Doi: 10.1007/s11589-010-0736-4

Optimization of a precise integration method for seismic modeling based on graphic processing unit*

Jingyu Li [1] Genyang Tang [2] and Tianyue Hu [1,♣]

[1] *School of Earth and Space Sciences*, *Peking University*, *Beijing* 100871, *China*
[2] *Department of Earth Sciences*, *Cambridge University*, *Cambridge CB*2 3*EQ*, *United Kingdom*

Abstract General purpose graphic processing unit (GPU) calculation technology is gradually widely used in various fields. Its mode of single instruction, multiple threads is capable of seismic numerical simulation which has a huge quantity of data and calculation steps. In this study, we introduce a GPU-based parallel calculation method of a precise integration method (PIM) for seismic forward modeling. Compared with CPU single-core calculation, GPU parallel calculating perfectly keeps the features of PIM, which has small bandwidth, high accuracy and capability of modeling complex substructures, and GPU calculation brings high computational efficiency, which means that high-performing GPU parallel calculation can make seismic forward modeling closer to real seismic records.

Key words: **precise integration method; seismic modeling; general purpose GPU; graphic processing unit**
CLC number: P315.69 **Document code**: A

1 Introduction

As the exploration of petroleum resources is developing continuously, depth of subsurface target and requirement for precision of exploration method are both growing, which causes rapid increase of data volume. Meanwhile, continuous cost increases in the field of geophysics data processing. For the sake of better recognizing the propagation and imaging of seismic wave in complex subsurface structures, it is essential to develop new modeling method with higher precision in order to carry out the forward modeling and inverse for seismic wavefields. Higher demand on the capacity of calculation devices has attracted more attention.

For the past 20 years, the performance of CPU is continuously growing with Moore's law, while the parallel calculation algorithm for multi-CPU system has been widely applied. CPU manufacturers also developed multi-core CPUs. Intel achieved six-core CPU in 2009. However, the problem of CPUs, which have relatively fewer processing units, has never had a fundamental change in their hardware architecture. Although super computers manage thousands of CPUs, and PC-Clusters are also set up for parallel computing, the contradiction is not still solved between CPUs' mainly serialized feature in processing, and huge volume geophysics data with relatively simple and repetitious calculating steps.

On the other hand, great demand for emulation technique of complex graphic processing forces faster development of graphic processing unit (GPU). Although the frequency of a GPU's processing unit is less than that of a CPU's, a GPU can handle much more processing units, which makes GPU's performance much better than CPU's of similar costs. Cutting-edge GPU's capacity of floating point arithmetic has reached 10^{12} FLOPS, advancing a six-core GPU's by two orders of magnitude (Zhang et al, 2009).

In this paper, we provide a precise integration method (PIM) to explore the application of GPU-based parallel calculation for difference modeling method. We enhanced the PIM calculation program by GPU-friendly optimization to fully exploit its calculation efficiency, and to illuminate the significance of GPU parallel calculation in practical production of petroleum

* **Received** 1 June 2010; **accepted in revised form** 23 June 2009; **published** 10 August 2010.
♣ **Corresponding author. e-mail**: tianyue@pku.edu.cn

exploration.

2 Precise integration method

Precise integration method (Tang, 2007; Yang, 2009) is a wave equation solving method designed for complex subsurface structures. This method can handle numerical dispersion and meet strict stability criteria of finite difference method (FDM). The basic idea is to employ a difference approximation in spatial domain, converting the wave equation into a set of differential equations with respect of time and then solve this equation system using an integration of a matrix of exponential functions. Theoretically this solution is accurate in the time domain, which solves the problem of numerical dispersion from the FDM, hence increasing the total accuracy. The whole spatial domain can be divided into several sub-domains. We use a single-point sub-domain to reduce the data bandwidth, with which we compromise between accuracy and efficiency. This method, with less data and higher stability and capable in heterogeneous medium and complex boundary conditions, has better results for complex subsurface structures and structures with relief surface.

For a 2D isotropic medium in *x*-*z* domain, the acoustic equation can be written as follows

$$\frac{\partial^2 u}{\partial t^2} = v^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2}\right), \tag{1}$$

where u is pressure perturbation propagating in the medium, and v is the acoustic velocity. For simplicity, the source function is omitted. All sources in this study is explosion, therefore the problem is completely 2-D.

Discretizing the spatial coordinates from equation (1), we get two-order difference equation

$$\ddot{\boldsymbol{U}} + \boldsymbol{K}\boldsymbol{U} = \boldsymbol{F}, \tag{2}$$

where $\boldsymbol{U} = [u_{i,j} \;\; u_{i+1,j} \;\; u_{i,j-1} \;\; u_{i-1,j} \;\; u_{i,j+1}]^{\mathrm{T}}$,

$$\boldsymbol{K} = \begin{bmatrix} 2v_{i,j}^2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta z^2}\right) & -\frac{v_{i,j}^2}{\Delta x^2} & -\frac{v_{i,j}^2}{\Delta z^2} & -\frac{v_{i,j}^2}{\Delta x^2} & -\frac{v_{i,j}^2}{\Delta z^2} \\ -\frac{v_{i+1,j}^2}{\Delta x^2} & v_{i+1,j}^2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta z^2}\right) & 0 & 0 & 0 \\ -\frac{v_{i,j-1}^2}{\Delta z^2} & 0 & 2v_{i,j-1}^2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta z^2}\right) & 0 & 0 \\ -\frac{v_{i-1,j}^2}{\Delta x^2} & 0 & 0 & 2v_{i-1,j}^2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta z^2}\right) & 0 \\ -\frac{v_{i,j+1}^2}{\Delta z^2} & 0 & 0 & 0 & 2v_{i,j+1}^2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta z^2}\right) \end{bmatrix},$$

$$\boldsymbol{F} = \begin{bmatrix} 0 \\ \frac{v_{i+1,j}^2 u_{i+2,j}}{\Delta x^2} + \frac{v_{i+1,j}^2(u_{i+1,j-1}+u_{i+1,j+1})}{\Delta z^2} \\ \frac{v_{i,j-1}^2 u_{i,j-2}}{\Delta z^2} + \frac{v_{i,j-1}^2(u_{i-1,j-1}+u_{i+1,j-1})}{\Delta x^2} \\ \frac{v_{i-1,j}^2 u_{i-2,j}}{\Delta x^2} + \frac{v_{i-1,j}^2(u_{i-1,j-1}+u_{i-1,j+1})}{\Delta z^2} \\ \frac{v_{i,j+1}^2 u_{i,j+2}}{\Delta z^2} + \frac{v_{i,j+1}^2(u_{i-1,j+1}+u_{i+1,j+1})}{\Delta x^2} \end{bmatrix}$$

This is a five-point sub-domain format, and the grid configuration is shown in Figure 1.

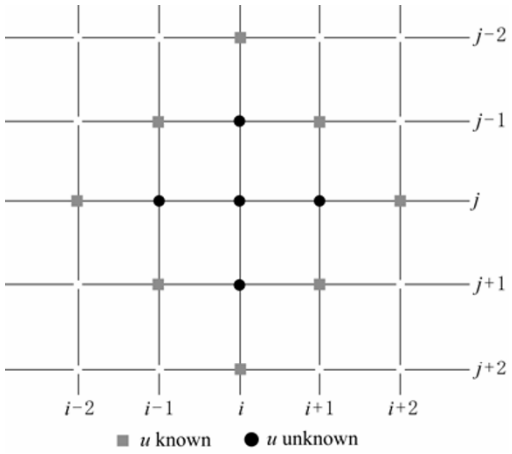

Figure 1 Configuration of grids for five-point sub-domain.

In time domain, considering the equations

$$\begin{cases} \boldsymbol{M}\ddot{\boldsymbol{U}} + \boldsymbol{C}\dot{\boldsymbol{U}} + \boldsymbol{K}\boldsymbol{U} = \boldsymbol{F} \\ \boldsymbol{U}(0) = \boldsymbol{U}_0 \\ \dot{\boldsymbol{U}}(0) = \dot{\boldsymbol{U}}_0 \end{cases}, \quad (3)$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}$ is damping matrix, $\boldsymbol{K}$ is stiffness matrix, and $\boldsymbol{F}$ is equivalent load, $\boldsymbol{U}_0$ and $\dot{\boldsymbol{U}}_0$ are both initial conditions of matrix $\boldsymbol{U}$. Introducing $\boldsymbol{P} = \boldsymbol{M}\dot{\boldsymbol{U}} + \boldsymbol{C}\boldsymbol{U}/2$, we obtain

$$\dot{\boldsymbol{V}} = \boldsymbol{H}\boldsymbol{V} + \boldsymbol{f}, \quad (4)$$

where

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{bmatrix}, \quad \boldsymbol{f} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{F} \end{bmatrix},$$

$$\boldsymbol{H} = \begin{pmatrix} -\boldsymbol{M}^{-1}\boldsymbol{C}/2 & \boldsymbol{M}^{-1} \\ \frac{1}{4}\boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{C} - \boldsymbol{K} & -\frac{\boldsymbol{C}\boldsymbol{M}^{-1}}{2} \end{pmatrix}. \quad (5)$$

Equation (4) has its analytic solution in time domain as

$$\boldsymbol{V}(t) = \mathrm{e}^{\boldsymbol{H}t}\boldsymbol{V}_0 + \int_0^t \mathrm{e}^{\boldsymbol{H}(t-\tau)}\boldsymbol{f}(\tau)\mathrm{d}\tau. \quad (6)$$

Considering constant time steps which is $t_n = n\Delta t$, introducing $\boldsymbol{T}(\Delta t) = \mathrm{e}^{\boldsymbol{H}\Delta t}$, and matrix $\boldsymbol{f}$ becomes $\boldsymbol{f}^n$ for time interval $[t_n, t_{n+1}]$, we can then get the explicit scheme from equation (6)

$$\boldsymbol{V}^{n+1} = \boldsymbol{T}\boldsymbol{V}^n + (\boldsymbol{T} - \boldsymbol{I})\boldsymbol{H}^{-1}\boldsymbol{f}^n. \quad (7)$$

For acoustic wave equation, we have $\boldsymbol{M}=\boldsymbol{I}$ and $\boldsymbol{C}=0$, equation (5) can be rewritten as

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{U} \\ \dot{\boldsymbol{U}} \end{bmatrix}, \quad \boldsymbol{H} = \begin{pmatrix} 0 & \boldsymbol{I} \\ -\boldsymbol{K} & 0 \end{pmatrix}, \quad \boldsymbol{f} = \begin{bmatrix} 0 \\ \boldsymbol{F} \end{bmatrix}.$$

After obtaining $\boldsymbol{V}$, we can get $\boldsymbol{U}$ from the first half of $\boldsymbol{V}$. Usually, the elements of $\boldsymbol{U}$ are considered as the final solution at the location (i, j), which is $\boldsymbol{U}=\{u_{i,j}\}$. The two-order difference equation (1) becomes

$$\frac{\partial^2 u_{i,j}}{\partial t^2} = v_{i,j}^2 \cdot \left(\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta z^2}\right). \quad (8)$$

Introducing weight coefficient $a^2 = \sum_{k=1}^{4} \alpha_k$, equation (8) becomes

$$\frac{\partial^2 u_{i,j}}{\partial t^2} + a^2 u_{i,j} = \sum_{k=1}^{4} \alpha_k u_{i,j}^{(k)}, \quad (9)$$

where

$$\begin{cases} \alpha_1 = \dfrac{v_{i,j}^{\ 2}}{\Delta x^2} \\ \alpha_2 = \dfrac{v_{i,j}^{\ 2}}{\Delta x^2} \\ \alpha_3 = \dfrac{v_{i,j}^{\ 2}}{\Delta z^2} \\ \alpha_4 = \dfrac{v_{i,j}^{\ 2}}{\Delta z^2} \end{cases}, \quad \begin{cases} u_{i,j}^{(1)} = u_{i-1,j} \\ u_{i,j}^{(2)} = u_{i+1,j} \\ u_{i,j}^{(3)} = u_{i,j-1} \\ u_{i,j}^{(4)} = u_{i,j+1} \end{cases}$$

Taking $u_{i,j}^{(k)n}$ as value $u_{i,j}^{(k)}$ at time n, we get the iterative solution

$$u_{i,j}^{n+1} = 2\cos(a\Delta t)(u_{i,j}^n - \frac{b_n}{a^2}) - u_{i,j}^{n-1} + \frac{2b_n}{a^2} - \frac{4d_n\left[1-\cos(a\Delta t)\right]}{a^4} + \frac{2d_n(\Delta t)^2}{a^2}$$

where

$$b_n = \sum_{k=1}^{4} \alpha_k u_{i,j}^{(k)n}, \quad d_n = \frac{1}{2}\sum_{k=1}^{4} \alpha_k \ddot{u}_{i,j}^{(k)n}, \quad e_n = \sum_{k=1}^{4} \alpha_k \dot{u}_{i,j}^{(k)n}$$

The initial conditions are expressed as

$$\begin{cases} u_{i,j}^0 = 0 \\ u_{i,j}^1 = \cos(a\Delta t)(u_{i,j}^0 - \dfrac{b_0}{a^2}) + \dfrac{b_0}{a^2} - \dfrac{2d_0\left[1-\cos(a\Delta t)\right]}{a^4} + d_0 + \dfrac{\partial u_{i,j}}{\partial t}\Big|_{t=0} \cdot \Delta t \end{cases}$$

The value at current time is expressed with those of the previous two time steps in this iteration solution. To reduce dispersion, under single-point sub-domain, the stability condition for two- and four-order difference equation is

$$\frac{v\Delta t}{\Delta x} \le \frac{\pi}{2\sqrt{2}} \approx 1.1109$$

Comparing with the stability condition of FDM, $v\Delta t/\Delta x \le 1$, PIM has advantage in stability. We use two-order difference equations for following numerical tests.

For source implementation, we just add source function in the iteration solutions, with the same calculation format with non-source areas. And for absorbing boundaries, we acquire a perfectly matched layer absorbing boundary condition for the second-order seismic wave equation (Komatitsch and Tromp, 2003).

3 GPU-based parallel algorithm

Under the compute unified device architecture (CUDA), GPU-based parallel calculating program can combine well with high level programming languages such as C and FORTRAN. There are a number of GPU -based high performing parallel algorithm in the field of scientific calculating.

Unlike the ideas of dividing model into several partitions for PC-Cluster, CUDA is designed with a two-stage thread managing system. The whole model is put into a grid, which contains numerous blocks, with no more than 512 threads in each block. During the procedure of processing, the program send all threads of a block into a stream multiprocessor (SM). The SM then assign each thread to a stream processor (SP). Every SM can have up to eight active blocks processing at the same time. A GPU contains dozens of SMs, and a SM has eight SPs. All the assignment work of blocks and threads are automatically completed by GPU (NVIDIA, 2008). The advantage of this managing system is that program designers do not have to care about the actual number of SMs and SPs or make specialized programs for each device, just like what they do for PC-Clusters. The same codes can run on every CUDA-supported GPU. In the other words, the program has better portability.

The best suitable strategy for GPU-based program designing is to divide the model into very small pieces, matching along with a large number of threads. The calculation steps for each thread should be kept simple. From numerical test we can see that the strategy is very efficient especially for models with larger scale.

Figure 2a shows the kernel calculating flow of CPU-based PIM program. Based on it, we applied several steps to transit the method on GPU. All data for calculating is stored in global memory of GPU, which can be accessed by all threads from both CPU and GPU, for reason that there's no data interchange between different threads. We assign every meshed units of the model to a thread. So the total number of threads equals to the number of meshed units, including absorbing boundaries. The number of threads in each block is limited to 512, while the number of threads in each dimension of grid is only limited to 65 535, so we set the block size to 16×16, and the number of blocks in the grid is decided by the actual scale of the model. Most often used constants such as time-unrelated item of source function, factors of absorbing boundaries and weighting coefficient, are stored in the constant memory of GPU, which has cache for faster access.

Figure 2b show the kernel calculating flow of GPU-based PIM parallel program. All the GPU tasks are located in the loop of time step (with gray background in Figure 2b). There's no data interchange between devices during that repetitive procedure, which increases the efficiency. For the aspect of instruction optimization, the program adopts asynchronous technique. While GPU is calculating, CPU write collected data from last shot into the resulting SEG-Y file. The synchronous technique also increases the efficiency.

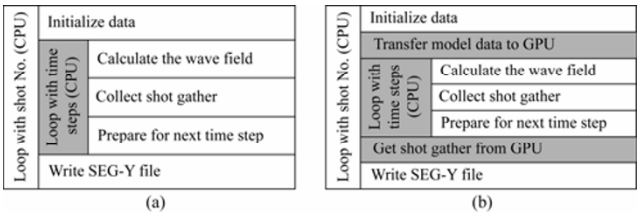

Figure 2 PIM program calculation flow for CPU (a) and GPU (b).

4 Efficiency analysis of GPU-based program

We test the relation between efficiency of the program and volume of model. All following numerical test results only indicate kernel calculation time, that is, they do not include the time of initializing. Environment Test includes Intel Core i7 CPU (clock frequency: 3.2 GHz), 4 GB RAM and a NVIDIA GTX 260 GPU (24 SMs, 192 SPs, clock frequency of SP: 1242 MHz, memory on GPU: 896 MB).

Table 1 PIM program calculation time of different volume of model

No.	Model volume	GPU blocks	CPU time/ms	GPU time/ms	Acceleration ratio
1	16×16	1	31.7	1.69×10^{2}	0.13
2	64×96	24	1.56×10^{3}	1.92×10^{2}	8.12
3	192×256	192	1.28×10^{4}	4.30×10^{2}	29.8
4	512×512	1024	5.48×10^{4}	1.47×10^{3}	37.3
5	1024×1024	4096	2.25×10^{5}	5.35×10^{3}	42.1
6	2048×2048	16384	8.18×10^{5}	2.12×10^{4}	38.4
7	4096×4096	65536	3.54×10^{6}	1.02×10^{5}	34.7

Testing model is a constant velocity model with one shot in the center, which is the minimum phase Ricker wavelet, the number of time steps is 1 000, and the time interval of the calculation is 1 ms. Table 1 shows the scale of the model (equivalent to total threads) and calculation time of both CPU and GPU.

Test No.1 uses only one block of GPU. As 'collecting shot gather' is to transfer data between devices, GPU program spends longer time than CPU. Test No. 2 has the same number of blocks as the number of SMs in the test device. There's only one active blocks in each SM. The scale of test No.3 is eight times as large as that of test No.2, which means all blocks are active in the test device. Only models with larger scale than test No.3 can be performed with full load. GPU-based program has apparent advantage in tests Nos.4–7. The acceleration ratio (the ratio of CPU time to GPU time) is steady at approximately 40. The acceleration ratio changes with the volume of the models are shown in Figure 3. In tests Nos.6 and 7, the program reduce the size of temporary data because of relatively small memory on GPU, which causes the time of data transfer increase and slows down the GPU performance.

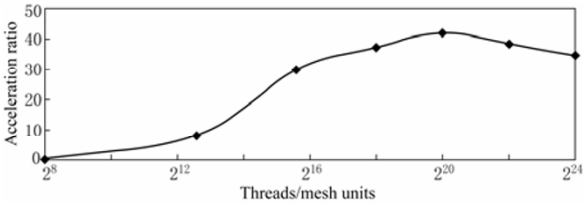

Figure 3 Acceleration ratio of different volume of model.

5 Case studies

We carry out two case studies of forward modeling to study the application of PIM to both CPU and GPU platforms. Case 1 images small-scale low-velocity caverns in a high-velocity layer. Case 2 contains large scale of complex structure which is of practical value.

Case 1 Caverns

The model shown in Figure 4 is a three-layer model with four small caverns in the high-velocity layer (Li and Zhao, 2006). The size of the largest cavern in the left is 40 m×40 m; the size of the two caverns in the center are both 30 m×30 m; and the smallest one in the right of the model has the size of 20 m×20 m. We calculate 60 shots altogether to acquire the seismic data with a shot interval of 50 m. Receivers are set to the right side of each shot in an off-end spread. The number of receivers is 100, with a receiver interval of 10 m. Offset ranges from 100 m to 1 090 m. With the size of calculation mesh unit set to be 5 m×5 m, we adopt a 15 Hz minimum phase Ricker wavelet as the source signature. The same procedure is calculated by both PIM programs on CPU and GPU.

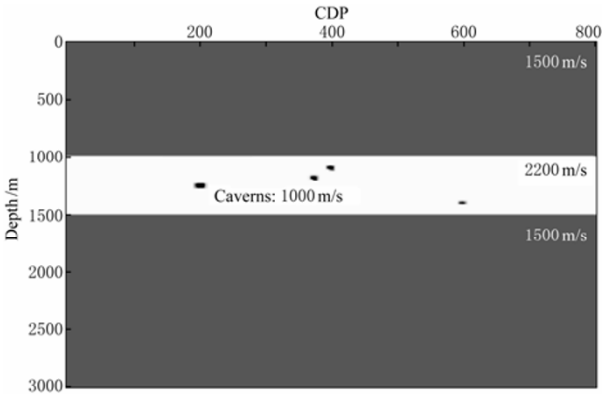

Figure 4 Caverns model.

The stacked sections from the results of both programs are shown in Figure 5. Even if they are calculated on different platforms, the results show no difference between the data from GPU and CPU platforms,

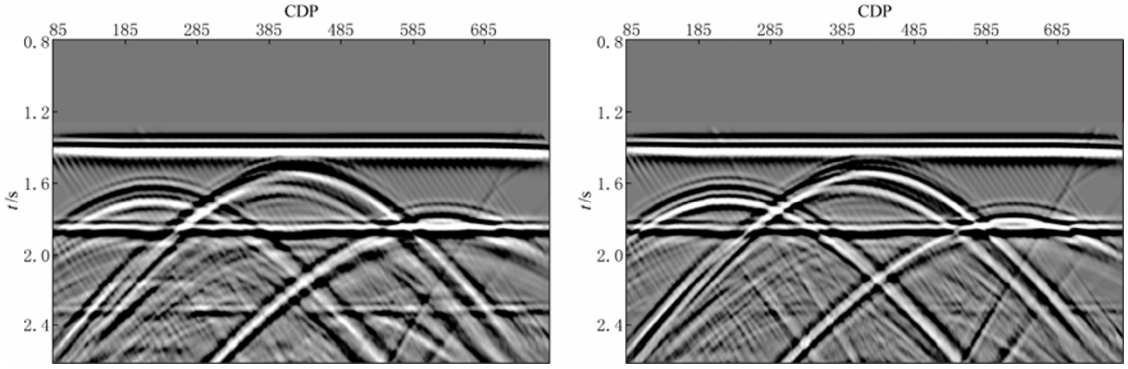

Figure 5 Stacked section of fracture model from forward modeling result of CPU (left) and GPU (right).

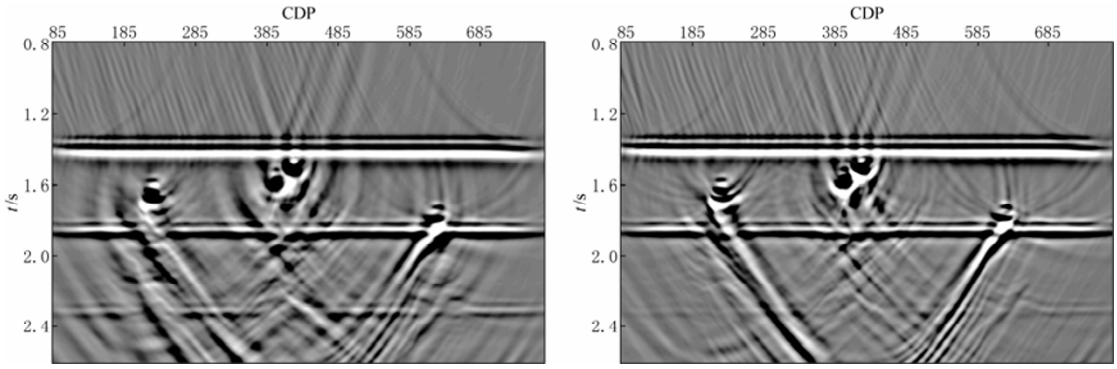

Figure 6 Kirchhoff time migration section of fracture model from forward modeling result of CPU (left) and GPU (right).

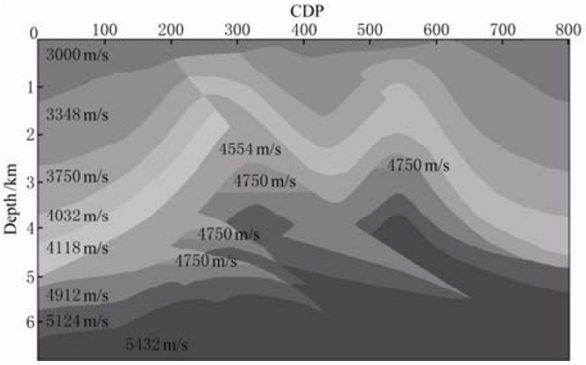

Figure 7 Kuche geological model.

which proves that GPU platform safely increases the time efficiency. Figure 6 is the post-stack Kirchhoff time migration section (from velocities of the original model). We can see that PIM clearly indicates three-layer subsurface structure and accurately locates the four caverns.

Case 2 Kuche geological model

Figure 7 shows Kuche velocity geological model, with a width of 16 km and a depth of 6.8 km. We calculate 75 shots with a shot interval of 160 m, with the same number of receivers on either side of a split spread. The maximum offset on either side are 2 km. A 15 Hz minimum phase Ricker wavelet is employed as the source, and the first shot goes at 2 km to the right of the top-left corner of the model. (CDP 400 in Figure 7).

We test different configuration of grids for different devices. For CPU program, we have 200 receivers with a receiver interval of 20 m. The model is meshed into 20 m× 10 m units. For GPU program, we have 400 receivers with a receiver interval of 10 m. The model is meshed into 5 m×5 m, which is of higher precision than that for CPU program.

Due to the difference of the configuration of grids, the computational cost of GPU program is eight times as large as CPU program, but GPU has an acceleration ratio of approximately 40, which makes GPU still faster than CPU. Figure 8 shows the stack sections from both CPU and GPU programs with different configuration of grids. The post-stack time migration sections and post-stack depth migration sections are shown in Figures 9 and 10.

As the calculation grids are set smaller in size for GPU, there is some slight dispersion which is not apparent in the result. GPU program provides clearer and more detailed information. Special subsurface structures in the deeper zone can be positioned more accurately. Therefore we can rely on the efficiency of GPU program to increase the precision by setting smaller grid size for the model.

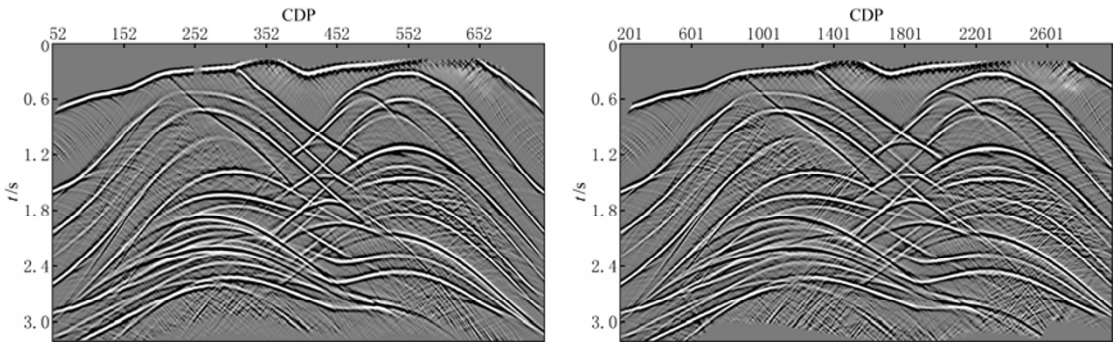

Figure 8 Stacked section of Kuche geological model from forward modeling result of CPU (left) and GPU (right).

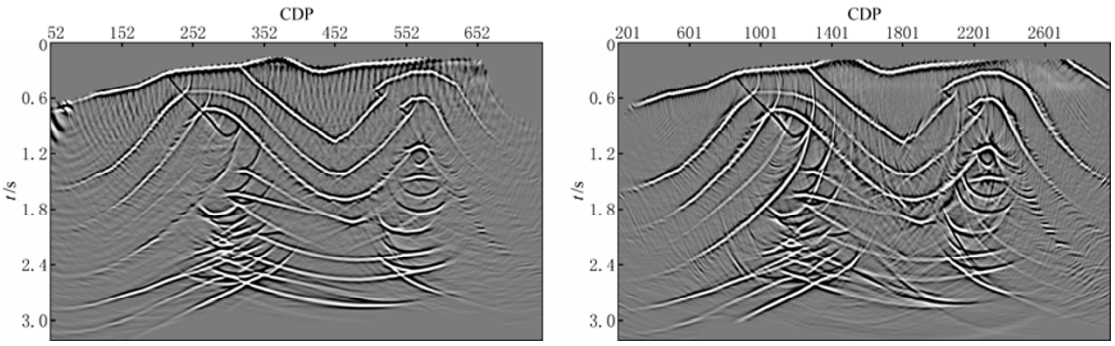

Figure 9 Kirchhoff time migration section of Kuche geological model from forward modeling result of CPU (left) and GPU (right).

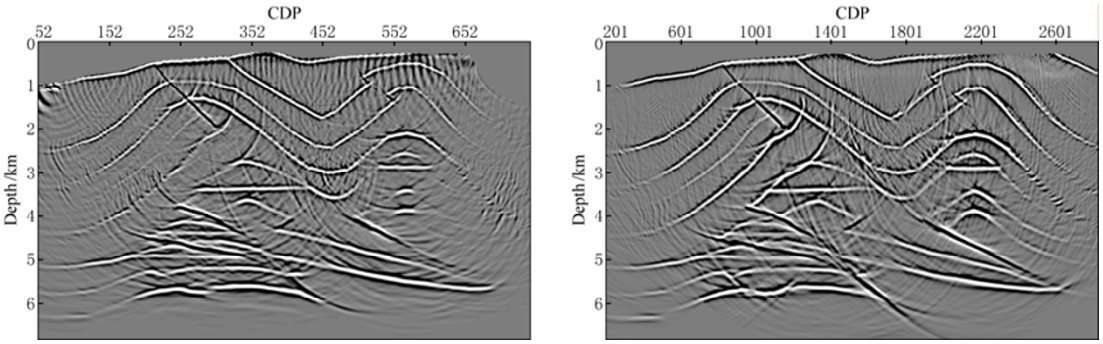

Figure 10 Post-stack depth migration section of Kuche geological model from forward modeling result of CPU (left) and GPU (right).

6 Discussion and conclusions

This paper introduces the GPU parallel calculation optimization of a precise integration method for seismic modeling. According to the numerical tests of simple model and geological model, GPU parallel calculation maintains the theoretical advantage of PIM, which has small data bandwidth, higher precision and better stability. Meanwhile, with proper program design and optimization, GPU parallel calculation will have a great performance on time efficiency, of which the acceleration ratio can as high as around 40. That will greatly reduce the cost of time and device in practical production.

This study just shows the advantage of GPU parallel calculation with a method of seismic modeling. A future improvement of GPU includes higher performance of SP clock frequency, higher precise processing, larger memory support, multi-GPU parallel calculation, etc. More GPU-oriented parallel calculating algorithm will be developed under the revolutionary multi-staged thread managing system. Both development of device and algorithm will be a strong force to the development of petroleum exploration calculating technology.

Acknowledgements This study was supported by the National Natural Science Foundation of China (Nos. 40974066 and 40821062) and by National Basic Research Program of China (No. 2007CB209602).

References

Komatitsch D and Tromp J (2003). A perfectly matched layer absorbing boundary condition for the second-order seismic wave equation. *Geophys J Inter* **154**(1): 146–153.

Li J and Zhao Q (2006). *Collection of Pictures of Physical Models for Petroleum Exploration Seismology.* Petroleum industry press, Beijing, 99 (in Chinese).

NVIDIA (2008). *CUDA Technical Training Volume* I: *Introduction to CUDA Programming*. http://developer.nvidia.com/object/gpucomputing.html, 2010-05-04.

Tang G, Hu T and Yang J (2007). Applications of a precise integration method in forward seismic modeling. *The 77th SEG Annual International Meeting*, *Expanded Abstract*, 2 130–2 134.

Yang J, Liu T, Tang G and Hu T (2009). Modeling seismic wave propagation within complex structures. *Appl Geophys* **6**(1): 30–41.

Zhang S, Chu Y, Zhao K and Zhang Y (2009). *High Performance GPU Calculating*: *CUDA*. China Water Power Press, Beijing, 6 (in Chinese)