Tu LHR1 10

Viscoelastic Forward and Adjoint Modeling with OpenCL on Heterogeneous Clusters

G. Fabien-Ouellet* (INRS-ETE), E. Gloaguen (INRS-ETE) & B. Giroux (INRS-ETE)

SUMMARY

Efficient seismic modeling is more and more needed because of the advent of full waveform inversion (FWI). For real case FWI, an efficient usage of the available computer resources is paramount. With the diversity of processor architectures found today, this is not a trivial task. In this study, we investigate the use of OpenCL to take advantage of large heterogeneous clusters in the context of FWI. The main objective is to present a scalable, multi-device code for the resolution of the viscoelastic wave equation that can compute the gradient of the objective function by the adjoint state method. We present several algorithmic aspects of our program in details, with an emphasis on its different levels of parallelism. The performance of the program is shown with several tests performed on large clusters with nodes containing three types of processors: Intel CPUs, NVidia GPUs and Intel Xeon PHI. We obtain a speed-up of more than 80 when using GPUs compared to a single threaded implementation and a linear scaling when computations are divided on separate nodes. Our results show that OpenCL allows a better usage of the computing resources available using a single source code for a multitude of devices.

Introduction

In the seismic community, General-Purpose Computing on Graphics Processing Units (GPGPU) has been applied most successfully to modeling wave propagation with Finite-Difference Time-Domain (FDTD) schemes. Several authors report an acceleration by a factor between 20 to 60 between a single-core and a single GPU implementations (Michéa and Komatitsch, 2010; Okamoto, 2011; Rubio et al., 2014; Weiss and Shragge, 2013). Multi-GPU implementations are also presented by these authors, all showing a sub-linear scaling.

Nearly all of the GPGPU seismic modeling codes have been implemented with the use of the CUDA standard (Nvidia, 2007). CUDA allows to easily program on Nvidia's GPUs; however a CUDA program cannot run on devices other than Nvidia's GPUs. On the other hand, OpenCL (Stone et al., 2010) is an open programming standard capable of using most existing types of processors and is supported by the majority of manufacturers like Intel, AMD and Nvidia, with performances comparable to CUDA (Du et al., 2012). Despite this advantage over CUDA, few published seismic modeling codes use OpenCL.

Efficient seismic modeling is more and more needed because of the advent of full waveform inversion (FWI). FWI is the process of recovering the Earth (visco)-elastic parameters by directly comparing raw seismic records to the solution of the wave equation (Tarantola, 1984; Virieux and Operto, 2009). Its main bottleneck is the numerical resolution of the wave equation that must be repeatedly performed for hundreds if not thousands of shot points for a typical survey. In addition, FWI requires the computation of the misfit gradient by the adjoint state method (Plessix, 2006), a method based on the same FDTD algorithm as the forward model. Hence, the benefit of a faster FDTD code is twofold.

In this study, we investigate the use of OpenCL for modeling wave propagation in the context of time domain FWI. The main objective is to present a scalable, multi-device and portable code for the resolution of the 2D and 3D viscoelastic wave equation that can additionally compute the gradient of the objective function used in FWI by the adjoint state method. First, the equations for viscoelastic wave propagation are briefly discussed. Then, different algorithmic aspects of the program are presented in details. Finally, numerical results are presented performed on large clusters with nodes containing three types of processors: Intel CPUs, Nvidia's GPUs and Intel Xeon PHI.

Theory

The goal of full waveform inversion is to estimate the viscoelastic parameters of the earth based on some records of the ground motion $\boldsymbol{d_i}$, usually in the form of particle velocities or pressure. This is performed by the minimization of a cost function, usually taken as the least-squares misfit of the raw seismic traces:

$$J(\boldsymbol{m}) = \frac{1}{2}\left(\boldsymbol{S}(v_i) - \boldsymbol{d_i}\right)^T \left(\boldsymbol{S}(v_i) - \boldsymbol{d_i}\right), \tag{1}$$

where $\boldsymbol{S}$ is the sampling operator. In this work, the forward model is given by the viscoelastic wave equation using the generalized standard linear solid (GSLS) as described by (Carcione et al., 1988; Robertsson et al., 1994):

$$\dot{v}_i = \frac{1}{\rho}\frac{\partial \sigma_{ij}}{\partial x_j} + f_i, \tag{2a}$$

$$\dot{\sigma}_{ij} = \left(M\frac{1+L\tau_p}{1+\alpha\tau_p} - 2\mu\frac{1+L\tau_s}{1+\alpha\tau_s}\right)\frac{\partial v_k}{\partial x_k}\delta_{ij} + \mu\frac{1+L\tau_s}{1+\alpha\tau_s}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) + \sum_{l=1}^{L} r_{ijl} + f_{ij}, \tag{2b}$$

$$\dot{r}_{ijl} = -\frac{1}{\tau_{\sigma l}}\left[\left(M\frac{\tau_p}{1+\alpha\tau_p} - 2\mu\frac{\tau_s}{1+\alpha\tau_s}\right)\frac{\partial v_k}{\partial x_k}\delta_{ij} + \mu\frac{\tau_s}{1+\alpha\tau_s}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) + r_{ijl}\right], \tag{2c}$$

where v are the velocities, σ the stresses, r the memory variables implementing viscous attenuation, ρ is the density, M is the P-wave modulus, μ is the shear modulus, $\tau_{\sigma l}$ are the relaxation times for the L Maxwell bodies and τ_p and τ_s are the attenuation levels.

The adjoint model equation has the same form as equation 2, with the source term equal to the data residuals reversed in time, i.e. $f = (\boldsymbol{S}(v_i) - \boldsymbol{d_i})|_{T-t}$. Once the forward and adjoint wavefields are computed, the misfit gradient can be computed by their cross-correlation (Tarantola, 1988).

OpenCL implementation

Equation 2 is is solved using a finite-difference approach with a standard staggered grid similar to Levander (1988) and Bohlen (2002). Spatial derivatives are approximated by finite-differences of order 2 to 12 and temporal derivatives are approximated by a stencil of order 2. In the classic leapfrog manner, velocities and stresses are updated sequentially. At each time step, the wavefield update is independent for each grid point. This conforms to the single-program multiple-data model and is the reason why FDTD computation is efficient on a GPU architecture. However, the computation of the spatial derivative requires a number of nearest neighbour grid points equal to the stencil order. This requires a careful usage of the GPU local memory.

The simplified algorithm implemented in OpenCL is described in algorithm 1. Several levels of parallelism exist in our code. First, nodes of a cluster are divided into groups, each performing the simulation on a subset of shots (task-parallel decomposition). Within each group, devices share computation by domain decomposition (Mattson et al., 2004), represented by the loops on lines 2 and 3 of algorithm 1. This decomposition requires memory communication that is implemented with MPI. To facilitate this communication, the update computations are split between grid points required in the communication and interior points (lines 6 to 11 of algorithm 1). Finally, there is a level of parallelism inside each OpenCL device, where the update of the wavefield is performed simultaneously for several grid points. These different levels of parallelism allow obtaining the best performances for large heterogeneous clusters containing any mix of GPUs, CPUs and accelerators.

Algorithm 1 Adjoint modeling

```
 1: procedure TIMESTEPPING(group, source)
 2:     for all $nodes \in group$ do
 3:         for all $devices \in node$ do
 4:             for $t \leftarrow 0, N_t$ do
 5:                 Inject source
 6:                 UPDATE($v$, $\sigma$, boundary)
 7:                 Transfer $v$ between devices, nodes
 8:                 UPDATE($v$, $\sigma$, interior)
 9:                 UPDATE($\sigma$, $v$, boundary)
10:                 Transfer $\sigma$ between devices, nodes
11:                 UPDATE($\sigma$, $v$, interior)

12: procedure UPDATE($v$, $\sigma$, domain)
13:     for all $local \in domain$ do
14:         for all $points \in local$ do
15:             Load $\sigma$ from global memory to local memory
16:             Compute $\partial_i \sigma$ from local memory
17:             Update $v$ in global memory
```

Performance

We present here two tests demonstrating the performance of the OpenCL program. The first test compares the running time of the OpenCL program with the single threaded implementation of Bohlen (2002), for three device architectures: a Nvidia Tesla K40 GPU, a CPU device containing two Intel Xeon E5-2680 v2 processors with 10 cores each at a frequency of 2.8 GHz and with 25 MB of cache and an Intel Xeon Phi 5110P accelerator. Results are shown in Figure 1. A maximum speed-up of 80 is attained with the GPU used in this study. The worst performing device is the Xeon PHI, probably because our program was not designed for this architecture. This shows that OpenCL still can require device specific optimization for maximum efficiency.

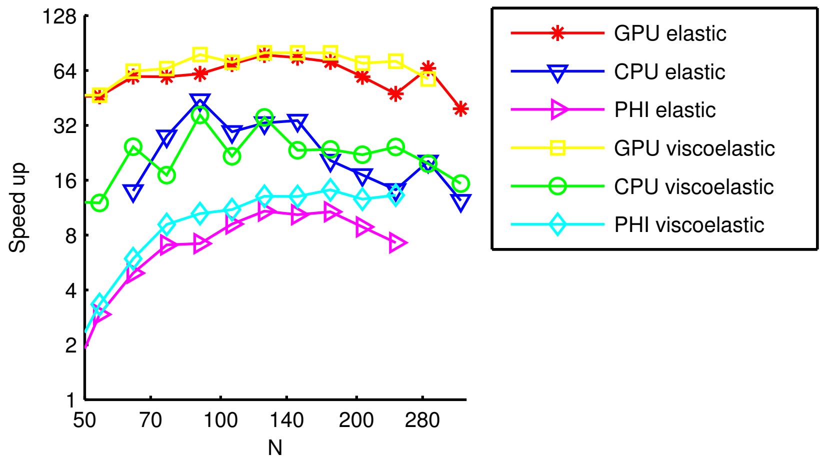

Figure 1 *Speedup over a single threaded CPU implementation for different model sizes in 3D.*

A strong scaling test is presented in Figure 2. This test consists in comparing the running time of the simulation for a fixed model size of 96x96x1000 for an increasing number of devices. This test was performed for three different configurations: on a single node containing 16 Nvidia GPUs, on multiple nodes containing 2 Nvidia GPUs and on multiple nodes containing CPUs only. The results show a quasi-linear scaling on nodes containing GPUs and a slightly worst scaling for nodes using CPUs. These results show that our OpenCL program is able to compute efficiently the wave equation solution for large 3D problems.

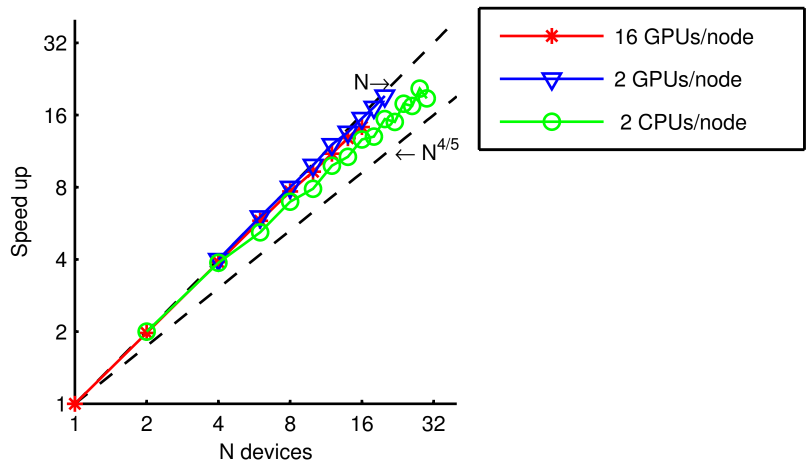

Figure 2 *Strong scaling tests for a grid size of 96x96x1000.*

Conclusions

In this work, OpenCL was used in the context of a large heterogeneous cluster for seismic modeling and for the adjoint computation of the misfit gradient in FWI. It was shown that OpenCL was able to take advantage of several processor architectures and lead to a significant speed-up when used with GPUs. An efficient integration of OpenCL and MPI allows the modeling of large 3D models over several nodes. Many aspects of the performance of OpenCL were not addressed in this paper, in particular, the efficiency of the different approaches to evaluate the wavefield cross-correlation during the gradient computation (in time or in frequency). Also, the impact of the finite-difference order was not addressed here. Further optimization for the Xeon Phi should also be the topic of future work.

Acknowledgements

This work was funded by a Vanier Canada Graduate Scholarship and supported by the Canada Chair in geological and geophysical data assimilation for stochastic geological modeling.

References

Bohlen, T. [2002] Parallel 3-D viscoelastic finite difference seismic modelling. *Computers and Geosciences*, **28**(8), 887–899.

Carcione, J.M., Kosloff, D. and Kosloff, R. [1988] Viscoacoustic wave propagation simulation in the earth. *Geophysics*, **53**(6), 769–777.

Du, P., Weber, R., Luszczek, P., Tomov, S., Peterson, G. and Dongarra, J. [2012] From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming. *Parallel Computing*, **38**(8), 391–407.

Levander, A.R. [1988] Fourth-order finite-difference P-SV seismograms. *Geophysics*, **53**(11), 1425–1436.

Mattson, T.G., Sanders, B.A. and Massingill, B.L. [2004] *Patterns for parallel programming*. Pearson Education.

Michéa, D. and Komatitsch, D. [2010] Accelerating a three-dimensional finite-difference wave propagation code using GPU graphics cards. *Geophysical Journal International*, no–no.

Nvidia, C. [2007] Compute unified device architecture programming guide.

Okamoto [2011] Accelerating large-scale simulation of seismic wave propagation by multi-GPUs and three-dimensional domain decomposition. *Earth, Planets and Space*, **62**(12), 939–942.

Plessix, R.E. [2006] A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, **167**(2), 495–503.

Robertsson, J.O.A., Blanch, J.O. and Symes, W.W. [1994] Viscoelastic finite-difference modeling. *Geophysics*, **59**(9), 1444–1456.

Rubio, F., Hanzich, M., Farrés, A., de la Puente, J. and María Cela, J. [2014] Finite-difference staggered grids in GPUs for anisotropic elastic wave propagation simulation. *Computers and Geosciences*, **70**, 181–189.

Stone, J.E., Gohara, D. and Shi, G. [2010] OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in science and engineering*, **12**(1-3), 66–73.

Tarantola, A. [1984] Inversion of seismic reflection data in the acoustic approximation. *Geophysics*, **49**(8), 1259–1266.

Tarantola, A. [1988] Theoretical background for the inversion of seismic waveforms including elasticity and attenuation. *Pure and Applied Geophysics*, **128**(1-2), 365–399.

Virieux, J. and Operto, S. [2009] An overview of full-waveform inversion in exploration geophysics. *Geophysics*, **74**(6), WCC1–WCC26.

Weiss, R.M. and Shragge, J. [2013] Solving 3D anisotropic elastic wave equations on parallel GPU devices. *Geophysics*, **78**(2), F7–F15.