Geophysical Prospecting, 2012, **60**, 906–918 doi: 10.1111/j.1365-2478.2011.01032.x

The issues of prestack reverse time migration and solutions with Graphic Processing Unit implementation

Hongwei Liu[1]*, Bo Li[1], Hong Liu[1], Xiaolong Tong[2], Qin Liu[2], Xiwen Wang[3] and Wenqing Liu[3]

[1]*Key Laboratory of Petroleum Resources Research, Institute of Geology and Geophysics, Chinese Academy of Sciences, Beijing 100029, China* [2]*Beijing Geostar Science and Technology Co. Ltd, Beijing 100029, China.* [3]*Research institute of Petroleum Exploration & Development-Northwest, Petrochina, Lanzhou 730020, China*

Received June 2011, revision accepted October 2011

ABSTRACT

Prestack reverse time migration (RTM) is a very useful tool for seismic imaging but has mainly three bottlenecks: highly intensive computation cost, low-frequency band imaging noise and massive memory demand. Traditionally, PC-clusters with thousands of computation nodes are used to perform RTM but it is too expensive for small companies and oilfields. In this article, we use Graphic Processing Unit (GPU) architecture, which is cheaper and faster to implement RTM and we obtain an order of magnitude higher speedup ratio to solve the problem of intensive computation cost. Aiming at the massive memory demand, we adopt the pseudo random boundary condition that sacrifices the computation cost but reduces the memory demand. For rugged topography RTM, it is difficult to deal with the rugged free boundary condition with the finite difference method. We employ a simplified boundary condition that avoids the abundant logical judgment to make the GPU implementation possible and does not induce any sacrifice on efficiency. Besides, we have also done some tests on multi-GPU implementation for wide azimuth geometries using the latest GPU cards and drivers. Finally, we discuss the challenges of anisotropy RTM and GPU solutions. All the jobs stated above are based on GPU and the synthetic data examples will show the efficiency of the algorithm and solutions.

Key words: RTM, GPU, Pseudo random boundary, Rugged topography, Anisotropy.

INTRODUCTION

The history of reverse time migration (RTM) can be traced back to 1978. Hemon (1978) proposed the basic idea of RTM by solving the wave equation using the finite difference (FD) method. But the first time that RTM was used in seismic exploration was in 1983 and it should be attributed to the work of several independent authors (Baysal, Kosloff and Sherwood 1983; Loewenthal and Mufti 1983; McMechan 1983). Originally, all these works were for post-stack migration. The expensive calculation cost and the massive memory demand limited the popularization of RTM and it was not used in the migration industry for a long time. On the contrary, the Kirchhoff integral method and one-way wave equation (OWE) method are widely used because of lower computation cost and smaller memory demand. In recent years, due to the fast development of computer technology and increasing demand of accurate imaging for a complex structure reservoir, RTM has regained attention in the seismic community. RTM has been developed from post-stack to prestack migration, from 2D to 3D migration (McMechan 1990), from acoustic to elastic wave equation migration (Chang and McMechan 1994) and from streamer data to VSP data migration

*E-mail: liuhw@mail.iggcas.ac.cn

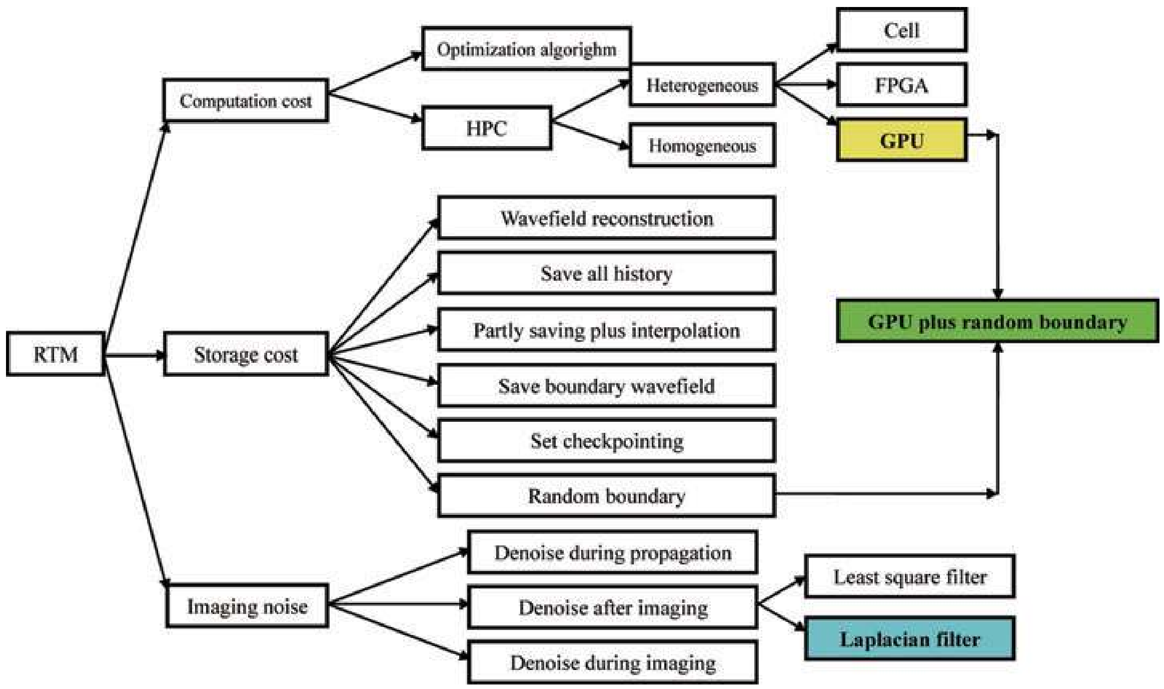


Figure 1 The bottlenecks of RTM and solutions. The bottlenecks include the highly intensive computation cost, massive memory demand and low frequency band imaging noise. We will mainly discuss the first two bottlenecks in this article. FPGA, Field Programmable Gate Array; HPC, High Performance Computing; RTM, Reverse Time Migration; GPU, Graphic Processing Unit.

(Neklyudov and Borodin 2009); from isotropic to tilted transversely isotropic (TTI) media migration (Zhang, Rector and Hoversten 2005; Du, Bancroft and Lines 2007; Fletcher, Du and Fowler 2009; Duveneck and Bakker 2011; Zhang, Zhang and Zhang 2011). But like any other imaging methods, RTM does have some limitations. As shown in Fig. 1, there are mainly three bottlenecks that prevent the wide use of RTM: highly intensive computation cost, massive memory demand and low-frequency band imaging noise. We will mainly discuss the first two in this article.

At present, there are mainly two ways to accelerate RTM: one way is to reduce the computational cost from the algorithm aspect and the other way is to use high-performance parallel computing to speed up RTM. In the former case, one mainly tries to use a larger time step and larger grid size. For example, Zhang and Zhang (2009) introduced a one-step method with a complex wavefield and square-root operator that reduces a second-order acoustic wave equation to a first-order partial differential equation similar to one-way wave equation (OWE). In this algorithm, the time step can be chosen up to Nyquist frequency without the limitation of a stability condition and time dispersion relation. Soubaras and Zhang (2008) pointed out that the pseudo-differential operator in the one-step method is difficult to generate and proposed a two-step method that overcomes the limits and maintains the high efficiency of the one-step method. Because of the limitations of the dispersion relation and stability condition, the first way to accelerate RTM could not completely solve the computation bottleneck. The second way to accelerate RTM relies on the improvement of the performance of computer hardware. Traditionally, RTM was done on PC-clusters with thousands of computation nodes. Villarreal and Scales (1997) implemented wave equation forward modelling using FD by domain decomposition on PC-clusters. Recently, the emergence of heterogeneous processors (Cell, Field Programmable Gate Array (FPGA), GPU and others) has provided new opportunities for RTM. Araya-Polo *et al.* (2009) implemented RTM on a cell processor; Micikevicius (2009) implemented the high order finite differences method using GPU; Foltinek *et al.* (2009) and Liu *et al.* (2011) made some useful attempts in RTM using GPU. In this article, we will focus on the use of GPU to accelerate RTM.

The memory demand is another bottleneck of prestack Reverse Time Migration. RTM needs the shot and receiver wavefields for the imaging computation at the same time. However, the extrapolation of the shot and receiver wavefields is along opposite time directions. This requires us to save the shot or receiver wavefield before the imaging computation that requires large disk space. For example, for a 500 × 1000 × 1000 3D grid, the number of modelling steps is usually in the order of several thousands, which means to save the whole propagation history we need about 20 TBs space. It is too big for the memory of the present hardware and so we must adopt an effective algorithm to reduce the memory demand. The usual practice is to compress the shot wavefield snapshots every few time steps and save these snapshots in the hard disk or random-access memory (RAM); when extrapolating the receiver wavefield along the reverse time direction, read back and decompress the shot wavefield snapshots and apply a proper imaging condition. For example, IBM (Perrone *et al.* 2011) used the Blue Gene/P (BGP) supercomputer, which has 1024 nodes and 4 TB RAM per rack to perform RTM. The RAM is so big that the algorithm does not need to compress the shot wavefield snapshots. Except for this usual practice, there are some other solutions. Symes (2007) used the optimized 'checkpointing' method that only saves the wavefields at some time checkpoints and recomputes the wavefields at other time points from these checkpoints. The 'recomputation ratio' is reduced by choosing optimal checkpoints. However, the recomputation ratio may be very high if the number of checkpoints is not enough and if too many checkpoints are chosen, the memory demand will be high too. Clapp (2009) used the pseudo random boundary method that does not need to save the shot wavefield for RTM. In this article, we will use the latter method to implement RTM using a GPU/CPU collaborative computation. Another implication of massive memory demand is that with the increasing acquisition aperture and increasing demand for high-resolution imaging, the size of data for a single shot reaches the order of tens of GBs. For example, the number of grids for a single shot of the SEG Advanced Model (SEAM) is 13.5 GB and one 3D TTI migration needs about 1 TB RAM. This is too much for GPU because even the latest GPU (Tesla M2090) has only 6 GB RAM. In this article, we will do some multi-GPU tests on 3D forward modelling using two Tesla C2050 GPU cards and the latest CUDA (Compute Unified Device Architecture) driver.

With the development of seismic exploration, areas with rugged topography and complex geological settings such as mountains, beaches and marshes have gained considerable attention. This has proposed new challenges for seismic processing: how to deal with the fluctuating surface and the complex near-surface velocity model; how to image complex subsurface structures such as strong fold, faults, steep tectonic and large changes in a stratum. It is difficult for traditional seismic imaging methods to accurately image these areas. There are mainly two ways to solve the rugged topography problem: the first one is to use the elevation static moveout method or wave equation datuming methods (Berryhill 1979; Berryhill 1984; Yilmaz and Lucas 1986; Bevc 1997) to redatum the data to a floating or fixed datum; the other way is to perform depth migration directly from rugged topography. In this article, we migrate the data directly from the rugged topography, which implicitly includes the static correction. This correction not only includes the vertical component of traveltime but also includes the horizontal component. Therefore, this method can solve the complex problem mentioned above. Compared to the ray based Kirchhoff integral method (Wiggins 1984) and one-way wave equation methods (Reshef 1991; Beasley and Lynn 1992), RTM directly solves the two-way wave equation and has no limitations of imaging angles and so far it is the most accurate imaging method and a better choice for imaging the complex problem mentioned above.

Due to certain geological reasons, most rocks have anisotropy or layer-induced anisotropy (Thomsen 1986). RTM needs to take anisotropy into account to obtain correct images. Shale layering that overlies dipping salt flanks can cause TTI anisotropy. Ignoring the tilted symmetry not only causes image blurring and mis-positioning of the salt flanks but also degrades and distorts the base of salt and subsalt images. In recent years, most researchers have adopted the pseudo acoustic anisotropy equations to perform anisotropic RTM (Zhang *et al.* 2005; Du *et al.* 2007; Fletcher *et al.* 2009; Duveneck and Bakker 2011; Zhang *et al.* 2011). Unlike the elastic wave equation, the wavefield in acoustic media is a scalar quantity rather than a vector. No splitting filters are needed to separate P-waves from S-waves when using this acoustic equation and the computation cost is much smaller (Alkhalifah 2000). Compared to the isotropic acoustic equation, the computation cost and memory demand of the TTI pseudo acoustic equation is several times larger. We will use GPU to speed up TTI RTM in this article.

GPU was designed as a device for computer display commonly known as 'display card'. In 2006, NVIDIA released CUDA version 1.0 (the latest version of the driver is 4.0 in 2011), which allows developers to write code to run on GPU, which is usually performed on CPU. Compared to traditional PC-clusters, GPU computing takes the advantages of multi-cores, lower cost of hardware, lower power consumption and smaller space. GPU has recently been widely used in life sciences, medical devices, industrial production, electronic design automation, manufacturing, finance and telecommunication industries. In the oil and gas industry, Geostar (Li, Liu and Liu 2009; Liu *et al.* 2009; Liu *et al.* 2010), Hess, CGGVeritas, Chevron, Headwave, Acceleware (Foltinek *et al.* 2009), Tsunami, Petrobras (Souza *et al.* 2011) and SeismicCity are among the first in this research area.

This article is organized as follows: first, we will discuss the implementation of prestack RTM and GPU acceleration and the BP 2004 (Billette and Brandsberg-Dhal 2005) synthetic data will be used to benchmark the speedup ratio; next, we will introduce the implementation of RTM based on the pseudo random boundary condition to reduce the memory demand; then, we will do some multi-GPU tests for 3D forward modelling using two Tesla C2050 GPUs and the latest CUDA driver; we then discuss the RTM from rugged topography, the free surface boundary condition and the process of GPU acceleration; finally, the anisotropy RTM and GPU acceleration will be discussed.

Prestack Reverse Time Migration and Graphic Processing Unit acceleration

Prestack RTM in the shot domain mainly contains three steps: propagating the shot wavefield along the positive time direction to the maximum recording time by solving equation (1)

$$\left(\frac{\partial^2}{\partial t^2} - v(\mathbf{x})^2 \nabla^2\right) p_s(\mathbf{x}, t) = \delta(\mathbf{x} - \mathbf{x}_s) f(t), \tag{1}$$

and saving the shot wavefield; propagating the receiver wavefield *R(x,y,t)* along the negative time direction by solving equation (2)

$$\begin{cases} \left(\frac{\partial^2}{\partial t^2} - v(\mathbf{x})^2 \nabla^2\right) p_r(\mathbf{x}, t) = 0, \\ p_r(x, y, z = 0, t) = R(x, y, t), \end{cases} \tag{2}$$

and reading back the shot wavefield; applying a proper imaging condition and summing up the results for all the shots to obtain the final imaging result. We use an explicit FD scheme to solve the acoustic wave equation due to its computation efficiency and it is straight forward to implement domain partition. To reduce numerical dispersion, we use a second-order FD scheme for the time derivative and a high order FD scheme for the spatial derivatives, as shown in equation (3),

$$\begin{aligned} p^{n+1}_{i,j,k} &= 2p^n_{i,j,k} - p^{n-1}_{i,j,k} + dt^2 v^2_{i,j,k} \\ &\times \left(\sum_{l=1}^{N/2} A_l \frac{p^n_{i+l,j,k} - 2p^n_{i,j,k} + p^n_{i-l,j,k}}{dx^2}\right. \\ &+ \sum_{m=1}^{N/2} A_m \frac{p^n_{i,j+m,k} - 2p^n_{i,j,k} + p^n_{i,j-m,k}}{dy^2} \\ &\left. + \sum_{n=1}^{N/2} A_n \frac{p^n_{i,j,k+n} - 2p^n_{i,j,k} + p^n_{i,j,k-n}}{dz^2}\right), \end{aligned} \tag{3}$$

Table 1 The device specifications of the GPU

Device Specifications (GPU)		
GPU Model	NVIDIA Tesla C1060	NVIDIA Tesla C2050
Global memory	4GB, 800MHz GDDR3	3GB, 1.5GHz GDDR5
Memory Bandwidth	102 GB/s	144 GB/s
Number of Scalar Processors	240	448
CUDA Driver Version	2.3	4.0
CUDA Capability revision	1.3	2.0
Clock rate	1.3 GHz	1.15 GHz
floating point performance (Single Precision)	933 GFlops	1.03 Tflops

where N is the order of finite-difference and the FD coefficients are computed using the method in Fornberg (1988). Chattopadhyay and McMechan (2008) proposed a systematic and comprehensive introduction to RTM imaging conditions. In this article, we choose the cross-correlation imaging condition equation (4)

$$I(\mathbf{x}) = \int p_s(\mathbf{x}, t)\, p_r(\mathbf{x}, t)\, dt, \tag{4}$$

because it is easy to implement, suitable for parallel computing and there is no stability problem compared to the decon-type imaging condition.

In the above equations, $p_s(\mathbf{x}, t)$ and $p_r(\mathbf{x}, t)$ are the source and receiver extrapolated wave-fields, respectively, $v(\mathrm{x})$ is the velocity function at the imaging location; $f(t)$ is the source function, $R(x, y, t)$ is the received seismic data function, $I(\mathbf{x})$ is the imaging function and A_l, A_m, A_n are FD coefficients. Being different from equation (1), equation (2) is a boundary value problem (BVP). To solve this BVP numerically, the boundary values should be set to $R(x, y, t)$ at the beginning of each time step wavefield extrapolation. Equations (3) and (4) indicate that during wavefield propagation and imaging, the computations at each grid point are independent. The computation at all the grid points are calculated in a parallel way and the parallel granularity is very small. Each thread only calculates one or several grid points. Different from the PC-cluster that is suitable for larger granularity, GPU has more cores and is more suitable for this problem. Table 1 shows the parameters of two types of GPU cards. In our first test we use the card of the first type – Tesla C1060. The card has 240 Scalar Processors (SP) and 4 GB GDDR3 memory and the peak floating point performance is 933GFlops. The CUDA driver version is 2.3 and the CUDA capability revision is 1.3. Micikevicius (2009) introduced the implementation progress

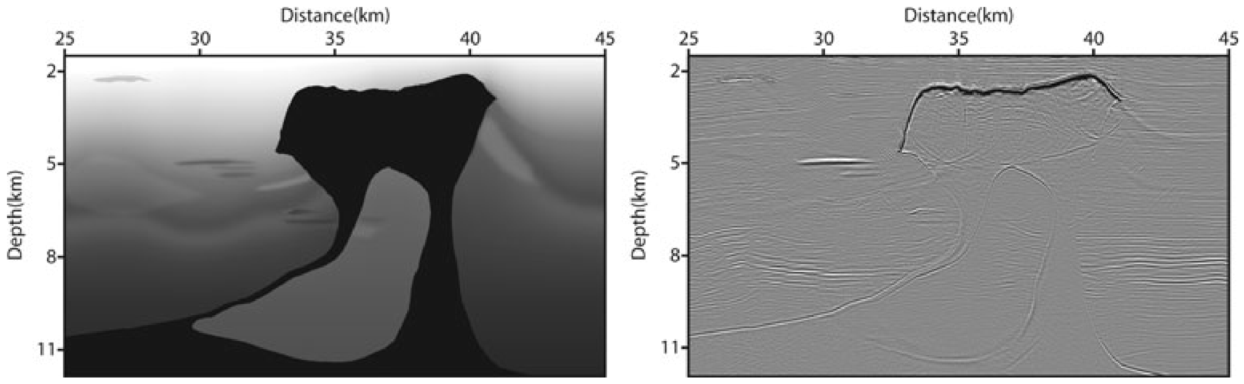


Figure 2 Migration test on the BP 2004 synthetic model. (a) The central part of the velocity model; (b) the result of Graphic Processing Unit based reverse time migration after removing the low frequency imaging noise using 88 Tesla C1060 GPU cards.

of equation (3) using GPU. The acceleration of RTM using GPU mainly contains two aspects:

(**1**) because GPU usually has many cores and the computation at independent grid points is computed in parallel, each core only needs to calculate one or several grid points;
(**2**) there are a larger number of memory accesses when using a high order FD scheme. Taking the case of a 3D eight-order FD as an example, to calculate the value of each grid point, we need to read 25 grid points around the target grid point and therefore the read redundancy is very high. To solve this problem we use the idea of a matrix slice and with the help of shared memory, the data are reused and the matrix read redundancy is reduced significantly.

We use the BP 2004 (Billette and Brandsberg-Dhal 2005) synthetic data as the benchmark test of the computation speed of GPU-based RTM. Figure 2(a) is the central part of the model and is a simplified representation of geologic features in the Eastern/Central Gulf of Mexico and off-shore Angola and the imaging challenge is the salt delineation. Figure 2(b) is the imaging result of the GPU-based RTM after removing low-frequency imaging noise. The salt body and vertical salt boundary at the right are clearly imaged. In our test, we use 88 Tesla C1060 GPUs and the computation time is ten minutes.

Storage and pseudo random boundary condition

To apply the imaging condition, Reverse Time Migration needs the shot and receiver wavefields at the same time. However, the shot and receiver wavefields are extrapolated along opposite time directions. This requires us to save the propagation history of the shot or receiver wavefield. The propagation history is usually too big for the memory of the present hardware and so we must adopt an effective algorithm to reduce the memory demand. The usual practice is to compress the shot wavefield snapshots every few time steps and save these snapshots in the hard disk or RAM; when extrapolating the receiver wavefield along the reverse time direction, read back and decompress the shot wavefield snapshots to compute the image. Clapp (2009) proposed the pseudo random boundary method that does not need to save the shot wavefield for RTM.

Figure 3(a) is the velocity model with pseudo random boundaries; the shot position is at the centre. Figure 3(b) is the wavefield snapshot at $t = 0.75$ s. On the one hand the wavefront has become random noise that would not form a continuous event during the imaging and on the other hand, the boundary values are used to reform the useful wavefield during back-propagation. Figure 4(a) is the original flowchart of GPU/CPU collaborative computing RTM; Fig 4(b) is the flowchart of GPU/CPU collaborative computing RTM using the pseudo random boundary condition. There are several advantages to using the random boundary condition in RTM:

(**1**) the shot wavefields in second time modelling and receiver wavefield modelling propagate along the reverse time direction simultaneously, so the history of the shot wavefield needs not to be saved;
(**2**) although the pseudo random boundary method increases the computation cost due to additional shot wavefield modelling, the communication cost between the GPU memory and the CPU memory is largely reduced, which is more important;
(**3**) traditional RTM needs to use the absorbing boundary condition to reduce boundary reflections. The absorbing boundary condition (ABC) equation is quite different from the acoustic wave equation and this will largely slow down the computation. When using the pseudo random boundary condition, we do not need to absorb the wavefield at the boundaries;

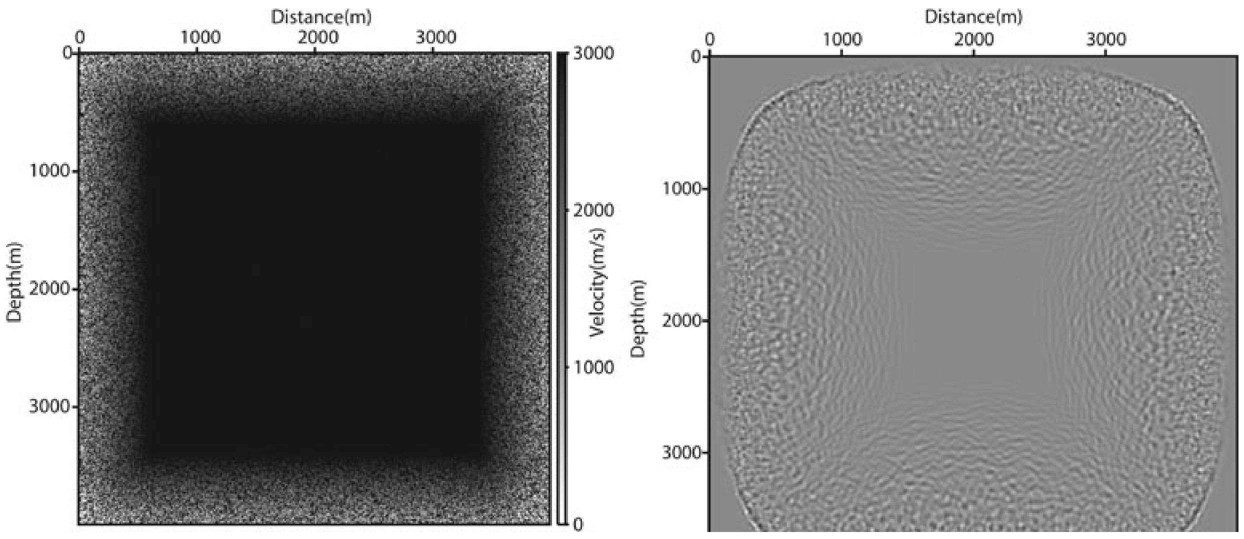


Figure 3 Random boundary model test. (a) The velocity model with 30 layers at the boundaries as pseudo random borders; (b) the wave-field snapshot at t = 0.75 s, the wave-field at each time step before 0.75 s could be reformed from this snapshot and they do not need to be stored.

(4) in the flowchart of the pseudo random boundary condition algorithm, the wavefields at all the time steps are accumulated because we do not need to save these wavefields in the memory; while in the traditional flowchart, one usually has to save the wavefields every several time steps because the entire wavefields at all the time steps are too big for the memory of the present hardware;
(5) the random noise will introduce some high-wavenumber noise during the imaging progress but these high wavenumber noises are removed during the stacking of all shots.

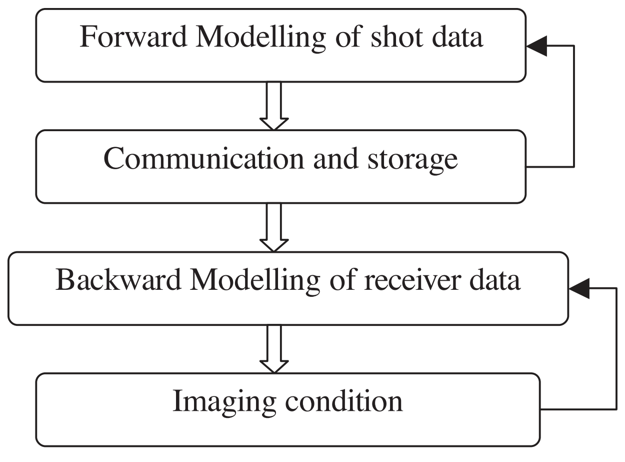
Figure 5(a) is the 2D salt velocity model; Fig 5(b) is the result of the traditional GPU/CPU collaborative computing RTM, which saves the propagating history of the shot wavefield and uses absorbing boundary condition for both the shot and receiver wavefields; Fig. 5(c) is the result of GPU/CPU collaborative computing RTM using the pseudo random boundary condition. The difference is negligible, which proves the effectiveness of the pseudo random boundary method.

Multi-Graphic Processing Unit test

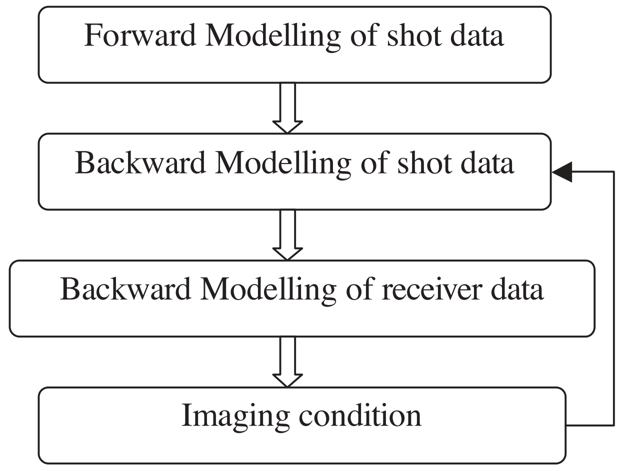
As shown in Fig. 6, the difference of the parallel computing technology between the GPU cluster and the CPU cluster is very evident. For the CPU cluster, the program still runs sequentially in each computation node; while for the GPU cluster case, the grid points in each node are also computed simultaneously. The problem of domain decomposition is that for each time step of wavefield extrapolation, the data are transferred between neighbouring nodes and the communication is very time consuming. For traditional GPU architecture, data communication between GPUs must recur to the host CPU RAM (copying the data from GPU A to the host RAM and copying the data from host RAM to GPU B) and this will slow down the program to a large extent. In 2011, NVIDIA released GPUDirect^{TM} technology based on T20 serial GPU cards, which can directly transfer data between two GPUs without the help of CPU RAM. Meanwhile, the Unified Virtual Addressing (UVA) technology greatly reduces the programming difficulty. These new technologies provide the basis for multi-GPU RTM.

For the explicit FD algorithm, the calculation of the wavefield at each time step only needs the snapshots at early time steps, which means the communication areas and the other ones are calculated separately. As shown in Fig. 7, the green areas need to be transferred, so they should be calculated first. After this, with the help of 'stream' technology, the data transfer from the green areas to the red areas and the calculation of the yellow areas are done concurrently. The computation reduces the communication latency to a large extent, which provides a better mechanism to enhance the performance of the algorithm.

The parameters of Tesla C2050 GPU are shown in Table 1. The card has 448 SPs, 3 GB GDDR5 memory and the peak floating point performance is 1.03 TFlops. The CUDA driver version is 4.0 and the CUDA capability revision is 2.0. This card supports the GPUDirect^{TM} and UVA technologies mentioned above. We use two Tesla C2050 GPU cards to perform a forward modelling test. The grid size is $400 \times 400 \times 200$ and


Forward Modelling of shot data
Communication and storage
Backward Modelling of receiver data
Imaging condition

(a)


Forward Modelling of shot data
Backward Modelling of shot data
Backward Modelling of receiver data
Imaging condition

(b)

Figure 4 The flowcharts of reverse time migration of different algorithms. (a) The traditional flowchart of GPU/CPU collaborative computing RTM which needs to store the wave propagation history; (b) the flowchart of GPU/CPU collaborative computing RTM using random boundary condition.

we use a twelfth-order explicit FD algorithm for the spatial derivatives to extrapolate the wavefield for 3520 time steps. The computation time on one card is 49 s while 26 s on two cards, and which we can see that the communication latency is significantly reduced.

Reverse Time Migration from rugged topography

The finite-difference (FD) method is widely used to solve the two-way wave equation since it is computationally efficient and easy to simulate seismic wave propagation in complex media. The drawback of this method is that it is difficult to deal with rugged topography because the grid points at the free boundary require special treatment, which involves many logical judgments. Different from CPU's processor micro-architecture, GPU sets a large portion of the transistors as an Arithmetic Logical Unit (ALU), while only a very few parts for dynamic storage. Because it significantly reduces the buffer space, GPU has difficulties in dealing with the problems of logical judgment and non-linear addressing. Therefore, for the problem with a large number of logic scientific computing, the advantage of GPU computing will be greatly reduced. Therefore, we need to find a new way to handle the free boundary condition to avoid the large number of logical judgments. The free surface boundary condition at rugged topography is as follows:

$$p_\Omega(x, y, z = 0) = 0, \tag{5}$$

where Ω represents the free surface. Traditionally, the free boundary grid points are divided into several different categories and different equations are used for the numerical free boundary condition for each category (for example, Yuan *et al.* 2011). This results in a lot of logical judgments. In this paper, we use a wavefield filter to implement a free boundary condition that is similar to the OWE (One-way wave equation) method in Reshef (1991).

$$p(x, y, z, t) = p(x, y, z, t)\, filt(x, y, z). \tag{6}$$

$$filt(x, y, z) = \begin{cases} 1 & z > z_0, \\ 0 & z < z_0. \end{cases} \tag{7}$$

In equation (7), z_0 denotes the elevation of the free surface and the positive direction is pointing downward. The difference between RTM and OWE is that the filter is applied in the frequency-domain for OWE while it is applied in the time domain for RTM. This approach not only satisfies the free boundary condition equation (5) but also avoids detailed classification of the boundary grids. As a result, the additional cost of the use of GPU to implement a free boundary condition is negligible.

Figure 8 is the flow chart of RTM using GPU for rugged topography. Different from the OWE method, RTM extrapolates the wavefield along the time direction and the receiver data are directly positioned on the true surface, which is beneficial for GPU implementation. It should be emphasized that, when extrapolating the wavefield along the time direction, we still use regular spatial grids and only need to apply the filter (equations (6) and (7)) during each time step.

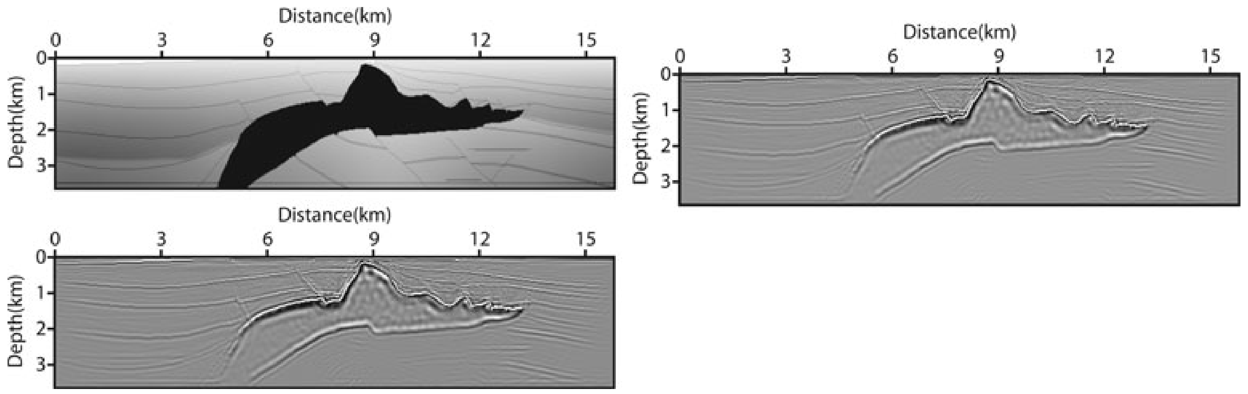


Figure 5 Test on 2D salt model. (a) The velocity model; (b) the result of the traditional GPU/CPU collaborative computing RTM using absorbing boundary condition; (c) the result of GPU/CPU collaborative computing RTM using random boundary condition.

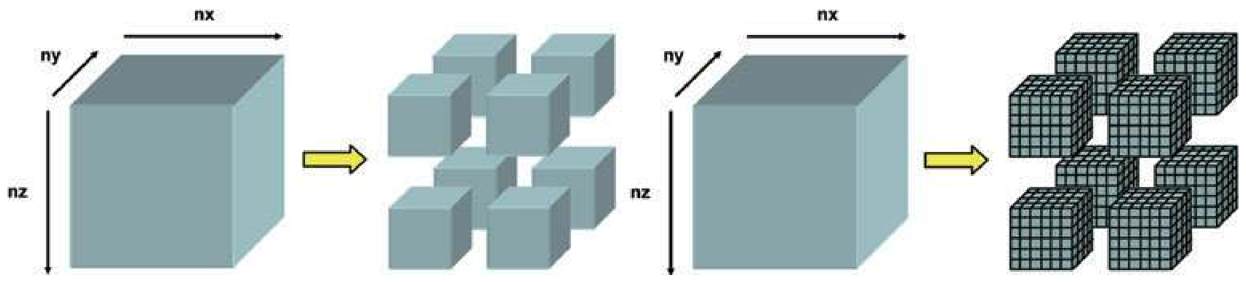


Figure 6 The difference between CPU and Graphic Processing Unit clusters for domain decomposition. (a) The CPU cluster case, the program still takes the serial strategy in each computation node; while for the GPU cluster case (b), the grid points in each node could also be computed simultaneously.

In this section we will test the proposed algorithm on the Canadian overthrust synthetic data set prepared by Gray and Marfurt (1995), which was widely circulated among Canadian contractor companies in the mid 1990s. The velocity depth model (as shown in Fig. 9a) is typical in north-eastern British Columbia. The model is 25 km long. The top of the model is 2000 m above sea level and the bottom of the model is 8000 m below sea level. The total relief of the earth's surface along the cross-section is approximately 1600 m. The lateral velocity variation is significant, the thrust faults are well developed and there are several negative structures. We test OWE and RTM on this model and the results are described as follows.

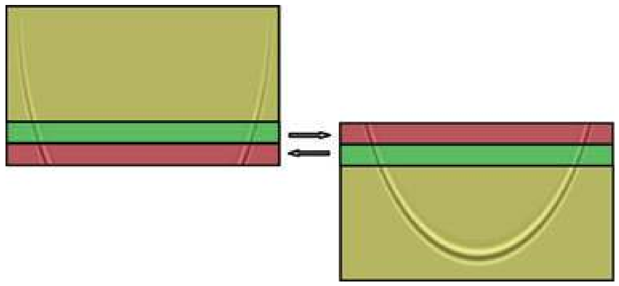


Figure 7 Multi-Graphic Processing Unit forward modeling test. The green districts are calculated firstly. After that, the communication from the green districts to the red ones and the calculation of the yellow districts could be issued concurrently. The computation could hide the communication latency to a large extent.

Figure 9(a) is the velocity model; Fig 9(b) is the RTM result of all the 277 shot data after removing low-frequency noise and Fig. 9(c) is the result of OWE using the phase-shift plus interpolation (PSPI) method (Gazdag and Sguazzero 1984) and Reshef's (1991) algorithm to deal with the rugged topography problem. The shallow part of the RTM result is clearer and the fault is more continuous. The seismic events of the A and B areas in Fig. 9(c) are discontinuous because of the propagation angle limitation, or the failure to properly handle the sharp velocity contrast by the PSPI method. RTM does not have these limits and the imaging result is improved significantly, as shown in Fig. 9(b). For the negative fault of the C area in Fig. 9(c), both RTM and OWE failed to obtain a good image because the design of the observing system prevents reflection and transmission and other useful information from being received at the surface.

In this test, the type of GPU is Tesla C1060 and the CUDA version is 2.3. The type of CPU is Pentium(R) Dual-Core CPU E5420 at 2.50 GHz with 24 GB DDR2 RAM.

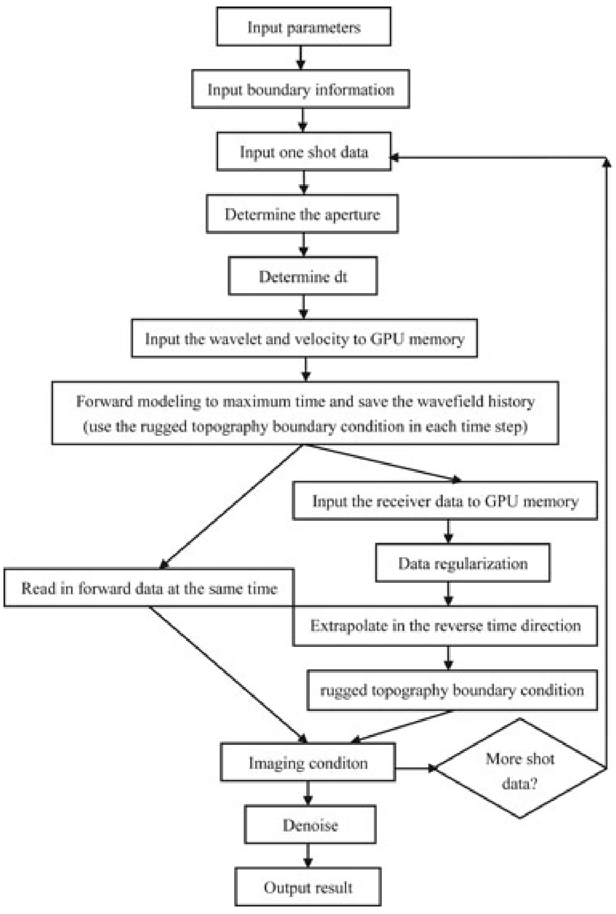

Input parameters
Input boundary information
Input one shot data
Determine the aperture
Determine dt
Input the wavelet and velocity to GPU memory
Forward modeling to maximum time and save the wavefield history (use the rugged topography boundary condition in each time step)
Input the receiver data to GPU memory
Data regularization
Read in forward data at the same time
Extrapolate in the reverse time direction
rugged topography boundary condition
Imaging conditon
More shot data?
Denoise
Output result

Figure 8 The flow chart of RTM using GPU for rugged topography. The receiver data could be directly positioned on the true surface which is beneficial for GPU realization. We still use regular spatial grids and only need to filter the data after each time step.

RTM steps along the time direction and needs 6005 recursive steps and the OWE method needs to migrate 234 frequency components. For the computation time, the GPU-based RTM operation needs 20 s for one shot, while the CPU-based PSPI method needs 860 s.

Anisotropy Reverse Time Migration and Graphic Processing Unit acceleration

Solving the eigenvalues of the Christoffel equations for homogeneous TI media gives three distinct wave modes: qP, qSV and qSH. The qSH mode decouples and for the coupled qP and qSV modes, we have the following fourth-order dispersion equation.

$$\begin{aligned}\omega^4 = &\left(\left(v_{px}^2 + v_{sz}^2\right)k_x^2 + \left(v_{pz}^2 + v_{sz}^2\right)k_z^2\right)\omega^2 - v_{px}^2 v_{sz}^2 k_x^4 \\ &- v_{pz}^2 v_{sz}^2 k_z^4 + \left(v_{pz}^2\left(v_{pn}^2 - v_{px}^2\right) - v_{sz}^2\left(v_{pn}^2 + v_{pz}^2\right)\right)k_z^2 k_x^2.\end{aligned} \tag{8}$$

This equation corresponds to an equation with a fourth-order time derivative and a fourth-order spatial derivative equation in the time space domain. Since it is difficult to directly solve this equation researchers usually reformulate this equation into two equations with second-order time and space derivatives (Zhou, Zhang and Bloor 2006a; Fletcher *et al.* 2009; Duveneck and Bakker 2011; Zhang *et al.* 2011). In this article, we use Fletcher's equation (Fletcher *et al.* 2009) as shown in equation (9) for the 2D-TTI RTM.

$$\begin{aligned}\frac{\partial^2 p}{\partial t^2} &= v_{px}^2 H_2 p + v_{pz}^2 H_1 q + v_{sz}^2 H_1 (p - q), \\ \frac{\partial^2 q}{\partial t^2} &= v_{pn}^2 H_2 p + v_{pz}^2 H_1 q + v_{sz}^2 H_2 (q - p).\end{aligned} \tag{9-1}$$

$$\begin{aligned}H_1 &= \sin^2\theta\frac{\partial^2}{\partial x^2} + \cos^2\theta\frac{\partial^2}{\partial z^2} + \sin 2\theta\frac{\partial^2}{\partial x\partial z}, \\ H_2 &= \cos^2\theta\frac{\partial^2}{\partial x^2} + \sin^2\theta\frac{\partial^2}{\partial z^2} - \sin 2\theta\frac{\partial^2}{\partial x\partial z}.\end{aligned} \tag{9-2}$$

In this equation, q is the auxiliary function, v_{pz} is the P-wave velocity in the direction normal to the symmetry plane; $v_{pn} = v_{pz}\sqrt{1+2\delta}$ is the P-wave normal moveout (NMO) velocity, again relative to the normal to the symmetry plane; $v_{px} = v_{pz}\sqrt{1+2\varepsilon}$ is the P-wave velocity in the symmetry plane; $\varepsilon$and $\delta$are the Thomsen dimensionless anisotropy parameters defined by Thomsen (1986); sgetting v_{sz} to zero in equation (9) leads to computation saving. However, it was reported that the simplified equations introduce numerical instability at the locations where sharp contrasts in the azimuth and dip models exist. A remedy to this problem can be found in Duvenneck and Bakker (2011) or Zhang *et al.* (2011). In Fletcher's equation, v_{sz} is defined as:

$$v_{sz}^2 = \frac{v_{pz}^2}{0.75}(\varepsilon - \delta). \tag{10}$$

Compared to the isotropic acoustic RTM, there are mainly two problems for TTI acoustic RTM. The first one is that the TTI RTM needs more RAM for more model parameters and the auxiliary function wavefields. The main challenge of TTI RTM is that there are cross-partial derivatives that need more memory access and computation cost compared with the single central derivatives for the FD method in the isotropic case. Take the 2D case as an example (we use a twelfth-order central FD scheme for the spatial derivatives); as shown in Fig. 10, the computation of the second-order spatial derivatives along a single direction only needs to read 13 grid points values, while the cross-derivatives (equation (11)) need to read 144 grid point values. The memory read redundancy is so high that it becomes the dominant part of

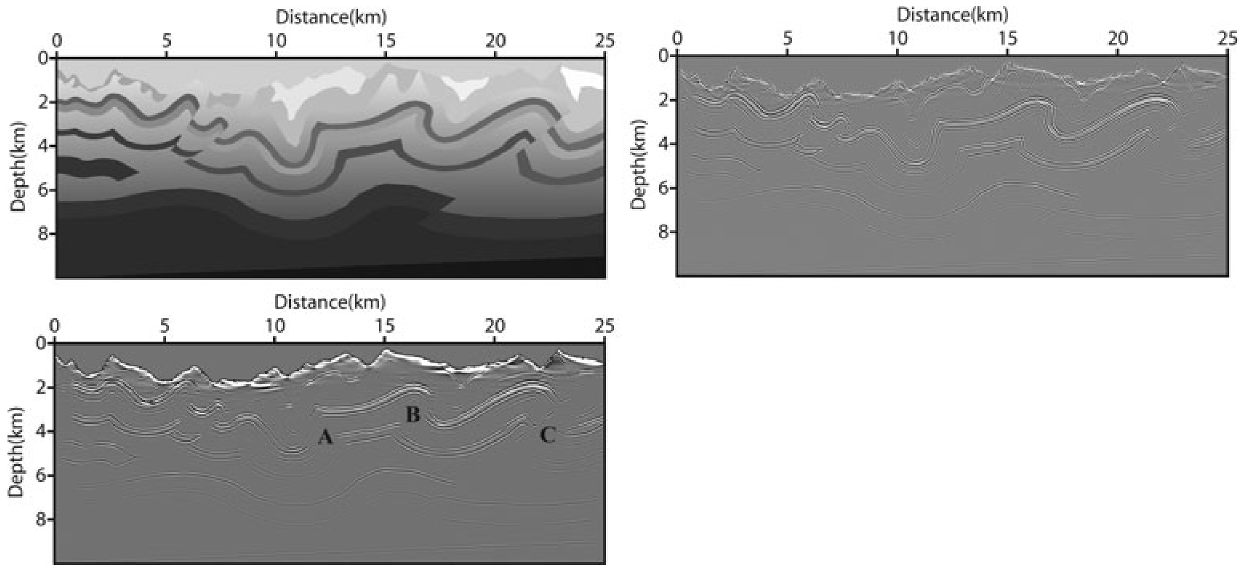


Figure 9 Migration test for rugged topography foothill model. (a) The velocity model; (b) the result of Reverse Time Migration after removing the low frequency noise; (c) the result of the PSPI OWE method. The seismic events of the A and B areas of the RTM result is better than the OWE method; while both of the two results of the C area are not ideal because the design of observing system.

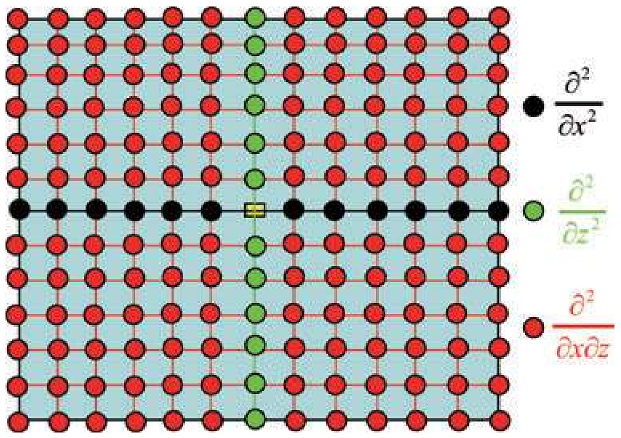

$\frac{\partial^2}{\partial x^2}$
$\frac{\partial^2}{\partial z^2}$
$\frac{\partial^2}{\partial x \partial z}$

Figure 10 The calculation of spatial derivatives in Graphic Processing Unit. The calculating of the second order spatial derivatives along a single direction only need to read 13 grid points values, while the cross derivatives need to read 144 grid points values.

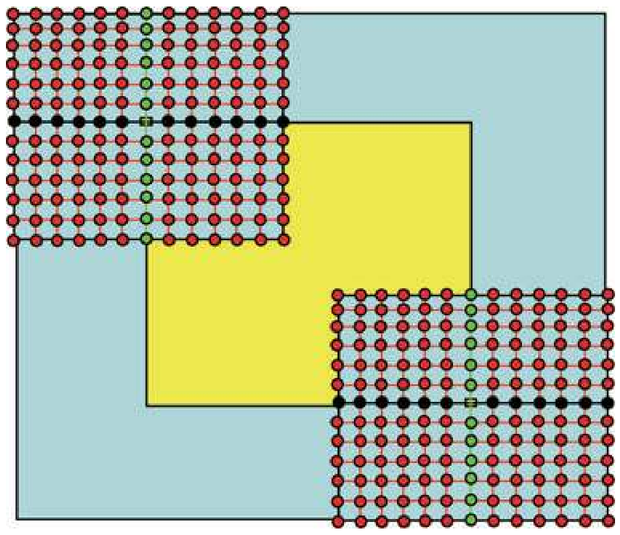


Figure 11 The use of shared memory to replace global memory. The yellow square is the size of per block and it could be 16 × 16; the blue square is the size of shared memory to be used for the block and the size should be 28 × 28. Each thread of the block only needs to read 4 grid points from the global memory so the read redundancy is reduced to a large extent.

the computation cost.

$$\left(\frac{\partial^2 p}{\partial x \partial z}\right)_{i,j} = \frac{1}{dxdz}\sum_{m=1}^{6}\sum_{n=1}^{6} a_m a_n \times \left(p_{i+n,j+m} - p_{i-n,j+m} - p_{i+n,j-m} + p_{i-n,j-m}\right). \quad (11)$$

The multilevel memory architecture makes it possible to reduce the memory read redundancy. During wavefield extrapolation, the temporary wavefield is stored in the global memory, which is big enough. There are 400–600 clock cycles of memory latency when accessing the global memory (NVIDIA 2009), while the shared memory space access is much faster than the global memory spaces because it is on-chip. As shown in Fig. 11, we use the shared memory instead of the global memory to compute the spatial derivatives. The

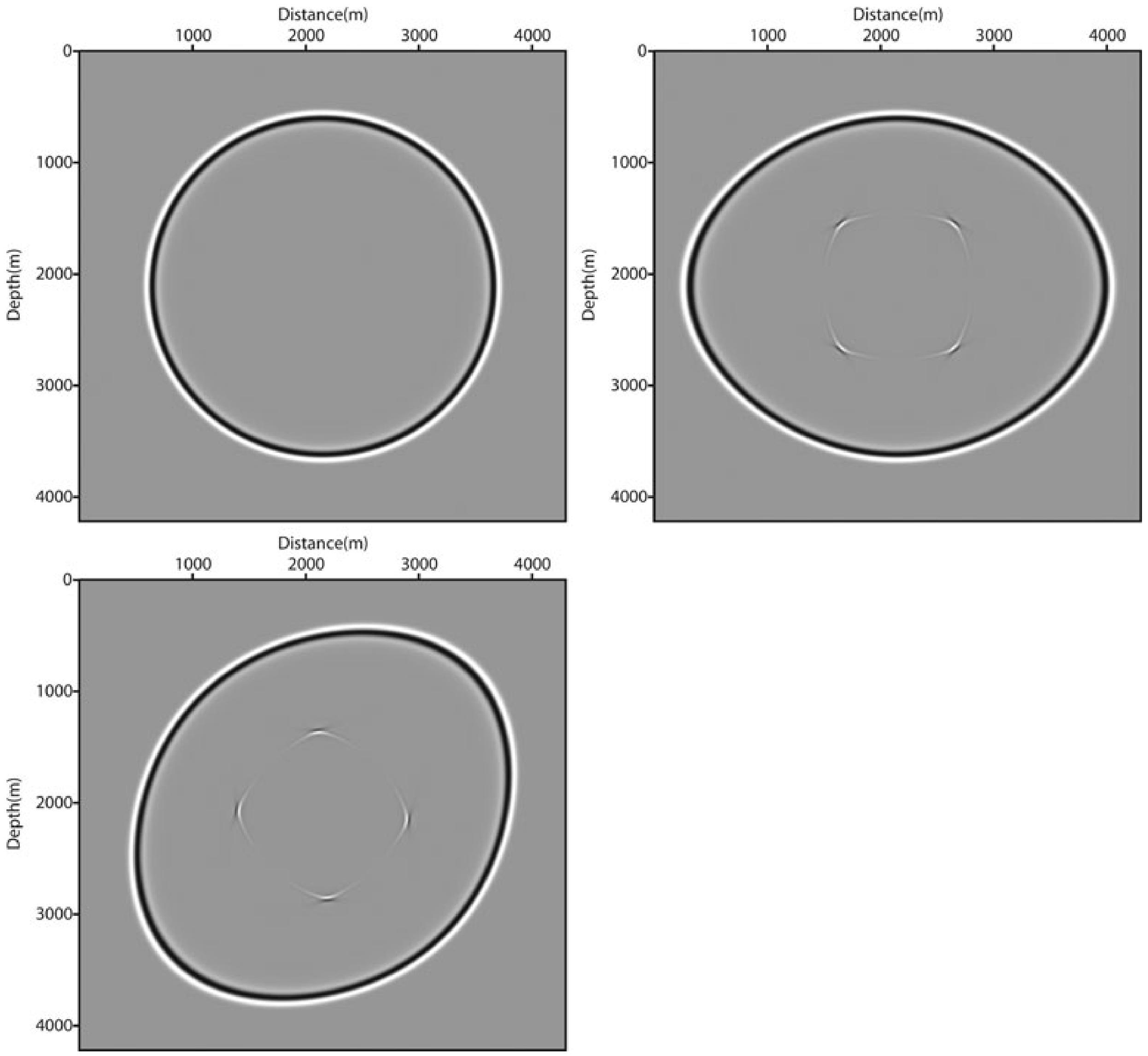


Figure 12 The snapshots of isotropic, VTI and TTI acoustic responses. The inner part of the VTI and TTI responses are shear wave, the shape is nearly a square while not a diamond because we do not set the shear velocity to zero.

yellow square is the size of one block and in our program it is 16×16; the blue square is the size of the shared memory to be used for the block and the size should be 28×28 because 6 layers should be padded on all the boundaries to calculate the derivatives. After using the shared memory, each thread of the block only needs to read 4 grid points from the global memory instead of 144 so the read redundancy is reduced to a large extent.

Figure 12 is the snapshots of isotropic, VTI and TTI acoustic responses. The inner part of the VTI and TTI responses are shear-wave modes; the shape is nearly a square but not a diamond because we use equation (10) to define the shear-wave velocity and this makes the algorithm stable in most cases (Fletcher *et al*. 2009).

Figure 13 is the TTI RTM result of the BP 2007 TTI synthetic model overlapped with the velocity model. The test is done using 88 Tesla 1060 GPU cards and the computation time is 6 hrs and 50 mins. We also tested the VTI and isotropic RTM and the computation time is 2 hrs and 13 mins for VTI and 57 mins for isotropic RTM. The computation ratio is consistent with other publications (for example, Jin, Fan and Ren 2010).

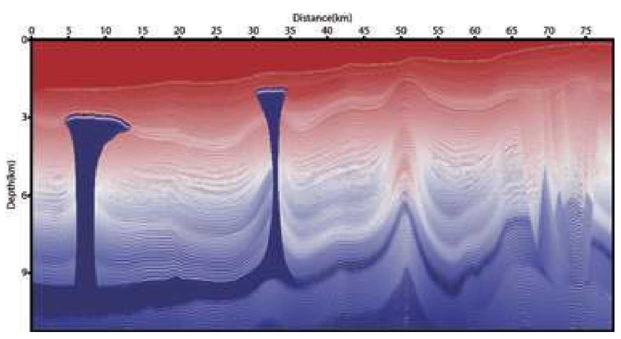


Figure 13 The TTI Reverse Time Migration result of BP2007 TTI model overlapped with the velocity model.

CONCLUSION AND DISCUSSION

The main bottlenecks of prestack Reverse Time Migration are the highly intensive computational cost and massive memory demand. In this article, we demonstrated that GPU, the pseudo random boundary condition and the explicit FD scheme provide a good combination for RTM implementation. This algorithm was used on several field data examples, it is very efficient and the results are better than ray-based Kirchhoff migration and the OWE method.

Using the pseudo random boundary method to implement the GPU-based prestack RTM reduces the memory demand but sacrifices some computation cost. Meanwhile, the communication between CPU and GPU is reduced, which further reduces the computation cost. Tests on synthetic data examples illustrate that the migration results using the pseudo random boundary condition are very close to those from the traditional method. The pseudo random boundary condition method works fine for stacked imaging but it might cause some problems for angle gathers if the coverage is not dense enough. We will do some tests in future research.

For rugged topography RTM, the filter approach, which is similar to the OWE method, is used at the free boundary grid points to avoid detailed classification, thus avoiding the large number of logical judgments. As a result, the additional cost of using GPU to achieve this is negligible and the numerical experiments on the synthetic data example show the effectiveness of this treatment.

For RTM in anisotropic media, we only show 2D implementation and examples. 3D TTI RTM needs to calculate three cross-derivatives. If we use the high order spatial FD scheme to solve the equation, the memory read redundancy would be very high. For example, for the twelfth-order FD scheme, we need to read 432 grid points values to calculate the cross-derivatives. One approach is to use a hybrid scheme: use the pseudo spectral method for horizontal derivatives and use the FD scheme for vertical derivatives. This will be part of our future research.

ACKNOWLEDGEMENTS

This research was supported by the National Science and Technology of Major Projects (grant No. 2011ZX05008-006-50 and 2011ZX05003-003), the Project of National Natural Science Foundation of China (Grant No.40830424 and 41104078) and the 48th batch China Postdoctoral Science Foundation general programme (20100480450). We acknowledge SEG for providing the Foothill synthetic data set and BP for providing the 2004 BP 2D isotropic and 2007 TTI synthetic data set. We thank Yu Zhang and three anonymous reviewers for reviewing this manuscript and giving constructive suggestions.

REFERENCES

Alkhalifah T. 2000. An acoustic wave equation for anisotropic media. *Geophysics* **65**, 1239–1250.

Araya-Polo M., Rubio F., de la Cruz R., Hanzich M., María Cela J. and Paolo Scarpazza D. 2009. 3D seismic imaging through reverse-time migration on homogeneous and heterogeneous multi-core processors. *Journal of Scientific Programming* **17**, 186–198.

Baysal E., Kosloff D.D. and Sherwood J.W.C. 1983. Reverse time migration. *Geophysics* **48**, 1514–1524.

Beasley C. and Lynn W. 1992. The zero-velocity layer: Migration from irregular surfaces. *Geophysics* **57**, no.11, 1435–1443.

Berryhill J.R. 1979. Wave equation datuming. *Geophysics* **44**, no.8, 1329–1344.

Berryhill J.R. 1984. Wave equation datuming before stack (short note). *Geophysics* **49**, no.11, 2064–2067.

Bevc D. 1997. Flooding the topography: Wave-equation datuming of land data with rugged acquisition topography. *Geophysics* **62**, no.5, 1558–1569.

Billette F. and Brandsberg-Dhal S. 2005. The 2004 BP velocity benchmark. 67th Annual Conference and Exhibition, EAGE, Expanded Abstracts, B035.

Chang W.F. and McMechan G.A. 1994. 3-D elastic prestack, reverse-time depth migration. *Geophysics* **59**, 597–609.

Chattopadhyay S. and McMechan G.A. 2008. Imaging conditions for prestack reverse-time migration. *Geophysics* **73**(3), S81–S89.

Clapp R.G. 2009. Reverse time migration with random boundaries. 79th Annual International Meeting, SEG Expanded Abstracts, 28, 2809–2813.

Du X., Bancroft J.C. and Lines L.R. 2007. Anisotropic reverse-time migration for tilted TI media. *Geophysical Prospecting* **55**, 853–869.

Duveneck E. and Bakker P.M. 2011. Stable P-wave modeling for reverse-time migration in tilted TI media. *Geophysics* **76**, S65–S75, doi:10.1190/1.3533964.

Fletcher R.P., Du X. and Fowler P.J. 2009. Reverse time migration in tilted transversely isotropic (TTI) media. *Geophysics* **74**, WCA179.

Foltinek D., Eaton D., Mahovsky J., Moghaddam P. 2009. Industrial-scale reverse time migration on GPU hardware. 79th Annual International Meeting, SEG Expanded Abstracts, 28, 2789–2793.

Fornberg B. 1998. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation* **51**(184), 699–706.

Gazdag J. and Sguazzero P. 1984. Migration of seismic data by phase-shift plus interpolation. *Geophysics* **49**, no.2, 124–131.

Gray S.H. and Marfurt K.J. 1995. Migration from topography: Improving the near-surface image. *Canadian Journal of Exploration Geophysics* **31**, 18–24.

Hemon C. 1978. Equations d'onde et modeles. *Geophysical Prospecting* **26**, 790–821.

Jin S.W., Jiang F. and Ren Y.Q. 2010. Comparison of isotropic, VTI and TTI reverse time migration: An experiment on BP anisotropic benchmark dataset. 80th Annual International Meeting, SEG, Expanded Abstracts, 3198–3203.

Li B., Liu G.F. and Liu H. 2009. A method of using GPU to accelerate seismic pre-stack time migration. *The Chinese Journal of Geophysics* (in Chinese) **52**(1), 245–252.

Liu H.W., Li B., Liu H., Tong X.L. and Liu Q. 2010. The algorithm of high order finite difference pre-stack reverse time migration and GPU implementation. *The Chinese Journal of Geophysics* (in Chinese) **53**(7), 1725–1733.

Liu H.W., Li B., Liu H., Tong X.L. and Liu Q. 2011. The problems of pre-stack reverse time migration and solutions with GPU implementation. 73rd EAGE Conference & Exhibition Expanded Abstracts, P268.

Liu G.F., Liu H., Li B., Liu Q., Tong X.L. 2009. Method of prestack time migration of seismic data of mountainous regions and its GPU implementation. *The Chinese Journal of Geophysics* (in Chinese) **52**(12), 3101–3108.

Loewenthal D. and Mufti I.R. 1983. Reverse time migration in spatial frequency domain. *Geophysics* **48**, 627–635.

McMechan G.A. 1983. Migration by extrapolation of time-dependent boundary values. *Geophysical Prospecting* **31**, 413–420.

McMechan G.A. and Chang W.F. 1990. 3D acoustic prestack reverse-time migration. *Geophysical Prospecting* **38**, 737–756.

Micikevicius P. 2009. 3D finite difference computation on GPUs using CUDA. *Proceedings of the 2nd Workshop on General Purpose Processing on Graphics Processing Units*, Washington, D.C., pp. 79–84.

Neklyudov D. and Borodin I. 2009. Imaging of offset VSP data acquired in complex areas with modified reverse-time migration. *Geophysical Prospecting* **57**, 379–391.

NVIDIA Corporation 2009. CUDA Programming Guide, version 2.3.1, P88.

Perrone M.P., Lu L., Liu L., Fedulova I., Semenikhin A. and Gorbik V. 2011. High performance RTM using massive domain partitioning. 73rd EAGE Conference & Exhibition, Expanded Abstracts, D034.

Reshef M. 1991. Depth migration from irregular surfaces with depth extrapolation methods (short note). *Geophysics* **56**, no.1, 119–122.

Soubaras R. and Zhang Y. 2008. Two-step explicit marching method for reverse time migration. 78th Annual International Meeting, SEG Expanded Abstracts, 27, 2272–2275.

Souza P., Teixeira T., Panetta J., Cunha C., Romanelli A., Roxo F., Pedrosa I., Sinedino S., Monnerat L., Carneiro L. and Albrecht C. 2011. Accelerating Seismic Imaging on GPUs. 73rd EAGE Conference & Exhibition, Expanded Abstracts, D035.

Symes W.W. 2007. Reverse time migration with optimal checkpointing. *Geophysics* **72**(5), SM213–SM221.

Thomsen L. 1986. Weak elastic anisotropy. *Geophysics* **51**, 1954–1996.

Villarreal A. and Scales J.A. 1997. Distributed three dimensional finite-difference modeling of wave propagation in acoustic media. *Computers in Physics, American Institute of Physics* **11**, 388–399.

Wiggins J.W. 1984. Kirchhoff integral extrapolation and migration of nonplanar data. *Geophysics* **49**(8), 1239–1248.

Yilmaz O. and Lucas D. 1986. Prestack layer replacement. *Geophysics* **51**, no.7, 1355–1369.

Yuan S.Y., Wang S.X., He Y.X. and Tian N. 2011. Second-order Wave Equation Modeling in Time Domain with Surface Topography Using Embedded Boundary Method and PML. 73rd EAGE Conference & Exhibition, Expanded Abstracts, P356.

Zhang L., Rector III J.W. and Hoversten G.M. 2005. Finite-difference modelling of wave propagation in acoustic tilted TI media. *Geophysical Prospecting* **53**, 843–852, doi:10.1111/j.1365-2478.2005.00504.x.

Zhang Y. and Zhang G.Q. 2009. One-step extrapolation method for reverse time migration. *Geophysics* **74**, A29.

Zhang Y., Zhang H.Z. and Zhang G.Q. 2011. A stable TTI reverse time migration and its implementation. *Geophysics* **76**, WA3–WA11, doi:10.1190/1.3554411.

Zhou H., Zhang G. and Bloor R. 2006a. An anisotropic wave equation for VTI media. 68th Conference and Exhibition, EAGE, Expanded Abstracts, H033.