Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2018-18-RC2, 2018

Geoscientific Model Development Discussions

Interactive comment on "The VOLNA-OP2 Tsunami Code (Version 1.0)" *by* Istvan Z. Reguly et al.

Anonymous Referee #2

Received and published: 11 July 2018

Summary:

This paper describes the design, implementation, and performance of VOLNA-OP2, a code for tsunami modelling based upon the shallow water equations solved by the finite volume method. VOLNA-OP2 is the result of long-standing interdisciplinary work, having the objective of porting the original C++ implementation, VOLNA (released in 2011), on top of OP2 (hence the name VOLNA-OP2), a framework for unstructured mesh applications. Thanks to the OP2 layer, VOLNA-OP2 can now run on a variety of computer architectures, including CPUs (e.g., Intel Xeon, Intel Xeon Phi) and GPUs. Support for both shared-memory parallelism (OpenMP, CUDA), distributed-memory parallelism (MPI), and SIMD vectorisation is provided. Therefore, there are two main benefits deriving from this work: a new codebase that drastically relieves the burden of extending

and maintaining VOLNA; higher performance as well as performance portability across architectures. These make VOLNA-OP2 an appealing candidate in the ecosystem of tsunami-modelling codes. An entire section discusses the performance achieved on several computer architectures.

Comment:

This work is a good example of what can be achieved through interdisciplinary research. VOLNA-OP2 overcomes the initial limitations of VOLNA by allowing the code to scale up to hundreds of cores and on different kinds of architecture, thus enabling science in new contexts and scenarios.

My questions, that should be addressed before publication, mostly concern the performance results section, as the rest already looks in solid shape (apart from some minor points listed below). Despite I understand that "showing how fast VOLNA runs on top of OP2" is not the main point of the article, I think that some of the following aspects should be clarified. This may also be helpful for people wanting to follow a code-modernisation approach similar to the one described in the submitted article.

* Use of uniform triangular meshes as opposed to non-uniform ones. Am I right if I say that the most typical use case of VOLNA-OP2 is with non-uniform meshes? If so, why using uniform ones? The mesh can have a drastic impact on the achieved performance, due to different MPI partitioning, load balancing, effectiveness of mesh renumbering etc. Is this a weakness of the analysis? if not, why?

* I know that with other OP2-based applications, traditionally, MPI+OpenMP has never really outperformed pure MPI (due to issues that have been extensively described in prior work), or at least not by a significant factor. Is the situation different with VOLNA? If so, can you say why? Speculation: is it because VOLNA is "more compute-bound" than other codes used in the past? I'm asking because I see that all numbers reported derive from MPI+OpenMP. By the way, I assume that you have taken care of pinning etc – can this be confirmed?

* Was data alignment enforced for directly accessed datasets ? I understand the same dataset can be accessed directly in one loop and indirectly in another loop, but perhaps there's hope to exploit some alignment in at least some of the memory-bounded loops?

* Can you be more precise as to why it was not possible to vectorise 'applyFluxes'? Can you share more details about how the vectorisation of the other loops has been achieved – is it via auto-vectorisation or something else?

* Which loops are compute-bound and which are memory-bound?

* Can you report the max memory bandwidth of *all* platforms? I think some are currently missing. Also, how was the memory bandwidth limit determined ? specs, STREAM, or...?

* Can you state the model of XeonPhi used? Some have 68 cores, not 64 like the one used in the experimentation.

* I think comparing the VOLNA-OP2 performance in different architectures is a bit unfair, at least for two reasons: 1- The number of "degrees-of-freedom" (or simply triangles) per core is generally different, and so is the proportion of time spent is computation and communication. Even data locality may have been impacted. 2- These architectures are profoundly different among them – in theoretical attainable peak performance, price, release date, etc. I don't like sentences such as "The GPU system with X1 nodes was Y times faster than the CPU system with X2 nodes"; I don't think they add much to the paper, and in fact they may be misleading. Perhaps you can normalise over a common metric, but again that's quite tough. Also, the Phi suffers a lot from the lack of the vectorisation in 'applyFluxes', and it's unclear whether this issue can be worked around or not.

* In the conclusions, I disagree with the sentence: "Through performance scaling and analysis of the code on (...), we have demonstrated that VOLNA-OP2 indeed delivers near-optimal performance on a variety of platforms (...)" . My point is that I don't think

that you have demonstrated that you're achieving near-optimal performance – it is true that some loops are relatively close to the architecture memory bandwidth limit, but others are not. We don't know the reason – in fact, we don't even know whether these loops are compute- or memory-bound (see point above). So I suggest to either drop that sentence, or rephrase it, or add a roofline plot.

* Of course, I assume that comparing the performance of VOLNA-OP2 to that of other codes is way too difficult, if not impossible

Code release and misc:

I see that version 1.0 has been "Zenodoized" – don't forget to add the DOI to the paper. Also, I suggest to move the code to its own organisation on GitHub so that permissions can be more easily set and contributing to VOLNA-OP2 gets easier. I also suggest to link VOLNA-OP2 from the 'apps' section of OP2.

On the language:

It might be just me, but it feels like that different sections of the paper have been written by different people. That is, the transition from some paragraphs to others is not as smooth as it might be. Also, there are some typos (e.g., outline -> outlined) and/or missing words ("the physical across") in various sections. All this should be improved prior to publication.

Minor notes:

* Section 3.1 might benefit from some figures. I know this is not the main point of the article, but I'm not sure that readers who are unfamiliar with OP2 will be able to understand how, for example, GPU parallelisation works. Maybe just cite some prior paper?

* In the performance section, Table 1 gives a name to three meshes: M1, M2, M3. These could be used systematically throughout the whole section, and in the plots as well, instead of referring to "the 1.4M mesh".

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-18, 2018.