Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2017-150-RC3, 2017

Geoscientific Model Development Discussions

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

Here I add some more minor specific comments for the text. I hope it helps!

pag. 7, line 10: "In order that we could safely add such directives we altered the GOcean infrastructure to allocate all field-data arrays with extents greater than strictly required": It is not clear to the reader why these larger extents are required in order to add this directive. pag. 7, line 22: "Although the Intel compiler does do" I would replace by "Although the Intel compiler can do" since it is not guarantee when the compiler will inline. section 2.1.5: the discussion makes the reader wonder whether attaching the grid to the field, which can be read only or not is the right choice. section 2.1.8: it is not

clear why inlining these copies gives better performance. Is it due to data reuse of the same fields that might be in cache? If so, that will depend on the domain size? Maybe a sentence clarifying that would help.

section 2.2.1: Aside note: It looks like other omp strategies like blocking would help increasing the data locality of computations among cores computing different blocks, or parallel do simd that would allow collapsing the loops and increase parallelism?

Figure 3: why is gnu still much slower? We would expect that PSyKAI would help the compiler adding the corresponding directives so that all compilers could deliver similar performance. Still is a factor 2x away from others. Can the reason for this be clarified?

page 15, line 5: Text says that 256^2 fits in cache. Which cache are the authors referring to? A single field with that domain size would take 512 KB (in double precision) which will be out of L1

Figure 5: same comment I made before. It seems that we can not attain same perf. with all compilers. Would be interesting clarify what is the intrinsic reason for this, if this is a real limitation of the compiler, or future work can improve the gnu performance.

page 18, line 9: text says that 32x32, due to limited amount of parallelism inhibits scaling, therefore focus is on 256x256 domain sizes. Is it clear that is parallelism what limits scalability? The curve shown by 32x32 might have a reasonable shape for memory bound codes, where with few cores one can saturate the memory bandwidth of the Ivy Bridge, and adding more cores might not provide better performance. On the contrary the large gap between 32x32 and the other domains suggests that the code is not cache oblivious, where the updated points/s would not (so strongly) depend on the domain size? See for example: https://arxiv.org/pdf/1410.5010.pdf

Figure 11: For domain size of 64, parallelism will be very small on the GPU, if collapse(2) is not used. If used, 4096 GPU threads could be active, still not 100% occupancy but much better. So it is not clear why the collapse(2) didnt help in performance?

Page 22, line 11: The statement says that the comparison between PSyKAI approach and the CUDA fortran implementation proves little overhead introduced by PSyKAI. While this is true, it is no proven that the implementation is efficient, since also a CUDA implementation could be inefficient. Figure 12 suggests that actually the GPU implementation provides an expected performance in comparison with memory bandwidths of the IB and K40. Adding the GPU numbers to the Roofline model of Figure 13 would help in proving efficency of the GPU code and make a more solid statements about the GPU performance.

Figure 13: font size and lines are small and hard to read. Increasing the font size and probably using markers would help.

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