C. R. Mecanique 339 (2011) 125–135

Contents lists available at ScienceDirect

Comptes Rendus Mecanique

www.sciencedirect.com

High Performance Computing / Le Calcul Intensif

Fluid–solid coupling on a cluster of GPU graphics cards for seismic wave propagation

Couplage fluide–solide sur un réseau de cartes graphiques GPU pour la propagation des ondes sismiques

Dimitri Komatitsch [a,b,*]

[a] *Université de Pau et des Pays de l'Adour, CNRS & INRIA Magique-3D, Laboratoire de Modélisation et d'Imagerie en Géosciences UMR 5212, Avenue de l'Université, 64013 Pau cedex, France*
[b] *Institut universitaire de France, 103 boulevard Saint-Michel, 75005 Paris, France*

ARTICLE INFO

Article history:
Available online 30 December 2010

Keywords:
Computer science
Numerical modeling
Finite elements
Seismic waves

Mots-clés :
Informatique, algorithmique
Modélisation numérique
Éléments finis
Ondes sismiques

ABSTRACT

We develop a hybrid multiGPUs and CPUs version of an algorithm to model seismic wave propagation based on the spectral-element method in the case of models of the Earth containing both fluid and solid layers. Thanks to the overlapping of communications between processing nodes on the computer with calculation by means of non-blocking message passing, we obtain excellent weak scalability of this finite-element code on a cluster of 192 GPUs and speedup factors of more than one order of magnitude compared to the same code run on a cluster of traditional CPUs. This enables us to show a new geophysical phenomenon concerning wave propagation of diffracted shear waves in a layer called D″ located at the base of the Earth's mantle, namely that in this layer the transverse and radial components of these waves can undergo a relative shift even in an isotropic Earth model, whereas this observation in real seismological data was interpreted until now as an indication of the presence of anisotropy in this layer.

RÉSUMÉ

Nous développons une version hybride multiGPUs et CPUs d'un algorithme de modélisation de la propagation des ondes sismiques fondé sur la méthode des éléments spectraux dans le cas de modèles de terre présentant à la fois des couches fluides et des couches solides. Grâce au recouvrement des communications entre noeuds informatiques par du calcul au moyen de passage de messages non bloquants nous obtenons un excellent passage à l'échelle faible (« *weak scalability* ») de ce code d'éléments finis d'ordre élevé sur un réseau (« *cluster* ») de 192 GPUs et des facteurs d'accélération de plus d'un ordre de grandeur par rapport au même code exécuté sur un cluster de CPUs classique. Ceci nous permet de démontrer un phénomène géophysique nouveau concernant la propagation des ondes de cisaillement diffractées dans une couche appelée D″ située à la base du manteau terrestre, à savoir que dans cette couche les composantes transverse et radiale de ces ondes peuvent subir un décalage relatif y compris dans un modèle de terre isotrope, alors que

* Address for correspondence: Université de Pau et des Pays de l'Adour, CNRS & INRIA Magique-3D, Laboratoire de Modélisation et d'Imagerie en Géosciences UMR 5212, Avenue de l'Université, 64013 Pau cedex, France.
E-mail address: dimitri.komatitsch@univ-pau.fr.

1631-0721/$ – see front matter
doi:10.1016/j.crme.2010.11.007

cette observation dans des données sismologiques réelles était jusqu'à présent interprétée comme un signe de la présence d'anisotropie dans cette couche.

1. Introduction regarding the seismological aspects

The seismological part of my research has been made in collaboration with Lev P. Vinnik (Russian Academy of Sciences, Moscow, Russia) and Sébastien Chevrot (CNRS, Observatoire Midi-Pyrénées, Toulouse, France).

The D″ (pronounced "D double prime") geological layer of the Earth, which is a layer located above the core–mantle interface and which has an average thickness of approximately 150 km, is a complex region that has been the subject of numerous seismological studies based on the propagation of seismic waves, which is our main way of studying the deep structure of the Earth (for a review see e.g. [1]). At a global scale, the deepest part of the Earth's mantle contains two large regions of low shear wave (also called S wave) velocity, called superplumes, one located beneath central Pacific and the other one beneath Southern Atlantic and Africa. The magnitude of large scale lateral variations of the speed of shear waves in the D″ layer is about $\pm 3\%$ [2].

The D″ layer is not only laterally heterogeneous but also anisotropic, at least in certain areas [3]. The form usually assumed for this anisotropy in D″ is hexagonal with a vertical symmetry axis (vertical transverse isotropy or VTI). Transverse isotropy leads to a difference in travel time of horizontally and vertically polarized shear waves (respectively called SV and SH). In seismic data recorded around the Earth following large earthquakes, the splitting of SV and SH waves is classically interpreted by seismologists as an indication of the presence of anisotropy in the D″ layer. However, in this work, using isotropic models of the Earth having a heterogeneous lower mantle, we examine the relative properties of the diffracted shear waves called SVdiff and SHdiff and show that their splitting can be mistaken for an indication of the presence of anisotropy in that layer of the Earth. We highlight a new feature of seismic waves propagating in the D″ layer, namely that even in a model without anisotropy we can obtain a very significant splitting of SV and SH waves, the vertically polarized seismic phase SVdiff arriving sometimes later than the horizontally polarized phase SHdiff.

2. Hybrid multiGPUs + CPUs computing for this problem

Since we are going to handle large numerical grids and simulate a large number of time steps for each geophysical model under study, we decided to resort to hybrid calculation using many Graphics Computing Units (GPUs) in parallel (with message passing based on the Message Passing Interface – MPI) and also using the CPUs of the nodes on which these GPUs are installed in order to carry out other operations. The goal of using GPUs is to drastically speed up the calculations because in the last few years graphics processors have quickly become important as a viable architecture to carry out scientific computations. Current GPUs can be seen as Single Instruction Multiple Data (SIMD) structures with a large number of processor cores and a hardware scheduler that simultaneously maintains thousands of threads active by effectively suspending tasks that are waiting for memory transactions. This part of my work has been made in collaboration with Gordon Erlebacher (Department of Scientific Computing, Florida State University, USA), Dominik Göddeke (Institut für Angewandte Mathematik, TU Dortmund, Germany), and David Michéa (INRIA Magique3D, Pau Bordeaux Sud Ouest, France).

We will see in the following that some meshes are too large to fit on the 192 GPUs of the hybrid part of the computer that we will use, and thus in this case currently our only option will be to go back to using multi CPUs + MPI calculations on a very large traditional cluster in order to be able to use more memory. However, in this first part, in the case of an implementation on a cluster of GPUs, a mesh coloring algorithm will enable us to carry out an effective summation on GPUs of the degrees of freedom in the assembly phase of a non-structured finite-element mesh. We will use non-blocking MPI message passing and will show that communications across the network and between CPUs and GPUs are almost entirely covered by calculation. We will also show that the weak scaling of our multiGPU code is excellent up to 192 GPUs, i.e. the whole 'Titane' machine of CCRT/CEA/GENCI, and that the acceleration factor (speedup) obtained is very significant compared to the same computer code implemented on a traditional cluster of CPUs.

In the literature, we can find some other work concerning seismic modeling and geosciences on GPUs. For example, [4] and [5] recently solved seismic migration problems in the time domain (seismic reverse time migration) for the oil industry on GPUs. They implemented a finite-difference method in the case in an acoustic geological medium with either constant or variable density and ran these calculations on a cluster of GPUs with MPI. In recent work, with colleagues we developed finite-difference or finite-element techniques on GPUs to model the propagation of seismic waves in a purely elastic medium [6–8]. Here, we develop an extension to the case of media that contain solid parts as well as fluid parts.

Regarding other work about using GPU clusters published in the literature, [9] described for the first time how an existing cluster (and the associated distributed-memory application based on MPI) could be improved significantly in terms of performance by using GPUs not for visualization but for calculation. More recently, [10] used a cluster of 160 GPUs and a code based on OpenGL to analyze the scalability, the price/performance ratio, power consumption and efficiency of multigrid finite-element solvers to solve a canonical Poisson's problem. A computational framework for molecular dynamics on clusters of GPUs was presented by [11], and [12] accelerated a solver for Euler's equations on a cluster of 16 GPUs.

3. Background

3.1. The spectral-element method and its resolution algorithm

We use the Spectral Element Method (SEM) to simulate numerically the propagation of seismic waves resulting from earthquakes of from active seismic acquisition experiments in the oil industry [13,14]. The SEM is one of the techniques widely used to model the propagation of seismic waves, together with the finite-difference method (see e.g. [15,16]). Its popularity is due to its great flexibility and accuracy. Another field in which the SEM is often used, at a completely different scale, is the simulation of ultrasonic experiments in a laboratory [17]. It is also often used to study anisotropic media (e.g., [18]). The SEM solves the variational (weak) form of the elastodynamics equation in the time domain on a non-structured mesh of elements, called the spectral elements, in order to compute the displacement vector at any grid point in the domain under study. This method is more flexible than traditional global pseudospectral techniques [19,20]. The SEM is a continuous Galerkin technique, which can easily be made discontinuous [21–25]; it is then close to a particular case of the discontinuous Galerkin technique [26–33], with optimized efficiency because of its tensorized basis functions. One of its advantages is that it can handle very distorted mesh elements accurately [34]. Note that in many (most?) geological models in the context of seismic wave propagation studies (except for fault dynamic rupture studies) a discontinuous mesh is not needed because material property contrasts are not drastic and thus a continuous formulation is sufficient.

We consider a linear elastic anisotropic rheology for a heterogeneous solid region of the Earth, and the seismic wave equation can thus be written in the differential (strong) form as:

$$\begin{aligned}
&\rho\ddot{\mathbf{u}} = \nabla\cdot\boldsymbol{\sigma} + \mathbf{f}\\
&\boldsymbol{\sigma} = \mathbf{C}:\boldsymbol{\varepsilon}\\
&\boldsymbol{\varepsilon} = \frac{1}{2}\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right]
\end{aligned} \tag{1}$$

where $\mathbf{u}$ is the displacement vector, $\boldsymbol{\sigma}$ is the symmetric second-order stress tensor, $\boldsymbol{\varepsilon}$ is the symmetric second-order strain tensor, $\mathbf{C}$ is the fourth-order stiffness tensor, ρ is density, and $\mathbf{f}$ is an external force representing the seismic source. The ':' symbol represents a double tensorial contraction operation, the T exponent represents transposition, and a dot over a symbol represents a derivative with respect to time. The material properties of the solid, $\mathbf{C}$ and ρ, can be spatially heterogeneous and are assumed to be known quantities that define the geological medium under study. $\mathbf{C}$ is positive definite and has the following symmetries (see e.g. [35]):

$$\begin{aligned}
&C_{ijkl} = C_{klij} \qquad \text{(major symmetries)}\\
&C_{ijkl} = C_{jikl} = C_{ijlk} \qquad \text{(minor symmetries)}
\end{aligned} \tag{2}$$

Let us denote by Ω and Γ respectively the domain under study and its boundary. One can then rewrite system (1) in the variational form by dotting it with an arbitrary test function $\mathbf{w}$ and integrating by part over the whole domain:

$$\int_\Omega \rho\mathbf{w}\cdot\ddot{\mathbf{u}}\,\mathrm{d}\Omega + \int_\Omega \nabla\mathbf{w}:\mathbf{C}:\nabla\mathbf{u}\,\mathrm{d}\Omega = \int_\Omega \mathbf{w}\cdot f\,\mathrm{d}\Omega + \int_\Gamma (\boldsymbol{\sigma}\cdot\hat{\mathbf{n}})\cdot\mathbf{w}\,\mathrm{d}\Gamma \tag{3}$$

The last term, i.e. the contour integral, vanishes because of the free surface condition, i.e. the fact that the traction vector $\boldsymbol{\tau} = \boldsymbol{\sigma}\cdot\hat{\mathbf{n}}$ must be equal to zero at the surface of the Earth, where $\hat{\mathbf{n}}$ is the normal to the contour.

In the case of fluid layers in contact with the elastic parts of the domain we use a scalar potential instead of displacement as the unknown of the problem in order to reduce computer storage by a factor of three by using a scalar instead of a vector. In the inviscid fluid, the equation of motion is

$$\rho\dot{\mathbf{v}} = -\nabla p \tag{4}$$

where $\mathbf{v}$ denotes the velocity in the fluid and the pressure p is determined by

$$\dot{p} = -\kappa\nabla\cdot\mathbf{v} \tag{5}$$

where κ is the bulk modulus of the fluid. To solve the system of Eqs. (4) and (5), we introduce a scalar potential χ such that

$$p = -\ddot{\chi} \tag{6}$$

From (4) and the initial conditions ($\mathbf{u} = \mathbf{0}$ at $t = 0$), we find that

$$\mathbf{u} = \rho^{-1}\nabla\chi \tag{7}$$

Upon substituting (6) and (7) into (5) we obtain a scalar equation for χ:

$$\ddot{\chi} = \kappa \nabla \cdot \left(\rho^{-1} \nabla \chi \right) \tag{8}$$

The weak form of this equation is obtained by multiplying it by a scalar test function w and integrating by parts over the volume Ω of the fluid layer:

$$\int_{\Omega} \kappa^{-1} w \ddot{\chi} \, \mathrm{d}\Omega = - \int_{\Omega} \rho^{-1} \nabla w \cdot \nabla \chi \, \mathrm{d}\Omega \tag{9}$$

Matching between the fluid and the solid regions in enforced explicitly based upon a coupling integral along the fluid–solid interfaces to impose the continuity of the traction $-p\hat{\mathbf{n}}$ and of the normal component of velocity $\hat{\mathbf{n}} \cdot \mathbf{v}$.

In the SEM technique, the physical domain under study is subdivided in mesh cells in which variables are approximated by high-degree interpolants. In order to get better accuracy, the edges of the elements honor the topography of the model as well as its main internal discontinuities such as interfaces between geological layers, or faults. A Jacobian transform then defines the mapping between the Cartesian points $\mathbf{x} = (x, y, z)$ inside a given deformed hexahedral element Ω_e and the reference cube. When modeling is carried out in a semi-infinite domain, absorbing layers called Convolution Perfectly Matched Layers (CPMLs) are usually used to absorb energy on the fictitious edges of the domain (e.g., [36,37]).

To represent the displacement field inside an element, the SEM technique uses Lagrange polynomials of degree 4 to 10 typically for the interpolation of functions [38,39]. The control points ξ_α are chosen to be the $n+1$ Gauss–Lobatto–Legendre (GLL) points because the combination of Lagrange interpolants with GLL quadrature drastically simplifies the algorithm because it leads to a perfectly diagonal mass matrix. This in turns leads to the use of fully explicit time schemes [14], which can very efficiently be implemented on very large parallel computers (e.g., [40]).

Time integration of the discrete system is usually carried out based on a second-order Newmark centered finite-difference time scheme (e.g., [41,14]), but higher-order time schemes such as symplectic schemes or a fourth-order Runge–Kutta algorithm can be used if necessary [42]. In the SEM algorithm, the sequential loop over all the time steps completely dominates the calculation cost because in almost all seismic wave propagation applications a large number of time steps is computed, typically between 5000 and 100,000. All the time steps have an identical cost because the grid is static (it does not change during the simulation) and the algorithm is completely explicit, which facilitates its optimization.

4. Porting our simulation package SPECFEM3D to a cluster of GPUs using CUDA

In the last decade, in collaboration with several colleagues, I developed SPECFEM3D, a software package that carries out the three-dimensional numerical simulation of seismic wave propagation resulting from earthquakes or from active seismic experiments in the oil industry, based on the spectral-element method. In order to study seismic wave propagation in the Earth at very high resolution (i.e. up to very high seismic frequencies) the number of mesh elements that one must use is very large. Typical calculations require a few hundred processor cores used in parallel for a few hours of elapsed time. Large simulations often require a few thousand processor cores, typically 2000 to 4000, and take two to five days of elapsed time [43]. To date, the largest calculation that we have carried out used nearly 150,000 processor cores and reached a sustained performance level of 0.20 petaflops [40].

To port this software package to a cluster of GPUs we use the NVIDIA CUDA programming language [44]. The implementation in CUDA, initially on a single GPU, follows the algorithm described in Section 3.1 by programming each of the three following stages in the form of separate CUDA kernels. The first stage updates the total displacement vector based on its value at the previous time step; the second stage carries out the calculation of elastic forces and the assembly of these forces within the grid of finite elements; and the last stage calculates the total acceleration vector. The first and last stages are intrinsically parallel because they apply to total vectors having a unique numbering of their degrees of freedom without indirect addressing, and they are thus easily programmed in CUDA with a perfect occupancy rate of the multiprocessors (multiprocessor occupancy) of 100% as well as perfectly coalesced memory accesses. In what follows we thus concentrate on the second stage, because tests not shown here demonstrate that this stage takes more than 85% of the computing time of each time step [6].

A significant advantage is that spectral-element codes to model linear seismic wave propagation are sufficiently accurate in single precision, as shown for instance in [14] and [6]. It is thus not necessary to resort to double precision calculations to solve this problem, which is an advantage on current GPUs because calculations in single precision are significantly faster, although the situation of double precision operations is currently improving, in particular on the NVIDIA FERMI architecture. On the GPU we associate each spectral element with a block of 128 compute threads, and employ a thread to handle each Gauss–Lobatto–Legendre quadrature point. As in our case each spectral element contains $(n+1)^3 = 125$ Gauss–Lobatto–Legendre points because we use polynomial basis functions of degree $n = 4$, 125 of the 128 threads perform useful work and the three others are used to implement "zero padding" and thus automatically obtain memory alignment on multiples of 16, which is optimal in CUDA. We first copy the total displacement vector corresponding to each spectral element in shared memory using indirect addressing to map the global numbering of the grid points to their local numbering. The derivative matrix associated with the Lagrange polynomials is stored in the so-called constant memory of the GPU, which has extremely fast access, and the CUDA calculation kernel then multiplies it with the local coefficients of the displacement vector at the GLL points. The third stage carries out numerical integration with the discrete Jacobian to obtain the local

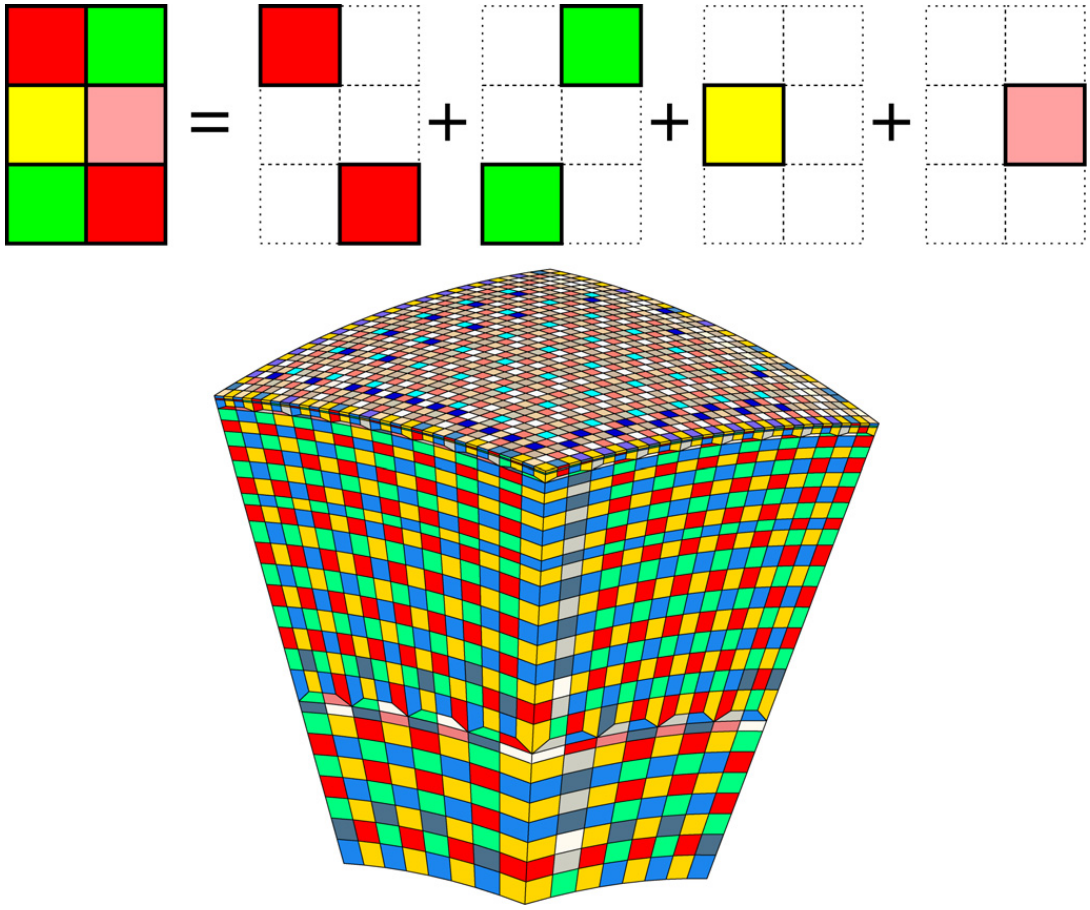

Fig. 1. Illustration of mesh coloring: a mesh of connected elements can always be split into subsets of disjoined elements; this removes the need for an atomic sum in CUDA, which would reduce the efficiency of our multiGPU code in a drastic way because some sections of the code would be serialized. Top: the principle of mesh coloring. Bottom: application to a real mesh slice.

gradient of the displacement vector. In the final stage, the contributions calculated within each element must be assembled at each global mesh point, since neighboring elements can share such mesh points. Thus, this assembly stage is similar to a reduction or scatter and therefore in principle requires an atomic sum, which would be very slow in CUDA because part of the sum in question would be serialized. We thus remove these dependencies between neighboring elements, which cannot easily be handled in parallel, based upon a mesh coloring technique to create subsets of independent elements (Fig. 1). This has the (fully acceptable) consequence that it is necessary to call the kernel CUDA once for each color of the mesh in a serial loop on all the colors (in practice for large meshes one usually obtains a maximum of a few tens of different colors only). This coloring of the mesh is carried out once and for all before the time loop, during the mesh creation phase.

4.1. MultiGPU parallel implementation

There are several challenges to address in order to perform this type of calculations on a cluster of GPUs. The elements that compose the mesh slices are in contact by a face, an edge or a common point. To allow for overlapping of communications between the nodes of the cluster with calculations on the GPUs, we create – inside each mesh slice – a list of all the 'outer' elements of the mesh slice, and a similar list of all the 'inner' elements (Fig. 2). We calculate the outer elements first, as is done classically [45]. Once these calculations performed, we copy the data associated with the MPI communication buffers that must be sent between neighboring mesh slices in contact, and then start the non-blocking MPI calls that initiate the communications and immediately return to the calling program. While the messages are traveling across the network, we compute the inner elements. Obtaining efficient overlapping of the communications requires that the ratio of the number of inner elements compared to the outer elements be sufficiently large, which is always the case for large meshes. Under these conditions of correct granularity, the transfer of MPI messages will statistically probably finish before the end of the calculation of the inner elements, and the cost of communications will thus be almost completely hidden. Let us note that to obtain efficient overlapping on a cluster of GPUs this ratio must be larger than on traditional clusters of CPUs because of the high speedup obtained on GPUs, which makes the duration of the calculation of the inner elements much shorter.

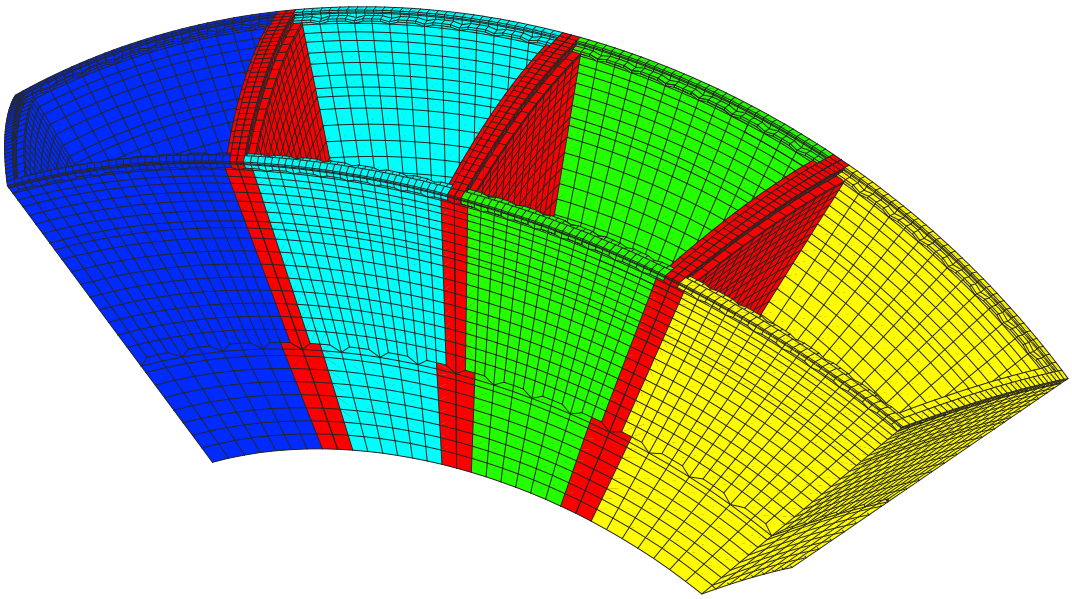

Fig. 2. Outer (in red) and inner (other colors) elements for part of a mesh. The elements in red share at least a point with an element of another mesh slice and must thus be calculated first, before launching non-blocking MPI communications. Here for clarity only the cut planes that define the mesh slices have been represented, but in reality the mesh is full.

5. Results on a multiGPU cluster

5.1. Test configurations

The machine we use is the 'Titane' cluster of 48 Teslas S1070 at CCRT/CEA/GENCI in Bruyères-le-Châtel, France; each Tesla S1070 has four GT200 GPUs and two PCI Express-2 buses (i.e., two GPUs share a PCI Express-2 bus). The GT200 cards have 4 GB of memory, and the memory bandwidth is 102 GB per second with a memory bus width of 512 bits. The Teslas are connected to BULL Novascale R422 E1 nodes with two quad-core Intel Xeon Nehalem processors operating at 2.93 GHz. Each node has 24 GB of RAM and runs Linux kernel 2.6.18. The network is Infiniband.

For the scalability tests, we use slices of 446,080 spectral elements each, out of which 122,068 are 'outer' elements, i.e., elements in contact with MPI cut planes by at least one mesh point, and 324,012 elements are 'inner' elements. The ratio between outer and inner elements is thus approximately 27.5% to 72.5%. Each slice contains approximately 29.6 million unique grid points, i.e., 88.8 million degrees of freedom, corresponding to 3.6 GB (out of 4 GB) of memory used per GPU. Indeed, for this kind of experiments in geophysics, researchers are usually interested in studying the largest possible meshes in order to reach the highest possible resolution for seismic waves; we thus carry out all our experiments using 90% of the memory of each GPU. The largest possible problem size, using all 192 GPUs in the cluster, is thus 17 billion unknowns. All our measurements correspond to the duration (i.e., elapsed time) of 1000 time steps, knowing that each time step consists of exactly the same numerical operations (see Section 3.1). To get accurate measurements, the nodes used to perform the calculations are not shared with other users and each run is executed three times to ensure that the timings are reliable and to see if there are any fluctuations.

5.2. Weak scalability and speedup

Fig. 3 shows the average elapsed time per time step of the SEM algorithm for simulations on 4 to 192 GPUs (i.e. the whole machine), by steps of four GPUs. Weak scaling is almost perfect. The small fluctuations that we observe are on the order of 2–3%. We repeat this experiment using a single GPU per node, and consequently we can only go up to 96 GPUs, keeping the load per GPU constant. Fluctuations are now entirely removed, which shows that they are caused by the sharing of the PCI-Express bus in each half-S1070 Tesla. However, the total duration of the calculation is on average only 3% faster when the PCI-Express buses are not shared, which means that the sharing of the PCI-Express bus implied by the hardware structure of the Tesla S1070 is not a significant bottleneck. This clearly shows that overlapping of non-blocking MPI communications and of PCI-Express bus transfers by calculations on the GPUs is excellent in our application.

To measure the acceleration factor (speedup), we repeat the weak scaling experiment with two different configurations on the CPUs. In the first one, we assign each mesh slice of 3.6 GB to a CPU core, and we assign two mesh slices to each CPU. In the second, we cut each mesh slice in two halves and assign four of these smaller slices to the four cores of each CPU. This second set of experiments requires only half of the compute nodes because a larger amount of memory is available on each full CPU than on each GPU card. Fig. 4 shows the weak scaling measurements that we obtain. Fluctuations are larger in the case of CPUs (because elapsed time is longer), but the relative amount of noise in the measurements is similar to

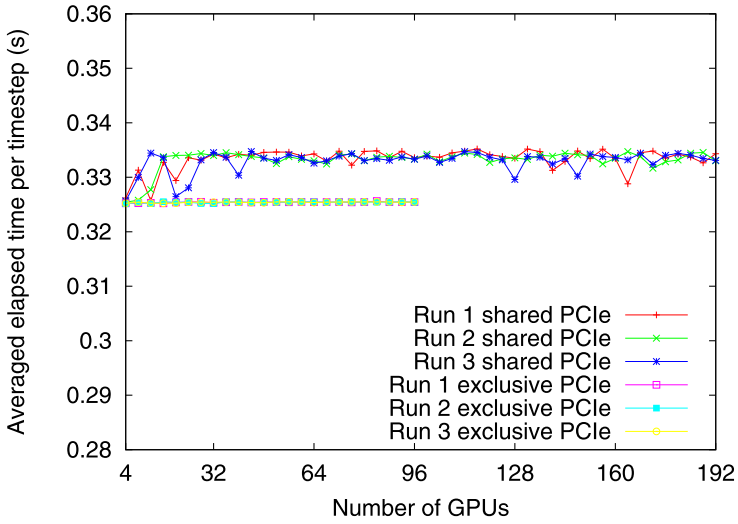

Fig. 3. Weak scalability measured using the GPUs, with or without sharing the PCI-Express bus.

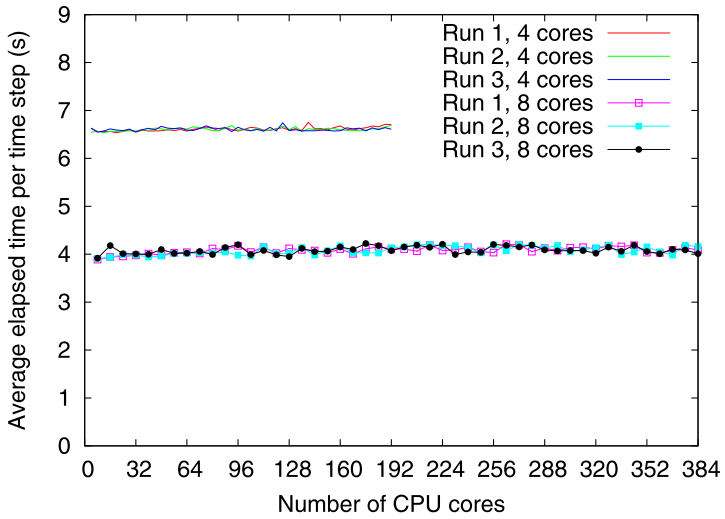

Fig. 4. Weak scalability obtained using only CPU cores. Let us note that the two configurations solve the same problem and that the size of the problem per full CPU is also the same (and is the same as the size per GPU of Fig. 3).

that for the case of calculations performed purely on GPUs. The configuration that uses four CPU cores per node to compute four mesh slices of half size is 1.6 times faster than that using only two cores with full mesh slices. We do not observe an ideal speedup factor of two because of resource sharing between the processor cores.

When we combine the measurements of Fig. 3 and those of Fig. 4, we can deduce an average speedup of 12.9 of a S1070 Tesla GPU compared to four Nehalem CPU cores for our application, and of a factor of approximately 20.6 compared to the case in which we use only two cores of each Nehalem CPU. These factors are both meaningful because when we divide the size of the mesh slices by two when using eight CPU cores per Nehalem node we change the ratio between the number of outer and inner elements and we also change the communication topology as well as the size and the number of MPI messages.

5.3. Adding fluid/solid coupling based on hybrid programming

A missing component in the above tests if we wish to study the propagation of Sdiff shear waves in the layer of the Earth called D″ is that these waves travel and are diffracted along the interface between the base of the Earth's mantle, which is solid, and the top of the outer core of the Earth, which is fluid. Thus, in our spectral-element technique, it is necessary to add coupling along this curved fluid/solid interface, keeping in mind that the solid part is discretized based upon the displacement vector whereas the fluid part is discretized based upon a displacement potential; it is thus necessary to impose the fluid/solid matching conditions between the two regions explicitly because they are discretized differently.

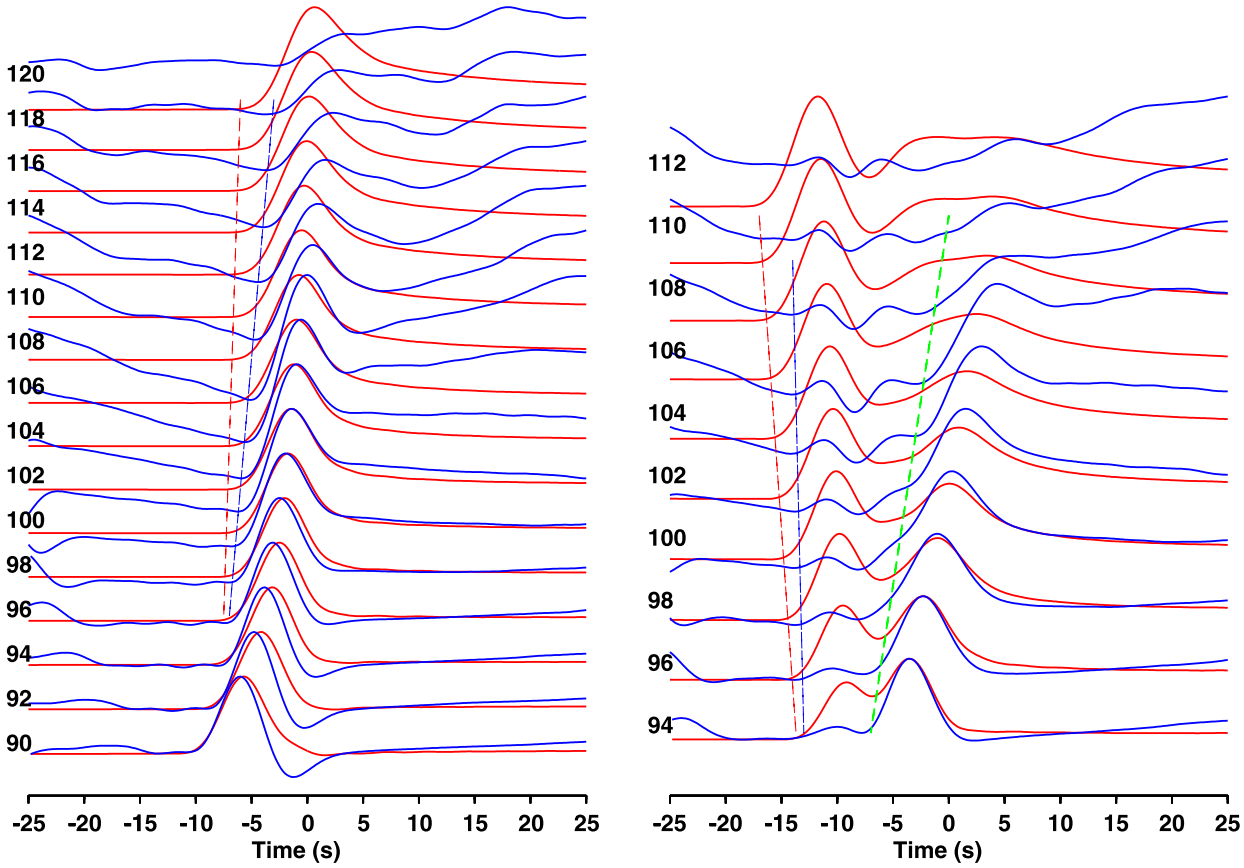

Fig. 5. Effect of a difference in terms of seismic properties in the layer located at the base of the Earth's mantle on the propagation of seismic waves in this layer (difference of 3% of these properties between the figure on the left and the figure on the right). In spite of the fact that we use all the 192 GPUs of the 'Titane' machine, because we can use "only" $192 \times 4 = 768$ GB of memory we see that there remains a relatively high level of numerical noise in our simulations (in particular on the blue curves) because the resolution of the mesh is not yet high enough. Thus, higher-resolution calculations will be performed on a very large cluster of CPUs and presented in Figs. 6 and 7.

To do that, we use a property of hybrid computation models which is that we can try to carry out part of the calculations on the CPU cores of the compute nodes while the rest of the calculations is carried out on the GPUs. In practice however such a model is difficult to implement efficiently because it requires one to implement transfers of arrays between GPUs and CPUs and also the other way around at each time step, and in an explicit time-stepping algorithm such as ours such transfers are difficult to perform in an asynchronous way to overlap them with calculation; they often rather need to be implemented in synchronous, i.e. blocking, mode. Thus when we add such coupling between the fluid and the solid, we observe a reduction of a factor of approximately two of the very high speedup measured in Section 5.2 and in Fig. 4. However, that could probably be improved in future versions of the code because so far we have not carefully optimized the implementation of this coupling integral.

An example of a simulation result for a fluid–solid model of the Earth is given in Fig. 5, in which we can clearly see the splitting of the shear waves that we wish to analyze; but owing to the fact that we can "only" use $192 \times 4 = 768$ GB of memory, we see that there remains a relatively high level of numerical noise in our simulations because the resolution of the mesh is not yet high enough. In order to perform larger calculations, the only current option is thus to switch back to calculations on more traditional clusters of CPUs, using several thousand CPU cores in order to have access to more memory. That will enable us to confirm these physical effects by using a finer mesh and thus removing most of the numerical noise, knowing that it is typically necessary to use a total of 5 or 6 terabytes of memory for that, which is currently accessible only on traditional clusters of CPUs. That will also enable us to remove spurious edge effects by modeling the whole Earth (i.e., the whole sphere) rather than only one chunk.

6. Modeling SHdiff and SVdiff waves at very high resolution on a large cluster of CPUs

The largest calculations that we can currently carry out for parametric studies (i.e. by comparing the results obtained for various models) can typically reach a minimum seismic period of 5 seconds [43], i.e. a frequency of one fifth of a Hertz, for the whole 3D elastic Earth. In this second part of our study, we wish that our simulations be precise at least until the same minimum period of 5 seconds. We thus mesh the whole Earth using 639,995,904 hexahedral spectral elements and use polynomial basis functions of degree 4 to discretize the wavefield inside each spectral element. This corresponds to roughly 42.2 billion grid points in the entire mesh and 115.1 billion degrees of freedom that need to be computed at each

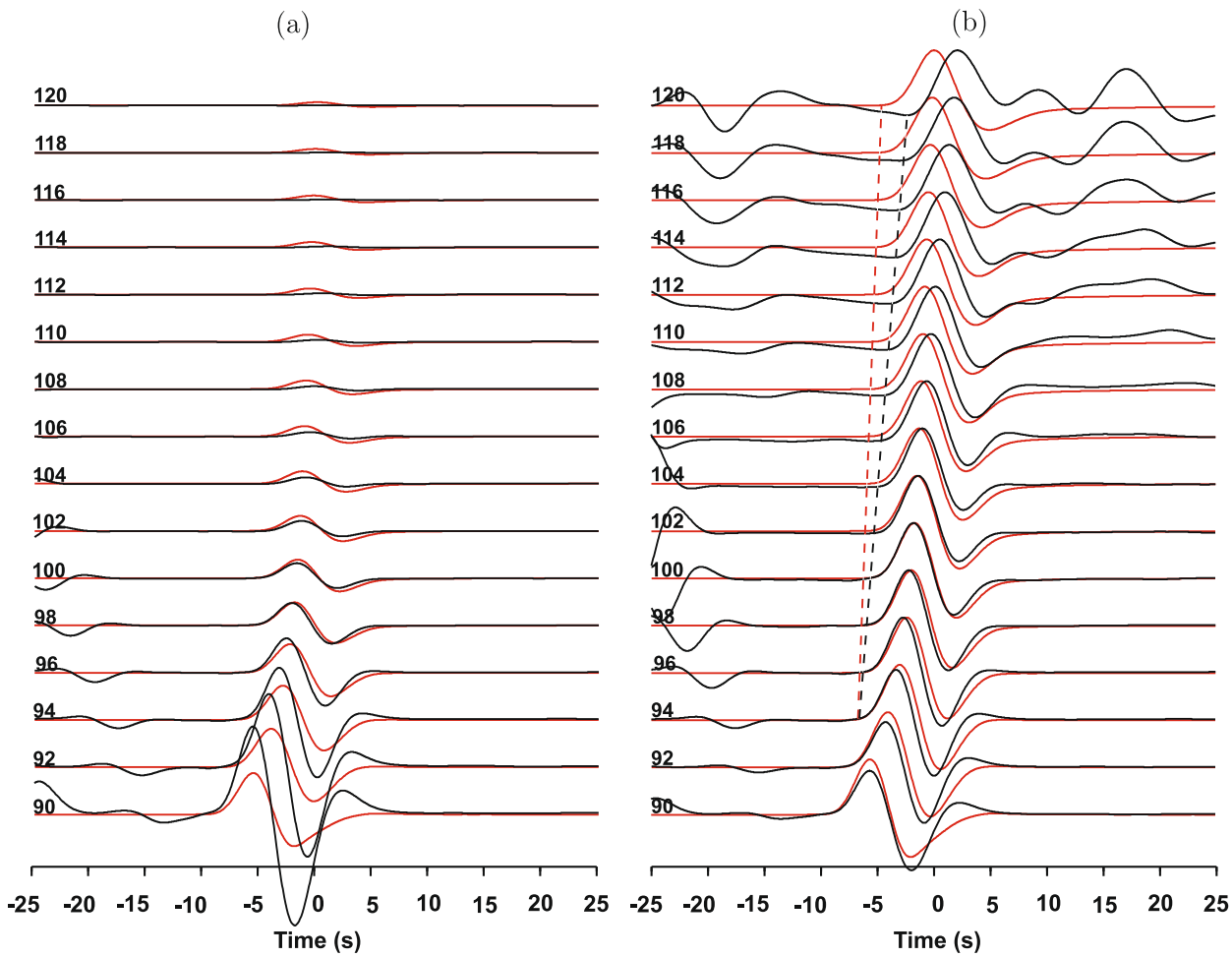

Fig. 6. Synthetic seismograms of the SH (red) and SV (black) components of the velocity vector along the equator of the Earth for geological model IASP91: with true amplitudes (a) and individually-normalized (self-normalized) amplitudes (b). The numbers on the left represent epicentral distance in degrees. Arrival times are shown with a reduction slowness of 8.3 s/°. The red and black dotted lines indicate the arrivals of the SHdiff and SVdiff diffracted shear waves, respectively. They represent the splitting of the SV and SH seismic shear waves that we clearly observe in our calculations for such an isotropic model.

iteration of the time loop. We carry out the calculations in parallel on 6144 processor cores of the 'Jade' supercomputer of CINES/GENCI in Montpellier, France, by dividing the mesh into 6144 parts having the same number of elements.

The simulations require 0.9 gigabyte of memory per processor core, i.e., a total of 5.5 terabytes. To simulate 33 minutes of seismograms (i.e. curves showing the variations of the components of the velocity vector with time at a given point in space) we calculate 47,200 steps of a duration of 42 milliseconds each, which requires approximately 22 hours of elapsed time for each calculation carried out on the 6144 processor cores used in parallel. The seismic source that we employ for the earthquake is located at the surface of the Earth at latitude 0° and longitude 0° and is a Gaussian function in time $e^{-t^2/6.5}$. We analyze the properties of the diffracted wavefields for some three-dimensional models with spherical symmetry, sometimes called 1-D models. The thickness of the D″ layer is 149 km. We analyzed many models (not all presented here) with perturbations of ±3% and ±1% of the speed of the S and P seismic waves in D″, respectively. The dominant period of the amplitude spectrum of our synthetic seismograms of the components of the velocity vector is $\pi\sqrt{6.5} = 8$ s, and they have significant energy up to a minimum period of approximately 5 seconds.

Figs. 6a and 7a show the synthetic seismograms for the standard model and the model with higher seismic wave speed using true amplitudes, and Figs. 6b and 7b show the same recordings but with individually-normalized amplitudes. In the standard model, the amplitude decay of the SVdiff wave with epicentral distance is large compared to that of SHdiff, and at a distance of 105° the amplitude of SHdiff becomes several times larger than that of SVdiff (Fig. 6a). However, with normalized amplitudes, one clearly observes SVdiff until a distance of 120° (Fig. 6b). The normalized seismograms of Fig. 6b show a delay of SVdiff compared to SHdiff that increases with distance from 0.0 s at a distance of 94° up to 2.4 s at a distance of 120°. The relative difference in slowness is ∼0.1 s/°. If we compare this Fig. 6b with Fig. 5, we note the positive impact of having increased the size of the mesh and thus the resolution of calculations owing to the fact that we can use more memory: the level of numerical noise present in calculations has significantly decreased, and as a result the new physical phenomenon that was guessed in Fig. 5 can be confirmed and observed with higher accuracy in Fig. 6b.

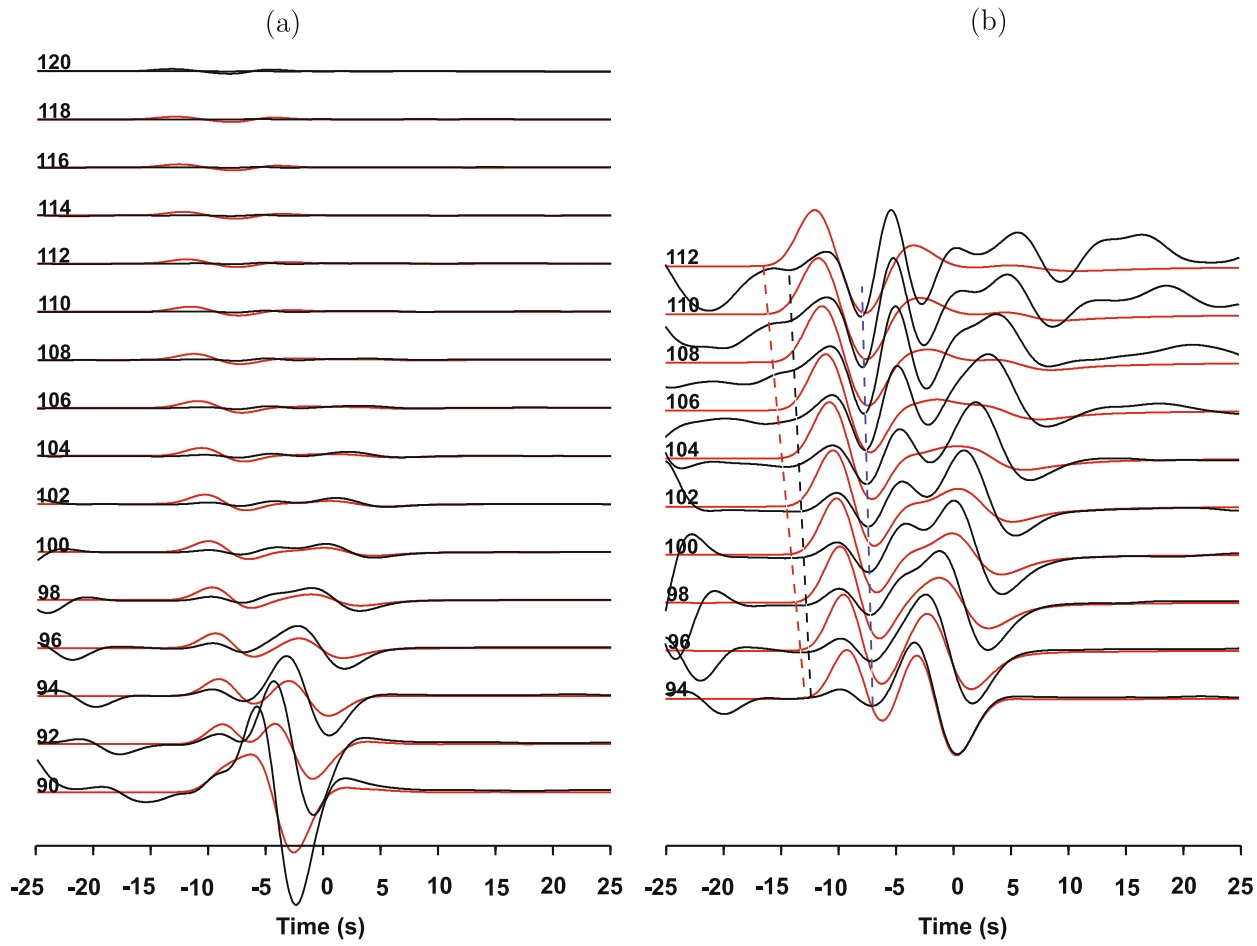

Fig. 7. Same as Fig. 6 but for a D″ geological layer having a higher wave speed. The SH and SV waves are represented in red and black, respectively.

7. Conclusions and future work

We have shown that even in an isotropic model of the Earth the SVdiff seismic shear wave can arrive later than the SHdiff wave with a delay of a few seconds (Fig. 6). We also observe this effect in 1-D models with a D″ geological layer having a higher wave speed (Fig. 7). These effects can be mistaken for effects of seismic anisotropy, which can lead to incorrect physical interpretations, because in the geophysical community the splitting of SV and SH waves is frequently used as an indication of the presence of seismic anisotropy.

In order to study these phenomena, we were led to develop a hybrid multiGPUs and CPUs version of our modeling software package called SPECFEM3D for Earth models having both fluid and solid layers. We showed the excellent weak scalability of this finite-element code on a cluster of 192 GPUs and obtained speedup factors of more than one order of magnitude compared to the same algorithm ran on a cluster of traditional CPUs. To efficiently overlap communications by calculation, we had to resort to non-blocking message passing.

In future work we wish to reimplement our approach using OpenCL to increase the range of hardware on which our code will be able to run, in particular to include AMD/ATI graphics cards as well as multicore processors. Another interesting option would be to use the HMPP parallelization directives [46]. The new features of the NVIDIA FERMI graphics cards should also be studied.

Our SPECFEM3D software package can be downloaded open source from www.geodynamics.org.

References

[1] M.E. Wysession, T. Lay, J. Revenaugh, Q. Williams, E.J. Garnero, R. Jeanloz, L.H. Kellogg, The D″ discontinuity and its implications, in: M. Gurnis, M.E. Wysession, E. Knittle, B.A. Buffett (Eds.), The Core–Mantle Boundary Region, American Geophysical Union, Washington, DC, USA, 1998, pp. 273–298.
[2] B. Romanowicz, Using seismic waves to image Earth's internal structure, Nature 451 (2008) 266–268.
[3] M. Panning, B. Romanowicz, Inferences on flow at the base of Earth's mantle based on seismic anisotropy, Science 303 (2004) 351–353.
[4] Paulius Micikevicius, 3D finite-difference computation on GPUs using CUDA, in: GPGPU-2: Proceedings of the 2nd Workshop on General Purpose Processing on Graphics Processing Units, Washington, DC, USA, March 2009, pp. 79–84.
[5] Rached Abdelkhalek, Henri Calandra, Olivier Coulaud, Jean Roman, Guillaume Latu, Fast seismic modeling and reverse time migration on a GPU cluster, in: Waleed W. Smari, John P. McIntire (Eds.), High Performance Computing & Simulation 2009, Leipzig, Germany, June 2009, pp. 36–44, http://hal.inria.fr/docs/00/40/39/33/PDF/hpcs.pdf.

[6] Dimitri Komatitsch, David Michéa, Gordon Erlebacher, Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using CUDA, J. Parallel Distributed Comput. 69 (5) (2009) 451–460.
[7] Dimitri Komatitsch, Gordon Erlebacher, Dominik Göddeke, David Michéa, High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster, J. Comput. Phys. 229 (20) (2010) 7692–7714.
[8] David Michéa, Dimitri Komatitsch, Accelerating a 3D finite-difference wave propagation code using GPU graphics cards, Geophys. J. Int. 182 (1) (2010) 389–402.
[9] Zhe Fan, Feng Qiu, Arie E. Kaufman, Suzanne Yoakum-Stover, GPU cluster for high performance computing, in: SC '04: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, November 2004, p. 47.
[10] Dominik Göddeke, Robert Strzodka, Jamaludin Mohd-Yusof, Patrick McCormick, Sven H.M. Buijssen, Matthias Grajewski, Stefan Turek, Exploring weak scalability for FEM calculations on a GPU-enhanced cluster, Parallel Comput. 33 (10–11) (2007) 685–699.
[11] James C. Phillips, John E. Stone, Klaus Schulten, Adapting a message-driven parallel application to GPU-accelerated clusters, in: SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, November 2008, pp. 1–9.
[12] Everett H. Phillips, Yao Zhang, Roger L. Davis, John D. Owens, Rapid aerodynamic performance prediction on a cluster of graphics processing units, in: Proceedings of the 47th AIAA Aerospace Sciences Meeting, January 2009, pp. 1–11.
[13] R. Vai, J.M. Castillo-Covarrubias, F.J. Sánchez-Sesma, D. Komatitsch, J.P. Vilotte, Elastic wave propagation in an irregularly layered medium, Soil Dyn. Earthquake Eng. 18 (1) (1999) 11–18.
[14] Jeroen Tromp, Dimitri Komatitsch, Qinya Liu, Spectral-element and adjoint methods in seismology, Commun. Comput. Phys. 3 (1) (2008) 1–32.
[15] P. Moczo, J. Robertsson, L. Eisner, The finite-difference time-domain method for modeling of seismic wave propagation, in: Ru-Shan Wu, Valérie Maupin (Eds.), Advances in Wave Propagation in Heterogeneous Media, in: Advances in Geophysics, vol. 48, Elsevier–Academic Press, London, UK, 2007, pp. 421–516 (Chapter 8).
[16] B. Lombard, J. Piraux, C. Gélis, J. Virieux, Free and smooth boundaries in 2-D finite-difference schemes for transient elastic waves, Geophys. J. Int. 172 (1) (2008) 252–261.
[17] Kasper van Wijk, Dimitri Komatitsch, John A. Scales, Jeroen Tromp, Analysis of strong scattering at the micro-scale, J. Acoust. Soc. Am. 115 (3) (2004) 1006–1011.
[18] S. Chevrot, N. Favier, D. Komatitsch, Shear wave splitting in three-dimensional anisotropic media, Geophys. J. Int. 159 (2) (2004) 711–720.
[19] J.M. Carcione, P.J. Wang, A Chebyshev collocation method for the wave equation in generalized coordinates, Comp. Fluid Dyn. J. 2 (1993) 269–290.
[20] D. Komatitsch, F. Coutel, P. Mora, Tensorial formulation of the wave equation for modelling curved interfaces, Geophys. J. Int. 127 (1) (1996) 156–168.
[21] C. Bernardi, Y. Maday, A.T. Patera, A new nonconforming approach to domain decomposition: the Mortar element method, in: H. Brezis, J.L. Lions (Eds.), Nonlinear Partial Differential Equations and Their Applications, in: Séminaires du Collège de France, Pitman, Paris, 1994, pp. 13–51.
[22] D.A. Kopriva, S.L. Woodruff, M.Y. Hussaini, Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method, Int. J. Numer. Meth. Eng. 53 (1) (2002) 105–122.
[23] E. Chaljub, Y. Capdeville, J.P. Vilotte, Solving elastodynamics in a fluid–solid heterogeneous sphere: a parallel spectral-element approximation on non-conforming grids, J. Comput. Phys. 187 (2) (2003) 457–491.
[24] D.A. Kopriva, Metric identities and the discontinuous spectral element method on curvilinear meshes, J. Sci. Comput. 26 (3) (2006) 301–327.
[25] Lucas C. Wilcox, Georg Stadler, Carsten Burstedde, Omar Ghattas, A high-order discontinuous Galerkin method for wave propagation through coupled elastic-acoustic media, J. Comput. Phys. 229 (24) (2010) 9373–9396.
[26] W.H. Reed, T.R. Hill, Triangular mesh methods for the neutron transport equation, Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, USA, 1973.
[27] Richard S. Falk, Gerard R. Richter, Explicit finite element methods for symmetric hyperbolic equations, SIAM J. Numer. Anal. 36 (3) (1999) 935–952.
[28] Fang Q. Hu, M.Y. Hussaini, Patrick Rasetarinera, An analysis of the discontinuous Galerkin method for wave propagation problems, J. Comput. Phys. 151 (2) (1999) 921–946.
[29] B. Rivière, M.F. Wheeler, Discontinuous finite element methods for acoustic and elastic wave problems, Contemp. Math. 329 (2003) 271–282.
[30] Peter Monk, Gerard R. Richter, A discontinuous Galerkin method for linear symmetric hyperbolic systems in inhomogeneous media, J. Sci. Comput. 22–23 (1–3) (2005) 443–477.
[31] Marcus J. Grote, Anna Schneebeli, Dominik Schötzau, Discontinuous Galerkin finite element method for the wave equation, SIAM J. Numer. Anal. 44 (6) (2006) 2408–2431.
[32] Marc Bernacki, Stéphane Lanteri, Serge Piperno, Time-domain parallel simulation of heterogeneous wave propagation on unstructured grids using explicit, nondiffusive, discontinuous Galerkin methods, J. Comput. Acoust. 14 (1) (2006) 57–81.
[33] M. Dumbser, M. Käser, An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes—II. The three-dimensional isotropic case, Geophys. J. Int. 167 (1) (2006) 319–336.
[34] S.P. Oliveira, G. Seriani, Effect of element distortion on the numerical dispersion of spectral element methods, Commun. Comput. Phys. 9 (4) (2011) 937–958.
[35] L. Brillouin, Tensors in Mechanics and Elasticity, 3rd edition, Academic Press, New York, USA, 1964.
[36] Roland Martin, Dimitri Komatitsch, Abdelaâziz Ezziani, An unsplit convolutional perfectly matched layer improved at grazing incidence for seismic wave equation in poroelastic media, Geophysics 73 (4) (2008) T51–T61.
[37] Roland Martin, Dimitri Komatitsch, Stephen D. Gedney, A variational formulation of a stabilized unsplit convolutional perfectly matched layer for the isotropic or anisotropic seismic wave equation, Comput. Model. Eng. Sci. 37 (3) (2008) 274–304.
[38] G. Seriani, E. Priolo, A spectral element method for acoustic wave simulation in heterogeneous media, Finite Elem. Anal. Des. 16 (1994) 337–348.
[39] Jonás D. De Basabe, Mrinal K. Sen, Grid dispersion and stability criteria of some common finite-element methods for acoustic and elastic wave equations, Geophysics 72 (6) (2007) T81–T95.
[40] Laura Carrington, Dimitri Komatitsch, Michael Laurenzano, Mustafa Tikir, David Michéa, Nicolas Le Goff, Allan Snavely, Jeroen Tromp, High-frequency simulations of global seismic wave propagation using SPECFEM3D_GLOBE on 62 thousand processor cores, in: Proceedings of the ACM/IEEE Supercomputing SC '2008 Conference, 2008, pp. 1–11 (Article #60, Gordon Bell Prize finalist article).
[41] Thomas J.R. Hughes, The Finite Element Method, Linear Static and Dynamic Finite Element Analysis, Prentice–Hall International, Englewood Cliffs, New Jersey, USA, 1987.
[42] Tarje Nissen-Meyer, Alexandre Fournier, F.A. Dahlen, A 2-D spectral-element method for computing spherical-earth seismograms – II. Waves in solid–fluid media, Geophys. J. Int. 174 (2008) 873–888.
[43] D. Komatitsch, L.P. Vinnik, S. Chevrot, SHdiff/SVdiff splitting in an isotropic Earth, J. Geophys. Res. 115 (B7) (2010) B07312.
[44] NVIDIA Corporation, NVIDIA CUDA Programming Guide version 2.3, Santa Clara, California, USA, July 2009, 139 pp.
[45] K.T. Danielson, R.R. Namburu, Nonlinear dynamic finite element analysis on parallel computers using Fortran90 and MPI, Adv. Eng. Software 29 (3–6) (1998) 179–186.
[46] R. Dolbeau, S. Bihan, F. Bodin, HMPP: A hybrid multi-core parallel programming environment, in: Proceedings of the Workshop on General Purpose Processing on Graphics Processing Units (GPGPU '2007), Boston, Massachusetts, USA, October 2007, pp. 1–5.