Performance and scalability of finite-difference and finite-element wave-propagation modeling on Intel's Xeon Phi

Elena Zhebel, Shell Global Solutions International B.V., Sara Minisini, Shell Global Solutions International B.V., Alexey Kononov, Source Contracting, Wim Mulder, Shell Global Solutions International B.V. and Delft University of Technology*

SUMMARY

With the rapid developments in parallel compute architectures, algorithms for seismic modeling and imaging need to be reconsidered in terms of parallelization. The aim of this paper is to compare scalability of seismic modeling algorithms: finite differences, continuous mass-lumped finite elements and discontinuous Galerkin finite elements. The performance for these methods is considered for a given accuracy. The experiments were performed on an Intel Sandy Bridge dual 8-core machine and on Intel's 61-core Xeon Phi, which is based on the Many Integrated Core architecture. The codes ran without any modifications. On the Sandy Bridge, the scalability is similar for all methods. On the Xeon Phi, the finite elements outperform finite differences on larger number of cores in terms of scalability.

INTRODUCTION

High-performance computing develops rapidly and new technologies keep appearing on the market. A few years ago, a new generation of graphics processing units (GPUs) appeared, offering tera-FLOPs performance on a single card. The GPU was originally designed to accelerate the manipulation of images in a frame buffer that was mapped to an output display. Later on, dedicated GPUs found use as general-purpose coprocessors suited for parallelizable applications. In 2012, Intel released its Xeon Phi coprocessor based on the Many Integrated Core (MIC) architecture.

With the changes in compute architectures, the algorithms for seismic modeling and imaging need to be reconsidered in terms of parallelization and scalability. Scalability refers to the ability of an algorithm to sustain increasing performance on an increasing number of cores.

The finite-difference method has become the workhorse for time-domain modelling of the wave equation with applications in acquisition optimization, development and testing of seismic processing algorithms, reverse-time migration and full waveform inversion. Its advantages are the relative ease of coding and parallelization, the use of high-order spatial discretization schemes and explicit time stepping, and its computational speed. The common opinion and experience is that finite differences perform well on rectangular domains with smooth velocity variations. However, in case of an irregular free surface or sharp contrasts in the medium properties, they lose their accuracy when using a Cartesian coordinate system. If the interface does not follow the grid, the staircasing effect generates first-order errors. Because the solution is continuous but not differentiable across an impedance contrast, a local second-order error will be incurred as well.

Finite-element methods have some advantages over finite differences because they can easily handle geometric or property discontinuities by using unstructured meshes and spatial local refinement. For instance, meshes can be used that consist of tetrahedral elements having their size scale with the local velocity. In this way the resulting meshes have less elements while maintaining the same number of points per wavelength in the computational domain. Finite elements that follow sharp interfaces do not suffer from a loss of accuracy. Moreover, they offer flexibility in mixing of discretization orders, mixing element geometries and deploying hybrid discretizations. The choice of a suitable time discretization scheme enables explicit time stepping. If the standard finite-element approach is used, a large sparse linear system of equations has to be solved at each time step, which has a negative impact on performance. There are a number of finite-element techniques that avoid the inversion of the large sparse matrix, for example, spectral elements, discontinuous Galerkin finite elements and continuous mass-lumped finite-elements. The last two will be considered here.

Several authors compared the various methods (Fornberg, 1987; Moczo et al., 2011; Chaljub et al., 2007; Wang et al., 2010; Pasquetti and Rapetti, 2004; De Basabe and Sen, 2007, e.g.). Zhebel et al. (2012a) considered the continuous mass-lumped and the discontinuous Galerkin finite elements in terms of accuracy, stability and computational cost. Numerical experiments on 3-D problems showed that both methods have similar stability conditions and require a comparable computational time to obtain a result with a given accuracy, assuming that the stiffness and mass matrices are pre-assembled. Here, we will recompute them on the fly to save storage. Another paper (Zhebel et al., 2013) compares the continuous mass-lumped finite elements to the finite-difference method. Zhebel et al. (2012b) cover the stability analysis.

The goal of the present paper is to compare finite-difference and finite-element algorithms in terms of scalability on many-core architectures. The next section reviews the methods. Then, we provide detail of the compute architecture, followed by the results of experiments with the three methods on different architectures.

METHODS

We consider the 3-D constant-density acoustic wave equation

$$\frac{1}{c}\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} - \frac{\partial^2 u}{\partial z^2} = s, \qquad (1)$$

DOI http://dx.doi.org/10.1190/segam2013-0861.1

on a bounded domain $\Omega \subset \mathbb{R}^3$, where $u(t,x,y,z)$ is the pressure wave field, $c(x,y,z)$ is the velocity of the medium and $s(t,x,y,z)$ is the source function, $x,y,z \in \Omega$. If the domain Ω has topography, the free surface can have reflecting (zero pressure) boundary conditions. Absorbing boundary conditions are imposed elsewhere.

By choosing a symmetric time-marching scheme with a time step Δt, for example, the leap-frog method, we obtain the following algebraic system of equations,

$$\mathbf{u}^{n+1} = 2\mathbf{u}^n - \mathbf{u}^{n-1} + \Delta t^2 \left(-\mathscr{L}\mathbf{u}^n + \mathbf{s}^n\right), \tag{2}$$

where $\mathscr{L}$ denotes a spatial discretization. The only unknown is the vector $\mathbf{u}^{n+1}$. The values of the solution at the previous time steps, n and $(n-1)$, are known. For the spatial discretization, we consider a finite-difference method (FD), symmetric interior penalty discontinuous Galerkin finite elements (DG) and continuous mass-lumped finite-elements (ML).

Finite differences

The problem 1 is discretized by central finite differences on a regular Cartesian grid in the rectangular domain represented by $\Omega = \{(x_i, y_j, z_k) \mid x_i = x_0 + i\Delta x,\ y_j = y_0 + j\Delta y,\ z_k = z_0 + k\Delta z,\ i,j,k \in \mathbb{N}\}$, where Δx, Δy and Δz are grid spacings in the x-, y- and z-directions, respectively, and $i = 0,1,\ldots,N_x-1$, $j = 0,1,\ldots,N_y-1$, $k = 0,1,\ldots,N_z-1$. With this, the acoustic wave equation in the second-order formulation at one point in space and time becomes

$$\begin{aligned} u^{n+1}_{i,j,k} = 2u^n_{i,j,k} - u^{n-1}_{i,j,k} + \Delta t^2 c^2_{i,j,k} \Big(&- [D_{xx}u^n]_{i,j,k} \\ &- [D_{yy}u^n]_{i,j,k} - [D_{zz}u^n]_{i,j,k} + s^n_{i,j,k} \Big), \end{aligned} \tag{3}$$

where D_{xx}, D_{yy} and D_{zz} denote the discretized second derivatives in the x-, y- and z-directions, respectively. Time is sampled at $t^n = T_{\min} + n\Delta t$, with $\Delta t = (T_{\max} - T_{\min})/N_T$ and $n = 0,\ldots,N_T$. The pressure field at the next time step, u^{n+1}, depends on the current n and the previous $(n-1)$ time steps.

The legacy finite-difference code makes use of distributed memory parallelization with MPI, also when used on a single many-core node. Grid partitioning is applied to divide the domain Ω into a number of subdomains. From the expression 3, it is clear that the computation of a single grid point (i,j,k) requires information from its neighbors in three directions. The subdomains require additional points for the calculations, called halos. At each time step, the points in the halos need to be exchanged to synchronize their values. Therefore, the parallelization for a distributed memory design requires extra memory and has a communication overhead.

Discontinuous Galerkin

We briefly review the formulation of the problem when using the symmetric interior penalty discontinuous Galerkin formulation. A detailed derivation can be found in Rivière (2008). We choose the symmetric and conservative form of the scheme. In this method, the solution is allowed to be discontinuous across element boundaries. Because in seismics, we want continuous solutions, a penalty is added to make the discontinuities small, of the order of the numerical discretization error.

To discretize the wave equation 1 with finite elements, we use the weak formulation

$$\begin{aligned} \int_\Omega \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}\, v\, d\Omega + \int_\Omega \nabla u\, \nabla v\, d\Omega - \int_\Gamma (\mathbf{n}\cdot\nabla u)\, v\, d\Omega = \\ \int_\Omega s\, v\, d\Omega, \end{aligned} \tag{4}$$

for all test functions v that are chosen as polynomials up to degree M. Here, $\mathbf{n}$ denotes the outward normal and Γ consists of internal and external boundaries of the domain Ω.

The domain Ω can be partitioned into tetrahedral elements τ. To have a certain number of points per wavelength, we require the diameter of the inscribed sphere to scale with the dominant wavelength, hence with the local velocity. The meshing approach has been described elsewhere (Kononov et al., 2012).

Since the solution is discontinuous across the internal boundaries, additional computations of so-called fluxes are needed to make the solution continuous. The term with the normal in 4 is a flux term that is given on the faces of elements. It consists of incoming and outgoing fluxes.

By discretizing the weak formulation 4 with leap-frog, we obtain for each element τ,

$$\mathbf{u}^{n+1}_\tau = 2\mathbf{u}^n_\tau - \mathbf{u}^{n-1}_\tau + \Delta t^2 \mathbf{M}^{-1}\left(-\mathbf{K}\mathbf{u}^n_\tau - \mathbf{F}^+\mathbf{u}^n_\tau - \mathbf{F}^-\mathbf{u}^n_{\tau^-} + \mathbf{s}^n_\tau\right),$$

where $\mathbf{M}$ and $\mathbf{K}$ are the local mass and stiffness matrix, respectively. Depending on the element degree M, the matrices will have different sizes, see Table 1. We also have the contribution of the fluxes. The term $\mathbf{F}^+$ denotes the sum of outgoing fluxes over the four faces in the given element and can be stored in one matrix. The second term, $\mathbf{F}^-$, contains incoming fluxes from the four neighboring elements, τ^-, and requires the storage of four matrices. Note that for elements which lie on the boundary, we need to store less outgoing flux matrices, since one or more neighbors are absent.

The implementation of DG uses shared memory parallelization with OpenMP on many-cores architectures. The parallelization is performed over the grid points in space. That means that, for each time step, the computations for the grid points (i,j,k) are performed in parallel. The information from the neighboring points is read from shared memory. To save memory, the matrices $\mathbf{M}$, $\mathbf{K}$ and $\mathbf{F}^+$ and the four $\mathbf{F}^-$ for each element are reassembled at each time step.

Continuous mass-lumped finite elements

The weak formulation of equation 1 for the continuous mass-lumped finite element method is given by equation 4, but now Γ only refers to the external boundaries, because the test functions and solution are assumed to be continuous. Discretizing equation 4 with mass-lumped finite elements in space and with second-order finite differences in time, we obtain a global fully algebraic system of the form

$$\mathbf{u}^{n+1} = 2\mathbf{u}^n - \mathbf{u}^{n-1} + \Delta t^2 \mathbf{M}^{-1}\left(-\mathbf{K}\mathbf{u}^n + \mathbf{s}^n\right). \tag{5}$$

The matrix $\mathbf{M}$ is diagonal, avoiding the need to solve a large sparse linear system of equations. Chin-Joe-Kong et al. (1999)

DOI http://dx.doi.org/10.1190/segam2013-0861.1

Table 1: The size of the matrices per element used in DG and ML is $N \times N$ and depends on the degree M.

degree M	N (DG)	N (ML)
1	4	4
2	10	23
3	20	50
4	35	–

describe the construction of the mass-lumped tetrahedral elements in detail.

The global discretized problem 5 can be reformulated element-by-element, improving parallelization of the time-stepping. In this case, depending on the element degree M, the matrices will have different sizes, see Table 1. The mass matrix was pre-assembled before the time stepping started, but the stiffness matrices were reassembled on the fly during the time stepping. The contribution per element was determined from the 6 pre-computed matrices of size $N \times N$ for the reference element and 6 scalar factors related to the local coordinate transformation of each element. In this way, only two vectors for the solution and one for the inverse mass matrix need to be stored, each of the size of the total number of degrees of freedom. The velocity, $c(x,y,z)$ can be absorbed in the inverse mass matrix. The shared-memory parallelization of ML uses OpenMP. During each time step, the elements τ are treated in parallel.

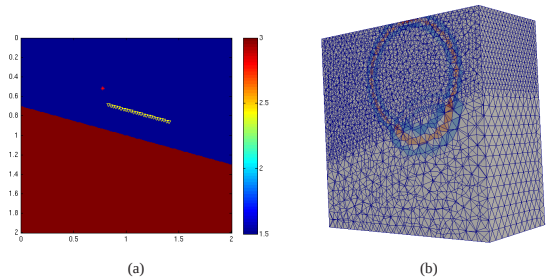
COMPUTE ARCHITECTURES

The codes were run on two architectures: Intel's standard x86 processor, Sandy Bridge, and a Xeon Phi based on Intel Many Integrated Core architecture.

The Sandy Bridge based compute nodes has two eight-core Intel ES-2670 2.6 GHz CPUs processors.

The Xeon Phi is a coprocessor based on Intel's Many Integrated Core architecture. The difference between a processor and coprocessor is that a processor does not require another compute device to be present in a working system. A coprocessor cannot be used as an independent device. It has to be connected to a processor (host). In our case, an Intel Xeon Phi card is connected to a host through a PCIe$\times$16 bus.

There are two ways to use the Intel Xeon Phi card. The first is to run the whole application on the Intel Xeon Phi, called its *native mode*. The second is to identify the parallel parts in the algorithm and run the program on the host, letting only those parallel parts run out-of-core on a MIC, also called *offload mode*. In our experiments, we only used the Xeon Phi in the first way by running the whole application on a single card. For this, the codes were recompiled with the latest Intel Compiler (ICS2013). The Intel Xeon Phi coprocessor runs Linux. Each card has his own IP address. Each Xeon Phi coprocessor combines 61 Intel cores on a single chip running with a frequency of 1.1 GHz. One core is responsible for the OS. The maximum number of threads per core is 4, allowing a total of 244. The Xeon Phi has 8 GB of DDR5 memory and a bandwidth of 320 GB/s.



(a) (b)

Figure 1: (a) Velocity model for the model problem. The source is denoted by a red star and receivers by yellow triangles. (b) Snapshot of the wave field.

The FD code, containing MPI calls, and the DG and ML code, parallelized with OpenMP pragmas, did not require any modification.

RESULTS

We consider seismic modelling 1, which represents the essential part of migration and inversion algorithms. The problem size has been chosen such that it would fit in the 8 GB of Xeon Phi memory and is kept the same also for the experiments on Sandy Bridge. This means we only consider *strong scaling*, as opposed to *weak scaling* where the problem size per core is kept constant.

The goal of the experiments is to compare finite-element and finite-difference algorithms in terms of their scalability on many-core architectures.

The model problem is the dipping interface shown in Figure 1(a). We consider a 3-D domain of size $(2\text{ km})^3$ with two halfspaces having velocities of 1.5 and 3.0 km/s. The interface runs from 0.7 to 1.3 km depth between 0 and 2 km in x-direction, so the dip angle is 16.7°. A shot is located at (779.7, 1000, 516.3) m, 350 m above the interface in the shallow low-velocity part of the model. The receivers are located 250 m above the interface and have offsets from 100 to 700 m with a 25-m interval, parallel to the interface in the down-dip direction of the source.

Figure 2 shows the errors as function of the total degrees of freedom N, using either the finite-difference scheme with a 4^{th}-order approximation of the second derivatives in each coordinate direction, the $M = 3$ type 2 continuous mass-lumped finite-element method, or the discontinuous Galerkin finite element of degree 3. The finite-element methods therefore also have 4^{th}-order spatial accuracy and a 2^{nd}-order temporal error. Note that 4^{th}-order accuracy of the finite-difference scheme is lost near discontinuities in the model. The finite-difference code was run on grids with 201^3, 251^3, 301^3, 351^3, 401^3 and 501^3 points. The continuous mass-lumped finite-element code and discontinuous Galerkin finite-element code ran on meshes with $294,508$, $567,071$, and $2,320,289$ elements, re-

DOI http://dx.doi.org/10.1190/segam2013-0861.1

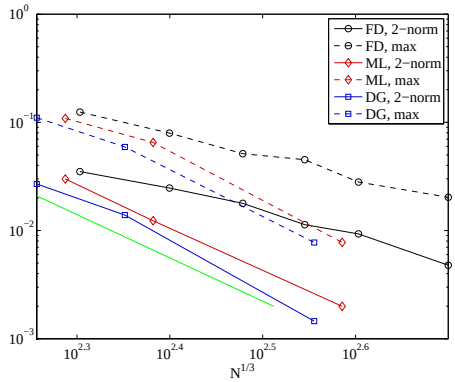


Figure 2: Errors for the model problem as a function of $1/h = N^{1/3}$, where N is the number of grid points for the finite-difference method (FD) or the number of degrees of freedom for the finite-element methods (ML and DG). The dashed lines represent the maximum norm, the drawn the 2-norm. The green line corresponds to a spatial fourth-order error. Second-order time stepping was used.

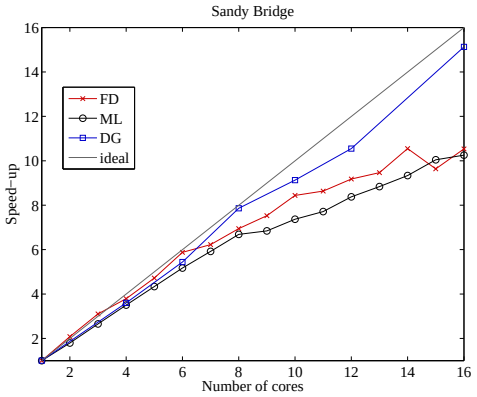


Figure 3: Speed-up on Intel's Sandy Bridge. The red line with crosses denotes FD, the black line with circles denotes ML and the blue line with squares denotes DG. The solid line represents ideal speed-up.

spectively. For finite differences and for finite elements, the direct wave was modelled on the same mesh but with a constant velocity of 1.5 km/s and then subtracted to only have the reflected/refracted event. It is obvious that, asymptotically, the ML and DG finite elements are more accurate than the finite differences because the mesh follows the interface.

For the experiments on the Intel Xeon Phi, we have selected the FD problem of size 301^3 and the mesh with 567,071 tetrahedra for the finite elements. For these problems, FD, ML and DG have a similar accuracy of about 3%. In total, ML had 28,353,550 and DG had 11,341,420 degrees of freedom, respectively. Figure 3 and 4 show the speed-up for the finite differences and the two finite-element methods on Intel's Sandy Bridge and Xeon Phi, respectively. The results on Sandy Bridge

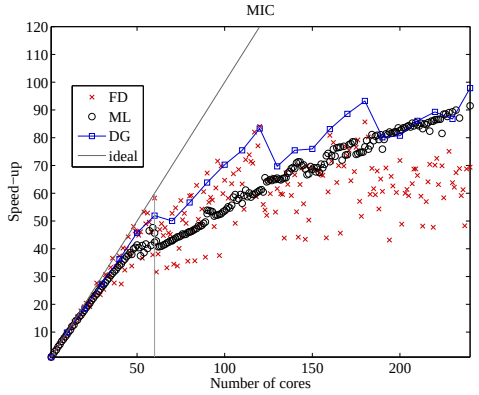


Figure 4: Speed-up on Intel's Xeon Phi. The red crosses denote FD, black circles denote ML and the blue line with squares denotes DG. The solid line presents ideal speed-up.

show that the speed-up measured on the nodes on one socket (up to 8 cores) is close to optimal for the three methods. To make sure that 1 process is running on the same core and is not dynamically relocated with cache losses, we have forced affinity. When using more than 8 cores, affinity is switched off. Since the DG code is more computational intensive that ML and FD, due to calculation of the fluxes, the speed-up is larger. The results on Xeon Phi show that the speed-up is almost linear with less than 50 threads. For a larger number of threads, the efficiency is reduced. Similar to the experiments with FD, we note a performance peak with 60, 120 and 180 threads, which are running at maximum capacity. This is especially obvious with DG. FD uses 1 thread for I/O and programme control, which is mostly idle during the computations. This one is excluded from the node count in the graphs. The code attempts to find a subdivision that favors cube-like structures, but the choice of the number of cores limits the options. The specific choice of subdivision impacts the performance. With the 301^3 points on the mesh, a subdivision on, for instance, 173+1 nodes into 173 subdomains in one coordinate direction does not make sense, so cases like that are absent from the graph.

CONCLUSIONS

We have compared the scalability of three seismic modeling algorithms: finite differences, continuous mass-lumped finite elements and discontinuous Galerkin finite elements. Their performance was considered for a given accuracy. Asymptotically, the ML and DG finite elements are more accurate than the finite differences when the finite-element mesh follows the interfaces between sharp contrasts in medium properties. The experiments were run on the Intel Sandy Bridge dual 8-core machine and Intel Xeon Phi based on the Many Integrated Core architecture. We observed that the finite elements have similar speed-up as finite differences, if a proper subdivision is chosen. Only beyond 120 cores, FD degrades. On Sandy Bridge, all methods show similar scalability.

DOI http://dx.doi.org/10.1190/segam2013-0861.1

http://dx.doi.org/10.1190/segam2013-0861.1

EDITED REFERENCES

Note: This reference list is a copy-edited version of the reference list submitted by the author. Reference lists for the 2013 SEG Technical Program Expanded Abstracts have been copy edited so that references provided with the online metadata for each paper will achieve a high degree of linking to cited sources that appear on the Web.

REFERENCES

Chaljub, E., D. Komatitsch, J. Vilotte, Y. Capdeville, B. Valette, and G. Festa, 2007, Spectral element analysis in seismology: *in* R.S. Wu and V. Maupin, eds., Advances in geophysics: Advances in wave propagation in heterogeneous media: Elsevier, 365–419.

Chin-Joe-Kong, M. J. S., W. A. Mulder, and M. van Veldhuizen, 1999, Higher-order triangular and tetrahedral finite elements with mass lumping for solving the wave equation: Journal of Engineering Mathematics, **35**, 405–426, http://dx.doi.org/10.1023/A:1004420829610.

De Basabe, J. D., and M. K. Sen, 2007, Grid dispersion and stability criteria of some common finite-element methods for acoustic and elastic wave equations: Geophysics, **72**, no. 6, T81–T95, http://dx.doi.org/10.1190/1.2785046.

Fornberg, B., 1987, The pseudospectral method: Comparisons with finite-differences for the elastic wave equation: Geophysics, **52**, 483–501, http://dx.doi.org/10.1190/1.1442319.

Kononov, A., S. Minisini, E. Zhebel, and W. A. Mulder, 2012, A 3D tetrahedral mesh generator for seismic problems: 74th Conference & Exhibition, EAGE, Extended Abstracts, 58922.

Moczo, P., J. Kristek, M. Galis, E. Chaljub, and V. Etienne, 2011, 3-D finite-difference, finite-element, discontinuous-Galerkin and spectral-element schemes analysed for their accuracy with respect to P-wave to S-wave speed ratio: Geophysical Journal International, **187**, 1645–1667, http://dx.doi.org/10.1111/j.1365-246X.2011.05221.x.

Pasquetti, R., and F. Rapetti, 2004, Spectral element methods on triangles and quadrilaterals: comparisons and applications: Journal of Computational Physics, **198**, 349–362, http://dx.doi.org/10.1016/j.jcp.2004.01.010.

Rivière, B., 2008, Discontinuous Galerkin methods for solving elliptic and parabolic equations: Theory and implementation: SIAM Frontiers in Mathematics 35.

Wang, X., W. W. Symes, and T. Warburton, 2010, Comparison of discontinuous Galerkin and finite difference methods for time domain acoustics: 80th Annual International Meeting, SEG, Expanded Abstracts, 3060–3065, http://dx.doi.org/10.1190/1.3513482.

Zhebel, E., S. Minisini, A. Kononov, and W. A. Mulder, 2012a, A comparison of continuous mass-lumped finite elements and finite differences for 3-D acoustics in the time domain: 74th Conference & Exhibition, EAGE, Extended Abstracts, 59474.

Zhebel, E., S. Minisini, A. Kononov, and W. A. Mulder, 2012b, On the time-stepping stability of continuous mass-lumped and discontinuous Galerkin finite elements for the 3D acoustic wave equation: Proceedings of the 6th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2012), CD-ROM, 1041–1051.

Zhebel, E., S. Minisini, A. Kononov, and W. A. Mulder, 2013, Personal communication.