Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2020-323-RC1, 2021

Geoscientific Model Development Discussions

Interactive comment on "The GPU version of LICOM3 under HIP framework and its large-scale application" *by* Pengfei Wang et al.

Anonymous Referee #1

Received and published: 6 January 2021

The paper presents implementation of LICOM3 using HIP framework targeting an HPC cluster with AMD GPUs. Atmospheric modeling in not my area of expertise, therefore I will not comment on this aspect of the paper; I will only comment on the model implementation and parallelization efforts. Here are some specific comments about individual sections.

Section 2.3: "The nodes of both partitions are interconnected through the high-performance InfiniBand (IB) networks." Which partitions? Prior system description does not mention any partitions, instead it states that the system consists of 7000 nodes with 4 GPUs per node.

Section 3.1: There is some repetition here, e.g., the authors state several times that HIP supports both AMD and NVIDIA GPUs.

Section 3.2: Good work on converting the original Fortran code to C and verifying the results to be identical between the two implementations. However, the authors do not say much about this ported code in terms of the optimizations applied to it, if the code is purely executed as a single thread on each node, if any 3rd party libraries have been used, etc. Also, how the execution time of this newly ported C version of the code compares to the execution time of the original Fortran code? It is important to have a well-performing CPU code as a starting point.

Section 3.3: From section 3.2, I understood that the entire code was ported to C. However, section 3.3 states that a mixed Fortran/C code was used here. Please clarify to resolve this inconsistency. Also, I do not quite follow the discussion in lines 222-227 about array size. I understand that the original Fortran code uses static arrays and these arrays need to be changed to be dynamically allocated in order to move them to the GPU global memory. Is this what you are implying here? I also have a hard time following the discussion about halo data packing and Fig. 4.

Section 3.4: What storage and file system do you use? These I/O performance numbers are not very meaningful without specifying the underlying storage architecture. It is possible that the low original performance of the I/O operations was solely due to a low-performing storage system and would not be an issue on a higher-performing storage. Also, how exactly is data loaded/stored to disk? The size of data transfers would play a huge role on the overall performance.

Section 4.1: I think the correct way to quantify the speedup is to use all CPU cores vs. the GPUs, not just a single CPU core. Also, it is not clear from the text how the GPUs are managed. For example, is each GPU managed by a single CPU thread, or is the same thread is used to manage all 4 GPUs? Section 5.1: Interesting discussion about failure rates. However, I wonder how realistic your assumptions are with regards to existing studies such as https://www.christian-engelmann.info/publications/gupta17failures.pdf.

Overall, the presented implementation is rather straightforward and not well-thought. The authors ported to GPU several time-consuming kernels within the existing HPC application. Such approach is typically not very productive as it limits the design choices and does not give enough flexibility to optimize the overall application. Before proceeding with such an effort, the authors should have analyzed the amount of time spent in each of the subroutines on the CPU with regards to all other aspects of the code, such as I/O and network communication. Figure 8 shows such kernel time distribution after porting, but I could not find much about the amount of time spent on cross-node communication. It is important to understand the potential benefits of one or another approach before starting the actual implementation effort. Next, there is no discussion about dominant computations in each of the kernels and how to best implement them on the GPU. There is no information in the paper that would indicate anything about the quality of implementation of these GPU kernels, e.g., how well they use GPU resources. I would like to see achieved FLOPS and memory bandwidth of these kernels with respect to the roofline model for this particular GPU to be convinced that the authors did a good job porting these kernels. The discussion about performance improvements is very convoluted by the fact that the authors start comparing performance at some rather large scale of 384 GPUs. What about performance of a single GPU on a much smaller model vs. performance of a single CPU socket, or a 4-GPU node vs. all CPU cores in that node? There is no discussion about what happens inside a single node, for example, how well are all 4 GPUs are utilized and if there is anything that one could benefit from the fact that these 4 GPUs have access to the same host memory. There is no discussion about how the halo exchanges are implemented at the MPI level, how much overlapping is happening for computation, communication, and I/O. The paper is short on many technical details, ranging from characteristics of the underlying hardware, to software/compiler environment, to implementation details, which makes it very difficult for others to put the results obtained by the authors in comparison with other work. Above all, the paper is hard to read due to a very poor use of language; almost every sentence needs corrections.

If the authors want their paper to be printed with the existing implementation, they must provide a much better description and analysis of this implementation.

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2020-323, 2020.