Geosci. Model Dev. Discuss., 6, C1373–C1375, 2013
www.geosci-model-dev-discuss.net/6/C1373/2013/

Geoscientific Model Development Discussions
Open Access

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

Anonymous Referee #2

Received and published: 1 September 2013

This paper shows several aspects (mainly the convergence behaviour on GPU) of a GPU implementation (using CUDA C and Fortran modules) for a previously published high order Discontinuous Galerkin numerical 2D solver of linear hyperbolic PDEs on unstructured meshes. Some of the main aspects of the CUDA implementation are briefly shown but they are not suitably argued and the authors do not explain the reasons behind the main design decisions.

It is noteworthy that the paper dedicates 11 pages to describe the already published numerical scheme and only 4 pages to describe the CUDA implementation of the scheme (the most important section together with the numerical experiments). Authors give a vague and brief desription of the CUDA C implementation of the numerical scheme. With this description, it is not possible to understand how the numerical scheme is

mapped onto the GPU and to evaluate the actual contribution of the paper.

Section 4 is dedicated to study the convergence behaviour of the GPU solver in a Fermi GPU (Tesla C2070) in comparison with several CPU solvers. This study (with single and double precision and testing the numerical scheme for different convergence orders) is interesting and the data analysis which is included can be useful to scientists and engineers that work in the GPU acceleration of similar high order numerical solvers. However, without a clear description of the implementation details and a more suitable performance analysis, this section loses meaning and relevance.

On the other side, the numerical results section does not include a suitable study and analysis of the performance of the GPU solver. Figures 9, 11,13,14 are interesting to show the convergence behaviour but make difficult for the reader to evaluate the speedup with respect to the CPU solvers. Additional Figures showing the speedup obtained (although the CPU solver is not optimized) would be useful (at least to watch the speedup with respecto to SeisSol in Figure 14 and the speedup for order 2 in the remaining tests). A brief description of the CPU implementations would also be necessary to understand why the authors think that it is not optimized. Moreover, the speedup with respect to the CPU solver (even considering non optimized versions) does not seem very high, taking into account the power of the considered GPU platform and the structure of the numerical algorithm. In order to evaluate the efficiency of the GPU solvers (for several orders and precision), the computation of the GFLOPS and the transfer rate in GB/s on GPU would make it possible to compare against the theoretical peak compute in GFLOPS ((in single and double precision)) and peak transfer rate in GB/s of the Tesla C2070 GPU.

It would be very important to include in Section 4, the characteristics of the host platform (CPU model and DRAM size) and the particular software tools used to process the languages employed in the implementations of the numerical scheme (compilers, runtime systems, compilation switches etc.) because this information makes it possible to evaluate suitably the experimental results. On the other hand, I think that Section

GMDD
6, C1373–C1375, 2013

3.1. should be extended and could be included as part of the Appendix (for readers that are not familiar with GPU programming).

The authors promise a more detailed description of the kernels but this description is not included in the appendix (I think that this information is extremely important and it should be included in Section 3). As a consequence, several important implementation details are unclear, such as:

- Why do the same execution configuration parameters are used in both kernels (TimeIntegrateDof and FluxComputation) and how the data locality is improved? - The mapping of the numerical scheme calculations to GPU threads is not given. The connection between the numerical scheme and the CUDA kernels should be clearly specified in the paper. - The shared work of the kernel threads of a block to recover the state of the corresponding element is not described.

In my opinion, the paper includes an interesting study about the convergence behaviour on GPU of a CUDA implementation of a relevant unstructured-mesh high order numerical scheme that can be used in several applications in Geosciences. However, this study loses meaning without a much more complete description of the implementation and a better performance analysis that make it possible to understand the details and quality of the mapping of the numerical scheme to the GPU platform.

Minor aspect: In Section 3.1, SIMD is considered as a programming model to program GPUs but SIMD is actually a parallel architecture model that matches well with the architecture of a CUDA-based GPU Streaming Multiprocessor. The programming model followed to program CUDA-based GPUs is the Single Program Multiple Data (SPMD) Programming style.

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