Computers & Geosciences 91 (2016) 64–76

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

GPU performance analysis of a nodal discontinuous Galerkin method for acoustic and elastic models

A. Modave [a,*], A. St-Cyr [b], T. Warburton [a]

[a] Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA
[b] Shell Global Solutions International B.V., Rijswijk, The Netherlands

ARTICLE INFO

Article history:
Received 25 November 2015
Received in revised form
2 February 2016
Accepted 7 March 2016
Available online 10 March 2016

Keywords:
Finite element
Discontinuous Galerkin
Seismic waves
Time domain
GPU
BLAS
Profiling

ABSTRACT

Finite element schemes based on discontinuous Galerkin methods possess features amenable to massively parallel computing accelerated with general purpose graphics processing units (GPUs). However, the computational performance of such schemes strongly depends on their implementation. In the past, several implementation strategies have been proposed. They are based exclusively on specialized compute kernels tuned for each operation, or they can leverage BLAS libraries that provide optimized routines for basic linear algebra operations. In this paper, we present and analyze up-to-date performance results for different implementations, tested in a unified framework on a single NVIDIA GTX980 GPU. We show that specialized kernels written with a one-node-per-thread strategy are competitive for polynomial bases up to the fifth and seventh degrees for acoustic and elastic models, respectively. For higher degrees, a strategy that makes use of the NVIDIA cuBLAS library provides better results, able to reach a net arithmetic throughput 35.7% of the theoretical peak value.

Published by Elsevier Ltd.

1. Introduction

High-performance compute resources are used intensively in modern computational seismology for applications including earthquake simulation and seismic imaging. State-of-the-art compute clusters consist of massively parallel many-core central processing units (CPUs) with graphics processing units (GPUs) or coprocessors to provide a performance boost. Computing on GPUs has been made accessible to the scientific community thanks to programming frameworks that expand languages already widely used (e.g. C, C++ or Fortran). However, to fully leverage the capabilities of GPUs, applications must be developed taking into account specifics of each architecture. It is therefore critical to develop computational algorithms that can be efficiently implemented on these architectures.

In computational seismology, several implementations have been proposed for numerical methods suitable for GPUs coprocessors. Examples include tuned implementations for finite-difference schemes (Michea and Komatitsch, 2010; Weiss and Shragge, 2013; Abdelkhalek et al., 2012; Rubio et al., 2014) or finite-element schemes (Heinecke et al., 2014; Komatitsch et al., 2009, 2010; Mu et al., 2013; Modave et al., 2015). Nowadays, the finite-difference schemes are the most widely used, and a large literature is available (see e.g. Virieux et al., 2011 for a review). However, the stencil based reconstruction used in finite difference methods is not ideally suited to resolving wave propagation in realistic physical heterogeneous media typically suffering from loss of accuracy (Symes and Vdovina, 2009). By contrast, finite-element methods based on unstructured meshes are better suited to handle such material interfaces. Several variants have been investigated, such as spectral finite-element methods (Komatitsch and Vilotte, 1998; Komatitsch and Tromp, 1999), continuous mass-lumped finite-element methods (ChinJoeKong et al., 1999; Cohen et al., 2001) or discontinuous Galerkin (DG) methods (Dumbser and Käser, 2006; de la Puente et al., 2007; Collis et al., 2010; Etienne et al., 2010; Krebs et al., 2014; Mercerat and Glinsky, 2015). In contrast to the standard schemes, the mentioned methods do not require the solution of large sparse linear systems of equations, and it is possible to use explicit time-stepping schemes. The DG approach, in particular, provides a framework that is both flexible for multi-scale modeling and appears to be well suited for GPU accelerated computing. First, it can easily handle local time-stepping strategies (Warburton, 2008; Dumbser and Käser, 2009; Baldassari et al., 2011; Minisini et al., 2013), and hybrid discretizations can be deployed by mixing different discretization orders and several kinds of elements (Kirby et al., 2000; Dumbser and Käser, 2009; Etienne et al., 2010; Chan et al., 2015). Then, the weak element-to-element coupling and the dense algebraic operations required per element are suitable for parallel multi-threading computations with GPUs (Klöckner et al., 2009).

* Corresponding author.
E-mail address: modave@vt.edu (A. Modave).

http://dx.doi.org/10.1016/j.cageo.2016.03.008
0098-3004/Published by Elsevier Ltd.

Discontinuous Galerkin schemes can be implemented in different ways for GPU computing. Klöckner et al. (2009) proposed an implementation for first-order wave systems discretized using a nodal DG scheme. This implementation has been successfully adapted to several physical contexts (Gödel et al., 2010; Gandham et al., 2015; Modave et al., 2015), and we have recently presented an application for reverse-time migration with multi-rate time-stepping and GPU clusters (Modave et al., 2015a). Alternatively, Fuhry et al. (2014) have proposed an implementation for two-dimensional problems with a modal DG scheme. All these implementations partition the computational work into tailored GPU kernels, which are optimized separately in order to improve performance. A key difference between the approaches of Klöckner et al. (2009) and Fuhry et al. (2014) is the programming strategy of kernels: each thread performs computations corresponding to one node and to one element, respectively. On the other hand, Witherden et al. (2014, 2015) recently developed an implementation based on a nodal DG method for applications of fluid dynamics. This implementation, named PyFR, makes use of external BLAS libraries that provide linear algebra routines optimized for hardware devices. Such a strategy has been successfully used for CPU computing (Chevaugeon et al., 2005; Hesthaven and Warburton, 2002; Marchandise et al., 2006; Vos et al., 2010).

In this paper, we investigate and evaluate three strategies to implement time-domain DG schemes for GPU computing. All these strategies have been compared for three-dimensional acoustic and elastic cases with a unique computational framework. They were programmed in C++ using CUDA 7.5 through the abstract framework OCCA (Medina, 2014), and tested on a single Nvidia GTX980. We have tested one-element-per-thread and one-node-per-thread strategies, as well as a strategy that makes use of an external BLAS library. In this work, we have used the generalized single precision matrix-multiplication (SGEMM) routine of NVIDIA's cuBLAS library (NVIDIA Corporation, 2015a). We show that, similarly to CPU implementations (Vos et al., 2010), the best GPU implementation depends on the polynomial degree. The one-node-per-thread tailored kernels provided the best runtime for small degree, while the implementation with SGEMM is better for higher degrees.

This paper is organized as follows. In Section 2, we present the mathematical models for acoustic and elastic wave propagation, and we describe the nodal discontinuous Galerkin method and the time-stepping scheme. In Section 3, key aspects of GPU hardware and GPU programming are summarized, and all the implementations are described. Section 4 is dedicated to performance results and comparisons. We discuss the optimization of the SGEMM routine for the DG operations, and we analyze its performance using the roofline model. The implementations are then compared, and all the kernels are systematically profiled.

2. Discontinuous Galerkin schemes

We consider acoustic and isotropic elastic wave models discretized with a nodal discontinuous Galerkin method in space and on a third-order Adam–Bashforth method in time. We assume the physical coefficients to be constant over each mesh cell and discontinuous at interfaces. The variational forms of the models and the schemes are presented in Sections 2.1 and 2.2, respectively.

2.1. Physical models and variational forms

Acoustic waves are governed by the pressure-velocity system, which reads

$$\frac{\partial p}{\partial t} + \rho c^2 \nabla\cdot\mathbf{v} = 0,$$
$$\rho\frac{\partial \mathbf{v}}{\partial t} + \nabla p = 0,$$

with the pressure $p(\mathbf{x}, t)$, the velocity $\mathbf{v}(\mathbf{x}, t)$, the density $\rho(\mathbf{x})$ and the phase velocity $c(\mathbf{x})$. For each mesh cell D_k, we consider the variational form

$$\int_{\mathrm{D}_k} \frac{\partial p}{\partial t}\psi \, d\mathbf{x} = -\int_{\mathrm{D}_k} \rho c^2 \nabla\cdot\mathbf{v}\psi \, d\mathbf{x} - \int_{\partial\mathrm{D}_k} p_p \psi \, d\mathbf{x}$$
$$\int_{\mathrm{D}_k} \frac{\partial \mathbf{v}}{\partial t}\psi \, d\mathbf{x} = -\int_{\mathrm{D}_k} \frac{1}{\rho}\nabla p \, \psi \, d\mathbf{x} - \int_{\partial\mathrm{D}_k} p_{\mathbf{v}}\mathbf{n} \, \psi \, d\mathbf{x}$$

where $\psi(\mathbf{x})$ is a test function, $p_p(\mathbf{x})$ and $p_{\mathbf{v}}(\mathbf{x})$ are penalty terms. The penalty terms corresponding to upwind fluxes provided by a one-dimensional Riemann solver are given by (see e.g. Hesthaven and Warburton, 2007; Modave et al., 2015a)

$$p_p = -(\rho c^2)^- k_c \alpha_c,$$
$$p_{\mathbf{v}} = c^- k_c \alpha_c,$$

with $\alpha_c = [[p]] - (\rho c)^+ [[v_n]]$, $k_c = 1/\{\rho c\}$ and the semi-jumps defined as $[[p]] = (p^+ - p^-)/2$ and $[[v_n]] = (\mathbf{v}^+ - \mathbf{v}^-)\cdot\mathbf{n}/2$. The brackets $\{\cdot\}$ denotes the mean value at the interface.

For isotropic media, elastic waves can be simulated with the velocity-stress system, which reads

$$\frac{\partial \boldsymbol{\sigma}}{\partial t} = \lambda(\nabla\cdot\mathbf{v})\mathbf{I} + 2\mu\mathrm{sym}\big(\nabla\mathbf{v}\big),$$
$$\rho\frac{\partial \mathbf{v}}{\partial t} = \nabla\cdot\boldsymbol{\sigma},$$

with the stress tensor $\boldsymbol{\sigma}(\mathbf{x}, t)$ and the Lamé parameters $\lambda(\mathbf{x})$ and $\mu(\mathbf{x})$. This system supports pressure waves and shear waves, which the phase velocities are

$$c_p = \sqrt{(\lambda + 2\mu)/\rho},$$
$$c_s = \sqrt{\mu/\rho},$$

respectively. We consider the variational form

$$\int_{\mathrm{D}_k} \frac{\partial \boldsymbol{\sigma}}{\partial t}\psi \, d\mathbf{x} = \int_{\mathrm{D}_k} \lambda(\nabla\cdot\mathbf{v})\mathbf{I} + 2\mu\mathrm{sym}\big(\nabla\mathbf{v}\big)\psi \, d\mathbf{x} + \int_{\partial\mathrm{D}_k} \mathbf{P}_{\boldsymbol{\sigma}}\psi \, d\mathbf{x},$$
$$\int_{\mathrm{D}_k} \frac{\partial \mathbf{v}}{\partial t}\psi \, d\mathbf{x} = \int_{\mathrm{D}_k} \frac{1}{\rho}(\nabla\cdot\boldsymbol{\sigma})\psi \, d\mathbf{x} + \int_{\partial\mathrm{D}_k} \mathbf{p}_{\mathbf{v}}\psi \, d\mathbf{x}$$

where the upwind fluxes provided by an exact Riemann solver are given by Wilcox et al. (2010)

$$\mathbf{P}_{\boldsymbol{\sigma}} = k_p\alpha_p\lambda^-\mathbf{I} + 2(k_p\alpha_p - k_s\alpha_s)\mu^-\mathbf{N} + 2k_s\mu^-\mathrm{sym}\big(\mathbf{n}\otimes\boldsymbol{\alpha}_s\big),$$
$$\mathbf{p}_{\mathbf{v}} = (k_p\alpha_p c_p^- - k_s\alpha_s c_s^-)\mathbf{n} + k_s c_s^-\boldsymbol{\alpha}_s,$$

with

$$\alpha_p = [\sigma_{nn}] + (\rho c_p)^+[v_n], \quad k_p = 1/\{\rho c_p\}, \quad [\sigma_{nn}] = \mathbf{n}\cdot(\boldsymbol{\sigma}^+ - \boldsymbol{\sigma}^-)\cdot\mathbf{n}/2,$$
$$\alpha_s = [\sigma_{nn}] + (\rho c_s)^+[v_n], \quad k_s = 1/\{\rho c_s\}, \quad [\boldsymbol{\sigma}\mathbf{n}] = (\boldsymbol{\sigma}^+ - \boldsymbol{\sigma}^-)\cdot\mathbf{n}/2,$$
$$\boldsymbol{\alpha}_s = [\boldsymbol{\sigma}\mathbf{n}] + (\rho c_s)^+[\mathbf{v}], \quad \mathbf{N} = \mathbf{n}\otimes\mathbf{n}, \quad [\mathbf{v}] = (\mathbf{v}^+ - \mathbf{v}^-)/2.$$

2.2. Numerical schemes

The approximate fields are built on a spatial mesh of the computational domain $\Omega \subset \mathbb{R}^3$, made of K non-overlapping tetrahedral cells, $\Omega = \bigcup_k \mathrm{D}_k$, where D_k is the kth cell. For the nodal discontinuous Galerkin method, all the scalar fields and the Cartesian components of vector and tensor fields are approximated by piecewise polynomial functions. The discrete unknowns correspond to the values of fields at nodes distributed over the surface and interior of an element (Hesthaven and Warburton, 2002,