Geosci. Model Dev. Discuss., 4, C1246–C1253, 2011
www.geosci-model-dev-discuss.net/4/C1246/2011/

Geoscientific Model Development Discussions

Interactive comment on "ASAMgpu V1.0 – a moist fully compressible atmospheric model using graphics processing units (GPUs)" *by* S. Horn

S. Horn

horn@tropos.de

Received and published: 27 December 2011

The author would like to thank the reviewer for raising a lot of important questions and will try to answer them in detail:

Q.1:He did not describe the hardware that was used or it's performance - where did the 10 TFlops peak performance come from?

A:The used hardware is mentioned only very briefly in the current manuscript version. The model runs on several different hardware configurations. For development smaller cases run on an Lenovo W500 equipped with an AMD Mobility Radeon HD 3650. For three dimensional larger domains we have two computation server. The motherboard we use, provide full wired 16 x PCIe slots for 4 GPUs. The older server is equipped

with two AMD Radeon HD 4870, and the newer one with four AMD Radeon HD 5870. The decision for AMD boards was a financial one, they provided the highest theoretical peak performance at the lowest price. The used CPU is an Intel(R) Xeon(R) CPU E5530 with 4 cores, so the 4 GPU threads can be distributed over the available cores. The 10 TFlops peak performance is an estimation based on the technical specifications of one AMD Radeon HD 5870 (http://www.amd.com/de/products/desktop/graphics/ati-radeon-hd-5000/hd-5870/Pages/ati-radeon-hd-5870-overview.aspx#2). One of these GPUs has an theoretical peak performance of 2.7 TFlops, with that four of them should be in the range of 10 TFlops. But this is a optimistic value disregarding communication cost between GPUs. The theoretical peak performance for the older AMD Radeon HD 4870 is 1.2 TFLops. The operating system is a standard linux with hardware accellerated xserver using the proprietary drivers by AMD.

Q.2:There were no performance numbers, no comparisons to CPU results, or scaling numbers for serial and multi-GPU runs.

A:First comparisons with a similar model (ASAM) running on CPUs (Intel(R) Xeon(R) CPU E7 - 4860 @ 2.27GHz) showed an estimated speedup of factor 80. That means an experiment on one ATI Radeon HD 4870 is about eighty times faster then the CPU version running on one core or two times faster then the CPU version running on 40 of these cores. But detailed comparisons were not performed yet and are definitely a task for future development. First scaling experiments were done with the DYCOMS-II case. The results are shown in Tab.1 in the manuscript and in the plot attached to this comment. The bottleneck of this GPU architecture can be identified there as well, the communication through the PCIe bus. That is why for small domains using multiple GPUs has no advantages then using one GPU.

Q.3:What percentage of peak did you get?

A:This is a tough question, because with the complex model structure including branching it is not easy to define how many floating point operations were performed during

one or more time steps. But the speed of about eighty times faster than one CPU core (with approx. 10-20 GFlops) indicates a performance in the range of 1 TFlop what is near the theoretical peak performance of 1.2 TFlops. But this is just an approximation and further research is needed to answer this question.

Q.4:Can it run on CPUs?

A:Currently it is not possible to run the model on CPUs only. With the migration of GPUs into the CPU and the development of many-core architectures it will be necessary to provide graphics device drivers for those solutions. Hopefully those will support OpenGL what then enables this model to run on such CPUs.

Q.5:If the author chose OpenGL for portability reasons, did he compare performance and accuracy on different platforms? Was it truly portable (no code changes)?

A:At the beginning of the development, we also tested the model on nVidia platforms. There we found a problem with the conservation of symmetry. The origin of the problem is located in the order some floating point operations are performed. This can be reproduced on CPUs as well. On the AMD boards this issue could be solved by rewriting the terms and changing the execution order using brackets. On nVidia this rewriting did not show the desired effect. Our current theory about that is that nVidia did some stronger optimizations on the shader code before compiling. Newer GLSL Versions include compiler flags to control the degree of optimization so it should be possible to solve the problem there as well. At the moment the code still needs minor changes to be run on nVidia. For example the value reported for the number of available texture devices is interpreted differently by the nVidia and AMD drivers, because they seem to have different definitions of what a "texture device" is. That causes some issues with memory allocation for the texture pointers for the input/output buffers, but can be easy solved by manual defining the number of available texture devices.

Q.6:There are no indications of communications costs in the multi-GPU environment.

A: The cost of communication can be observed in Tab.1 in the manuscript and is also mentioned in the text, but I agree this is a very important point that should be stated out more strongly. This question is directly connected to a Figure requested by reviewer 1, a plot for the very first scaling experiments (Fig.1, attached to this comment). The first impression is, that for smaller domains which fit into the memory of one GPU, only small speedup is possible using two GPUs, most of the time is used for communication between the GPUs. I think we have some space for optimization there. For larger domains more GPUs are necessary because of the restricted amount of on-board memory. And it is hard to extrapolate scaling behavior from such small numbers of blocks, because with higher numbers of GPU threads and CPU nodes, more and more communication can happen independently from other blocks and simultaneously. So for systems with more than nine GPUs communication overhead is expected to not increase with domain size (except perhaps due data collisions in the connecting network). So there is a lot of future work that has to be done during the further development of this model approach.

Q.7:The parallelization strategy was vague with little discussion of the code design, efficiency, readability, etc.

A: There are two different levels of parallelization in this model approach. The first one is a fine grained parallelization done by the GPU which is transparent to the developer. This is done by the device driver that splits the textures/buffers into a large number of blocks each processed by one stream processor individually. In contrast to CUDA where the developer can define the number of threads and block sizes used, in OpenGL the developer has no influence on this aspect of parallelization. To use more then one GPU the second level of parallelization has to be implemented. This is done using the MPICH2 library. At the current state for every GPU one thread is created with it's own OpenGL context. These threads communicate with each other using MPI_Send and MPI_Recv functions. Therefore a static connection between those threads is established. The model is parallelized using domain decomposition. That means the

domain is splitted in two or more large blocks handled by one GPU each. Therefor it requires one data exchange per acoustic wave time step at the boundaries between two blocks. This includes copying the necessary data from GPU memory to main memory, send them to the destination block, and uploading to the on-board memory of the destination GPU and vice versa. Those data transfers are asynchronous calls, so one thread could send data over MPI-functions to a second thread while copying data from the GPU to main memory for a third thread. The code written in C++ and structured in different classes, mainly shader, texture and transporter class. The source and destination textures are connected with the shader using pointers. A single function call initiates the computation on the GPU for the complete texture set. The computation may be a single operation per element but hence every shader call invokes a CPU function call, it is recommended to perform as much operations per shader call as possible. For example all sinks and sources for momentum (like pressure, gravity, viscosity, divergence damping, friction and damping layers) can be handled in one shader. The transporter class encapsulates the MPI functions and transfer functions between GPU and main memory. The class structures, the pointers for data flow and the transparent fine grain parallelization lead to good readability.

Q.8:There are no references indicating the author knows about other GPU parallelization efforts for weather and climate models. There is good work being done in the community.

A: Definitely there were a lot of good work done in this field. For example in the weather research forecast model (WRF) several parts of the model like the microphysical parameterization, passive tracer advection and the chemical solver were migrated to the GPU [Michalakes, 2008]. If parts of a complex model are migrated to GPUs, problems with high data transfer costs are documented, limiting possible speedups. There is also the model GALES [Schalkwijk, 2012] with a similar approach to the work presented here, a model running completely on the GPU with great interaction possibilities written using CUDA. Those references will be added in the introduction in a revised

manuscript.

Q.9:There were also no references to CUDA, OpenCL, GPU compilers, or an indication why OpenGL was chosen. Was it because the author was using an older GPU which did not support the HPC languages?

A:The main reason for OpenGL was the already mentioned platform independence. It is an established widely supported open standard. When the development of this model started, 4 years ago, OpenCL was not supported very well yet, meanwhile this could be an alternative and migration from GLSL to OpenCL should be possible with relative low effort. Another nice side effect is the transparency of the fine grain parallelization which allows developers to concentrate on model physics and numerics. Last but not least OpenGL as a graphics library can be used for visualization and rendering, reducing the amount of data that has to be saved on hard disk to produce high resolution animations.

Q.10:The scientific research seems promising though I am not a meteorologist so it's hard to evaluate merit. However, the scope seems limited to success running some idealized test cases. Is there a CPU version of the model so results can be evaluated? If not, can it be evaluated to other models of similar scale? When will real data be used?

A:The author presented mainly idealized test cases because those can be compared to simulations by other workgroups and can be used to evaluate the model results. The model is still far away from current real weather forecast models like WRF. The next important steps on this way would be the implementation of orographic structures, a soil and vegetation model and a radiation code. Nevertheless, there were some experiments using more or less real boundary condition data. One example is the simulation of the heat island effect of Cap Verde Island with comparison to doppler lidar data (Engelmann et al. 2011) or a still experimental simulation of a supercell (http://www.youtube.com/watch?v=bJqdQe2HVqI). Also different stratocumulus

and cumulus cloud cases based on real data (BOMEX, RICO, DYCOMS) from the GEWEX Cloud System Study (GCSS) were successfully simulated but not published yet.

Q.11:The author also did not research other scientific efforts to develop and run non hydrostatic models of this type. There are plenty of models already developed that are at the scale indicated and years of work have been done in this area - so again, the scientific content does not appear new or innovative. While this topic is of great interest to the community, I cannot see any benefit to publishing this paper without major revisions.

A:There are some good points that all together make this model something new. The first one is that the model calculations are completely done on the GPU. The second is it can use several GPUs in parallel. And the third one is OpenGL, allowing the community to use whatever graphics platform they like.

references:

Engelmann, R., Ansmann, A., Horn, S., Seifert, P., Althausen, D., Tesche, M., Esselborn, M., Fruntke, J., Lieke, K., Freudenthaler, V., and Gross, S.: Doppler lidar studies of heat island effects on vertical mixing of aerosols during SAMUM2, Tellus B, 63, 448-458, 2011. 2652

Michalakes, J. and M. Vachharajani, “GPU Acceleration of Numerical Weather Prediction”, Parallel Processing Letters Vol. 18 No. 4. World Scientific. Dec. 2008. pp. 531âĂŤ548,

J. Schalkwijk; Griffith, E.; Post, F; and Jonker, H.J.J.; High performance simulations of turbulent clouds on a desktop PC: exploiting the GPU, Bulletin of the American Meteorological Society, 2012 (in press)

Interactive comment on Geosci. Model Dev. Discuss., 4, 2635, 2011.

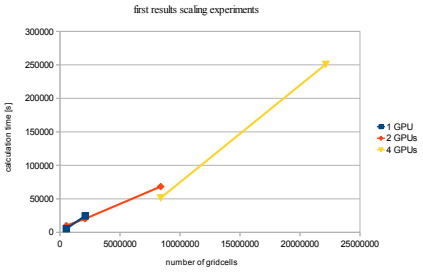

Fig. 1.