Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2018-218-AC1, 2019

Geoscientific
Model Development
Discussions

Interactive comment on "HOMMEXX 1.0: A Performance Portable Atmospheric Dynamical Core for the Energy Exascale Earth System Model" *by* Luca Bertagna et al.

Luca Bertagna et al.

lbertag@sandia.gov

Received and published: 1 February 2019

We thank the reviewer for their comments and remarks. Here, we summarize how we addressed their comments.

1) RC: I think the introduction and problem description is clear for someone from other fields in numerical methods to follow without too much difficulty. For completeness, the authors could consider adding mathematical equations for the differential operators.

AC: Equations that describe HOMME are added at the beginning of the HOMME section.

2) RC: I feel Section 3.3 needs some improvement. I found the part describing how parallel_for loops map to different execution policies slightly unclear. I suggest describing the kernel with pseudo-code of nested loops, decorated with execution policy choice.

AC: We added one generic example in section 3.1, and one example specific to HOMME in section 3.4.2.

3) RC: Have the authors verified that vectorization on CPU is effective, potentially by looking at the generated assembly code?

AC: We have. To explain our approach, we have expanded the material on vectorization to explain our analysis and results. We have provided a new figure to show performance as a function of the pack length and operator implementation approach.

4) RC: I'm curious that if the authors encountered any limitations for the vector data types, e.g. for maths function calls, conditionals etc.

AC: We added some sentences about conditionals. HOMME has very few low-level conditionals, unlike, for example, physics parameterizations.

5) RC: Could the authors elaborate more on "reuse of subviews is important to minimize index arithmetic" on CPU (page 11 line 9)? I don't quite follow what is "... the number of connections per elements..." (page 11 line 12).

AC: We expanded the sentence in 3.4.1 to explain what we meant by that.

6) RC: In Section 4, HOMME and HOMMEXX have very similar performance on Haswell, but using different (if I understand correctly) strategy, could the authors explore a bit more on the reason behind it?

AC: On conventional CPU, such as Haswell, and especially in the serial build (1 thread/core, 1 MPI rank/core), the two implementations are quite similar. In this case, our goal is to match HOMME's performance, with the primary challenges being (i) matching HOMME's excellent auto-vectorization and (ii) not slowing down this run configuration due to strategies used for other architectures.

7) RC: Subtle point: is Turbo-Boost a potential source of randomness in the experiments?

AC: Turbo-Boost and other throttling in either direction, such as decreasing the clock speed on Skylake when AVX512 instructions are being run, is unlikely to be any more a source of randomness than other effects, such as network physical topology. Indeed, experience on supercomputers is that the interconnect tends to be the greatest source of run-to-run variability. To minimize the impact on code comparability of noise effects in general, side-by-side comparison was done always in the same job submission, so exactly the same computer nodes are used within a temporally constrained period of time to obtain comparison results. To clarify this point, a sentence on jobs submission was added at the beginning of section 4.

8) RC: One thing I feel the paper is missing is that we do not know if the achieved performance is "good enough". The paper could be improved (by a lot) by e.g. showing the percentage of peak performance achieved and/or roofline model of the hardware. This is especially helpful because the experiments are carried on a large range of hardware with very different characteristics, and finding some common metrics to compare and contrast between them would help the authors in organizing the presentation of experimental results.

AC: First, for clarity, we would like to emphasize that the metric of "(thousands of elements-timesteps)/(node or GPU)/second", while complicated, is a useful efficiency metric. Data in these units can be directly compared across architectures (e.g., GPU vs KNL vs HSW vs SKX) and across scaling regimes (many elements/compute resource vs. few elements/resource). Second, we agree that careful systems-oriented metrics would be interesting. These would characterize inter-node bandwidth and latency; on-node bandwidth, cache performance, GPU kernel launch latency, CPU-GPU bandwidth, and many other GPU metrics. However, any set of metrics requires a lot of

work to collect and analyze. We have chosen to prioritize the key metrics of interest: 1. Does HOMMEXX match HOMME wherever HOMME can run? 2. Does HOMMEXX vectorize well? 3. Across scaling regimes, both isolating on-node performance and accounting also for inter-node communication, how does the the end-to-end performance vary? 4. What is the performance on nonconventional archictectures relative to conventional ones? 1, 3, and 4 are already answered in the original manuscript, and we have improved the exploration of question 2 in the revised manuscript, in response to thoughtful questions regarding vectorization. That said, the substance of this question essentially points to the research topic of speeding up the dycore independently of implementation strategy. That is, can we isolate an important section of code, collect careful systems data, and use it to speed it up, even just in the original HOMME Fortran code? This is a great question. We do not attempt to answer that question in this paper.