Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2017-150-RC2, 2017

Geoscientific Model Development Discussions
Open Access
EGU

Interactive comment on "Portable Multi- and Many-Core Performance for Finite Difference Codes; Application to the Free-Surface Component of NEMO" *by* Andrew Porter et al.

C. Osuna (Referee)

carlos.osuna@meteoswiss.ch

Received and published: 28 August 2017

This is a well written paper, that describes a solution to the portability and performance portability problem in multiple computing architectures. The paper contains a good description of the separation of concerns of PSyKAI, provided as three different layers. This design helps isolating the computing architecture dependent optimizations in the middle layer, PSy, that sits between the Algorithm and the Kernel layers. The paper provides a good description of the application, their equations and type of computational patterns and an excellent description of all the optimizations evaluated and integrated within PSyKAI. The results are consistent and described in a very compre-

Printer-friendly version

Discussion paper

hensive manner with several comparisons and scalability measurements. The application of PSyKAI to the free surface component of NEMO demonstrates the applicability of the separation of concerns techniques to finite difference models as a real solution to attain a single source code that can efficiently run in multiple architectures. Therefore the reviewer thinks that this paper provides a very valuable contribution to the problem of portability of weather modes, and brings insight into new programming models that could be adopted in the future by the community.

Below I make some general comments, with the aim to help the authors increase the quality of the presentation. I will make other small specific points in a separate comment:

While the mathematics and equations are well described, the reviewer is missing some snippet of code that is representative of the target model. Showing some simplified codes, like the dominant momentum section, could help the reader to understand the computational patterns (see some floating point operations performed, extensively discussed in section 3.3), and, in general, better reason about sections that discuss the relevant characteristics of the code, like AI, or ILP.

The reviewer thinks PSyKAI provides a very relevant contribution that tackles the problem of separation of concerns and portability. Although PSyKAI is not the only approach to solve similar problems, it has some particularities and interesting novel contributions when compare to others. A small "Related Work" section where other approaches are explained and compared would help and can be used to highlight the novelties of this work. Similarly a section or paragraph describing the limitations would be valuable. Particularly a discussion on the design choices of PSyKAI in three levels, where all the computing architecture dependent optimizations happen in the PSy layer. Is it the right choice, could the author imagine other optimizations that would need to be applied to the Kernel layer?

Section 3.3. and Figure 13 shows a very relevant roofline model to evaluate the effi-

Printer-friendly version

Discussion paper

ciency of the implementations discussed in the results section. The text points at some fundamental issues with the way the kernel is implemented. Then other theoretical upper bounds are constructed based on how NEMO is implemented. However the issues in the NEMO implementation that limit the performance and makes not possible to achieve the upper bound of the roofline are not explained or at least not clear to the reader. Deriving a new upper bound (different than the memory bandwith upper bound of the roofline line) suggest that the authors understand the reasons for the poor performance of the implementation. I would recommend explaining them in the text. A hint is given in the text that the implementation does not exhibit a good balance between floating point operations. However the CPU ceiling derived with Habakkuk even for the case where there is good balance and the pipeline is perfectly utilized gives a bound that is 8 times away from the limit provided by the Roofline. Assuming the underlying bottleneck is in the Kernel, since PSy would provide a proper parallelism, memory layout, etc I think this is an interesting effect that is not clear to the reader and would require some more clarification. What is there in the implementation, in terms of floating point operations that makes this code perform badly? Also in the comparison of SSE and AVX, it is said that SSE and AVX versions of the division vector instruction does not provide any benefit. That would explain why they do not perform well, but not why SSE performs better than AVX.

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2017-150, 2017.

Printer-friendly version

Discussion paper