Journal of Computational Science 11 (2015) 91–101

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/© 2015 The Author. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

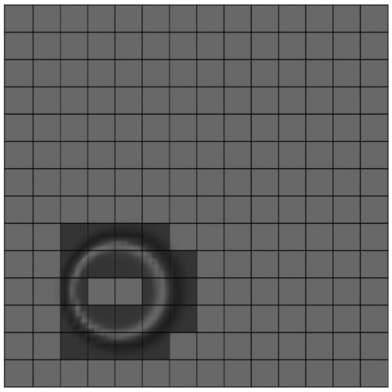

Fig. 1. A snapshot of a propagating wave. The domain is divided into 196 sub-domains. Only 21 (labeled dark) of 196 sub-domains need to be active to compute the next time step. Therefore, in the traditional approach 89 percent of the GPU devices are running idle in the computation of the current time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