Scalability of Acoustic Full Waveform Inversion on Graphics Processing Units
Santosh Dhubia[1,2], Chintan Prajapati[1], P.R. Mohnaty[2] and P.H. Rao[1]
santoshdhubia@gmail.com
1 – Gujarat Energy Research and Management Institute, Gandhinagar
2 – Indian Institute of Technology (ISM) Dhanbad

Keywords
Full Waveform Inversion, Graphic Processing Units, Velocity Model

Summary
Full Waveform Inversion (FWI) is the state of art algorithm for inverting observed seismic data to get velocity model. The industrial application of FWI require high computational cost as well as memory for fast turn-around time.

We have implemented the time domain full waveform inversion on Graphic Processing Units (GPU) and tested on 2D Marmousi velocity model. We have analyzed the scalability of the FWI in acoustic medium (AFWI) on GPU’s with different architecture. Numerical example shows that our implementation scales well with GPU of different architecture and provides good quality velocity model.

Introduction
Full Waveform Inversion (FWI) is the iterative error minimization algorithm, which minimizes the error between the observed seismic and forward modeled seismic data from current velocity model. The velocity model is updated by following equation:

\[ V_{i+1} = V_i + \alpha_i \cdot d_i \] (1)

Where, \( V_{i+1} \) is the velocity model at (i+1)th iteration and \( V_i \) is the velocity model and \( d_i \) is the direction for decent at the iteration \( i \) and \( \alpha_i \) is the step size computed using following relation (Pica et al., (1990)):

\[ \alpha_i = \frac{\langle J_i d_i, P_{\text{observed}} - P_{\text{mod}} \rangle}{\langle J_i d_i, J_i d_i \rangle} \] (2)

Where, \( P_{\text{observed}} \) is the observed data and \( P_{\text{mod}} \) is the forward modelled data using current velocity model.

The decent direction is computed using the minimization of following L2 norm objective function:

\[ O(V) = \frac{1}{2} \sum_{k=1}^{N_k} \sum_{l=1}^{N_l} \int_0^{N_t} \partial p(x_{k,l},z_{k,l},t)^2 \, dt \] (3)
Scalability of Acoustic Full Waveform Inversion on Graphics Processing Units

Where, the subscript $k,l$ denotes the sum over all the receivers & sources. And, $\frac{\partial p(x_k, z_l, t)}{\partial t}$ is the difference between the observed seismic data and forward modelled data at the given source-receiver geometry at time $t$.

The forward modelled wavefield are computed using the following acoustic wave equation (constant density):

$$\frac{\partial^2 P(x, z, t)}{\partial t^2} = V_i^2 \nabla^2 P(x, z, t) + f(x_l, z_l, t)$$  \hspace{1cm} (4)

Where, $P(x, z, t)$ is the pressure wavefield at location $(x,z)$ with source at position $(x_l, z_l)$ in the current velocity model $V_i$ at the $i$th iteration. $\nabla^2$ is the laplacian and solved with fourth order finite difference scheme.

The solution in time is done using second order finite difference scheme with absorbing boundary condition.

To update the velocity model based on the gradient descent direction, a Non-Linear conjugate gradient descent method (Hager and Zhang, 2006) is used as shown below:

$$\{d_i = \begin{cases} \nabla O(V_i), & i = 1 \\ \nabla O(V_i) + \gamma_i \ast d_{i-1}, & i > 1 \end{cases} \}$$  \hspace{1cm} (5)

Where, $\nabla O(V_i)$ is the gradient of the objective function defined in Eq. (2). $\gamma_i$ is computed using following equation:

$$\gamma_i = \max(0, \min(\gamma_{i}^{HS}, \gamma_{i}^{DV}))$$  \hspace{1cm} (6)

And $\gamma_{i}^{DV}$ can be given by following equations:

$$\gamma_{i}^{DV} = \frac{\|\nabla O(V_i) \ast \nabla O(V_{i-1})\|}{\|d_{i-1} \ast \nabla O(V_{i-1}) - \nabla O(V_{i-1})\|}$$  \hspace{1cm} (7)

$$\gamma_{i}^{HS} = \frac{\|\nabla O(V_i) \ast \nabla O(V_{i-1})\|}{\|d_{i-1} \ast \nabla O(V_i) - \nabla O(V_{i-1})\|}$$  \hspace{1cm} (8)

The calculation of $d_i$ using the Eq. (5) ensures the automatic direction reset and over-correction in iterations of conjugate gradient.

The gradient of the objective function $\nabla O(V_i)$ is calculated as follows:

$$\nabla O(V_i) = \frac{2}{v(x, z)^3} \sum_{k=1}^{N_r} \sum_{l=1}^{N_s} \int_0^{N_t} \frac{\partial^2 P_{\text{Forward}}(x_{k,l}, z_{k,l}, t)}{\partial t^2} P_{\text{Residual}}(x_{k,l}, z_{k,l}, t) dt + EPS$$  \hspace{1cm} (9)

Where, $P_{\text{Forward}}$ is the forward modelled data at the given source-receiver geometry for current velocity model $V_i$. $P_{\text{Residual}}$ is the residual wavefield back propagated in time.

A preconditioning (Gauthier et al., 1986 and Pyang et al. 2015) of gradient can be applied using source normalization as follows:

$$\nabla O(V_i) = \frac{\nabla O(V_i) \ast \sum_{k=1}^{N_r} \sum_{l=1}^{N_s} P_{\text{Forward}}(x_{k,l}, z_{k,l}, t)^2 + EPS}{\sqrt{\sum_{k=1}^{N_r} \sum_{l=1}^{N_s} P_{\text{Residual}}(x_{k,l}, z_{k,l}, t)^2}}$$  \hspace{1cm} (10)

Where, EPS is the stability factor to avoid division by zero.

The workflow for Full Waveform Inversion is shown in Figure 1.

Implementation of AFWI on Graphic Processing Unit (GPU)

GPU's have high compute power and their development as fully programmable pipelines make them appropriate choice for accelerating general purpose massively-parallel applications like Reverse Time Migration and Full Waveform Inversion. (Foltinek et al., 2009; Yang et al., 2014, Shin et al., 2014 and Yang et al. 2015).

AFWI is implemented in this study using C++ and CUDA (Nvidia, 2010) programming language. CUDA is a proprietary programming language developed by Nvidia targeting its GPUs and the C language version comes with the hardware. All the file IO including shot gather, velocity file reading are done utilizing Central Processing Unit (CPU) with C++ programming language.
Scalability of Acoustic Full Waveform Inversion on Graphics Processing Units

The high compute power of GPU’s are mainly suffered by very low data transfer. So, in this project we have utilized the method proposed by the Clapp (2008) and Dussaud et al., (2008) for wave field reconstruction using saved boundaries. The input model and the grid for computation is converted into number of blocks using following formula:

```
No. of Blocks = RoundUp((Model Size-1)/(Block Size)+1)
```

Dimensions of the thread block is kept as 32 in X and Z direction. The list of GPUs used in the study are shown in Table 1.

![Table 1 : List of GPUs used in the study](image)

**Table 1 : List of GPUs used in the study**

<table>
<thead>
<tr>
<th>S.NO</th>
<th>GPU</th>
<th>Quadro 600</th>
<th>GeForce GTX 745</th>
<th>Tesla M60</th>
<th>Quadro P5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Compute Capability</td>
<td>2.1</td>
<td>5</td>
<td>5.2</td>
<td>6.1</td>
</tr>
<tr>
<td>2</td>
<td>Global Memory (Mbyte)</td>
<td>1023</td>
<td>4070</td>
<td>8129</td>
<td>16275</td>
</tr>
<tr>
<td>3</td>
<td>CUDA Core</td>
<td>96</td>
<td>384</td>
<td>2048</td>
<td>2560</td>
</tr>
<tr>
<td>4</td>
<td>GPU Clock rate</td>
<td>1280</td>
<td>900 Hz</td>
<td>1178</td>
<td>1734</td>
</tr>
</tbody>
</table>

**Numerical Example and Performance Analysis**

The implemented AFWI in time domain has been tested on the Marmousi model (Figure 2). The size of the studied model is 9km in lateral and 3km in depth (Figure 2). The sampling interval is taken as 12.5m in horizontal and vertical directions. 90 shots with 320 receivers are simulated at the station interval of 12.5m and source interval of 100m in split-spread setting. The shot-receiver geometry on the velocity model is shown in Figure 3.

Synthetic shots using ricker wavelet with dominant frequency of 10 Hz are generated using forward modeling for true model with 2nd order finite difference in time and fourth order in space. The total time for shot recording is 4s at the sampling interval of 1ms. The four shots are displayed in the Figure 4.

![Figure 2 : Velocity Profile of Marmousi Model](image)

![Figure 3: Source Receiver geometry with first shot and shot 58. Green symbol represent source and receivers are shown with dashed line. For the first shot, source is at the starting of the model and at first channel. As the shot increases with 100 m, the shot position shifted with 8 stations till it reaches the split-spread configuration. The last shot has the source at the last channel.](image)

![Figure 4 : Four shots (shot 15(a), 35 (b), 55 (c), 75 (d)) are displayed from the generated 90 shots. Ricker wavelet with dominant frequency of 10 Hz, was used for the generation of shots.](image)
Full Waveform Inversion on GPU’s were performed using a smoothen version of Marmousi model. The smoothening is done iteratively with 10 points in X & Z direction. The initial model is shown in Figure 5(a). The total number of iteration for Full waveform Inversion is kept as the 250. The inverted model at various iterations and final AFWI model at 250th iteration is shown in Figure 5. The implemented AFWI able to invert the model from highly smoothen initial model. As, the iteration increases the resolution of inverted velocity model also increases. The workflow shown in Figure 1, with model at 50th iteration, and observed, forward modelled and residual data for shot 45 is shown in Figure 6 for illustration of results at various stages.

Figure 5 : AFWI inverted velocity models at different iteration. The input smoothened model is shown in (a). Inverted model at 1st iteration is shown in (b), inverted model at 50th & 100th iteration is shown in (c) & (d) respectively. The final inverted velocity model at 250th iteration is shown in (e). It can be clearly seen that as the iteration increases the resolution of inverted model increase and converges close to the true model.
Scalability of Acoustic Full Waveform Inversion on Graphics Processing Units

Figure 6: Adapted workflow with snapshots of velocity model at 50th iteration, observed data, forward modelled data and residual data at shot 45 and final velocity model.

Performance Analysis
The AFWI implemented on GPU are tested on the different GPU’s as shown in Table 1. Different model sizes as shown in Table 2 are considered for analysis of the scalability of the AFWI algorithm on GPU. The total time for AFWI was 5s with time step of 1 ms. The run-time taken by AFWI for different models on selected GPU’s for the study is shown in Table 3 and plotted in Figure 7. The run time indicates that as the model-size increases the time taken by GPUs to process per grid point reduces. The speedup over different GPU’s are shown in Figure 8 with Quadro 600 GPU has taken as the base case. As the model size increases the speedup over base case increases, and the Quadro P5000 provides ~19 times and Tesla M60 provides ~11 times speedup over for model size of 1024X1024.

Both the GPUs can complete the AFWI job in relatively lesser amount of time as compared to other GPUs used in the study.

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Nx (Grid point in X direction)</th>
<th>Nz (Grid point in Z direction)</th>
<th>N (Total Grid Points)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>128</td>
<td>128</td>
<td>16384</td>
</tr>
<tr>
<td>2</td>
<td>128</td>
<td>256</td>
<td>32768</td>
</tr>
<tr>
<td>3</td>
<td>256</td>
<td>256</td>
<td>65536</td>
</tr>
<tr>
<td>4</td>
<td>256</td>
<td>512</td>
<td>131072</td>
</tr>
</tbody>
</table>

Table 2: Grid points in X and Z directions for different models used in analysis.

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Model Size Multiple</th>
<th>Quadro 600</th>
<th>GeForce GTX 745</th>
<th>Tesla M60</th>
<th>Quadro P5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.47</td>
<td>1.04</td>
<td>0.27</td>
<td>0.25</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2.71</td>
<td>1.74</td>
<td>0.37</td>
<td>0.29</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5.17</td>
<td>3.38</td>
<td>0.61</td>
<td>0.39</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>10.06</td>
<td>6.42</td>
<td>1.11</td>
<td>0.69</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>20.06</td>
<td>12.70</td>
<td>2.03</td>
<td>1.21</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>39.31</td>
<td>24.88</td>
<td>3.74</td>
<td>2.15</td>
</tr>
<tr>
<td>7</td>
<td>64</td>
<td>78.90</td>
<td>49.10</td>
<td>7.10</td>
<td>4.00</td>
</tr>
</tbody>
</table>

Table 3: Average run time per shot taken by AFWI on different GPUs for different model sizes (multiple of 128X128).

Figure 7: Average run time (sec) taken by AFWI for different models with multiples of 128X128.

Figure 8: Speedup over Quadro 600 GPU as compared to other GPUs. It can be seen that Tesla M60 and Quadro P5000 always performs better than other...
Scalability of Acoustic Full Waveform Inversion on Graphics Processing Units

Conclusions

In this paper we have shown the implementation of 2D time domain Full Waveform Inversion in acoustic medium on Graphics Processing Units. Finite Difference Time Domain methods are used for forward modelling, back propagation of residual and computation of gradient function on GPU. The performance analysis of implemented AFWI was carried out on different GPUs, and the results demonstrate the scalability of the codes on different GPU architecture. The numerical test validates the implementation of AFWI. The current implementation is for single GPUs in 2D medium. Further implementation for higher dimension and use of Multi-GPU may be used to achieve higher speedup and industrial scale solutions.

Acknowledgment

This work was carried out at Data interpretation Center, Petroleum Research wing, GERMI. We thank Prof. Anirbid Sircar, Director General, GERMI for his Support during this study. The entire team of Data Interpretation Center is acknowledged for their support at every stage of this project.

References