Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2017-63-RC2, 2017

Geoscientific Model Development Discussions
Open Access
EGU

Interactive comment on "GPU accelerated atmospheric chemical kinetics in the ECHAM/MESSy (EMAC) Earth system model (version 2.52)" *by* Michail Alvanos and Theodoros Christoudias

Anonymous Referee #2

Received and published: 12 June 2017

The authors present the implementation and evaluation w.r.t. performance and accuracy of a source-to-source translation tool for KPP kernels in the EMAC earth system model. CUDA kernels and corresponding calling code is generated by parsing Fortran code generated by KPP. A representative benchmark is evaluated on 2 different platforms using 2 different generations of Nvidia GPUs. The performance impact of various optimizations is studied using microbenchmarks. Furthermore, the tool is released open source under a permissive MIT license and the specific release used to produce the manuscript is deposited on zenodo with a DOI, both of which are commendable.

Printer-friendly version

Discussion paper

1 Remarks / questions

1. Why did you decide for CUDA in favour of OpenCL (which is also mentioned on p. 4)?
2. Have you considered directive based approaches e.g. OpenACC?
3. Why is it necessary to parse the code generated by KPP? Could the integration be done at an earlier stage e.g. AST, before actually generating code and parsing it again? (p. 5)
4. Have you considered using an existing source-to-source translation framework e.g. the ROSE compiler framework, which does have a CUDA backend? (p. 5)
5. Did you consider other ways of subdividing the computation than columns? (p. 5)
6. Could you have refactored the solver to split matrices among threads such that they fit in on-chip memory? (p. 8)
7. Where does the 2.5x speedup figure in Table 4 come from? And where is there a 1.75x speed of 40 vs. 20 MPI processes?

2 Minor comments

1. What are the scalability limitations of the EMAC model? (p. 2)
2. Does KPP generate *either* Fortran or C code (equivalent) or *both* Fortran and C code calling each other? (p. 3)

3. What do you mean by "the linear algebra can be solved off-line before the execution of the application"? (p. 4)

4. "the design architecture for memory allocation is subject to change when the impact of coalesced memory access is more prominent" (p. 6): Do you mean with future GPU generations? Do you not expect memory access limitations to be lifted further?

5. Why is more memory required for Pascal? (p. 7)

6. Lower occupancy results in a reduced number of concurrent thread blocks per SM (p. 8)

7. Do you mean thread local variables rather than scalar variables for privatization? (p. 9)

8. The description of the strategy for reducing load imbalance (p. 9) is not very clear: what do you hope to gain by "offloading" more processes than available GPUs?

9. Can yo elaborate how "representative benchmark" was chosen? (p. 10)

10. What do you mean by "the performance gain of these optimizations in the P100 accelerators is limited within the margin error"? (p. 14)

3 Grammar / wording (page / line)

1/14: to the CPU-only code

2/28: executes the gas phase chemical kinetics independently because these are independent of

3/1: in terms of portability

3/4: KPP provides ... analysis routines

3/15: execution times

4/7: parallel instructions

4/25: assigns an equal number of

5/24: GPU threads

Algorithm 2: Step 2: Call kernels

7/5: for future GPU architectures if applicable

7/6: to the GPU

7/15: the limiting factor for memory allocation

7/16: at least

8/5: on the GPU

8/11: remove "the occupancy of"

8/15: Launch Bound

8/21: remove "width of"

9/8: computational efficiency

9/30: for each GPU thread

10/14: scalability within a compute node

11/7: remove "with"

13/10: EMAC model consists of

15/3: a better memory subsystem

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2017-63, 2017.

Printer-friendly version

Discussion paper