GPU accelerated 2-D staggered-grid finite difference seismic modelling

Zhangang Wang[1], Suping Peng[1], Tao Liu[2]
[1]State Key Laboratory of Coal Resources and Mine Safety, China University of Mining and Technology, Beijing, China
Email: millwzg@163.com

[2] School of Earth and Space Sciences, Peking University, Beijing, China
Email: liuluot@126.com

***Abstract*—The staggered-grid finite difference (FD) method demands significantly computational capability and is inefficient for seismic wave modelling in 2-D viscoelastic media on a single PC. To improve computation speedup, a graphic processing units (GPUs) accelerated method was proposed, for modern GPUs have now become ubiquitous in desktop computers and offer an excellent cost-to-performance-ratio parallelism. The geophysical model is decomposed into subdomains for PML absorbing conditions. The vertex and fragment processing are fully used to solve FD schemes in parallel and the latest updated frames are swapped in Framebuffer Object (FBO) attachments as inputs for the next simulation step. The seismic simulation program running on modern PCs provides significant speedup over a CPU implementation, which makes it possible to simulate realtime complex seismic propagation in high resolution of 2048*2048 gridsizes on low-cost PCs.**

***Index Terms*—seismic, wave propagation, finite difference, viscoelastic media, parallel algorithms, GPU**

I. Introduction

The staggered-grid finite difference (FD) method has long been a popular computational method for the simulation of seismic waves propagating in geophysical exploration. FD can solve geophysical models to identify geological structures with arbitrary geometries comprised of acoustic, elastic or even viscoelastic materials. The main drawback of the FD method is that simulations with large model spaces or long nonsinusoidal waveforms can require a tremendous amount of floating point calculations and run times. In recent years the spatial parallelism on high-performance PC clusters makes it a promising method for seismic numerical computing in realistic (complex) media [1-3]. These implementations are achieved through an explicit FD method by the rational domain decomposition and considerable communication between multiple CPUs. Each processor solves the problem within its small subdomain and communicates with neighboring processors to update wavefield information at each time step. However, the efficiency of parallel implementations is limited by the computational load balance and the communication between processors.

Manuscript received January 7, 2010; revised March 20, 2010; accepted July 1, 2010.

The most challenging problem is doubtlessly the computational ability of the single node in the parallel environment, which prompts a need to investigate methods of increasing computational speed for these simulations.

Modern GPUs have now become ubiquitous in desktop computers and offer an excellent cost-to-performance-ratio, exceeding the performance of general purpose CPU by many times, due to their powerful and fully programmable parallel processing architectures. In the past few years, programmability of GPU and increased floating point precision has allowed GPU to perform general purpose computations. Applications include numeric processing [4], pattern recognition [5], computer vision and image processing [6].

In this article, we have proposed a new implementation under the current GPU architecture to improve the real-time FD simulation of wave propagation in viscoelastic media. The rest of the paper is organized as follows. First, an overview of using GPU for general purpose computation is given. Next, we provide a brief description of the viscoelastic staggered-grid FD method with PML boundary conditions. Then, we present our method for achieving real-time simulation of seismic wave propagation on GPU in detail, including domain decomposition and processing techniques. Finally the experimental results and related discussion are given in Section 5 and 6.

II. General Purpose Computation on GPUs

For a few years now, GPUs have been used for general purpose computation [7, 8]. In terms of their fully programmable parallel processing architectures and raw floating point computational capability, they are many times more powerful than general purpose CPUs and have much higher memory bandwidth.

GPUs are optimized for kernel-oriented data-parallel implementation [9], or in other words, general purpose computation on GPUs assumes all work is done in parallel without any data interdependence. Explicit data parallelism allows GPUs to largely exclude control logic and to execute similar operations on large vectors or streams of data in parallel, and data streaming with high-speed memory interfaces eliminates the need for large on-die caches.

This paradigm of data-parallel computing also implies that the internal computation is split up among the available fragment processors and the user-programmable shaders cannot control the order in which fragments are processed, and we have to utilize the "address", the coordinates (pixel coordinates) in the target texture where an individual dataflow will end up. Our implementation of the seismic wave propagation that uses the parallel processing capabilities for general tasks undergo the typical phases as described in Section 4. Fig.1 contains an abstract view of the hardware rendering pipeline and data streaming.

Programmable parallel processors—GPUs have two types of programmable processors: vertex and fragment processors [7, 9].

Vertex processors handle streams of vertices (composed of positions, colors, normal vectors and other attributes), which are elements that compose 3D geometric models. Vertex processors apply vertex shader programs to transform each vertex based on its position relative to the viewpoint.

The fragment processors apply fragment shader programs to each output pixel to determine its final color. Vertex and fragment processors are fully programmable and can execute an instruction simultaneously on four different values. That's because graphics primitives are either positions (x, y, z, w) or colors (red, green, blue, alpha).

Shader programs can be used to iterate over the elements of a stream (stored in a texture), processing them automatically with the instructions in the parallel computing body. The amount of parallelism in this computation depends on the number of processors on the GPU.

The rasterization stage executes the fundamental step of transforming the geometry into discrete pixel-based ones by generating pixel-like entities called fragments for screen pixel locations covered by geometry. The rasterizer is not user-programmable, but the rasterizer can be thought of as an address interpolator and a data amplifier because it generates many pixels from a few vertices.

Render-to-texture functionality is essential for scientific computing. To achieve feedback on the GPU, render-to-texture must be used to write (render) the results of a fragment program to a texture memory for further processing in a subsequent rendering pass. The OpenGL extension FrameBuffer Object (FBO) implements direct feedback of GPU output to input without going back to the host processor.

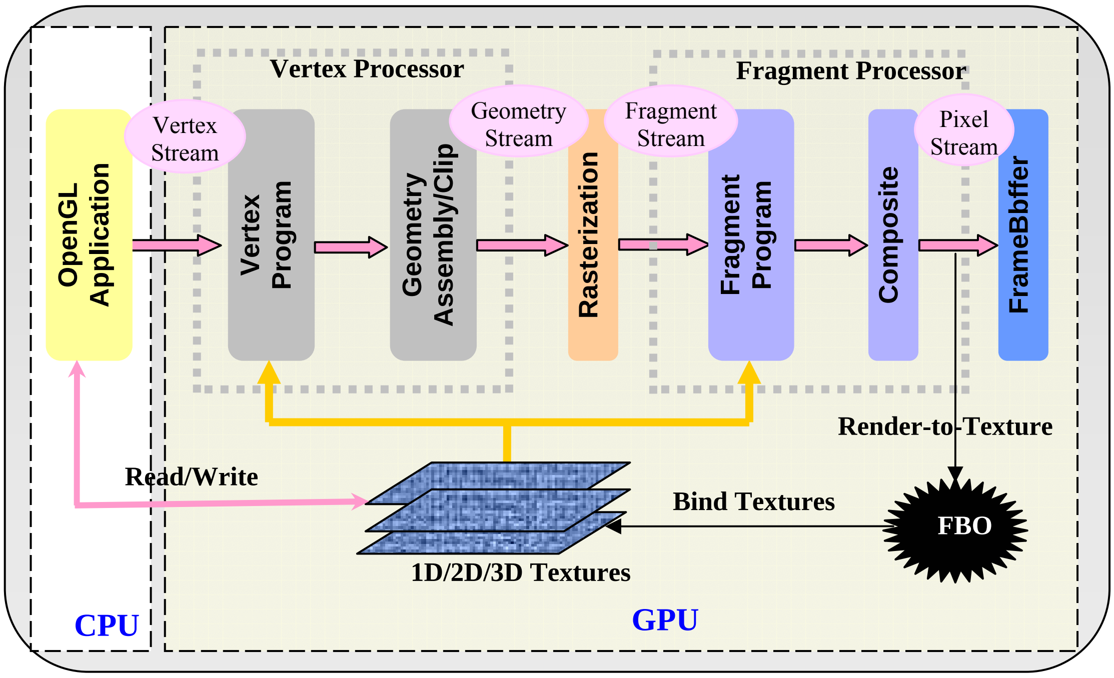


Fig.1. The hardware rendering pipeline implemented in GPUs can be mapped to the stream model. The steam formulation of non-graphics applications expresses all data as streams and computational stages. The first User-programmable part on the pipeline processes vertex primitives, the second part operates on fragment (pixel) data. FrameBuffer Object (FBO) allows reusing the framebuffer content as a texture for off-screen feedback in render-to-texture pass.

III. Theory

Explicit finite difference methods have been widely used to simulate seismic wave propagation, because of their ability to accurately model seismic waves in complex media. Early researches on FDM for elastic wave modeling in complex media gradually formulated finite difference schemes based on a system of high-order coupled elastic equations where the variables were stresses and velocities [10, 11, 12], rather than displacements [13, 14] and used a staggered-grid method [15], for staggered-grid operators are more accurate than standard grid to perform first derivatives for high frequencies close to the Nyquist limit [16]. In such studies, however, the earth's anelastic behaviours such as fluid-saturation and viscosity have been ignored and synthetic seismograms fail to model attenuation and dispersion of seismic waves. Day and Minster [17] made the first attempt to incorporate anelasticity into a 2-D time-domain modeling methods by applying a Pade approximant method. An efficient approach [18] was proposed based on the rheological model called "generalized standard linear solid" (GSLS), which was shown to explain experimental observations of wave propagation through earth materials [19]. Then a staggered grid finite difference technique [20] was proposed based on the GSLS to model the propagation of seismic waves in 2D/3D viscoelastic media. A staggered-grid FD method bases on Biot's equations [21] was given to simulate wave propagation in poroacoustic media [22].

The velocity-stress formulation of differential equations used in this article can be found in [20] and [23].

A. Staggered-grid Finite difference formulation

The first-order velocity-stress equations of viscoelastic wave propagation are given by

$$\begin{cases} \dfrac{\partial \sigma_{xy}}{\partial t} = \mu \dfrac{\tau_\varepsilon^s}{\tau_\sigma}(\dfrac{\partial v_x}{\partial y} + \dfrac{\partial v_y}{\partial x}) + \gamma_{xy} \\ \dfrac{\partial \sigma_{xx}}{\partial t} = \pi \dfrac{\tau_\varepsilon^p}{\tau_\sigma}(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}) - 2\mu \dfrac{\tau_\varepsilon^s}{\tau_\sigma} \cdot \dfrac{\partial v_y}{\partial y} + \gamma_{xx} \\ \dfrac{\partial \sigma_{yy}}{\partial t} = \pi \dfrac{\tau_\varepsilon^p}{\tau_\sigma}(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}) - 2\mu \dfrac{\tau_\varepsilon^s}{\tau_\sigma} \cdot \dfrac{\partial v_x}{\partial x} + \gamma_{yy} \\ \dfrac{\partial \gamma_{xx}}{\partial t} = -\dfrac{1}{\tau_\sigma}[\gamma_{xx} + \pi(\dfrac{\tau_\varepsilon^p}{\tau_\sigma} - 1)(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}) - 2\mu(\dfrac{\tau_\varepsilon^s}{\tau_\sigma} - 1) \cdot \dfrac{\partial v_y}{\partial y}] \\ \dfrac{\partial \gamma_{yy}}{\partial t} = -\dfrac{1}{\tau_\sigma}[\gamma_{yy} + \pi(\dfrac{\tau_\varepsilon^p}{\tau_\sigma} - 1)(\dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y}) - 2\mu(\dfrac{\tau_\varepsilon^s}{\tau_\sigma} - 1) \cdot \dfrac{\partial v_x}{\partial x}] \\ \dfrac{\partial \gamma_{xy}}{\partial t} = -\dfrac{1}{\tau_\sigma}[\gamma_{xy} + \mu(\dfrac{\tau_\varepsilon^s}{\tau_\sigma} - 1)(\dfrac{\partial v_x}{\partial y} + \dfrac{\partial v_y}{\partial x})] \\ \dfrac{\partial v_x}{\partial t} = \dfrac{1}{\rho}[\dfrac{\partial \sigma_{xx}}{\partial x} + \dfrac{\partial \sigma_{xy}}{\partial y} + f_x] \\ \dfrac{\partial v_y}{\partial t} = \dfrac{1}{\rho}[\dfrac{\partial \sigma_{xy}}{\partial x} + \dfrac{\partial \sigma_{yy}}{\partial y} + f_y] \end{cases} \quad (1)$$

The moduli i *π, μ* can be expressed as

$$\pi = v_{p_0}^2 \rho R^2 \sqrt{\frac{1}{1 + \frac{i w_0 \tau_\sigma}{1 + i w_0 \tau_\sigma} \tau_\varepsilon^p}}, \quad \mu = v_{s_0}^2 \rho R^2 \sqrt{\frac{1}{1 + \frac{i w_0 \tau_\sigma}{1 + i w_0 \tau_\sigma} \tau_\varepsilon^s}} \quad (2)$$

Where v_{P0}, v_{S0} denote the phase velocity at the centre frequency of the source(w_0) for P- and S-waves, respectively. The symbol *R* denotes the real part of the complex variable. The constants of stress relaxation times for both P-and S-waves, can be calculated by quality factor *Q* and angular frequency *w*.

$$\tau_\sigma = \frac{1}{w}(\sqrt{\frac{1}{1+Q_p^2}} - \frac{1}{Q_p}) \quad (3)$$

$$\tau_\varepsilon^p = \frac{1}{w^2 \tau_\sigma}, \quad \tau_\varepsilon^s = \frac{1 + w\tau_\sigma Q_s}{wQ_s - w^2 \tau_\sigma} \quad (4)$$

A second-order centered difference scheme is applied to approximate the time derivatives, and a fourth-order staggered scheme with centered differences to approximate the spatial derivatives. From Eqs.1, for examples, $\sigma_{xy}^{n^+}$, $r_{xy}^{n^+}$ and v_x^{n+1} can be approximated as:

$$\sigma_{xy}^{n^+}(i^+, j^+) = \sigma_{xy}^{n^-}(i^+, j^+) + \frac{\tau \cdot \pi(i^+, j^+)\tau_\varepsilon^p(i^+, j^+)}{\tau_\sigma(i^+, j^+)}(\frac{\partial v_x^n(i^+, j)}{\partial y} + \frac{\partial v_y^n(i, j^+)}{\partial x}) + \frac{\tau}{2}(r_{xy}^{n-}(i,j) + r_{xy}^{n^+}(i,j)) \quad (5)$$

$$r_{xy}^{n^+}(i^+, j^+) = (1 + \frac{\tau}{2\tau_\sigma(i^+, j^+)})^{-1}((1 - \frac{\tau}{2\tau_\sigma(i^+, j^+)}) \cdot r_{xy}^{n-}(i^+, j^+) - \frac{\tau \cdot \mu(i^+, j^+)}{\tau_\sigma(i^+, j^+)}(\frac{\tau_\varepsilon^s(i^+, j^+)}{\tau_\sigma(i^+, j^+)} - 1) \cdot (\frac{\partial v_x^n(i^+, j)}{\partial y} + \frac{\partial v_y^n(i, j^+)}{\partial x})) \quad (6)$$

$$v_x^{n+1}(i^+, j) = v_x^n(i^+, j) + \frac{\tau}{\rho(i^+, j)}(\frac{\partial \sigma_{xy}^{n^+}(i^+, j^+)}{\partial y} + \frac{\partial \sigma_{xx}^{n^+}(i, j)}{\partial x} + f_x^{n^+}) \quad (7)$$

Where *i, j, k, n* are the indices for the three spatial directions and time, respectively. τ denotes the size of a timestep.

The discretion of the spatial differential operator, for example, is given:

$$\frac{\partial v(i^+, j)}{\partial x} = \frac{1}{l}(c1 \cdot (v(i+1, j) - v(i, j)) + c2 \cdot (v(i+2, j) - v(i-1, j))) \quad (8)$$

Where *l* denotes grid spacing and *c1, c2* denote the differential coefficients.

B. PML absorbing boundary condition

In order to simulate an unbounded medium, an absorbing boundary condition (ABC) must be implemented to truncate the computational domain in numerical algorithms. The paraxial approximation [24] was used to make the boundary transparent to outgoing waves. Since the highly effective perfectly matched layer (PML) method [25] for electromagnetic waves was proposed, the PML has been widely used for finite-difference and finite-element methods [26, 27]. The effectiveness of the PML for elastic waves in solids and the zero reflections from PML to the regular elastic medium were proved [28, 29].

In this work, we exploit the elastic wave formulation in the ABC region, not considering the viscosity. For the first-order velocity equations:

$$\frac{\partial v_x}{\partial t} = \frac{1}{\rho}(\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y}) \quad (9)$$

The PML equations in x direction are obtained:

$$\begin{cases} \dfrac{\partial v_x^1}{\partial t} + d(x) \cdot v_x^1 = \dfrac{1}{\rho} \cdot \dfrac{\partial \sigma_{xx}}{\partial x} \\ \dfrac{\partial v_x^2}{\partial t} = \dfrac{1}{\rho} \dfrac{\partial \sigma_{xy}}{\partial y} \end{cases} \quad (10)$$

Then velocity in x direction becomes $v_x = v_x^1 + v_x^2$. Similar equations can be obtained in the same way for other components in Eqs.1. The outgoing waves are absorbed by the PML via high attenuation.

IV. Implementation on GPUs

A. Overview

To compute the staggered-grid FD and PML equations on graphics hardware, we divide the geophysical model into regions represented as textures with the input and output parameters. Fig.2 shows the division of a 2D model into a collection of regions. Two sets of alternating textures binding with a FBO are used to represent the parameters for computing frame at time t and t+1 respectively. All the other input-only variables are stored similarly in 2D textures. To update FBO textures at each time step, we render quadrilaterals (regions) mapped with the textures in order, so that a fragment program running at each texel computes the FD equations. After each frame, FBO attachments are swapped and the latest updated textures are then used as inputs for the next simulation step (loops go on). The simulation is totally implemented using the current fragment and vertex processing stages of the GPU.

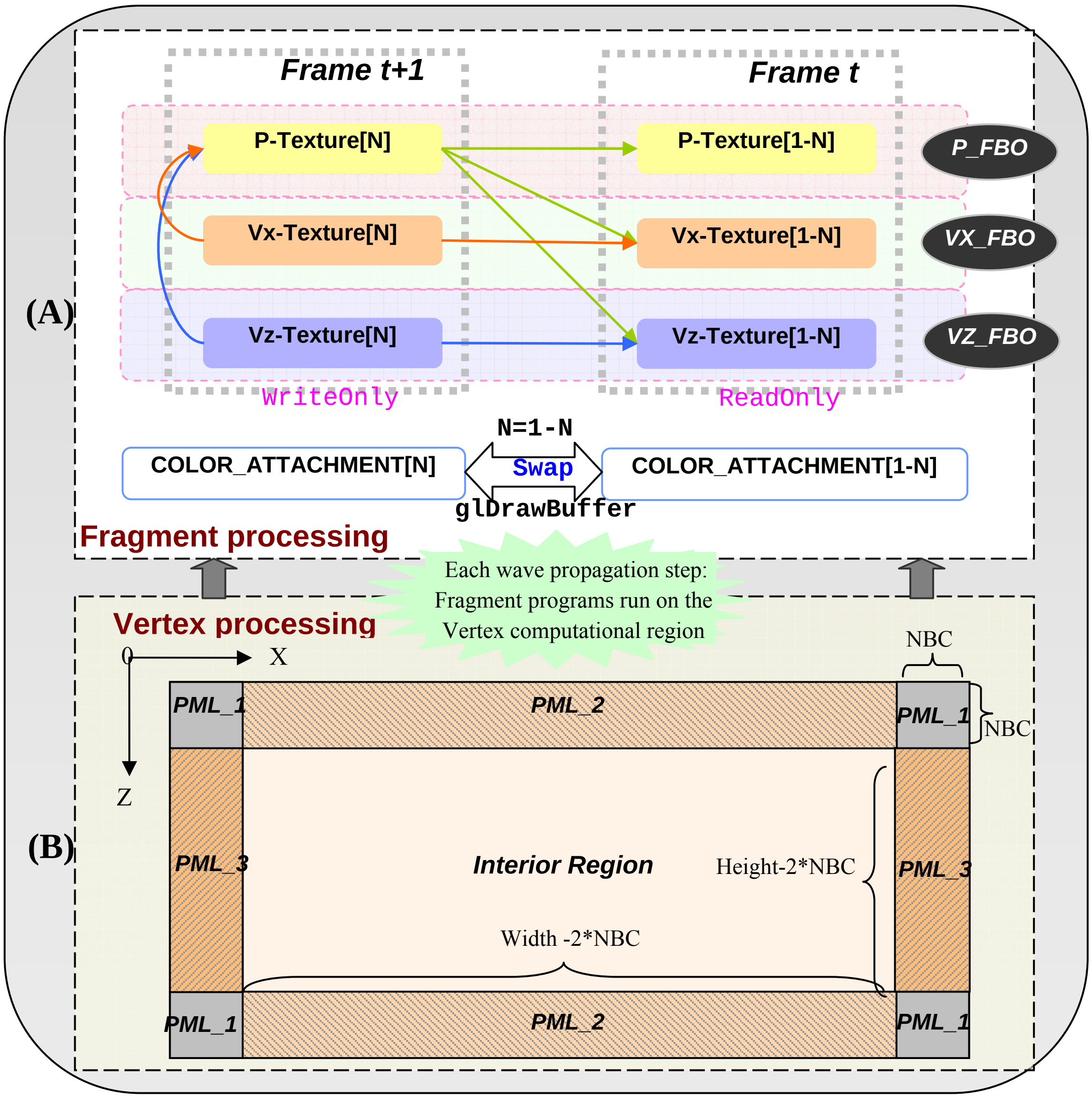


Fig.2. Implementation of staggered grid FD method with PML conditions on GPUs. Vertex processing switches computing regions and render quadrilaterals (regions) mapped to textures; fragment program run at each texel for computing frame at time t and t+1. FBO attachments are swapped and the latest updated textures are then used as inputs for the next simulation step (loops go on).

B. Domain decomposition

The computational domain is divided into an interior region and PML regions. The outgoing waves are absorbed by the PML via high attenuation. For 2D model, the size of the interior region is N1*N2, denoting the interior part of the whole textures, where N1 = width-2*NBC, N2 = Height-2*NBC. PML region covers 8 blocks and can be divides three types based on the different computing process: PML1, processing absorbing condition both in x and z direction, with the size NBC*NBC; PML2, processing absorbing condition only in z direction, locating in top and bottom of the whole region and the size is N1*NBC; PML3, processing absorbing condition only in x direction, locating in left and right of the whole region and the size is NBC*N2. The vertex processors will transform on computational domains and the rasterizer will determine all the pixels in the output buffer they cover and generate fragments.

C. Data representation

For 2D viscoelastic FD schemes of Eqs.1, computation on interior region at each frame is to get P_X, P_Z, P_{XZ}, V_X,

V_Z, R_X, R_Z and R_{XZ}; computation on PML regions needs two extra components for P_X, P_Z, P_{XZ}, V_X or V_Z on each of 4 absorbing boundaries. Considering the time t and t+1, there are actually (8+5*2*4)*2 = 96 arrays storing these components to run in the whole process.

In order to reduce to the number of textures and to switch computational domain quickly, 16 textures are used to store interior stress-velocity parameters, called *whole- texture* with the size width*height as shown in Fig.3, width and height denoting grid-point numbers in x and z direction respectively. It equals the size of the rendering viewport. The input seismic parameters Vp, Vs, Qp, Qs, ρ are also stored in *whole- texture*. 5*2 extra textures, called *x-direction-textur*e, are used to express all velocity and stress parameters in x absorbing direction; and the size of each texture is (2*NBC, height), NBC defining the gridpoints of the absorbing boundary. In our work, NBC is 10. Simultaneously, there are 5*2 extra arrays in z absorbing direction, called *z-direction-texture*, and the size is (width, 2*NBC).There are actually another 20 arrays at time t+1.

From above analysis, the memory requirements *MEMREq* for 2D viscoelastic modelling is obtained by,

$$MEMREg = (N^2 * C + N * NBC * D) * Fz / 1024^2 \quad (11)$$

where the total number C of the whole-texture is 21 and the total number D of x-and z- direction- texture is 40. Fz of float size equals 4 bytes.

Table1 shows 2D viscoelastic model which is only less than 1024^2 or 2048^2 gridsizes can be supported on current GPUs directly, since graphics video memory size is about 256-512M in general and the texture maximum size is 4096^2.

TABLE I
MEMORY REQUIREMENTS FOR 2D VISCOELASTIC MODEL

Gridsize(N^2)	Memory Requirements
512^2	22.56M
1024^2	87.12 M
2048^2	342.25 M
4096^2	1356.50 M

D. Rendering

To update textures at each time step, we render quadrilaterals (regions) in order to generate fragments mapped with those textures described above. Vertex domain in *interior region* operates on *whole-texures*; PML2 runs on *x-direction-textures* and switches viewport twice on the left and right half of the texture. PML3 runs on *z-direction-textures* and switches viewport twice on the top and bottom half of the texture. PML1 runs on either *z-direction-textures* or *x-direction-textures*.

In our approach, velocity and stress fields are staggered in spatial and time domain. Therefore, the velocity field is updated after the stress field is updated and the priority of subdomains processing is interior region, PML2 (3), PML3 (2), PML1.

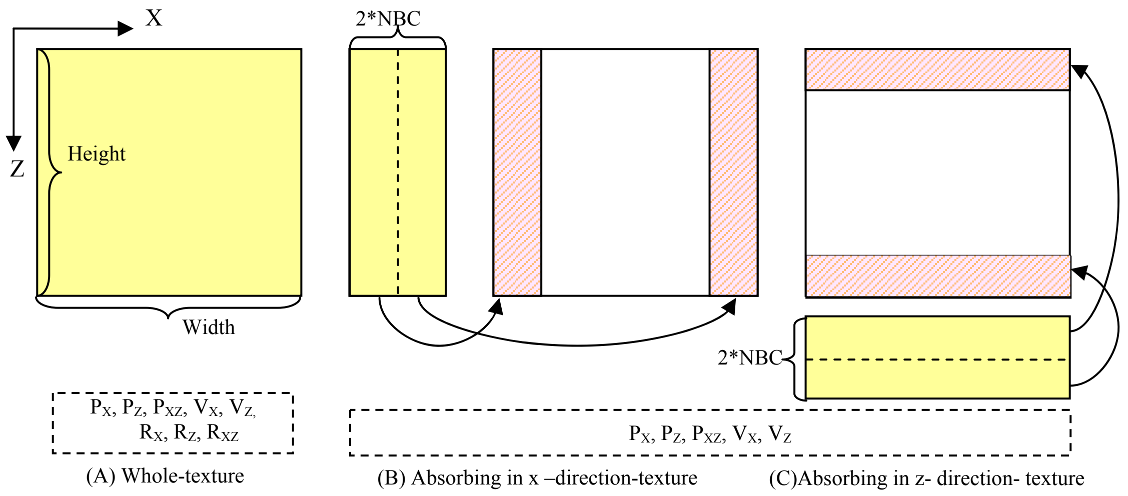

X
Z
Height
Width
2*NBC
2*NBC
Px, Pz, Pxz, Vx, Vz, Rx, Rz, Rxz
Px, Pz, Pxz, Vx, Vz
(A) Whole-texture
(B) Absorbing in x –direction-texture
(C)Absorbing in z- direction- texture

Fig.3. Three texture types at each time step: whole-texture denotes the all computational domain; x-direction-texture and z-direction-texture store absorbing results in x-and z-direction.

E. Fragment processing

The results of each step are written to a two-dimensional memory location called framebuffer. But the characteristics of the GPU prohibit the usage of the buffer for reading and writing operations at the same time. Fortunately, the OpenGL extension FrameBuffer Object (FBO) allows reusing the framebuffer content as a texture for further processing in a subsequent rendering pass and the “ping-pong" method allows two buffers swap their role after each write operation [9]. In order to generate and store intermediate results for each simulating step on GPUs, our strategy is to bind each texture that is rendered to onto one attachment on an assigned FBO; two textures binding on the same FBO are used to represent one stress-velocity component at time t and t+1, respectively. The pseudo FBO initialization codes for each component are given as follows.

```
glGenFramebuffersEXT(1,&m_fbo);
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, m_fbo);
FOR each N = 0, 1
  setupTexture (Texture[N], Width, Height)
  glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT,
  COLOR_ATTACHMENT[N],
  GL_TEXTURE_RECTANGLE_ARB, Texture[N], 0);
END FOR
```

In each rendering pass, *glBindFramebufferEXT()* is used to active the assigned FBO, and *glDrawBuffer()*

used to switch attachment points, making sure the current texture is to be written. To update FBO textures at each time step, we render quadrilaterals (regions) mapped to the computational domain in order, so that a fragment program running at each texel computes the FD equations. After each frame, FBO attachments are swapped and the latest updated textures are then used as inputs for the next simulation step while N = 1- N (N = 0, 1).

```
glBindFramebufferEXT (GL_FRAMEBUFFER_EXT, m_fbo );
glDrawBuffer ( COLOR_ATTACHMENT[N] );
cgGLBindProgram ( m_cgprog );
DrawRegion();
```

Alternatively, when data demand for simulating exceeds the capacity of the graphics video memory, the intermediate results can be transferred out into the host memory and for each frame be reloaded into the texture memory, and thus the video memory bottleneck is alleviated.

V. Numerical Examples

We have experimented with our GPU accelerated method implemented using C++/OpenGL and Cg shader language on a PC with Intel Core(TM)2 Duo E7400 2.8GHz, 2G main memory. The graphics chip is NVidia GeForce 9800GT with 512M video memory and 550MHz core frequency. The OS is Windows XP. All the results related to the GPU are based on 32-bit single-precision computation of the rendering pipeline. For comparison, we have also implemented a software version of the seismic simulation using single-precision floating point, and we have measured their performance on the same machine.

TABLE 2
FD Properties of The Grid Model

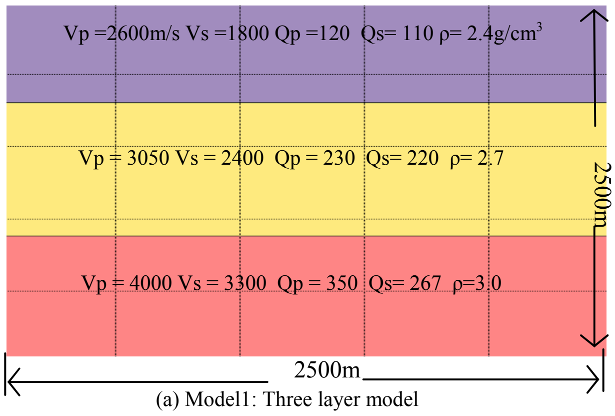
Gridsize(N^2)	△X*△Z	T/△T
256^2	10m*10m	2s/1ms
512^2	5m*5m	2s/1ms
1024^2	2.5m*2.5m	1s/0.5ms
2048^2	1.2m*1.2m	0.5s/0.25ms

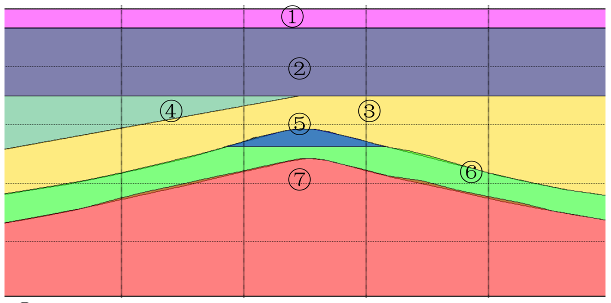
Two samples are used. One is a three-layer model (model1) and another is similar to Kelly model [14] (model2). The source is located at the top of the medium (Fig.4) and propagates downwards ricker wave with a dominant frequency of approximately 50 Hz.

The Models are discretized in various grid resolutions as shown in table2. The computation runs 2000 steps on both GPU and CPU, respectively. Note that the FD simulation is known to handle arbitrary models in accuracy. We show only the speedup of the method on the GPU.

The results for the two models are plotted in Fig.5. The time of viscoelastic wave simulation is defined as a function of the grid size. In order to compare the performance, GPU based method do not transfer the velocity-stress field or the parameters between the main memory and the graphics memory. The time spent on advecting and rendering the regions is negligible with respect to the simulation. The time of staggered-grid FD method is only dependent on the grid size, independent on geology structure complexity, which is shown in both GPU (Fig. 5a) and CPU (Fig.5b) based implementation.

The performance of the GPU based implementation increases significantly with the increasing number of the gridpoints. The speedup factor, which is defined as the ratio of CPU's time to GPU's time on the same grid size, increases from about 2 (128^2) to 50 (2048^2), illustrated in Fig. 5c. That is mainly due to the advantages of GPU's parallel computing, and the staggered grid FD method is amenable to an implementation by the rendering pipeline. For the simulation only runs in several minutes, geophysicists can use the new method to visualize the each frame of the computing process in real time, which helps to analysis the wave propagations easily.


Vp =2600m/s Vs =1800 Qp =120 Qs= 110 ρ= 2.4g/cm3
Vp = 3050 Vs = 2400 Qp = 230 Qs= 220 ρ= 2.7
Vp = 4000 Vs = 3300 Qp = 350 Qs= 267 ρ=3.0
2500m
2500m
(a) Model1: Three layer model



①Vp=1200	Vs=800	Qp=80	Qs=60	ρ=1.4
②Vp=2440	Vs=1330	Qp=100	Qs=90	ρ=2.2
③Vp=3210	Vs=2389	Qp=219	Qs=190	ρ=2.3
④Vp=3500	Vs=2173	Qp=200	Qs=170	ρ=2.5
⑤Vp=3225	Vs=1800	Qp=294	Qs=220	ρ=2.5
⑥Vp=3822	Vs=2980	Qp=490	Qs=340	ρ=2.6
⑦Vp=4785	Vs=3800	Qp=450	Qs=400	ρ=3.4

(b) Model2: Modified Kelly model

Fig. 4 Two test models: both are the same size of 2500m*2500m.

On the other hand, GPU based method improves more requirements of graphics card to use the programmable shaders. Now, only the latest nVidia or ATI card (Geoforce 6 above) can support the method. In additon, the available grid size is limit by the GPU texture maximum size. Now, the texture maximum size is 4096^2, so difference grid will not surpass the maximum size and the simulation accuracy is restricted. Graphics video memory size is about 256-512M in general, which limit the computing resources and only less than 2048^2 grid sizes can be supported directly.

VI. Conclusion

We have presented a GPU accelerated staggered-grid finite difference method to simulate the seismic propagation in 2D viscoelastic media model with complex earth structure geometry. The physical model is divided into subdomains for PML absorbing conditions and the implementation is totally implemented using the current fragment and vertex processing stages on GPUs. The simulation program runs on Modern PCs with popular low-cost GPUs, yet it is much faster than CPU-based simulation. Our experimental results have shown that the GPU version is significantly faster (for example, 3-50 times faster) for large grid sizes of 2D Models. The method makes it possible to simulate realtime complex seismic propagation in high resolution of 1024^2 – 2048^2 gridsizes on low-cost PCs.

Our future work will focus on GPU accelerated 3D staggered-grid FD in real seismic media and in order to overcome the imitation of video memory capacity, we will setup a high performance parallel environment with multiple GPUs and CPUs, in which CPUs are used to partition computing task and pass messages between tasks and GPUs are appointed to solve the FD equations.

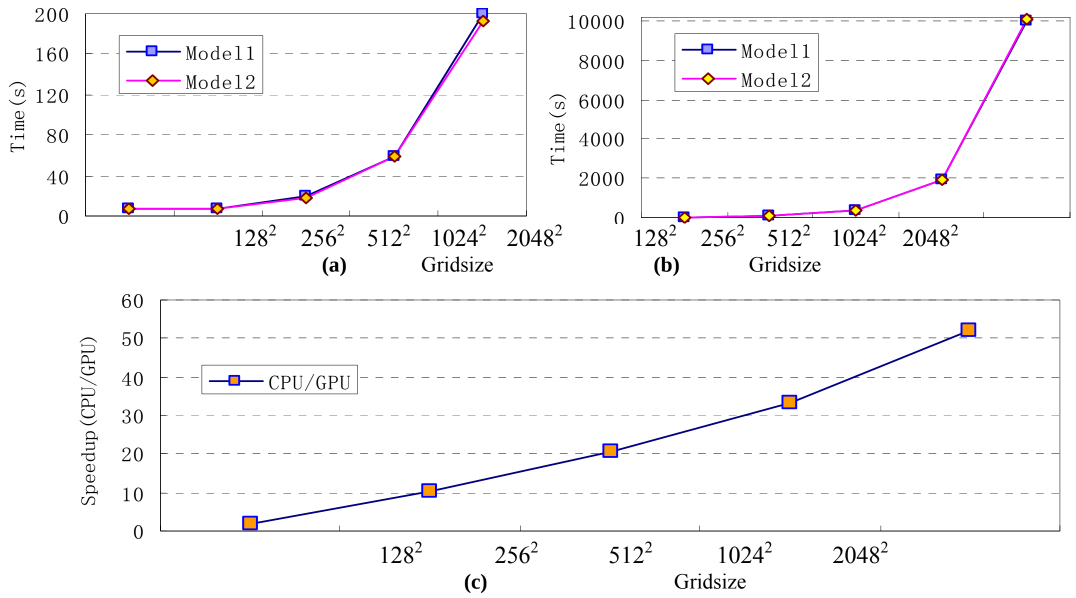


Fig.5. Comparative Analysis of GPU and CPU based staggered-grid FD method: (a) GPU based method using model1 and model2, (b) CPU based method using model1 and model2, (c) speedup analysis by CPU/GPU using model1.

Acknowledgment

The work was supported by China Postdoctoral Foundation (No.20090450466) and the National Key Basic Research and Development (973) Program of China (Grant No. 2010CB732002, 2009CB724601, 2005CB221505).

References

[1] T. Bohlen, "Parallel 3-D viscoelastic finite difference seismic modelling," *Computers & Geosciences*, vol. 28, no.8, pp.887–899, October 2002

[2] D.H. Sheen, K. Tuncay, C.E. Baag, and P. J. Ortoleva, "Parallel implementation of a velocity-stress staggered-grid finite-difference method for 2-D poroelastic wave propagation," *Computers & Geosciences*, vol. 32, no.8, 2006, pp.1182–1191.

[3] W. Sun, J.W. Shu, and W. Zheng, "Parallel Seismic Propagation Simulation in Anisotropic Media by Irregular Grids Finite Difference Method on PC Cluster," Gervasi et al. (Eds.): *ICCSA 2005*, LNCS 3483, 2005, pp.762 – 771

[4] J. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kruger, A. Lefohn and T. Purcell, " A survey of general-purpose computation on graphics hardware," In *Eurographics 2005*, State of the Art Reports, pp. 21-51.

[5] J. F. Ohmer, N.J. Redding, "GPU-Accelerated KLT Tracking with Monte-Carlo-Based Feature.Reselection," *IEEE Computer Society, Digital Image Computing: Techniques and Applications, DICTA 2008*, pp. 234-242.

[6] S. Tzeng and L.Y. Wei, "Parallel white noise generation on a GPU via cryptographic hash," ACM:In *Proceedings of the 2008 symposium on Interactive 3D Graphics (SI3D 2008)*, 2008, pp. 79–87.

[7] H. Mark, "Mapping Computational Concepts to CPUs," *GPU Gems2, Addison-Wesley*. NVIDIA, 2005, Chapter 47, pp.493-508

[8] J. F. Ohmer, F. Maire and R. Brown, "Implementation of kernel methods on the GPU," *IEEE Computer Society, In Digital Image Computing: Techniques and Applications*, DICTA 2005, pp. 543-550

[9] D. Göddeke. *GPGPU Basic Math Tutorial*. Available: http://www.mathematik.unidortmund.de/~goeddeke/gpgpu /tutorial.html

[10] J. Virieux, "SH-wave propagation in heterogeneous media: velocity-stress finite-difference method," *Geophysics*, vol. 49 issue 11, 1984, pp.1933–1957.

[11] J. Virieux, "P-SV wave propagation in heterogeneous media: velocity-stress finite-difference method," *Geophysics*, vol. 51 issue 4, 1986, pp.889–901.

[12] A.R. Levander, "Fourth-order finite-difference P-SV seismograms," *Geophysics*, vol. 53, issue 11, pp.1425–1436, 1988

[13] K.R Kelly, "Numerical study of love wave propagation," *Geophysics*, vol.48, issue 7, 1983, pp.833–853

[14] K.R. Kelly, R.W. Ward, S. Treitel, and R.M. Alford, "Synthetic seismograms: a finite-difference approach," *Geophysics*, vol. 41, issue 1, 1976, pp.2–27
[15] R. W. Graves, "Simulating seismic wave propagation in 3D elastic media using staggered grid finite differences," *Bull. Seism. soc. Am.*, vol. 86, 1996, pp.1091–1106
[16] G. Kneib and C. Kerner, "Accurate and efficient seismic modeling in random media," *Geophysics*, vol. 58, issue 4, 1993, pp.576–588
[17] S.M. Day and J.B. Minster, " Numerical simulation of wavefields using a Pade approximant method," *Geophysical Journal of the Royal Astronomical Society*, vol. 78, 1984, pp.105–118
[18] H. Emmerich, M. Korn, "Incorporation of attenuationinto time-domain computations of seismic wave fields". *Geophysics*, vol. 52, issue 9, 1987, pp.1252–1264.
[19] H.P. Liu, D.L. Anderson and H. Kanamori, "Velocity dispersion due to anelasticity: implications for seismology and mantle composition," *Geophysical Journal of the Royal Astronomical Society*, vol. 47, 1976, pp.41–58
[20] J.O. A , Robertsson, J.O. Blanch, and W.W. Symes, "Viscoelastic finite-difference modeling," *Geophysics*, vol. 59, issue 9, 1994, pp.1444–1456.
[21] M.A. Biot, "Mechanics deformation and acoustic propagation in porous media," *Journal of Applied Physics,* vol. 33, 1962, pp.1482-1498
[22] Y.Q. Zeng and Q.H. Liu, "A staggered-grid finite-difference method with perfectly matched layers for poroelastic wave equations," *The Journal of the Acoustical Society of America*, vol. 109, 2001, pp. 2571–2580.
[23] J..M. Carcione, D. Kosloff, and R. Kosloff, "Wave propagation simulation in a viscoelastic medium," *Geophysical Journal of the Royal Astronomical Society,* vol. 93, 1988, pp.597–611
[24] R. Clayton and B. Engquist, "Absorbing boundary conditions for acoustic and elastic wave equations," *Bulletin of the Seismological Society of America*, vol. 67, 1977, pp.1529–1540
[25] J.P. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," *Journal of Computational Physics,* vol. 114, 1994, pp.185–200
[26] F. D. Hastings, J. B. Schneider, and S. L. Broschat, "Application of the perfectly matched layer (PML): absorbing boundary condition to elastic wave propagation," *J. Acoust. Soc. Am*, vol. 100, 1996 , pp.3061–3069
[27] Q. H. Liu and J. Tao, "The perfectly matched layer for acoustic waves in absorptive media," *J. Acoust. Soc. Am*, vol. 102, 1997, pp.2072–2082
[28] W.C. Chew and Q.H. Liu, "Perfectly matched layers for elastodynamics: a new absorbing boundary condition," *Journal of Computational Acoustics*, vol. 4, 1996, pp.72–79
[29] Q. H. Liu, "Perfectly matched layers for elastic waves in cylindrical and spherical coordinates," *J. Acoust. Soc. Am.* Vol. 105, 1999, pp. 2075–2084

Zhangang Wang was born on November 4, 1980 in Ningxia, China. He received the B.S. and PhD degree from Peking University in 2003 and 2008, respectively. He was currently a lecturer of applied geophysics in China University of Mining and Technology, Beijing. His research interests include high performance computing for geosciences.

Suping Peng is an academician of China National Engineering. He serves as Director of the State Key Laboratory of Coal Resources and Mine Safety, China. He actively conducts research in the3D-3C seismic exploration of coal resources.

Tao Liu is a PhD candidate in Peking University. He received the B.S. degree from Peking University in 2006 and now researches elastic wave propagation.