Geosci. Model Dev. Discuss., 7, C2930–C2934, 2015
www.geosci-model-dev-discuss.net/7/C2930/2015/

Geoscientific
Model Development
Discussions
Open Access

Interactive comment on “Intel Xeon Phi accelerated Weather Research and Forecasting (WRF) Goddard microphysics scheme” *by* J. Mielikainen et al.

Anonymous Referee #1

Received and published: 14 January 2015

1 General remarks

In this paper, the authors optimized one of the microphysics schemes in the Weather Research and Forecasting (WRF) model which is a popular open source weather prediction system worldwide. The purpose is to take advantage of Intel Many Integrated Core Architecture (MIC) hardware - a new HPC product that integrates many cores on one die and provides large SIMD width. This work represents the authors’ first step towards a full acceleration of WRF on Intel MIC since the scheme they chose is actually a small portion (2355 out of $>$ 500000 lines) of the whole WRF code. I appreciate the authors’ efforts to exploit the capabilities of the new accelerator for WRF, which is an in-

teresting work and can be potentially beneficial for the large community of WRF users. However, I would not recommend publication of this paper on GMD in the current form mainly due to the following reasons:

1. The optimization strategies used are mostly common practice that can be found on Intel website. They are not, as the title may indicate, closely related to the specific microphysics scheme. So one may not learn too much from the experience presented in the manuscript. It is noteworthy that WSM5 seems to be the first microphysics scheme fully optimized on Intel MIC. But there are not enough details regarding the previous work on WSM5 for one to judge how much contribution this paper adds. Futhermore, the performance of Xeon Phi 7120P does not seem to be impressive as compared to that of Xeon E5-2670 CPU. The optimization of the Goddard scheme reduces the final processing time for the CONUS 12km benchmark data set from 223 ms to 47.8 ms on Intel MIC and from 207.8 ms to 73.7 ms on 2 socket Xeon CPU. It turns out that MIC becomes only 1.54x faster than Xeon CPU for this specific test case. The performance gain would be even less impressive considering the difference in theoretical peak FLOPS, memory bandwidth and prices of these two platforms. Thus it is a doubt whether Intel MIC is a promising solution for accelerating WRF or memory-bound modules in WRF. Of course, the speedup may vary depending on the problem size. The CONUS 12km test case may not be ideal for Intel MIC. How does Intel MIC perform on other data sets?
2. I do not agree with the statements about GPU on page 8945 “The data transfer overhead for a WRF module on a GPU is usually enough to kill any potential speedup for the whole WRF. Only when the whole code has been translated to run on GPU can WRF get any speedup on GPUs”. Many of the authors’ previous work on GPU have shown that significant speedup can be obtained with GPU for many WRF modules even when I/O cost is taken into account. In a presentation given by one of the authors http://on-demand.gputechconf.com/supercomputing/

2013/presentation/SC3133-CUDA-Weather-Research-Forecasting-Model.pdf, it is mentioned that the speedup of “Goddard microphysics” on GPU is 1311x for the same test case used in this paper. If it is the same scheme studied in this paper, then this work may be just another example proving the disappointing performance of Intel MIC. I would suggest the authors include GPU for comparison since GPU has been a very mature accelerator and extensively studied for WRF. The successful history of GPU may also shred light on whether a full acceleration of WRF on MIC is necessary. There would be a possibility that compute-intensive portions of code in WRF are suitable for GPU while data-intensive portions are a better fit for MIC. If the Goddard microphysics scheme is a memory-bound data-intensive module with large memory footprint, the authors should provide evidence (e.g. instructions or operations per byte) for it. To better show the advantages of Intel MIC, it is also desirable to find a case for which GPU may not perform well but Intel MIC can, or a case that Intel MIC can give easy speedups with much less effort than GPU, e.g. rely more on compiler features such as auto-vectorization instead of extensive manual modification of code.

3. Code validation would be very important for WRF. However, no results in terms of errors in the output or data visualization have been presented. Furthermore, the validation is done with precise math compiler options, not the options actually used when evaluating speedups. This makes the speedups meaningless. I find that the compiler options used are very aggressive. It would be reasonable to also investigate the results for less aggressive options.
4. Scaling results are not discussed at all. They are essential for evaluating many-core supercomputing environments.

2 Specific remarks

1. Whenever a speedup factor is given, the baseline over which the number is computed needs to be pointed out.
2. Citations are necessary for figures and tables if they are not made by the authors.
3. Line 12 on Page 8944: “are already exists of” should be “already exists”.
4. Line 20 on Page 8944: “dynamics” should be “dynamic”.
5. Line 23 on Page 8944: “fully acceleration” should be “full acceleration” or “fully accelerating”.
6. The description of Goddard microphysics scheme in section 2 is too sketchy. More importantly, the differences or similarities with other microphysics schemes are not discussed. Readers may want to know why this particular scheme was picked and if other schemes in WRF can benefit from this research.
7. The compiler options listed in the first paragraph of section 4 miss explanations.
8. In section 4.2, the descriptions as well as figure 8 are too vague. Details on how and why the i-loops were split into nine parts should be provided. Are there data dependencies in the i-loop?
9. Line 2 on Page 8950 “the next” should be “this”.
10. In section 4.3, it is not explained why only 16 data elements are assigned for each thread. This is a very important parameter that can affect the performance significantly. It may be worthwhile to try other numbers such as 32 and 64 to find out the optimal choice.

11. Why are section 4.2 and 4.3 combined into one step (v3)? What if they are separated into two steps?
12. In section 4.4, please explain why this step has a bigger effect for CPU than MIC.
13. Line 16 on Page 8951: it is inappropriate to say “compiler optimizations” since manual optimizations have also been used.
14. Line 24: a word is missing after “number of ”.
15. Line 2 on Page 8952: a punctuation is missing before“Instruction count”.
16. The description in section 4.8 does not match the information in Figure 11. And the purpose of this analysis is unclear.
17. In section 4.10, it is not clear how many threads are used and to what extent vectorization has been performed for both Xeon CPU and Intel MIC. Vectorization is the key to superior performance of MIC. It would be necessary to give an analysis about the behavior of complier auto-vectorization. The analysis would answer these questions such as, what percentage of code was successfully auto-vectorized by the compiler before manual optimization? what is the number after the code adjustments? is there any room for further improvement?
