Geosci. Model Dev. Discuss., 6, C1619–C1625, 2013
www.geosci-model-dev-discuss.net/6/C1619/2013/

Geoscientific Model Development Discussions

Interactive comment on "CUDA-C implementation of the ADER-DG method for linear hyperbolic PDEs" *by* C. E. Castro et al.

C. E. Castro et al.

ccastro@uta.cl

Received and published: 6 October 2013

We appreciate that the reviewer understands and honors our approach to investigate the numerical behavior (i.e. convergence and accuracy) of a realistic algorithm ported to GPU architectures. We realize that we give much more attention to the formulation of the numerical method than to the GPU implementation. After this review process however, we observe that for specialized computer scientists more information regarding the CUDA-C code is needed and we plan to provide this. Our first thought was that properly explaining the numerical method would simplify the description to the reader on how to code considering the GPU architecture. But we acknowledge that more detail is necessary. We also apologize for not providing the two kernels used to code the method as it was the original plan and stated in the manuscript.

Our main research question was whether the GPU hardware was capable of providing the expected convergence orders and accuracy reported for CPU versions of the method. From this work we can conclude that the ADER-DG method applied to linear hyperbolic PDEs is a good candidate to be ported to GPU architectures and could be further optimized to gain more advantages of this technology. Moreover, in this work we empirically prove the expected order of convergence for the GPU implementation and show evidence that in the single precision mode this convergence could be limited and not obtainable. This is relevant when we see recent publications like Mu et al. "Accelerating the discontinuous Galerkin method for seismic wave propagation simulations using the graphic processing unit (GPU)âĂŤsingle-GPU implementation" Computers & Geosciences 51 (2013) 282–292, where the only metric seems to be the speed up factor while accuracy investigation is limited to the statement "have no distinguishable differences". As mentioned in the previous paragraph we plan to provide more details of the implementation in order to better document the results. In particular the two kernels will be available in the revised version of the manuscript.

We are providing specific figures with the speed up as asked, however this information can be extracted from Figures 9, 11 and 14. As mentioned before, we did not intend to spread the simple message of gaining 10x speed up factor, but wanted to investigate convergence and accuracy. We realize that we should state more clearly what was the speed up gained in this study when using the GPU algorithm therefore this information will appear in a revised version. As a reference, our approach requires high-order accuracy, low coding effort and reduced time computation which we represent in Fig. 1.

The CPU solver has not been optimized by computer scientists but was developed by geoscientists trying to solve realistic problems like simulating the propagation of seismic waves in an elastic medium, or the advection of a tracer and tsunami propagation (linear case) where the presented numerical method is applicable. This means – as an example – that several different access-patterns to the unknowns are used: edge-wise, or cell-wise in the same functional unit. In order to describe the coding strategies

with more detail in a 1-to-1 comparison between CPU and GPU code, and yet retain readability of the manuscript, we suggest to include pseudo-code examples as given in an Author Comment in the GMDD web system.

We feel that computing GFLOPS rates for our algorithm is unrealistic as we consider a complex numerical algorithm with different processing paths including I/O, depending on the data. We would prefer to present our results in a more qualitative perspective, that is:

- It is possible to implement a GPU version of the ADER-DG numerical method with the expected accuracy.
- The GPU version of the algorithm presented in this manuscript is faster for the test problems presented.

We provide two speed up factor figures. Figure 2 for test problems 1 and 2 and a Fig. 3 for test problem testing the elastic wave equation.

The SeisSol CPU runs have been performed on shared-memory machine with 16 cores, 4 Intel Xeon X7350 Tigerton Quad-Cores with clock speed: 2.93 GHz, L2 cache: 2 x 4MB and 128GB main memory. We compiled the CPU version with Intel's Fortran compiler 13 and optimization flag -O2. Even though we used a multi-core architecture, all runs have been computed in serial to avoid the influence of MPI communication overhead. The rest of the CPU runs have been performed in serial on a shared memory machine with 24 cores, 4 Intel Xeon X5660 Six-Cores with clock speed: 2.67 GHz. We compiled the CPU version with Gfortran compiler version 4.6.3 and optimization flag -O3. The GPU version of the code was compiler with the CUDA compilation tool nvcc release 4.2.

We acknowledge that the description of our GPU optimization strategy has not been highlighted in enough detail. This is partly due to the fact that our code was not included

in the submission, as promised, and we apologize for that. To a greater part it is due to our decision to not give the technical details in the text in order to keep it readable. We propose therefore, to include pseudo-code examples for the important parts of our CUDA implementation. By this, we hope to give satisfying answers to the questions mentioned by the reviewer.

Regarding the SIMD versus SPMD nomenclature: To the author's knowledge, SIMD is often used synonymously to SPMD and this was our intention. We did not mean to refer to a parallel architecture model but to the corresponding programming paradigm. But we are happy to adopt the suggested – more accurate – convention given by the reviewer.

Interactive comment on Geosci. Model Dev. Discuss., 6, 3743, 2013.

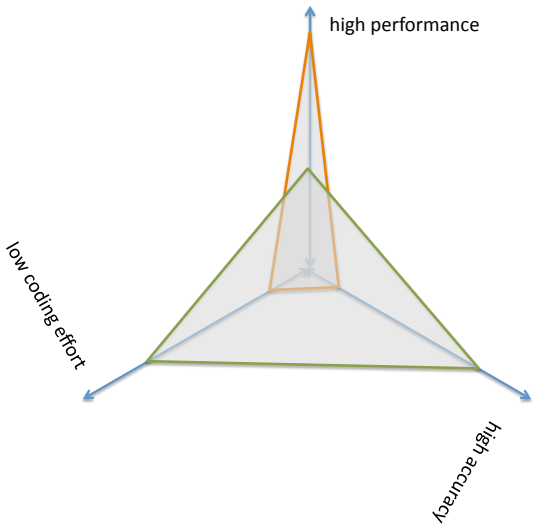

Fig. 1. Optimization problem for transforming original code into GPU enabled code. Our approach tries to maximize the area of the triangle (green) while many authors so far have focused on the speed-up factor

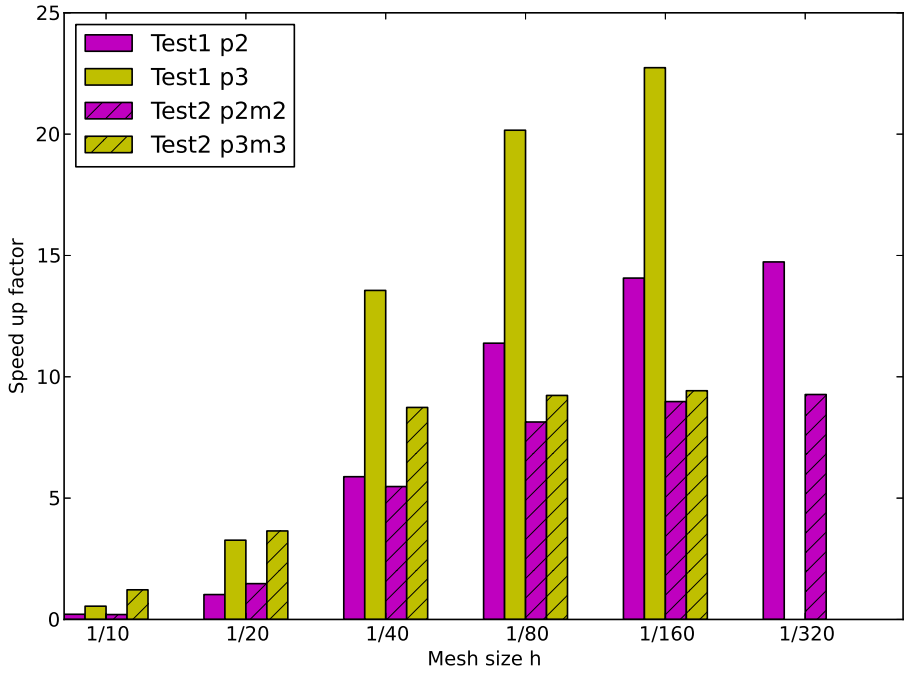

Fig. 2. Speed up for Test 1 and 2

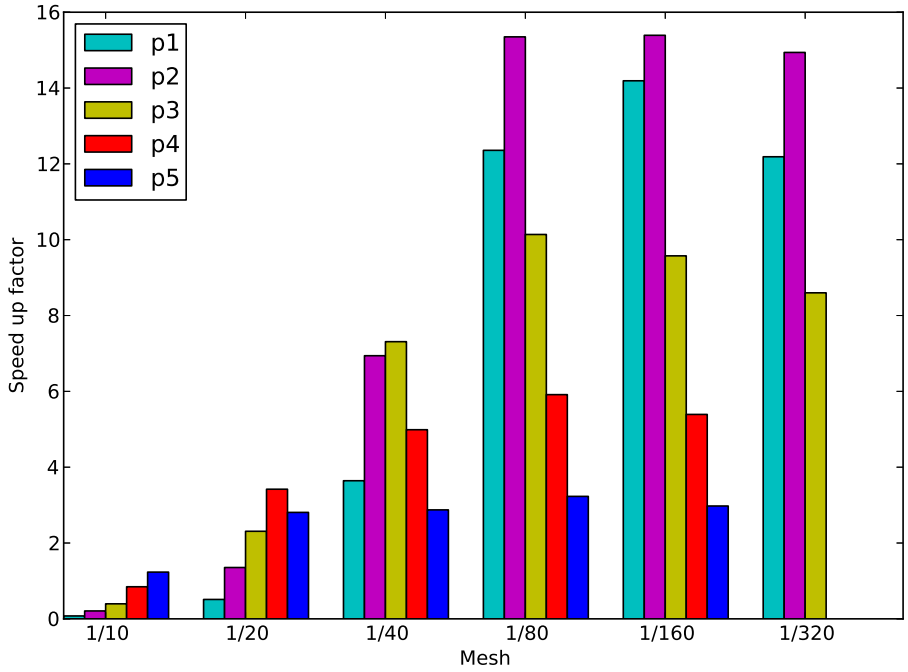

Fig. 3. Speed up for elastic wave equation