Geosci. Model Dev. Discuss.,
doi:10.5194/gmd-2016-307-RC1, 2017

Geoscientific Model Development Discussions
Open Access
EGU

Interactive comment on "Accelerating the Global Nested Air Quality Prediction Modeling System (GNAQPMS) model on Intel Xeon Phi processors" *by* Hui Wang et al.

Anonymous Referee #1

Received and published: 20 March 2017

General Comments

The paper describes the work done by the authors on optimizing the GNAQPMS model for Intel Knights Landing. The paper is well organized. I agree with the authors that there are still options to further optimize the code but the work performed so far is a good first step and worth publishing. There are a few changes that need to be addressed. Also the language could use some general improvement. I will first give some specific comments that I believe need to be addressed before publication. After that I list a few (optional) suggestions for future optimizations.

Specific Comments

Printer-friendly version

Discussion paper

(a) section 1, page 3: according to the introduction Section 3.1 is supposed to discuss where the bottlenecks of the code are. I assume that this refers to the runtime measurements shown in Figure 2? Under the term "bottleneck" I would have expected a discussion whether the code is bound by memory bandwidth by showing measurements of memory bandwidth and flops/s and comparing them with the peak performance obtained for the STREAM and LINPACK benchmarks. I understand that hardware counters are not very accurate on Intel architectures but you could still count them with an emulator or by hand. The paper does not show in its current state what the bottlenecks of the different parts of the code are.

(b) section 4.2, page 9: comparing the patterns and values without giving any specific relative difference between base and optimized version of the code is not enough to claim that the results are identical. The authors need to show some precise numbers like the total mass of the air and the different chemical species in both versions and the difference between optimized and base version.

(c) section 4.3, page 9: how did you measure the power consumption? Do you trust VTune to give you these measurements or did you measure the power consumption yourself?

(d) How many OpenMP threads were used per MPI process? Did you try different configurations (like 2, 4, 8, 16, 32, 64 threads per MPI process)?

(e) The term "manual vectorization" is used many times throughout the paper. This is very misleading. Manual vectorization would be in my opinion if the code was rewritten with avx512 vector intrinsics in C! The vectorization used in this paper still relies on the compiler.

Suggestions

(f) As mentioned before the paper does not investigate what the real bottleneck of the code is and how far the code is from optimal performance. I highly recommend to

create a theoretical and measured roofline model. This would allow to answer how much potential for further optimization should still be possible.

(g) I would love to know how many of all floating point operations are vectorized. I understand that counting floating point operations on Intel architectures through hardware counters is not very accurate. Maybe you could give some estimates from what the vectorization report shows you?

Interactive comment on Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-307, 2017.