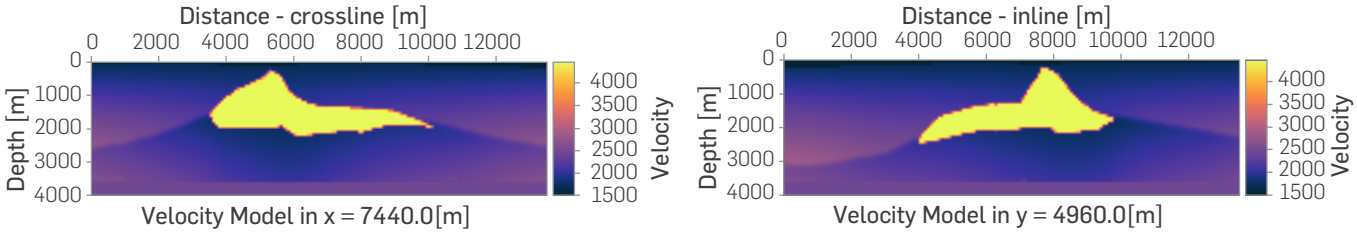

Figure 4. Left: Cross-line of the SEG velocity model at 7.44km. Right: In-line of the SEG velocity model at 5.12km.

STRATEGY 1

This strategy only stores some snapshots of the source wavefield and the missing snapshots are recomputed from the last stored checkpoint. To analyze the required memory and the execution time, the imaging condition was calculated on each time-step, but the interval to set our checkpoints range from kc=4 to kc=20. The memory resources and the execution times used by this strategy, for different kc values, are shown in **Table 3**. The resulting image is shown in **Figure 5**. These images are used as reference to compare with the images obtained with the following strategies.

STRATEGY 2

The second strategy only stores the boundaries of the source wavefield. The L2-error norm obtained with this strategy is 2.09 $\times 10^{-6}$ and the computational resources are also shown in **Table 3**. The migrated image is shown in **Figure 6** -left and -right. Note that the error in the quality of the migrated image is tolerable, and the balance between the GPU memory requirements and execution time makes this strategy very convenient for large size 3D models.

STRATEGY 3

The last strategy presented herein required only 0.16 GB of device memory, but the L2-error norm is 4.48×10^{-4} compared to the migrated image obtained with strategy 1. The migrated image obtained with this strategy is shown in **Figure 7**. It must be noted that, despite the numerical error obtained with this strategy, the image locates adequately the reflectivity map as there is incoherent random boundaries energy reflection and, therefore, the forward and backward wavefields correlate poorly, minimizing coherent artifacts produced by the boundaries. The greatest error is caused by the top boundary that is very close to the source and generates strong random reflections and diffractions at first modeling time snapshots. Different solutions can be used to solve this problem, such as applying PML boundary conditions to the top part of the model or increasing the random boundary length at the top. Those solutions, however, would increase the memory amount requirements of strategy 3.

Figure 8 and **Figure 9** compare the results of strategies 2 and 3 respect to the reference: Strategy 1. The magnitude of the difference is larger in strategy 3, according to the L2 error calculated on **Table 3**. The error in Strategy 2 is located on the top of the model as the

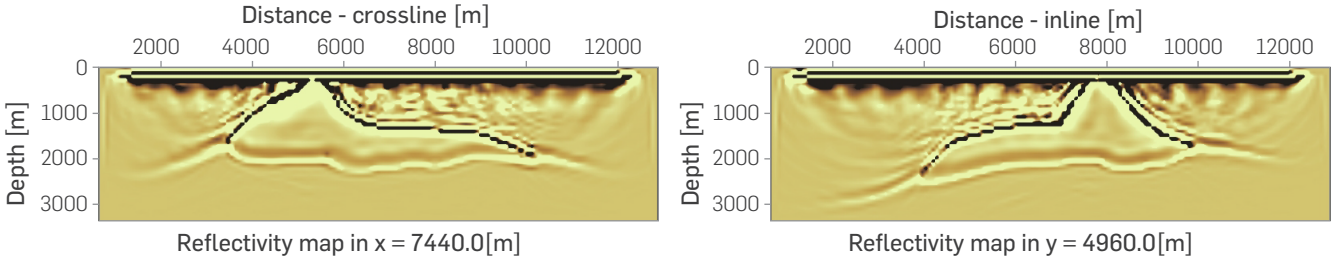

Figure 5. Left: Crossline of the migrated image at 7.44 km, and Right: Inline of the migrated image at 5.12 km; using RTM implementation with strategy 1.

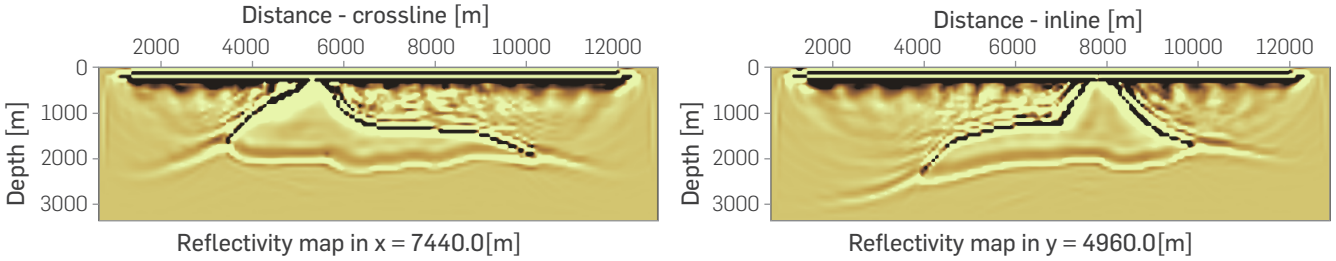

Figure 6. Left: Crossline of the migrated image at 7.44 km, and Right: Inline of the migrated image at 5.12 km; using RTM implementation with strategy 2.

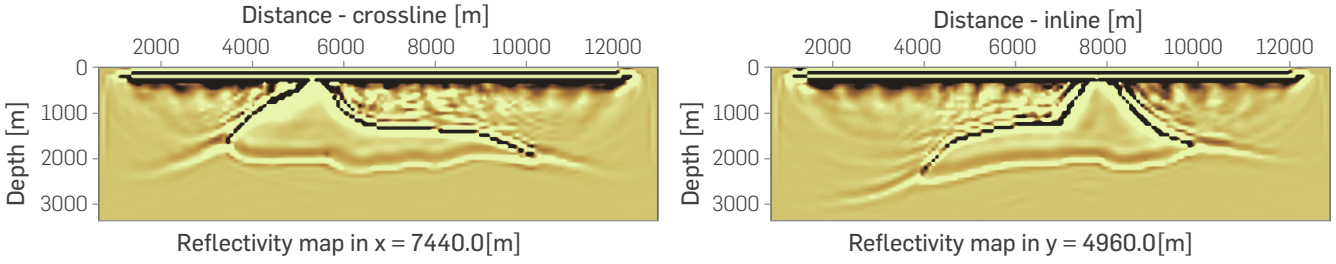

Figure 7. Left: Crossline of the migrated image at 7.44 km, and Right: Inline of the migrated image at 5.12 km; using RTM implementation with strategy 3.

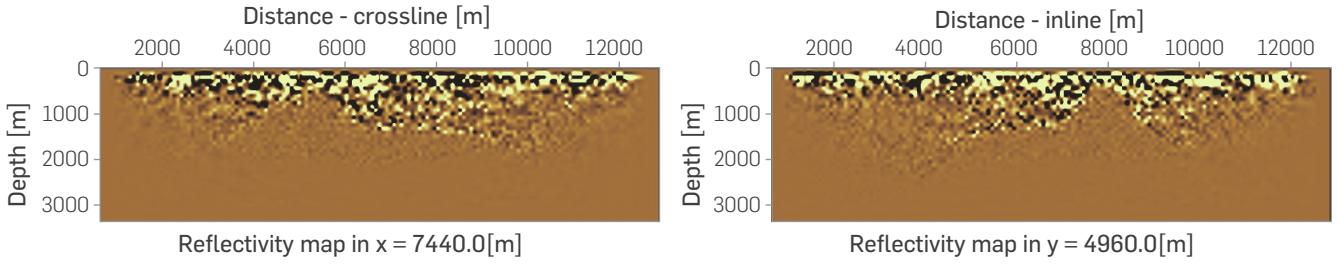

Figure 8. Left: Crossline of the difference between the migrated images at 7.44 km, and Right: Inline of the difference between the migrated images at 5.12 km; using the RTM implementation strategies 1 and 2. The amplitude of the images is amplified 10^6 compared to Figure 6.

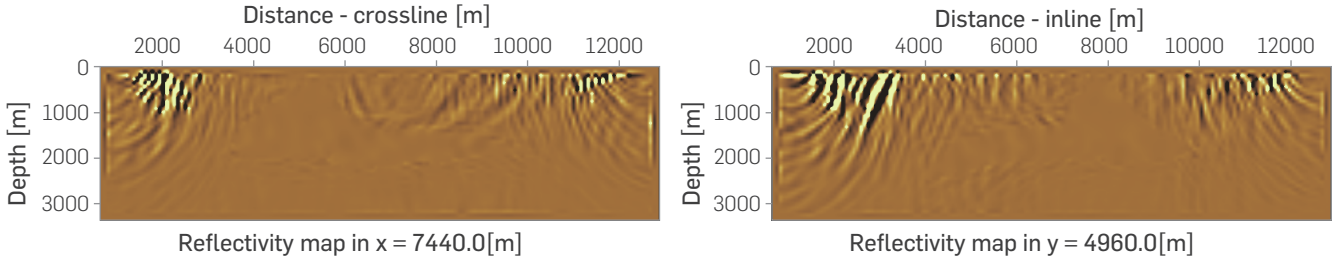

Figure 9. Left: Crossline of the difference between the migrated images at 7.44 km, and Right: Inline of the difference between the migrated images at 5.12 km; using the RTM implementation strategies 1 and 3. The amplitude of the images is amplified 10^3 compared to Figure 7.

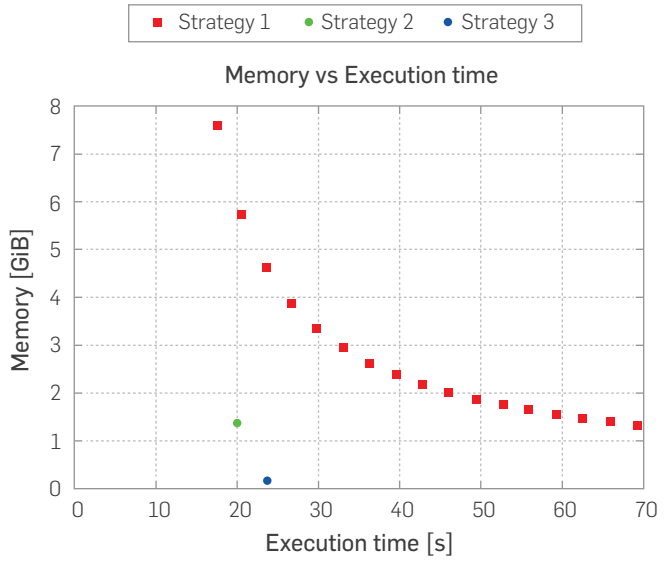

Figure 10. : Comparison of the memory usage (in GiB) for all strategies as a function of the execution time (in secs.).

reconstruction of the source wavefield is less accurate on the early time steps, near to the source location. This means that the error is mainly produced by numerical rounding due to the recomputation the source wavefield. Furthermore, strategy 3 locates the error near the lateral random boundaries.

Finally, Figure 10 presents a comparison of memory usage for all strategies as a function of the execution time. The best execution time is achieved by Strategy 1, but it also requires the largest memory. This strategy trades off between memory and computation time, depending on the *kc* parameter. This makes this strategy a very flexible option to fit the algorithm on any memory constrained architecture. For the specific experiment, Strategy 2 has good performance in terms of memory and execution times at the same time. However, in a large scale problem, the GPU memory available can be exceeded when the boundaries are large. Strategy 3 obtains the best ratio between execution time and memory, but due to the boundary conditions method, it introduces an error in the image. However, the quality of the image is acceptable in terms of reflection events and the artifacts can be attenuated applying post-processing filters to reduce incoherent noise in the image.

CONCLUSIONS

In this work, we evaluate three RTM implementation strategies focused on managing the memory, such that the seismic RTM algorithm can be executed on a single GPU device. The GPU is a memory-constrained architecture that reaches its maximum performance when memory transfers from (or to) CPU are minimized. In the strategies studied herein, each shot is migrated independently by making only memory transfers at the beginning and end of the migration. The entire source wavefield cannot be fully stored in any GPU available. Furthermore, the three strategies analyzed in this work fix this problem by recomputing portions of the source wavefield.

Strategy 1 can use all the memory available on the GPU by adapting the checkpoint parameter; however, the computing time increases. Strategy 2 stores the boundary of the source wavefield to recompute it backwards in time and gets good balance between the GPU memory requirements and the execution time; however, if the GPU memory available is not enough to store the boundary and the two last snapshots, this strategy cannot be used. Finally, Strategy 3 has

the lower execution time and requires less memory as compared with other strategies. Nevertheless, the reconstruction of the source wavefield introduces numerical error in the migrated image. Strategy 3 is therefore the recommended strategy to implement in the GPU architecture for large data and the numerical error in the final image produced by the random boundaries can be removed with post-processing techniques such as FX-decon [11], TFD [12] and structure tensors [13].

To conclude this analysis, the maximum frequency that can be migrated with the same parameters, the same velocity model, and the same GPU used in this article are determined. The results show that Strategy 1 can migrate up to 30 [Hz], Strategy 2 can migrate up to 30 [Hz] and Strategy 3 can migrate up to 70 [Hz]. Then, we extend this analysis to a hypothetic advanced seismic acquisition, such as broadband, which provides both low and high frequencies to obtain better resolution in imaging. In this case, RTM migration has the challenge to process data with frequencies above 100 [Hz]. Thus, we propose a marine survey acquisition with 18 cables 8000 [m]. The maximum frequency that each strategy can migrate, using the same GPU, is: 3 [Hz] for strategy 1, 3 [Hz] for strategy 2 and 12 [Hz] for strategy 3.

The frequency parameter quickly increases the memory required as it reduces spatial sampling of the velocity model and the timestep propagation. This analysis validates our recommendation about using Strategy 3 for migrations with high memory requirements.

ACKNOWLEDGEMENTS

This work is supported by Ecopetrol and Colciencias as part of the research project grant 0266 of 2013.
The authors gratefully acknowledge the support of CPS research group at Universidad Industrial de Santander.

REFERENCES

[1] Baysal, E., Kosloff, D. D., & Sherwood, J. W. Reverse time migration. Geophysics, 1983, 48 (11), 1514-1524. doi:10.1190/1.1441434

[2] NVIDIA Corporation. CUDA C best practices guide, 2017, [On line]. Avaliable at: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[3] Clapp, R. G., Haohuan Fu, & Lind, O. Selecting the right hardware for reverse time migration. The Leading Edge, 2010, 29(1), 48-58. https://doi.org/10.1190/1.3284053

[4] Dussaud, E., Total, E., Symes, W. W., Williamson, P., Lemaistre, L., Singer, P., Sa, T. Computational strategies for reverse-time migration. SEG Las Vegas 2008 Annual Meeting, 2008, 2267-2271. https://doi.org/10.1190/1.3059336

[5] Clapp, R. G. Reverse time migration with random boundaries. SEG Houston 2009 International Exposition and Annual Meeting, 2009, 2809-2813. https://doi.org/10.1190/1.3255432

[6] Pasalic, D., & McGarry, R. Convolutional perfectly matched layer for isotropic and anisotropic acoustic wave equations. 2010 SEG Annual Meeting, 2010, 2925-2929. Available at: http://www.onepetro.org/mslib/servlet/onepetropreview?id=SEG-2010-2925

[7] Griewank, A. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. Optimization Methods and Software, 2011, 35-54. doi:10.1080/10556789208805505

[8] Symes, W. W. (9 de 2007). Reverse time migration with optimal checkpointing. Geophysics, 2007, 72(5), SM213--SM221. doi:10.1190/1.2742686

[9] Bo, F., & Huazhong, W. Reverse time migration with source wavefield reconstruction strategy. Journal of Geophysics and Engineering, 2012, 9 (1), 69. [On line] Available at: http://stacks.iop.org/1742-2140/9/i=1/a=008

[10] Yang, P., Brossier, R., Métivier, L., & Virieux, J. Wavefield reconstruction in attenuating media: A checkpointing-assisted reverse-forward simulation method. Geophysics, 2016, 81(6), R349-R362. doi:10.1190/geo2016-0082.1

[11] Gulunay, N. FXDECON and complex wiener prediction filter. SEG Technical Program Expanded Abstracts 1986, 1986, 279-281. https://doi.org/10.1190/1.1893128

[12] Naghizadeh, M. Seismic data interpolation and denoising in the frequency-wavenumber domain. Geophysics, 2012, 77(2), V71-V80. https://doi.org/10.1190/geo2011-0172.1

[13] Gholami, A., Haghshenas, H. Curvelet-TV regularized Bregman iteration for seismic random noise attenuation. Journal of Applied Geophysics, 2014, 109, 233 - 241. https://doi.org/10.1016/j.jappgeo.2014.08.005