Contents lists available at ScienceDirect

Journal of Computational Science

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

A two-scale method using a list of active sub-domains for a fully parallelized solution of wave equations

Marcus Noack[a,b,c,*]

[a] Kalkulo AS, P.O.Box 134, 1325 Lysaker, Norway
[b] Simula Research Laboratory, P.O. Box 134, 1325 Lysaker, Norway
[c] Department of Informatics, University of Oslo, Gaustadalleen 23 B, 0373 Oslo, Norway

ARTICLE INFO

Article history:
Received 24 March 2015
Received in revised form
14 September 2015
Accepted 16 October 2015
Available online 20 October 2015

Keywords:
GPU
Wave propagation
Two-scale methods

ABSTRACT

Wave form modeling is used in a vast number of applications. Therefore, different methods have been developed that exhibit different strengths and weaknesses in accuracy, stability and computational cost. The latter remains a problem for most applications. Parallel programming has had a large impact on wave field modeling since the solution of the wave equation can be divided into independent steps. The finite difference solution of the wave equation is particularly suitable for GPU acceleration; however, one problem is the rather limited global memory current GPUs are equipped with. For this reason, most large-scale applications require multiple GPUs to be employed. This paper proposes a method to optimally distribute the workload on different GPUs by avoiding devices that are running idle. This is done by using a list of active sub-domains so that a certain sub-domain is activated only if the amplitude inside the sub-domain exceeds a given threshold. During the computation, every GPU checks if the sub-domain needs to be active. If not, the GPU can be assigned to another sub-domain. The method was applied to synthetic examples to test the accuracy and the efficiency of the method. The results show that the method offers a more efficient utilization of multi-GPU computer architectures.

1. Introduction

Wave propagation plays a central role in many fields such as physics, environmental research and medical imaging to model acoustics, solid state physics, seismic imaging and cardiac modeling [1–5]. Different methods have been proposed for stable and accurate solutions of the wave equation, but the computational costs remain a problem for most applications [1].

The most commonly used methods to solve the wave equation can coarsely be divided into finite-element methods [6,7], including spectral element methods [8], and explicit and implicit finite difference methods [9,10]. The finite difference method is especially suitable for GPU acceleration because of the simple division into independent operations [11]. The solution in the current time step depends only on solutions of the previous time steps; hence, all nodes can be computed in parallel. The numerical solution of the wave equation is a memory demanding process since desired frequencies, model sizes and wave velocities lead to a large number of wavelength in the domain which imposes large grid sizes.

Two examples should be mentioned here. The first example is in the field of acoustics [1,2], where the model size rarely exceeds 100 m. Mehra et al. [1] presented the problem of a cathedral, where the sound velocity and the desire for a large range of frequencies requires a grid size of 22×10^6 nodes. Seismic imaging represents the second example, where the model dimensions are often in the order of a few hundred kilometers [12–15] in lateral and vertical extension. For minimal wave velocities of 300 m/s and frequencies of 10 Hz, the final grid size is around 16×10^9 nodes. For stability reasons it is not possible to choose the step size freely, which increases the computational cost further. Current GPUs have a global memory of 24 gigabytes maximum (K80 Tesla GPU); therefore, they can store around 6.4×10^9 single precision floating point numbers.

Since the resulting array is not the only data that has to be stored in the global memory of the GPU, the actual possible problem size is much smaller. Additionally, demands for accuracy and domain size are growing constantly and will always exceed the available resources. A solution to the problem is distributing the workload and data to different GPUs. The traditional approach is to assign one GPU to one specific sub-domain. For the entire computation, this assignment is static; therefore, most GPUs remain idle during the largest period of the computing time (see Fig. 1) [11,14,15].

* Correspondence to: Kalkulo AS, P.O.Box 134, 1325 Lysaker, Norway.
E-mail address: mcn@simula.no

http://dx.doi.org/10.1016/j.jocs.2015.10.008
1877-7503/

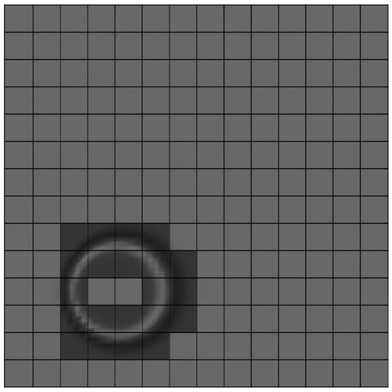

Fig. 1. A snapshot of a propagating wave. The domain is divided into 196 sub-domains. Only 21 (labeled dark) of 196 sub-domains need to be active to compute the next time step. Therefore, in the traditional approach 89 percent of the GPU devices are running idle in the computation of the current time step.

To address this issue, a list of active sub-domains can be used, as described in the following section.

The idea of considering exclusively the active part of a computation to save computing resources is not new. Di Gregorio et al. [16] employed the concept of active and inactive regions for wildfire susceptibility mapping (see also [17]). A rectangular bounding box distinguishes active from non-active regions and only active regions are computed. The bounding box method is also used in [18] for flow simulation on GPU computer architectures. Teodoro et al. [19] proposed a method for an efficient wavefront tracking that only uses active elements which form the wavefront. The advancements in this case enable an efficient image processing. Zhao et al. [20] used local grid refinement to restrict the computation to active regions of interest.

2. A list of active sub-domains

Gillberg et al. [21] introduced a list of active sub-domains for the simulation of geological folds by solving a static Hamilton-Jacobi equation. In the proposed method, the idea of Gillberg et al. [21] is adapted and used for the solution of the wave equation on multiple GPUs. The solution process for static Hamilton-Jacobi equation is very different from the solution process of the wave equation and the application of the idea in Gillberg et al. [21] is therefore neither on domain nor on sub-domain level straightforward. The main differences are the dimensionality of the problem, the solution process on sub-domain level, e.g., the required stencil shapes, and the desired employment of multi-GPU computer architecture.

The solution of a static Hamilton-Jacobi equation in [21] is found by a fast sweeping method on sub-domain level which sweeps until convergence to find the viscosity solution. In order to parallelize the solution process, a pyramid-shaped stencil is used to compute nodes of an entire plane independently. Different stencil shapes require different ghost-node configurations and, therefore, different communication schemes. Since the solution of the wave equation is not an iterative process that needs to converge to a minimum, the activation patterns for sub-domains and the solution process on sub-domain level are very different in Gillberg et al. [21] from the method proposed herein. Furthermore, the method in [21] is not developed to be used on a multi-GPU computer architecture; it is rather made to solve problems where strongly bent characteristic curves of the static Hamilton-Jacobi equation occur.

The adaption of the method in Gillberg et al. [21] included among other things the following: the establishment of an efficient communication between multiple GPUs, the adjustment of the activation pattern for sub-domains to the wave equation, implementing a different synchronization process, handling the fourth dimension and the employment of a different ghost-node configuration. However, the nomenclature is based on the one in Gillberg et al. [21] to simplify the comprehension for the reader.

The new proposed method distributes the workload and data efficiently on different GPUs by activating sub-domains in which the wave exhibits amplitudes larger than a given threshold and adding these sub-domains to a list. Only the sub-domains on this list are distributed over available GPUs. During the computation on the sub-domain level, each GPU checks if the computed sub-domain needs to be active and, therefore, locks the domain for computation if the wave has traveled out of the domain boundaries. Therefore, the effective problem size can be decreased by orders of magnitude depending on the problem itself and the computing capacities.

The proposed approach is able to decrease the demands of computing resources for a given desired computational performance since it avoids idle GPUs. In case of an abundant number of GPUs, the method allows to increase the number of sub-domains and hence improves the accuracy of the solution. More sub-domains also offer a more accurate isolation of active from inactive regions and, therefore, increase the performance (see Fig. 2).

The method was implemented for the acoustic wave equation but can simply be adapted to more complicated scenarios. It should also be mentioned that the main scope of the proposed method are multi-GPU computer architectures. However, every single GPU can be divided into independent parts to simulate a GPU cluster. This duality makes the method applicable on every parallel computer architecture and was used for all presented experiments. Furthermore, the method was developed for GPU computer architectures but the used principle leads to a speedup on all kinds of parallel computer architectures.

The remainder of the paper is organized as follows. The theory section gives an overview of the basic methods and the main principles of the algorithm, beginning with a summary of the mathematics and physics of the wave equation, followed by the description of the implementation. The method was applied to synthetic examples with different grid sizes.

3. Theory

The goal of the proposed method is to solve the wave equation, given by

$$\begin{aligned}\frac{\partial^2 u(\mathbf{x},t)}{\partial t^2} &= c(\mathbf{x})^2\nabla^2 u(\mathbf{x},t)\\ u(\mathbf{x},0) &= f(\mathbf{x})\\ \frac{\partial u(\mathbf{x},0)}{\partial t} &= 0,\end{aligned} \tag{1}$$

where $u(\mathbf{x})$ is a scalar function, $c(\mathbf{x})$ is the wave velocity at point $\mathbf{x}$ and ∇^2 is the Laplacian operator, on large grid sizes as efficient as possible. It has to be said that the proposed method is designed to solve all kinds of wave equations as efficient as possible. The acoustic wave equation is chosen here as an example for simplicity. To solve Eq. (1) with the help of an explicit finite difference scheme, it is mandatory to derive the finite difference approximation for the wave equation, given by

$$u_{ijk}^{t+1} = v_{ijk}^2 dt^2 \nabla^2 u + 2u_{ijk}^t - u_{ijk}^{t-1}. \tag{2}$$

Note that all nodes in the time step $t+1$ are independent of all other nodes in the same time step. All values depend only on the values of past time steps; thus, the solution process exhibits abundant parallelization. The computed wave field $u(\mathbf{x})^{t+1}$ in a certain time step will be the needed wave field $u(\mathbf{x})^t$ in the next time step and $u(\mathbf{x})^t$ will be the required $u(\mathbf{x})^{t-1}$ in the subsequent time step. Therefore, provided that the computation takes place only on one GPU, only

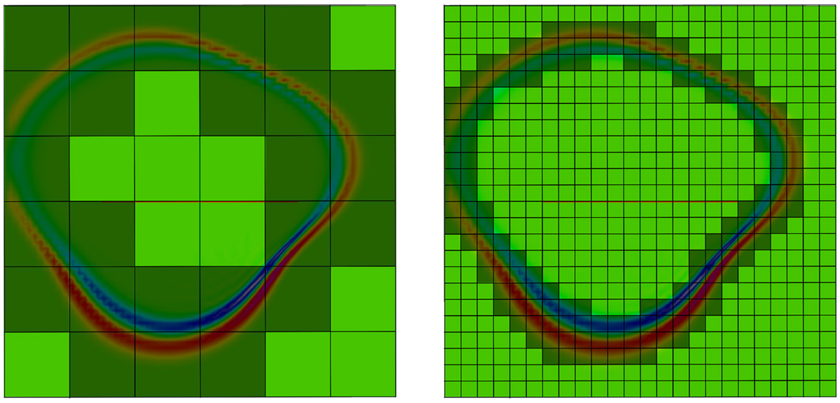

Fig. 2. Effective problem size compared to actual problem size for two different examples. The domain is divided into 36 sub-domains on the left side. The ratio of sub-domains to active sub-domains (labeled dark) is 1.44. The right side shows the same problem with a division in 576 sub-domains. The ratio of sub-domains to active sub-domains in this case is 3.81. The example shows that the computational costs benefit from more sub-domains since active regions can be separated from non-active regions more accurately. This principle has a limit: The sub-domains need to be large enough to fully utilize the GPU device. Therefore, in the case of an abundant number of GPUs the number of nodes in a sub-domain and hence the resolution can and should be enlarged to make use and utilize all available GPUs optimally.

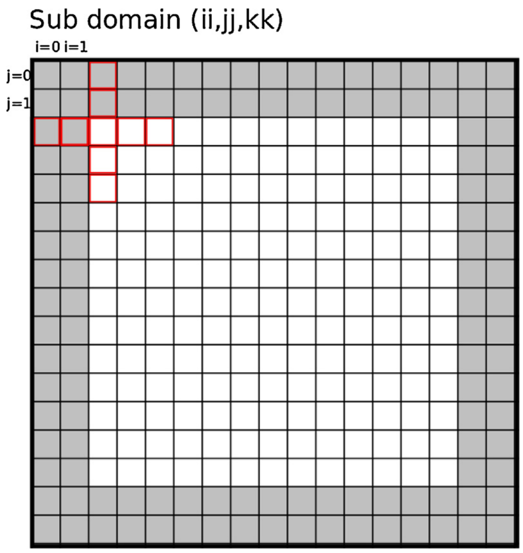

Fig. 3. The domain for the computation on one GPU or the CPU. The extent of the stencil for the Laplacian is shown in red. Two layers of nodes on each side cannot be computed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

data has to be copied to the device in the initialization step. This advantage is preserved in the case of multi-GPU computation. The algorithm checks if GPU devices and the data set on their global memory can be reused. If so, pointers are redirected one time step backward; therefore, no copying of new data is necessary as long as no new sub-domain is activated.

To guarantee the possibility for a correctly working communication between the sub-domains and to eliminate the need for communication during the computation, the incorporation of a sufficient amount of ghost-nodes around each sub-domain is necessary. Ghost-nodes are copies of nodes in adjacent domains (see Fig. 4) [21,22]. For accuracy reasons, in the proposed approach, a central finite difference scheme of fourth order was used for the second derivatives of the Laplacian

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} \approx \frac{-u_{i+2}+16u_{i+1}-30u_i+16u_{i-1}+u_{u_2}}{12\Delta x_i}. \tag{3}$$

Computing on the CPU or on one GPU, Eq. (3) requires the domain setting illustrated in Fig. 3. Two layers of nodes cannot be computed because of the spatial extent of the Laplacian. The communication between the sub-domains works with the same (sub-)domain setting. Therefore, the sub-domains for the multi-GPU computation are padded by two ghost-node layers at each side as illustrated in Fig. 4 [15]. The use of different stencil shapes for the computation of the Laplacian requires the adjustment of the ghost-node configuration.

Algorithm 1 shows the top level structure of the implementation of the method. It consists of a loop over all time steps. In every time step, the algorithm computes all tasks of the current schedule, synchronizes the sub-domains and builds a new schedule.

Algorithm 1. Pseudo code for the top level structure of the proposed algorithm. The first two time steps (0 and 1) must be given, therefore the loop starts with $i=2$.

```
Initialization;
BuildSchedule(List,CL);
for $i=2,\ldots,TimeSteps$ do
    ComputeSchedule(CL,List,NumbSched);
    SyncSd(CL);
    BuildSchedule(List,CL);
```

In the pseudo-code presented in this paper, the number of sub-domains is denoted by s_x,s_y and s_z, respectively. The size of a sub-domain is denoted by b_x, b_y and b_z, respectively. The wave field array and the velocity array are stored by sub-domain. Therefore, the velocity array is a four dimensional array. The first three dimensions describe the sub-domain (ii, jj, kk), and the last dimension represents a flattened array that describes the position in the sub-domain $((i*(b_y+4)*(b_z+4))+(j*(b_z+4))+(k))$, where the 4 originates from the ghost-node layers. The wave array is handled in a similar way with an additional time dimension. Thus, the wave array is five dimensional $(u(timestep, ii, jj, kk, pos.\ in\ sub\ domain))$. Since the last dimension for the sub-domain array is flattened, the treatment with CUDA is very straightforward.

3.1. Building the list of active sub-domains

In the initialization step, the wave field is defined for the first two time steps in accordance to Eq. (1). If a node gets a value assigned larger than a given threshold, then the corresponding sub-domain is activated. Activation means that the corresponding value in a Boolean array (CL in the pseudo-code) gets the

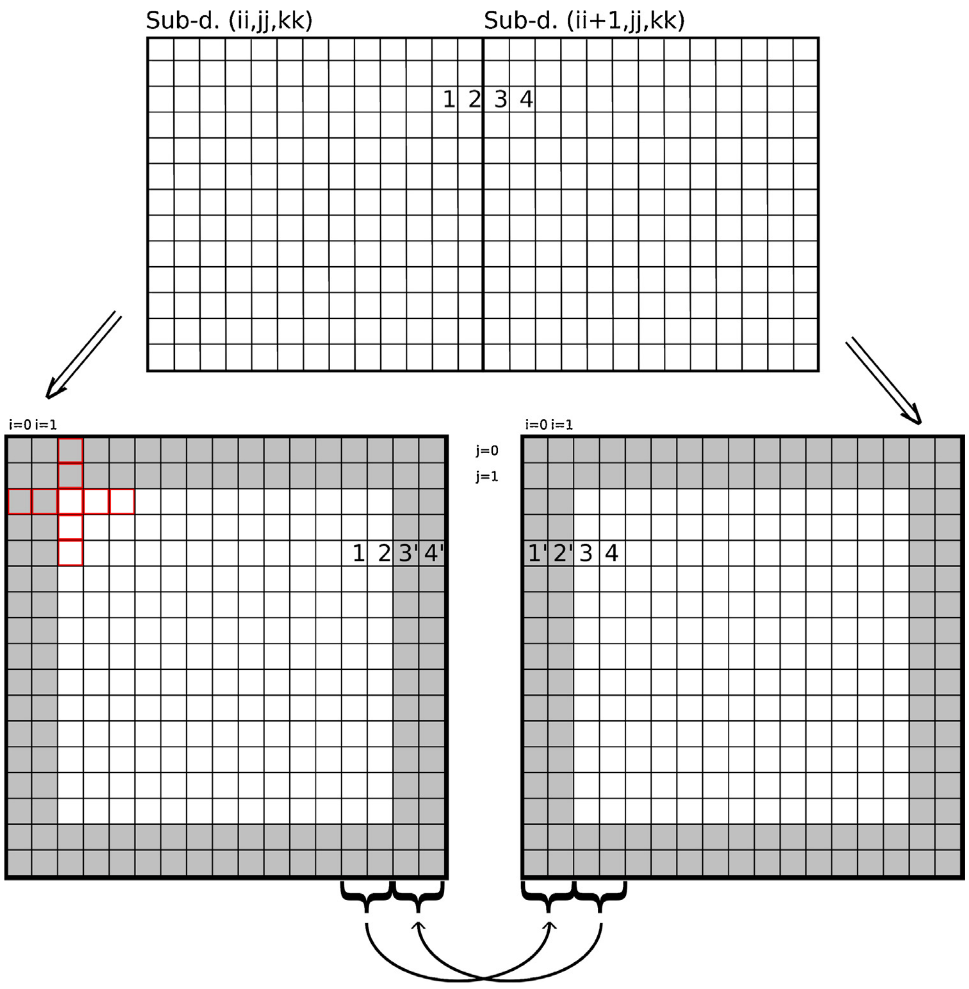

Fig. 4. For a correctly working communication, the sub-domains (top) are surrounded by two layers of ghost-nodes in our computation in accordance with the extend of the stencil for the Laplacian. Ghost-nodes (X') are copies of the corresponding node (X). During synchronization the values of the nodes 1 and 2 are first copied to the corresponding nodes in case their values are bigger than the given threshold. Afterwards the values of the nodes 3 and 4 are copied to the corresponding location in case of sufficiently large amplitudes. The sub-domain is activated if at least one node in the sub-domain gets a new value assigned.

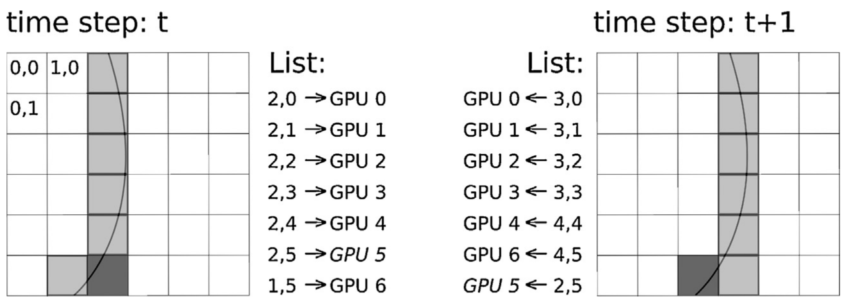

Fig. 5. Two time steps of a propagating wave. For simplicity, the wave is represented as a wave front. Active sub-domains are written in the list. At time step t seven sub-domains are active, hence the work is distributed on seven GPUs. In the time step $(t+1)$ the wave front moved out of some sub-domains into others. Hence, different sub-domains are active (in gray). Note that sub-domain (2,5) is active in both time steps (dark gray). Therefore, data can be reused and does not need to be copied. If the number of available GPUs is smaller than seven in this example the number of active sub-domains can be divided into groups of the size of the number of GPUs.

value "true" assigned. The coordinates of the sub-domains (denoted by *ii*, *jj*, *kk*) are written into a list. This list gives the method its name and can be seen as a schedule for the next computation. The sub-domains in the list are referred to as tasks. In each time step the available GPUs are optimally assigned to the tasks in the schedule, considering the least necessary data transfer (for more explanation see Fig. 5). Computing on the sub-domain level and synchronizing can change the activation of sub-domains; hence, it is important to build a new schedule after computation and synchronization.

Algorithm 2. BuildSchedule(LIST,CL).

```
NumbSched=0;
for $ii = 0; < s_x; ++ii$ do
    for $jj = 0; < s_y; ++jj$ do
        for $kk = 0; < s_z; ++kk$ do
            if CL[ii][jj][kk]== "true" then
                NumbSched=NumbSched++;
                LIST[NumbSched][0]=ii;
                LIST[NumbSched][1]=jj;
                LIST[NumbSched][2]=kk;
            return (NumbSched);
```

3.2. Computation of the schedule

After a list containing the schedule is built, every available GPU is assigned a task from the schedule, where one task equals one sub-domain. The corresponding sub-domains are copied to the different devices, where the next time step is computed in parallel. If a GPU is active a second time step in a row, data is not transferred again but reused to save computing time. During computation each GPU checks if at least one node in the sub-domain gets assigned an amplitude which is larger than a given threshold. If not, the corresponding GPU tells the host that the sub-domain may be deactivated. Since several sub-domains are computed simultaneously and the computation on the sub-domain level is in parallel, the algorithm exhibits a two-level parallelization.

Algorithm 3. ComputeSchedule(CL,List,NumbSched).

```
in parallel
for all tasks in schedule do
    select GPU device;
    Solve(CL,List,TaskInSchedule);
```

Algorithm 4. Solve(CL,List,TaskInSchedule).

```
allocation of memory;
cuda copy host to device;
DeviceFunction<<< $blocks, threads$ >>>();
cuda copy device to host;
if device function discovers no amplitude > Threshold then
    CL[ii][jj][kk]="false"
```

3.3. Synchronization and activation of sub-domains

After the computation of one time step, all sub-domains must be synchronized. For that, all ghost-nodes have to be copied to their corresponding position in the adjacent sub-domain. This process is taken take of by sweeps in positive and negative axial directions, one direction at a time, to avoid memory interference. A ghost-node is only copied to its corresponding position in the adjacent sub-domain if its value is larger than a given threshold. If a value of a node is copied to the adjacent sub-domain, this sub-domain is activated for the computation of the next time step. An if-condition makes sure that only sub-domains which were active in the last time step are synchronized to save computational costs.

Algorithm 5. SyncSd(CL).

```
/*Synchronization in positive x-direction*/
in parallel
for $ii = 0; ii < sx - 1; ++ii$ do
    for $jj = 0; jj < sy; ++jj$ do
        for $kk = 0; kk < sz; ++kk$ do
            if CL[ii][jj][kk] == "true" then
                for $i = bx; i < bx + 4; ++i$ do
                    for $j = 0; j < by + 4; ++j$ do
                        for $k = 0; k < bz + 4; ++k$ do
                            if $|u[timestep][ii][jj][kk][i, j, k]| >= Threshold$ then
                                $u[timestep][ii + 1][jj][kk][i - bx, j, k] = u[timestep][ii][jj][kk][i, j, k]$;
                                CL[ii+1][jj][kk] = "true";
```

4. Results

To prove the functionality of the proposed method, four key features were investigated. Firstly, to ensure that the accuracy of the traditional finite difference computation is preserved when applying the proposed method, resulting wave fields were compared. Secondly, computing times were measured to show that the list building step, which is additional work compared to the traditional method, only contributes a small amount of the overall computing time. Thirdly, overall computing times were compared. Finally, the ability of the new method to decrease the effective problem size is shown by means of a real life situation. The first three key features were investigated on the basis of two different experiments that are introduced in the following sections. The fourth key feature was investigated on the basis of one experiment which was created to resemble a real life seismological problem. The available computer architecture consists of two GeForce GTX 770 M GPUs. All experiments were designed to simulate a GPU cluster when necessary to obtain informative results by dividing one of the available GPUs into many processing units.

4.1. Experiment 1

Experiment 1 was designed to offer comprehensibility and clarity of the presented results. For experiment 1 a domain of $248 \times 248 \times 248$ nodes was divided into $2 \times 2 \times 2$ sub-domains of $124 \times 124 \times 124$ nodes. The velocity was chosen to be homogeneous in the entire domain. Accounting for the ghost-nodes the resulting problem size was $256 \times 256 \times 256$ nodes. The initial condition was chosen to be a narrow Gaussian function. Due to the small problem size, it is possible to map the entire domain on one of the available GPUs.

4.2. Experiment 2

Experiment 2 was designed to investigate the performance of the method based on a real-life example. For experiment 2, a domain of $308 \times 308 \times 308$ nodes was divided into $11 \times 11 \times 11$ sub-domains of $28 \times 28 \times 28$ nodes. The small sub-domain size makes it possible to simulate a computer architecture with 1331 GPUs on one of the available GPUs (not accounting for MPI communication). The velocity field was given by

$$v(\mathbf{x}) = 400 + (50 \times \sin(|x| \times 38)) \tag{4}$$

and is illustrated in Fig. 6. The chosen velocity field exhibits high frequencies and gradients of the velocity. It therefore represents a proper challenge for the proposed method. Accounting for the ghost-nodes, the resulting problem size was $352 \times 352 \times 352$ nodes. The initial condition was chosen to be a narrow Gaussian function.

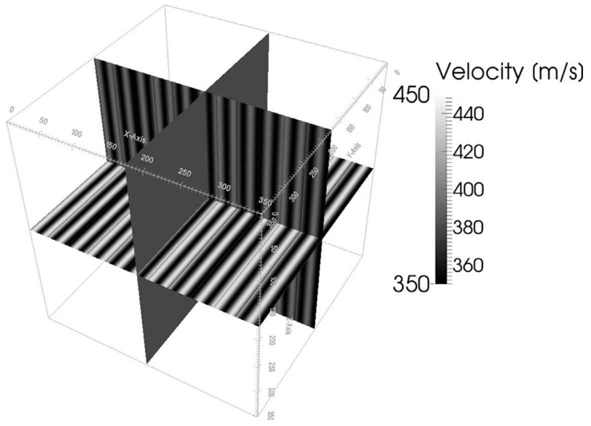

Fig. 6. The velocity used to assess the accuracy of the proposed method.

4.3. Experiment 3

Experiment 3 was designed to prove the validity of the main essence of the proposed method: saving effective problem size. For the experiment 3, a domain of $924 \times 924 \times 924$ nodes was divided into $33 \times 33 \times 33$ sub-domains of $28 \times 28 \times 28$ nodes. Accounting for the ghost-nodes the resulting problem size was $1056 \times 1056 \times 1056$ nodes. To make the result relevant for a real life application, the velocity field was chosen to represent a geological setting. The velocity model is shown in Fig. 7.

4.4. Comparison of solutions

Since sub-domains are activated only if the amplitude of an approaching wave is larger than a certain threshold, one has to make sure that the lost information does not degrade the final solution. Therefore, the solution of the acoustic wave equation computed on the CPU using the traditional method was compared to the solution obtained with the new proposed method. For an elaborated analysis of the numerical accuracy the L_1 and the L_2 norm, defined by

$$||u(\mathbf{x}, t)||_{L_1} = \frac{\sum_{ijk} |u^t_{ijk} - \hat{u}^t_{ijk}|}{N} \tag{5}$$

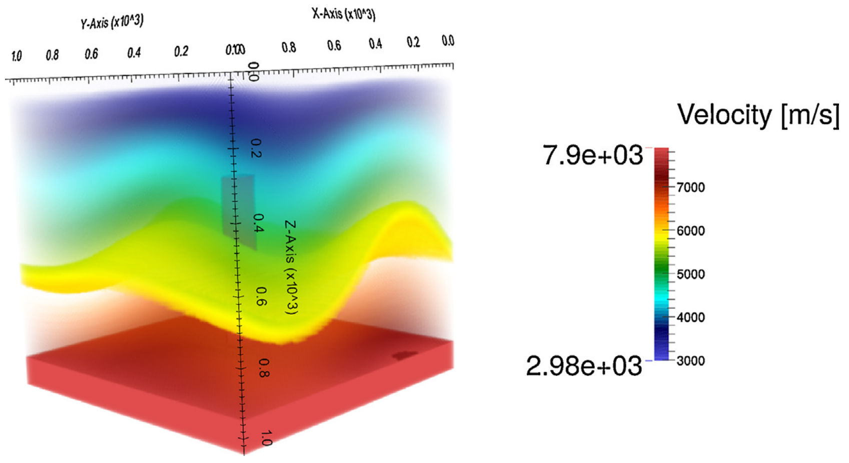

Fig. 7. The velocity model chosen for experiment 3. It is based on a real life geological setting.

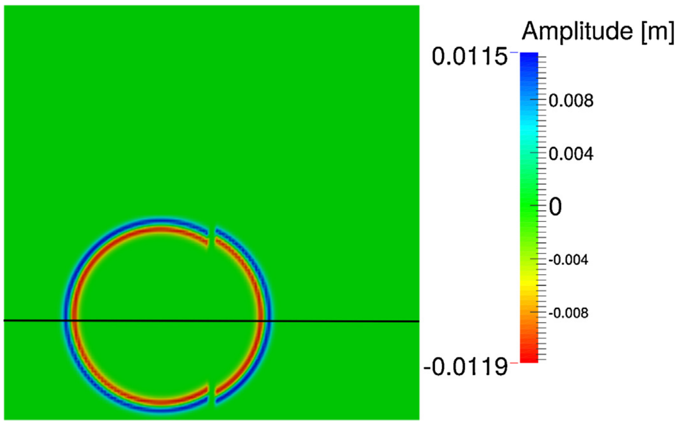

Fig. 8. The position and orientation of the cross section shown in Fig. 9. The solution is shown in the domain that includes ghost-nodes, hence an offset is visible. This offset is not a numerical error and does not affect the final solution.

and

$$||u(\mathbf{x},t)||_{L_2} = \frac{\sqrt{\sum_{ijk}(u^t_{ijk}-\hat{u}^t_{ijk})^2}}{N} \quad (6)$$

respectively are presented. u^t_{ijk} in Eqs. (5) and (6) represent the solution of the proposed method and $\hat{u}^t_{ijk}$ represents the solution computed on the CPU without division into sub-domains. At first, the solution of experiment 1 was compared with the solution on the CPU along a one-dimensional cross section (see Figs. 8 and 9). For this example, the threshold was chosen to be 0.001% of the amplitude of the initial condition. The L_1 and L_2 error norms for different thresholds are presented in Fig. 10. Next, experiment 2 was conducted and compared to the corresponding computation on the CPU using the traditional method. The L_1 and L_2 error norms of the solution of experiment 2 are presented for different thresholds in Fig. 11.

In order to determine the threshold, estimated amplitudes in the area of interest and numerical errors must be considered. For example, in a seismic scenario, the amplitude in the area of interest is important; there is no point in considering waves with an amplitude of 0.1 mm if the computation is used to assess the risk of earthquake damage to buildings. However, for a computation of many time steps in a domain which is divided into many sub-domains, smaller thresholds should be considered for accuracy reasons.

4.5. *Time measurement*

In the current implementation, the computation of one time step consists of the solution of the acoustic wave equation, a synchronization of all active sub-domains and the building of a new schedule. To establish the proposed method as a standard way to solve the wave equation on multi-GPU computer architectures, it must be proven that the additional list building step does not take the majority of the overall computing time. In the synchronization step, the values of the ghost-nodes are copied to adjacent sub-domains and hence to other GPUs. The synchronization step is a necessary step in the traditional approach too and does therefore not need to be justified. However, in the current implementation, this step is not simultaneous to the solution process on the GPU. It is therefore included in the following measurements. For experiment 1, the costs of synchronizing the sub-domains and building the new list amounts to 2% of the overall computational costs in the case of sequential synchronization. The synchronization in one direction can be a parallelized loop; thus, the synchronization and list building only takes about 0.5% of the overall computing time on a 4-core CPU machine (Intel Core i7-4800MQ CPU @ 2.70 GHz). The percentage of the computational costs of the list building and synchronization step compared to the computation mainly depends on the ratio between the ghost-nodes and the overall number of nodes. The current implementation includes a condition to ensure that only active sub-domains are synchronized, which lowers the computational costs and represents an advantage compared to the traditional approach where all sub-domains (and hence all GPUs) have to communicate during the entire computing time, independent whether there is information to exchange or not. As a worst case scenario for the proposed method, experiment 2 was conducted and the computing time of the list building and synchronization steps was measured. The small sub-domains result in a low ratio of overall nodes to ghost-nodes which maximizes the synchronization time. For experiment 2, the list building and synchronization steps needed 3.56% of the overall computing time using a sequential synchronization. In case of a parallelized synchronization on a 4-core CPU machine the list building and synchronization steps need below 1% of the overall computing time.

4.6. *Computing time*

The new proposed method reaches full potential on multi-GPU clusters when the number of GPUs equals the maximum number of active sub-domains during the computation. Here, since the mentioned GPU cluster was not available, the problem size of experiment 1 and 2 was chosen to simulate a GPU cluster which is able to communicate between GPUs in an instant.

Firstly, the computing time of experiment 1 is presented. The computation was firstly performed in the traditional way, meaning that the entire domain was mapped on one GPU without using sub-domains. This result was compared to the same computation on one GPU and two GPUs using the proposed method. As soon as the number of active sub-domains exceeds the number of available GPUs, the computations becomes partly sequential. One GPU computed 100 time steps in 14.7 s using the traditional method. The new proposed method employed on one GPU only needed 4.84 s, which makes the computation 3.02 times faster. The proposed method needed 4.61 s for the same computation when two GPUs were used, resulting in a speedup of 3.19. The speedup in this example is due to the fact that the effective problem size was reduced to $124 \times 124 \times 124$ nodes for the first 80 time steps before the wave front propagated into adjacent sub-domains. Experiment 1 showed the functionality for small numbers of sub-domains. For a more elaborated investigation of the computing times, experiment 2 was conducted and compared to the traditional method. For experiment 2, one of the GPUs was divided into 1331 processing units to simulate a cluster of GPUs. To make the statement clear, the conditions for the traditional method were optimized. As described, since the traditional computation takes place on one GPU there is no communication step. Even for these optimized conditions for the traditional method, the speedup is significant compared to the proposed method. One GPU computed 150 time steps in 36.62 s on the mentioned grid. The new proposed method used only maximal 120 active sub-domains and needed 7.88 s, which makes the computation 4.64 times faster. For 300 time steps, the same computation takes 58 s using the proposed method and 73 s using the traditional approach. The speedup in this case amounts to 1.26 times. A slice of the wave field is shown in Fig. 12. The computing times are summarized in Table 1.

4.7. *Saving computing resources*

Experiment 3 was conducted to show how efficient the algorithm is able to save computing resources in a real life situation.

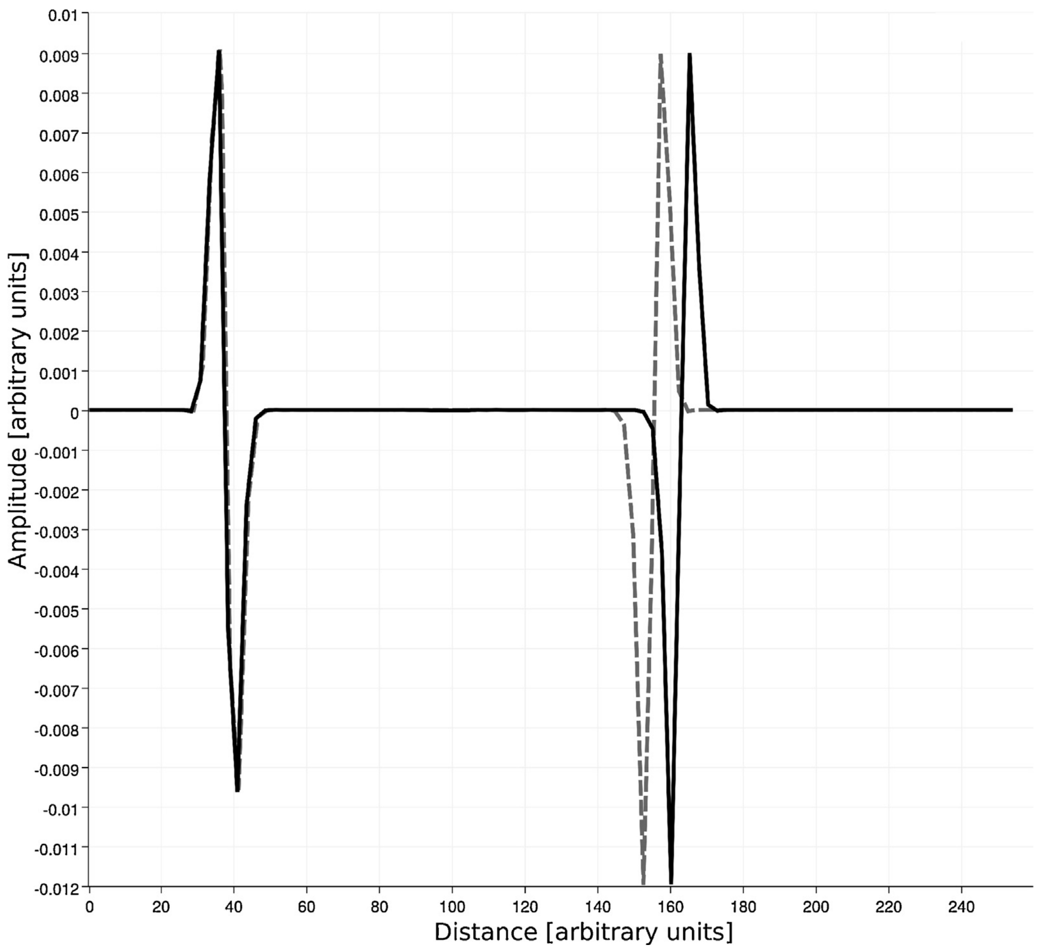

Fig. 9. The comparison of the solution of the acoustic wave equation on one GPU (solid line) with the solution with the proposed method (dashed line) along a cross section (Fig. 8). The wave form is similar. Note the occurrence of the phase shift because of the ghost-node layer which is purposely included. The phase shift is no numerical error and does not affect the final solution. The threshold for this example was chosen to be 0.001% of the amplitude of the initial condition.

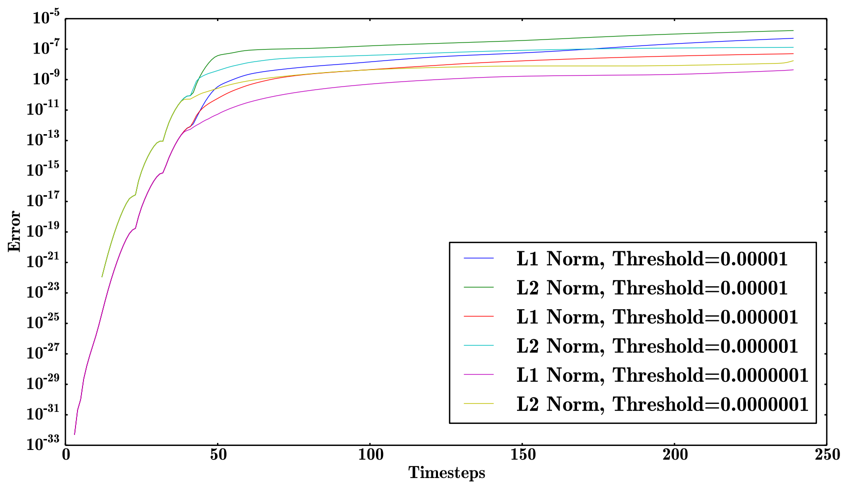

Fig. 10. Representation of error norms L_1 and L_2 of the solution for the homogeneous velocity field for different time steps. The first deflection marks the first transition of the wave front into an adjacent sub-domain. Note that the error reacts strongly to the reduction of the threshold.

2000 time steps were computed enabling the wave front to travel through all sub-domains. The number of active sub-domains in each time step for two different thresholds is shown in Fig. 13.

The maximum number of active sub-domains was 13,700 or 25,086 and on average 6563 or 11,232 sub-domains were active for the thresholds 10^{-4} or 10^{-5} respectively. To obtain a meaningful measure to compare the efficiency of the traditional and the proposed method, the number of overall computed nodes can be considered. Using the traditional approach, 229.76×10^{10} nodes were computed. Using the new proposed method, 58.88×10^{10} nodes were computed for a threshold of 10^{-4} and 106.24×10^{10}

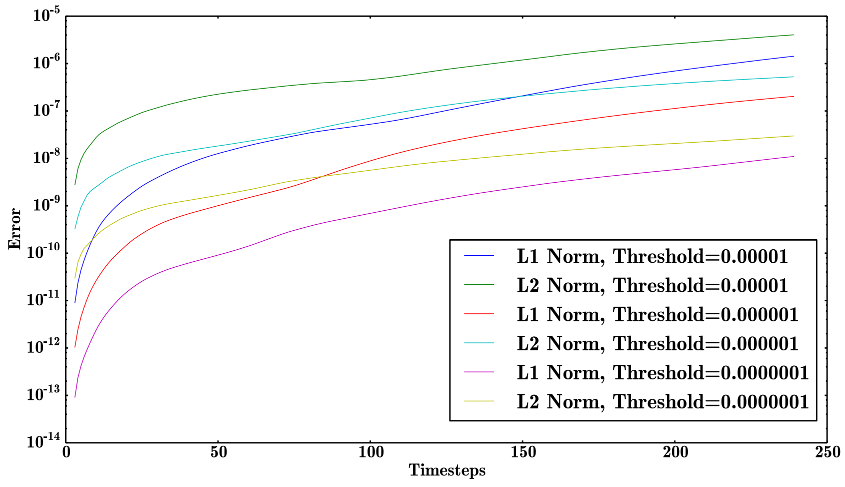

Fig. 11. Representation of error norms L_1 and L_2 of the solution for the velocity field shown in Fig. 6 for different time steps. Note once again that the error reacts strongly to the reduction of the threshold. The wave source in this experiment was located close to a sub-domain border, hence the first deflection is close to the origin.

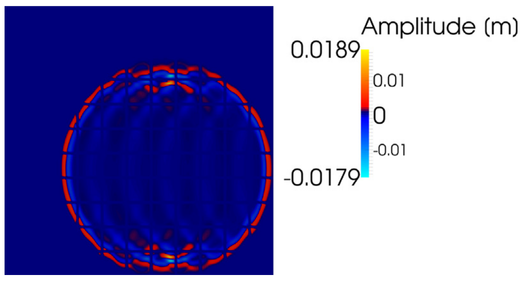

Fig. 12. A slice of the resulting wave field computed with the new proposed method. Note that reflections make it impossible to deactivate sub-domains downwind of the wave front for the given configuration.

Table 1
Computing time in seconds for different experiments and computer architectures.

Exp.	Trad. M. (s)	No. Sub-d.	Architecture	Comp. Time (s)	Speedup
1	14.7	8	1/2 GTX 770M	4.84/4.61	3.02/3.19
2	36.52	11^3	1 GTX 770M	7.88	4.64

Table 2
Number of computed nodes for the traditional method and two different thresholds, and the percentage of saved resources.

Threshold	Nodes Trad. method	Nodes Prop. method	Saving (%)
10^{-4}	229×10^{10}	58.88×10^{10}	74.3
10^{-5}	229×10^{10}	106.24×10^{10}	53.7

nodes for a threshold of 10^{-5}. These results indicate that, using the novel approach in experiment 3 74.4% (for a threshold of 10^{-4}) or 53.7% (for a threshold of 10^{-5}) of computing resources can be saved. The results are summarized in Table 2.

5. Discussion and perspective

The results section showed that the new proposed method computes the same result as the computation on one single GPU with a

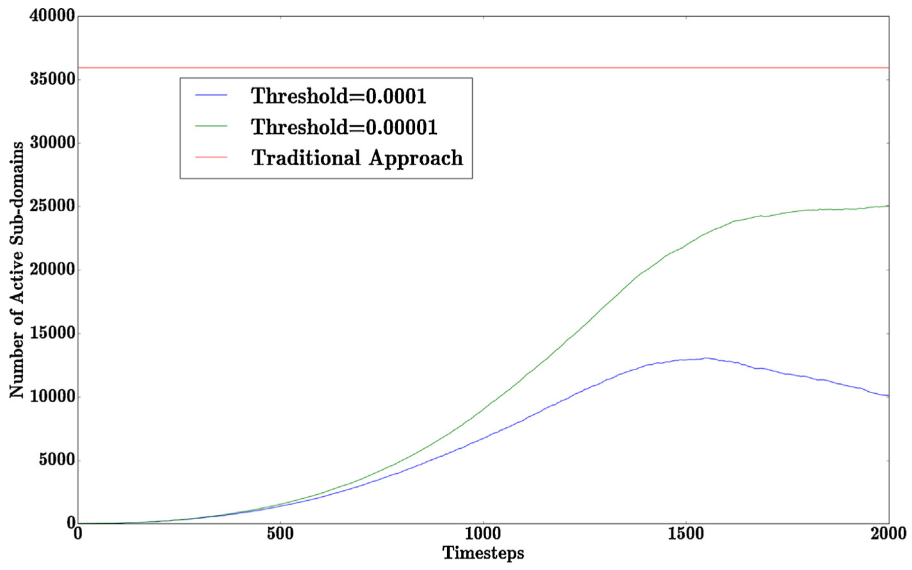

Fig. 13. The number of active sub-domains as a function of the time steps is shown for two different thresholds and the traditional approach. The maximum number of active sub-domains is 13,700 for a threshold of 10^{-4} and 25,086 for a threshold of 10^{-5}. The traditional method employs 35,937 sub-domains for the entire computing time. Note that the green graph approaches the maximum number of active sub-domains and the gradient approaches zero. Therefore, the number of new activated sub-domains per time step is close to the number of deactivated sub-domains in the same time step. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

significant improvement of computational efficiency. Fig. 9 shows that there is only a negligible difference in amplitude. The phase shift is included intentionally to show the effect of the ghost layers and can easily be removed. L_1 and L_2 error norms show how the error introduced by ignoring small values developed over time. The error increases strongly in the begging but reaches a stable value after some time. It was shown that the error strongly reacts to the reduction of the threshold size. Therefore, for large problems smaller thresholds should be chosen. In these scenarios, the proposed method maintains its superiority over the traditional method since more sub-domains mean a more accurate separation of inactive from active zones. Larger problem sizes also allow for a bigger ratio of inactive to active zones since commonly emerging wavelength are smaller compared to the problem size. In other words: the larger the model size compared to the emerging wave lengths, the higher the possibility for inactivating most of the model space especially when using a very time limited source term. This fact allows smaller thresholds when computing larger problems without loss of benefit. The comparison of the error norms of experiment 1 and experiment 2 also showed that the error increases only slightly for complex problems.

The beneficial effect of the method is obvious: regions where the amplitude of the wave is smaller than a certain threshold are not part of the computation and do not waste computing resources. This principle leads to a significant speedup even for an example that is not perfectly suited for the method. Instead of one GPU dealing with $256 \times 256 \times 256$ nodes the algorithm activates only one sub-domain in the beginning leading to a much smaller effective problem size. In later time steps, the adjacent sub-domains are activated. Since the number of sub-domains exceeds the number of GPUs the computation is partly sequential; however, the speedups of 3.02 times using one GPU and 3.19 times using two GPUs are still promising and in the expected range. For this example, a bounding box method would have yielded the same speedup because of the limited problem size and sub-domain number, which make it impossible to deactivate sub-domains behind the traveling wave. For more complex problems, sub-domains are deactivated as soon as the wave has traveled outside and the proposed method outperforms the bounding box method. In experiment 1, eight GPUs would not have been much faster since the activation of most sub-domains happens in the last 20 time steps. Hence, most of the time the GPUs would have been idle. Furthermore, the proposed method makes the division into eight sub-domains using one or two GPUs possible in the first place. The speedup is mainly due to the fact that the effective problem size is reduced by a factor of eight for a large part of the computation. The rest of the computation is subject to a partly sequential computation due to the chosen problem size and hardware. Therefore, the measured speedups are in a reasonable range.

Experiment 2 simulates a real life example computed on a GPU cluster equipped with 1331 GPUs. Each GPU can compute the solution of the wave equation on a grid of $28 \times 28 \times 28$ nodes. Since the problem size is manageable by one GPU the simulated cluster does not need to communicate when performing the traditional approach, therefore giving it an unrealistic advantage. During the computation using the traditional method, most of the simulated 1331 GPUs are waiting most of the time for their turn. On the other hand, the proposed method checks for active sub-domains and reduces the efficient problem size significantly to a maximum of 120 sub-domains in the first 150 time steps. The result is a 4.64 times faster computation. It has to be said, that the conducted experiment shows the traditional method at its best and the new proposed method at its worst since the high ratio of ghost-nodes to overall nodes maximizes the time for synchronization and list building steps. Even in this worst case scenario, the computing time for the list building and synchronization steps are small because only active sub-domains are synchronized with their neighbors and the synchronization can be performed in parallel. The same experiment conducted for 300 time steps showed a 1.26 times faster computation using the proposed method. The smaller speedup for 300 time steps is due to the special character of the velocity field. The high-frequency, periodic velocity field causes many reflections which make it impossible to inactivate sub-domains when using the given setting (see Fig. 12). In this case, a larger grid and more time steps would be beneficial since the amplitude of the reflected waves would decay below the threshold at some point. Experiment 3 proved the ability of the new method to save computing resources on the basis of a real life application. Instead of 35,947 active sub-domains used by the traditional method, the new algorithm only activated a maximum number of 13,700 or 25,086 sub-domains depending on the size of the threshold. On average 6563 or 11,232 sub-domains were active. The overall number of computed nodes showed that the saving of computer resources is significant for the chosen experiment for both thresholds.

The success of the method highly depends on problem specific parameters, like source definition, velocity model and problem size, and on the used computer architecture. However, all wave propagation algorithms can benefit from the proposed algorithm in the beginning of the wave propagation. When the active wave field is only small, all GPUs can be used for a higher resolution and hence higher accuracy of finite different approximations around the source. The proposed algorithm loses all its benefits as soon as a wave is active in all sub-domains. In this case the consumption of computing resources is the same as with the traditional method excluded the list building step. However, this scenario is rare in practice.

In the future, the sub-domains could be irregularly shaped and thus better isolating active from inactive zones. Furthermore, automatic tools that define sub-domains depending on wave activity and the number of available GPU devices could be very beneficial. Such a tool could divide the active regions into as many sub-domains as possible, resulting in higher resolution and/or computational performance. The goal is to optimally distribute computing resources only on active regions and not wasting them on regions in the domain where the wave exhibits negligible amplitudes.

6. Conclusion

A method for distributing the workload of solving the wave equation optimally on a multi-GPU computer architecture is proposed. The proposed algorithm can save computing resources by deactivating areas where the amplitude of the wave undergoes a defined threshold. The available computing resources are entirely utilized for regions where the wave is active; hence, no GPUs are running idle. Therefore, smaller clusters can perform equally well as larger ones. Using the proposed algorithm, one can divide the domain in more sub-domains than available GPU devices and still obtain good performance. In cases when enough GPUs are available, increasing the number of nodes (and thus the resolution of the solution) without losing computing time is possible. The proposed algorithm offers more efficient and accurate wave form modeling by optimizing the workload distribution on GPU clusters and has therefore a large potential impact on industry and research.

Acknowledgements

The presented work was funded by Kalkulo AS and the Research Council of Norway under grant 238346. The work has been conducted at Kalkulo AS, a subsidiary of Simula Research Laboratory.

I would like to thank Stuart Clark, Are Magnus Bruaset, Christian Tarrou and Xing Cai for beneficial comments and support.

References

[1] R. Mehra, N. Raghuvanshi, L. Savioja, M.C. Lin, D. Manocha, An efficient GPU-based time domain solver for the acoustic wave equation, Appl. Acoust. 73 (2) (2012) 83–94.

[2] D. Botteldooren, Finite-difference time-domain simulation of low-frequency room acoustic problems, J. Acoust. Soc. Am. 98 (6) (1995) 3302–3308.

[3] A.P. Sarvazyan, O.V. Rudenko, S.D. Swanson, J.B. Fowlkes, S.Y. Emelianov, Shear wave elasticity imaging: a new ultrasonic technology of medical diagnostics, Ultrasound Med. Biol. 24 (9) (1998) 1419–1435.

[4] J.F. Clœrbout, Imaging the Earth's Interior, 1982.

[5] C. Kittel, P. McEuen, Introduction to Solid State Physics, Wiley, New York, 1976.

[6] J. Lysmer, L.A. Drake, A finite element method for seismology, Methods Comput. Phys. 11 (1972) 181–216.

[7] T. Toshinawa, T. Ohmachi, Love-wave propagation in a three-dimensional sedimentary basin, Bull. Seismol. Soc. Am. 82 (4) (1992) 1661–1677.

[8] G. Seriani, E. Priolo, Spectral element method for acoustic wave simulation in heterogeneous media, Finite Elements Anal. Des. 16 (3) (1994) 337–348.

[9] H. Igel, P. Mora, B. Riollet, Anisotropic wave propagation through finite-difference grids, Geophysics 60 (4) (1995) 1203–1216.

[10] A. Fichtner, Full Seismic Waveform Modelling and Inversion, Springer, 2011.

[11] P. Micikevicius, 3D finite difference computation on GPUs using CUDA, in: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, ACM, 2009, pp. 79–84.

[12] S.P. Grand, R.D. van der Hilst, S. Widiyantoro, Global seismic tomography: a snapshot of convection in the Earth, GSA Today 7 (4) (1997) 1–7.

[13] S. French, V. Lekic, B. Romanowicz, Waveform tomography reveals channeled flow at the base of the oceanic asthenosphere, Science 342 (6155) (2013) 227–230.

[14] D. Komatitsch, G. Erlebacher, D. Göddeke, D. Michéa, High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster, J. Comput. Phys. 229 (20) (2010) 7692–7714.

[15] K.B. Olsen, R.J. Archuleta, J.R. Matarese, Three-dimensional simulation of a magnitude 7.75 earthquake, Science 270 (1995) 8.

[16] S. Di Gregorio, G. Filippone, W. Spataro, G.A. Trunfio, Accelerating wildfire susceptibility mapping through GPGPU, J. Parallel Distrib. Comput. 73 (8) (2013) 1183–1194.

[17] D. D'Ambrosio, S. Di Gregorio, G. Filippone, R. Rongo, W. Spataro, G.A. Trunfio, Fast assessment of wildfire spatial hazard with GPGPU, in: SIMULTECH, 2012, pp. 260–269.

[18] J. Zheng, X. An, M. Huang, GPU-based parallel algorithm for particle contact detection and its application in self-compacting concrete flow simulations, Comput. Struct. 112 (2012) 193–204.

[19] G. Teodoro, T. Pan, T.M. Kurc, J. Kong, L.A. Cooper, J.H. Saltz, Efficient irregular wavefront propagation algorithms on hybrid CPU–GPU machines, Parallel Comput. 39 (4) (2013) 189–211.

[20] Y. Zhao, F. Qiu, Z. Fan, A. Kaufman, Flow simulation with locally-refined LBM, in: Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, ACM, 2007, pp. 181–188.

[21] T. Gillberg, A.M. Bruaset, Ø. Hjelle, M. Sourouri, Parallel solutions of static Hamilton-Jacobi equations for simulations of geological folds, J. Math. Ind. 4 (1) (2014) 10.

[22] T. Gillberg, Fast and accurate front propagation for simulation of geological folds, Falculty of Mathematics and Natural Sciences, University of Oslo, 2013 (Ph.D. thesis).

Marcus Noack got his degree in geophysics from the Friedrich-Schiller-University in Jena. Currently working as researcher at Simula Research Laboratory in Oslo, he is able to pursue his interests in the theory of wave propagation and inversion. For the past two years he has been connecting his knowledge in theoretical and numerical geophysics to High Performance Computing to create efficient methods to model wave propagation.