A hybrid methodology for a 3D Full Waveform Inversion in time domain using GPUs

David L. Abreo [1], Sergio A. Abreo [1], Ana B. Ramirez [1].
[1] Universidad Industrial de Santander, Colombia.

Abstract

Full Waveform Inversion (FWI) is an iterative method that allows to determine the subsurface parameters from the observed data at surface and an initial model. The main drawback of a 3D FWI implementation in the time domain is the computational cost because the required memory and the execution time are very expensive. In serial platforms, the main constraint of a 3D FWI implementation is the execution time. On the other hand, in Parallel platforms such as GPUs the main constraint is the available memory resources. In this paper, we designed and implemented a strategy that takes advantage of both platforms, serial and parallel, using MPI and CUDA-C to resolve both problems. The new implementation has a speedup factor of 1.84x and a 76% of reduction of the required memory. This methodology makes feasible the 3D FWI implementation using a GPU cluster.

Introduction

In the last years, the Full Waveform Inversion method (FWI), used for estimate the model's properties, has gained attention in the geophysical community, because allows high quality reconstructions of the properties.

The FWI is a nonlinear data adjustment procedure that aims to obtain estimates subsurface properties from the observed data d_{obs}. The process starts with an initial estimate of the model's parameters $\mathbf{m}_0$. Then the modeled data d_{mod} are predicted by the solution of the wave equation (Equation 2) with $\mathbf{m}_0$. The new estimated parameters are updated to reduce the mismatch between d_{obs} and d_{mod}. The process is repeated iteratively until the error between d_{mod} and d_{obs} is smaller than a reference value or until the stopping criterion has been reached (Virieux and Operto, 2009).

One of the main disadvantages of a 3D FWI implementation in time domain is the computational cost because it is required to compute and save both the forward and backward fields to calculating the gradient (Plessix, 2006). In addition, a massive amount of operations are required, which implies a high execution time.

In Abreo et al. (2015) is implemented a 2D FWI in time domain using GPUs (NVIDIA, 2005) to reduce the computational cost. Although these devices are efficient in massive data processing (ideal for three-dimensional processing), they have difficulties when large amounts of information are stored.

In this paper is proposed a hybrid methodology using MPI and CUDA-C (NVIDIA, 2010) to reduce both the memory required and the computation time of a 3D FWI implementation. The proposed methodology makes feasible the 3D FWI implementation taking advantage of a GPU cluster.

Method

The FWI method uses the observed seismic data at the surface d_{obs} to estimate the subsurface velocity model $\mathbf{m}$, through the minimization of the difference between d_{obs} and d_{mod} (Tarantola, 1984) as,

$$\phi = \arg\min_{\mathbf{m}} ||d_{mod}(\mathbf{m}) - d_{obs}(\mathbf{m})||_2^2, \tag{1}$$

where $||\cdot||_2^2$ is the L_2-norm square operator.

During a seismic acquisition, d_{obs} are the obtained traces from the real model $\mathbf{m}$. And d_{mod} are the obtained traces from an estimated model $\hat{\mathbf{m}}$ using the operator

$$\mathbf{F}\{\cdot\} \triangleq \left\{ \frac{1}{\hat{\mathbf{m}}^2(x,y,z)} \frac{\partial^2 \mathbf{p}}{\partial t^2} = \frac{\partial^2 \mathbf{p}(x,y,z,t)}{\partial x^2} + \frac{\partial^2 \mathbf{p}(x,y,z,t)}{\partial y^2} + \frac{\partial^2 \mathbf{p}(x,y,z,t)}{\partial z^2} + \mathbf{s}(x,y,z,t) \right\}, \tag{2}$$

where $\mathbf{p}(x,y,z,t)$ denotes the pressure field; x, y and z are the spatial variables; t is the time variable and $\mathbf{s}$ is the source. In this paper we used *Convolutional Perfectly Matched Layer* method (CPML) (Pasalic et al., 2010) to avoid the reflections that come from the non-natural boundaries as was used by Abreo et al. (2015).

Equation 2 is implemented using centered finite differences in time domain (FDTD). A second order stencil is used for the time derivative and an eighth order stencil is used for the space derivatives (Sullivan, 2013).

The second order stencil is defined as

$$\frac{\partial^2 \mathbf{p}}{\partial t^2} \approx \frac{\mathbf{p}_{i,j,k}^{n+1} - 2 \cdot \mathbf{p}_{i,j,k}^{n} + \mathbf{p}_{i,j,k}^{n-1}}{\Delta t^2}, \tag{3}$$

where i,j,k and n represent the discretized variables for x,y,z and t, respectively; and Δt is the time step. (see Figure 1, red circles).

The eighth order stencil for the component x is defined as

$$\frac{\partial^2 \mathbf{p}}{\partial x^2} \approx \frac{\sum_{c=-4}^{4} a(4+c) \cdot \mathbf{p}_{i+c,j,k}^{n}}{\Delta x^2}, \tag{4}$$

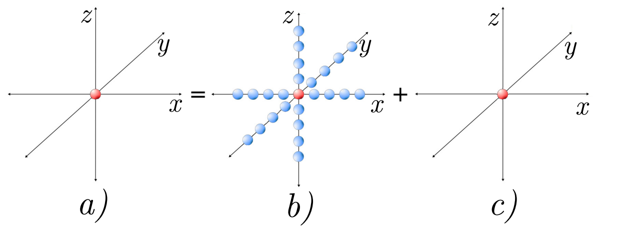
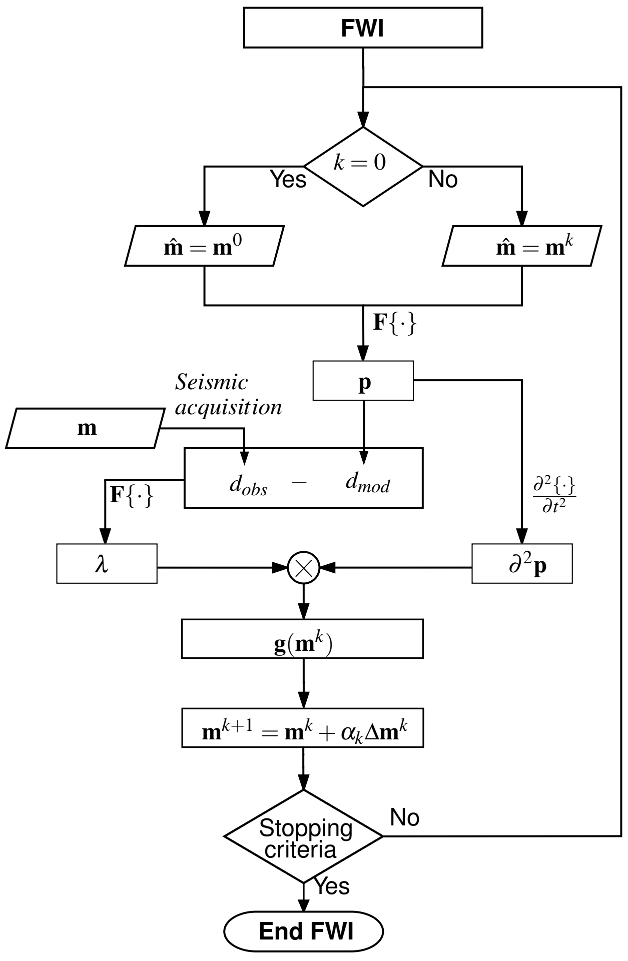


Figure 1: Graphical representation FDTD. a)$\mathbf{P}^{n+1}$ b)$\mathbf{P}^{n}$ c)$\mathbf{P}^{n-1}$

where $a = [\frac{-1}{560}, \frac{8}{315}, \frac{-1}{5}, \frac{8}{5}, \frac{-205}{72}, \frac{8}{5}, \frac{-1}{5}, \frac{8}{315}, \frac{-1}{560}]$ are the weights of the centered approximation and Δx is the spatial step in the x direction (see Figure 1-b).

A general scheme of an FWI implementation is shown in Figure 2. The algorithm iteration k starts with a zero value which is increased by each iteration step.



FWI
$k=0$
Yes
No
$\hat{\mathbf{m}} = \mathbf{m}^0$
$\hat{\mathbf{m}} = \mathbf{m}^k$
$\mathbf{F}\{\cdot\}$
$\mathbf{p}$
Seismic acquisition
$\mathbf{m}$
$d_{obs} - d_{mod}$
$\mathbf{F}\{\cdot\}$
$\frac{\partial^2\{\cdot\}}{\partial t^2}$
λ
$\partial^2\mathbf{p}$
$\mathbf{g}(\mathbf{m}^k)$
$\mathbf{m}^{k+1} = \mathbf{m}^k + \alpha_k\Delta\mathbf{m}^k$
Stopping criteria
No
Yes
End FWI

Figure 2: General scheme of an FWI implementation

In the first iteration $\hat{\mathbf{m}}$ takes an initial model $\mathbf{m}^0$ and for the next ones takes the model $\mathbf{m}^k$. The time derivative of the wave field $\partial^2\mathbf{p}$ is calculated using Equation 2 with $\mathbf{s}$ as a punctual source at surface. The wave field λ is calculated with the same Equation 2 using the residual traces obtained from of d_{obs} and d_{mod} as a source. This is possible because $\mathbf{F}\{\cdot\}$ is a self-adjoint operator.

The update for the velocity model can be obtained using Newton-like methods (Goldstein, 1965), as

$$\mathbf{m}^{k+1} = \mathbf{m}^k + \alpha_k \Delta\mathbf{m}^k, \quad (5)$$

where α_k is the step size and $\Delta\mathbf{m}^k$ at the k^{th} iteration is given by

$$\Delta\mathbf{m}^k = -[\mathbf{H}(\mathbf{m}^k)]^{-1}\mathbf{g}(\mathbf{m}^k), \quad (6)$$

with the gradient $\mathbf{g}(\mathbf{m}^k)$ of cost function ϕ and the inverse of the Hessian matrix $[\mathbf{H}(\mathbf{m}^k)]^{-1}$, both evaluated at $\mathbf{m}^k$.

The Hessian matrix is replaced by the identity matrix to reduce the computational cost of this operation. The gradient can be calculated using the Plessix (2006) expression

$$\mathbf{g}(\mathbf{m}^k) = \sum_{N_s}\int_0^T \frac{\partial^2\mathbf{p}(x,y,z,t)}{\partial t^2}\cdot\lambda(x,y,z,T-t)dt \quad (7)$$

where N_s is the number of sources and T is the propagation time.

Strategies to compute the Gradient

Figure 3 illustrates the gradient computation where the blacks arrows $(*)$ indicate the operations flow and the green squares represent the memory saved inside the GPU. First, the wave field $\mathbf{p}$ and its derivate are calculated. Second, the residuals are computed from d_{obs} and d_{mod}. Third, the backward field λ is computed using the residuals. Fourth, the wave fields are multiplied point to point, $(**)$, to find a temporal gradient G_{temp}. Fifth, G_{temp} is accumulated for all the time steps $(***)$ to get the gradient $\mathbf{g}(\mathbf{m}^k)$.

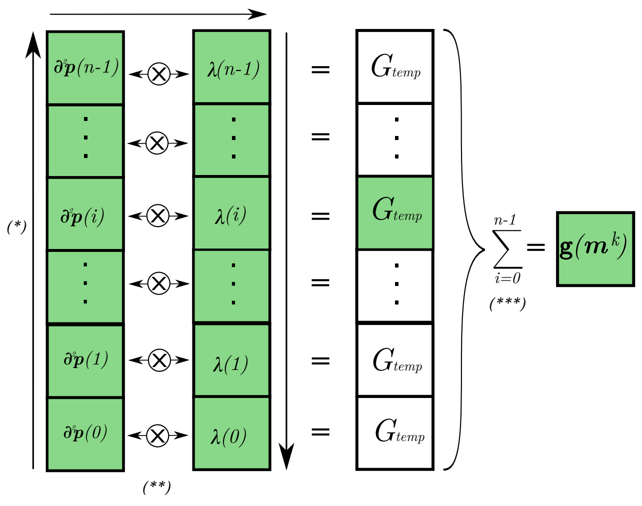


Figure 3: Gradient calculation saving two field

The memory required by the 2D FWI implementation in time domain of Abreo et al. (2015) is defined as

$$\begin{aligned} Ram\ size_{2D} = {} & \frac{\beta}{1024^2 \cdot 8}\cdot(N_s\cdot N_t\cdot N_x + 11\cdot N_x\cdot N_z + \\ & N_t + 2\cdot N_x\cdot N_z\cdot N_t + 4\cdot N_x + 4\cdot N_z + 2\cdot N_t\cdot N_x), \end{aligned} \quad (8)$$

where Nx and Nz are the number of spatial steps in x and z directions, respectively; N_t is the number of time steps, N_s

is the number of shots and β is the type of precision (single or double).

An approximate expression for a 3D FWI implementation saving both fields, $\mathbf{p}$ and λ, inside a GPU is

$$Ram\ size\ _{3D} = \frac{\beta}{1024^2 \cdot 8} \cdot (N_s \cdot N_t \cdot N_x \cdot N_y + 11 \cdot N_x \cdot N_y \cdot N_z + N_t + 2 \cdot N_x \cdot N_y \cdot N_z \cdot N_t + 4 \cdot N_x + 4 \cdot N_y + 4 \cdot N_z + 2 \cdot N_t \cdot N_x \cdot N_y). \quad (9)$$

where Ny is the number of spatial steps in the y direction.

This expression allows to compute the theoretical memory required to compute the gradient storing $\mathbf{p}$ and λ. In a small model $(Nx = 100, Ny = 100, Nz = 100, Nt = 2000, Ns = 15)$ the memory necessary to implement the 3D FWI is 16.5 GiB leaving disable our GPU Tesla k40c (NVIDIA, 2011) with only 12 GiB of RAM.

Due to this problem, it is necessary to design and implement strategies to reduce the memory required. In this paper two strategies are defined to perform the 3D FWI in time domain implementation.

First strategy

The first strategy, is to solve the gradient storing only $\partial\mathbf{p}$ inside the GPU. Then G_{temp} is calculated at each time step through the multiplication of $\lambda(i)$ with $\partial^2\mathbf{p}(i)$. Finally, $\mathbf{g}(\mathbf{m}^k)$ is obtained performing a summation over G_{temp} (Figure 4). This strategy keeps the computational cost reducing the memory required in a 50 %.

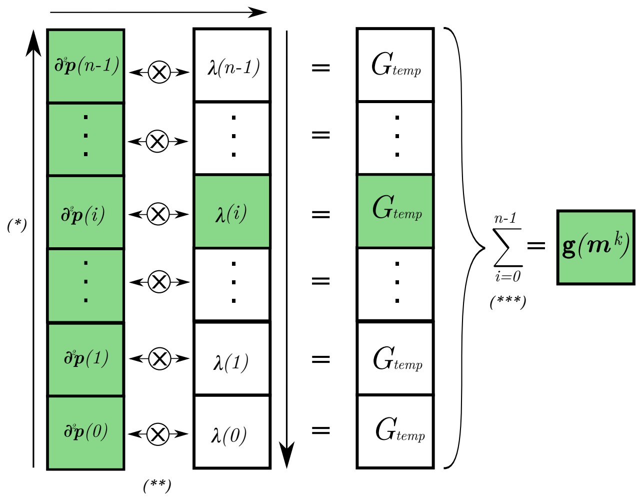


Figure 4: Gradient calculation saving one field

Second strategy

The second strategy does not save any field. λ is reconstructed in inverse sense (i.e since the snapshot $i = 0$ to $i = n-1$) and $\mathbf{p}$ with its derivate are calculated two times (only saving the actual snapshot $\partial^2\mathbf{p}(i)$). This strategy is implemented in five steps. First, $\mathbf{p}$ is calculated to obtain d_{mod}. Second, the residuals are computed from d_{obs} and d_{mod}. Third, λ is computed using the residuals saving only the boundary data, $\lambda(0)$ and $\lambda(1)$. Fourth, $\lambda(i)$ and $\partial^2\mathbf{p}(i)$ are calculated at the same time step (black arrows) and they are multiplied point to point to find a temporal gradient G_{temp} remembering that $\lambda(i)$ is reconstructed from the boundary data. Fifth, G_{temp} is accumulated for all time steps to get the gradient $\mathbf{g}(\mathbf{m}^k)$. Figure 5 shows this strategy.

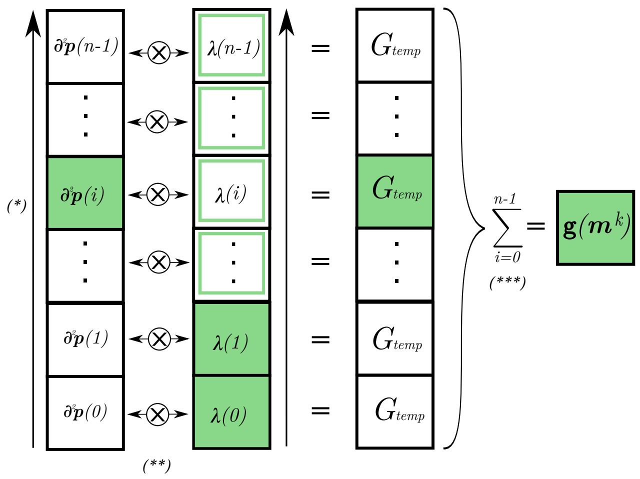


Figure 5: Gradient calculation without save any field

Hybrid implementation using MPI

A scheme of the FWI using MPI is described in Figure 6.

The number of shots N_s are equally distributed inside the GPU cluster using MPI. Each GPU receives the i_s^{th} shot to compute the $\mathbf{g}_{i_s}$ gradient. G_{i_G} accumulates $\mathbf{g}_{i_s}$ per GPU. This process is repeated until $i_s < N_s$. MPI collects all the computed gradients to obtain $\mathbf{g}(\mathbf{m}^k)$. The new velocity model is computed using only one GPU and shared to all the cluster to repeat the process. The loop stops when the stopping criteria is reached.

This scheme keeps the same memory requirements reaching a speedup factor of

$$Speedup = \frac{N_S + T_{serial}}{ceil(N_S/N_G) + T_{serial}}, \quad (10)$$

where the T_{serial} is the time needed to calculate $\mathbf{g}(\mathbf{m}^k)$ and $\mathbf{m}^{k+1}$.

Results

The tests are performed in a cluster with a CPU Intel Xeon E5-2609 with 256 GiB of RAM and two GPUs Tesla K40 with 12 GiB of RAM (NVIDIA, 2011).

The model chosen to implement the FWI 3D is a diffracting cube (2500 m/s) inside a velocity constant volume of 2000 m/s. In Figure 7, a slice in-line of both the truth velocity model and the starting point for the FWI 3D are illustrated. Fifteen sources distributed at surface and a model of $Nx = 211, Ny = 101, Nz = 68$ are used.

Figure 8 illustrates a 3D visualization of the FWI results using the hybrid methodology. The sources are marked as red points at surface.

The image is edited to eliminate the constant velocity volume to highlight the reconstruction and the sources effect. The reconstruction is obtained after a multi-scale implementation (Bunks et al., 1995) using a Ricker wavelet with three central frequencies (3, 10 and 15 Hz), 30

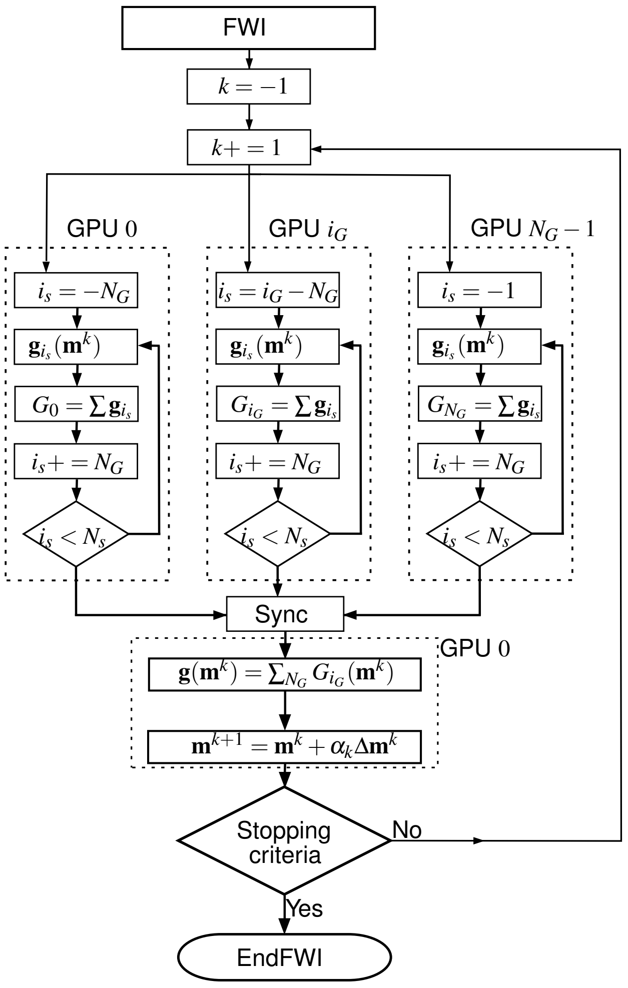

FWI
$k = -1$
$k += 1$
GPU 0
GPU i_G
GPU $N_G - 1$
$i_s = -N_G$
$i_s = i_G - N_G$
$i_s = -1$
$\mathbf{g}_{i_s}(\mathbf{m}^k)$
$G_0 = \sum \mathbf{g}_{i_s}$
$G_{i_G} = \sum \mathbf{g}_{i_s}$
$G_{N_G} = \sum \mathbf{g}_{i_s}$
$i_s += N_G$
$i_s < N_s$
Sync
GPU 0
$\mathbf{g}(\mathbf{m}^k) = \sum_{N_G} G_{i_G}(\mathbf{m}^k)$
$\mathbf{m}^{k+1} = \mathbf{m}^k + \alpha_k \Delta \mathbf{m}^k$
Stopping criteria
No
Yes
EndFWI

Figure 6: Schema FWI using MPI.

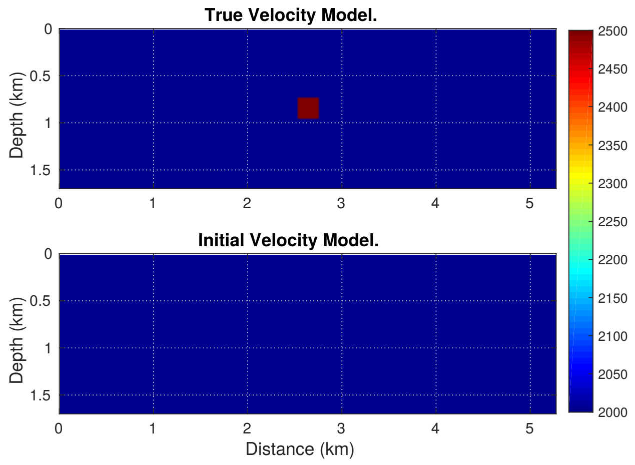


Figure 7: True and initial velocity models

iterations per frequency step and $\alpha = 90,\ 130$ and 85, respectively.

Figure 9 is a 2D slice in-line of the reconstruction at the same position of Figure 7. Figure 10 shows the 1D profile of the initial, final and true velocity models extracted from the center of the x, y surface.

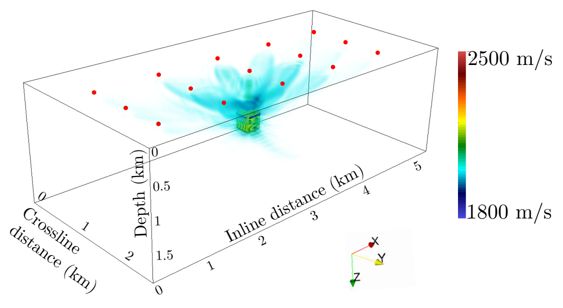


Figure 8: Final velocity model.

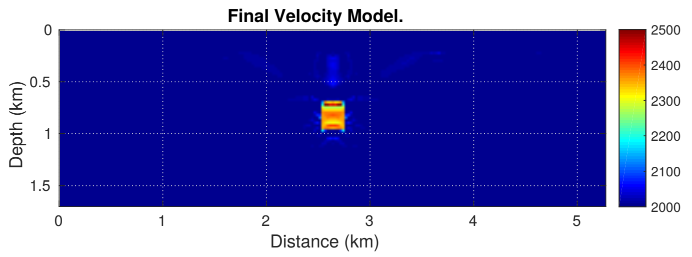


Figure 9: 2D section of the final velocity model.

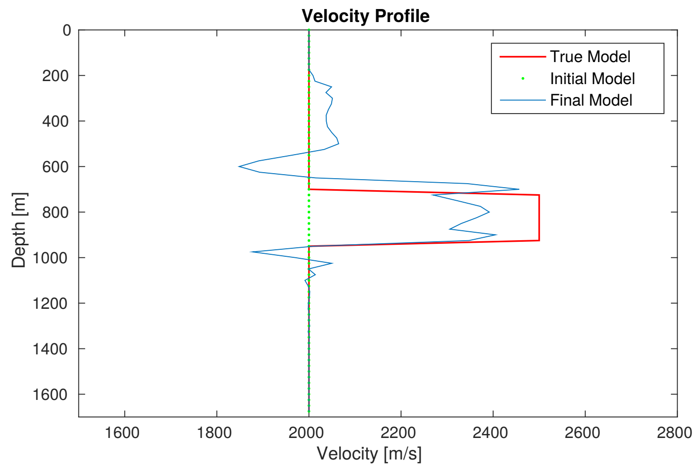


Figure 10: Comparison of velocity profile

In Table 1 the execution time and memory requirements per GPU are shown using both strategies running on one or two GPUs with MPI. The time is measured in minutes.

Table 1: Execution time and memory requirements

Strategy	# GPUs	Time [m]	Memory [GiB]
I	1	635	8.64
I	2	345	8.64
II	1	984	2.09
II	2	537	2.09

Conclusions

In this paper is implemented the FWI 3D in time domain using two strategies to calculate the gradient and a hybrid implementation to reduce the execution time using a cluster with 2 or more GPUs.

According to Table 1, the execution time of the first strategy is 35% faster than the second strategy because the last one requires to perform an additional propagation to save the data at boundaries.

On the other hand, the second strategy only needs the 24% of the memory required by the first strategy. This is because in the first strategy is saved one field and in the second strategy is only saved the data at boundaries.

In the hybrid implementation the runtime is reduced in a 46% because the shot gathers are equally distributed between the number of GPUs inside the cluster.

The memory required using MPI does not change compared with the stand alone implementation because each GPU needs all the information to calculate the gradient.

Acknowledgements

This work is supported by Colombian Oil Company ECOPETROL and COLCIENCIAS as a part of the research project grants No. 0266-2013. The authors gratefully acknowledge Industrial University of Santander.

References

Abreo, S., A. Ramirez, D. Abreo, O. Reyes, and H. Gonzales, 2015, A practical implementation of acoustic full waveform inversion on graphical processing units: Ciencia, TecnologÃa y Futuro, **6**, 5–16.

Bunks, C., F. M. Saleck, S. Zaleski, and G. Chavent, 1995, Multiscale seismic waveform inversion: Geophysics, **60**, 1457–1473.

Goldstein, A. A., 1965, On newton's method: Numerische Mathematik, **7**, 391–393.

NVIDIA, 2005, Gpu computing revolutionizing high performance computing: www.stanford.edu/sep/prof/bei11.2010.pdf. (Revisado: December 2016).

——, 2010, What is cuda: www.developer.nvidia.com/what-cuda. (Revisado: December 2016).

——, 2011, Gpu tesla k40 computing revolutionizing high performance computing: www.nvidia.com/content/PDF/kepler/nvidia-tesla-k40.pdf. (Revisado: December 2016).

Pasalic, D., R. McGarry, et al., 2010, Convolutional perfectly matched layer for isotropic and anisotropic acoustic wave equations.

Plessix, R.-E., 2006, A review of the adjoint-state method for computing the gradient of a functional with geophysical applications: Geophysical Journal International, **167**, 495–503.

Sullivan, D. M., 2013, Electromagnetic simulation using the fdtd method: John Wiley & Sons.

Tarantola, A., 1984, Inversion of seismic-reflection data in the acoustic approximation: Geophysics, **48**, 1259–1266.

Virieux, J., and S. Operto, 2009, An overview of full-waveform inversion in exploration geophysics: GEOPHYSICS, **74**, WCC1–WCC26.