EGUsphere, referee comment RC2
https://doi.org/10.5194/egusphere-2022-781-RC2, 2022

Comment on egusphere-2022-781

Anonymous Referee #2

Referee comment on "Parallelized Domain Decomposition for Multi-Dimensional Lagrangian Random Walk, Mass-Transfer Particle Tracking Schemes" by Lucas Schauer et al., EGUsphere, https://doi.org/10.5194/egusphere-2022-781-RC2, 2022

This paper highlights a multi-dimensional, parallel domain decomposition method for mass-transfer particle tracking methods. The authors focus on demonstrating the parallel scalability of a two- and three-dimensional "checkerboard" partitioning, and highlight a theoretical analysis and prediction of scalability at high core counts as their main novel contribution. A grand claim of reducing a "5-hour simulation to 8 seconds" is made to highlight the near-linear scalability of the chosen method.

The paper is well written and gives a detailed and concise overview of the problem definition and theoretical context in which the problem is set. A clear description of the particular class of particle tracking algorithm is provided and appropriate comparisons to other related fields are made. The performance analysis and theoretical prediction of scalability, however, neglects certain established principles (for example weak and strong scaling) and attempts to re-derive some well-known performance analysis steps; for example using an arbitrary 100-core baseline to normalise super-linear speed-ups, instead of normalising relative speed-up over memory sockets or nodes.

Unfortunately, however, the lack of any performance quantification of the baseline result, in particular with respect to shared-memory scaling, casts doubt over the achievements presented. Achieving near-perfect scaling is, after all, much easier with an unoptimised baseline. [1] While the authors are clearly aware of the redundant ghost particle communication within a shared memory space, no attempt is made to quantify single-node/single-socket performance. In particular, despite several of the dominant algorithm components being likely to contain memory-intensive operations (sparse matrix solves, KD-tree lookups, potentially redundant communication buffer copies), no attempt is made to measure or quantify memory-bandwidth utilisation, which could potentially explain some of the observed scaling behaviour (eg. the drop-off in Fig. 9 possibly?). To compound this further, several of the

performance graphs are plotting wall time against number of particles per partition, rather than time vs number of core/sockets/nodes, which makes reasoning about scaling behaviour counter-intuitive; and the description of the benchmarking hardware is very loose.

Nevertheless, the authors clearly demonstrate that their implementation of the dominant KD-tree lookups scales as N log(N), and that at high core counts their predicted parallel efficiency can be achieved. The theoretical derivation and comparison of parallel efficiency and scaling behaviour and subsequent mapping to achieved results adds value to the paper.

Overall, I find the topic of the paper and its novel contribution, as stated in the introduction, to be relevant for GMD. The paper has strong potential, but requires significant revisions before I can recommend acceptance, in particular due to the quality of the performance analysis. While I appreciate that fitting shared-memory parallelism is beyond the scope of the benchmarking code, I believe the paper would be significantly improved if the authors would establish single-node performance more rigorously, especially with regards to memory bandwidth utilisation, before focusing on scaling behaviour across multiple sockets/nodes and modelling of algorithmic scaling behaviour.

[1] https://blogs.fau.de/hager/archives/5260

Detailed list of comments:

==========

* Overly strong statements, like "reducing a 5-hour simulation to 8 seconds" (pg 2, l. 17) are undermined by the fact that single-node parallelism is not explored, benchmarked or optimised for.

* Section 2.3: The test hardware configuration is less than ideal for the given analysis, as the test nodes vary significantly ("8-24 cores with clock speeds ranging from 2.5GHz-3.06GHz"). For a rigorous performance analysis a fixed subset or partition of nodes with identical core types should be used, and the CPU model, as well as available memory needs be stated here.

* While I appreciate that availability may be a limiting factor here, industrial compilers (Intel / Cray / Nvidia) are likely to achieve better single-node performance than gfortran -O3. If available on the test system, investigating reporting whether or not that changes the baseline (single core / single node) runs would add weight to the performance investigation. This can should be considered optional.

* As clearly stated in section 5, accesses to particle of a neighbouring processors still go through a full MPI-driven halo exchange protocol, thus likely incurring significant memory movement that could further be optimised. Without appropriate treatment of shared memory parallelism, either via OpenMP or p-threads, or explicit MPI shared memory programming (one-sided) this is likely to incur significant performance overheads that should be accounted for and evalauted in scalability studies such as this.

* Section 6.1 Cost analysis: This seems like a rather lengthy way to explain and derive commonly known scaling behaviour (linear weak scaling with constant per-processor work

and KD-tree lookup scaling as n log(n)). I wonder if the manuscript can be made more concise by stating these facts and offering evidence that indeed the implementation behaves as expected?

* Figure 6: While Fig. 6 (a) does appear to match the predicted growth very well, the slope in Fig. 6 (b) is arguably steeper than the prediction. Could this be clarified or commented on?

* Line 335-338; this concept is commonly referred to as "weak scaling" and should be named as such.

* Figure 8: The x-scale should be inverted. Furthermore, this figure does not actually add any value to the discussion, as the inverse relationship between total particles and local particles is trivial.

* Similarly, Fig 7. shows multiple instances of weak-scaling snapshots that are nowhere near the regime limits. Please consider replacing Fig 7. and Fig. 8 with an actual strong-scaling graph (wall time vs. number of cores) for total particle values chosen in Fig 7.

* Figure 9.: It is increasingly hard to think about scaling behaviour with an implicit number of processors. Please plot wall time against number of cores when assessing strong scaling!

* Fig 9: Are the drop-off points related to scaling beyond a single memory socket? The near-linear scaling beyond that point (counter-intuitively to the left!) suggests that scaling behaviour is memory-bandwidth limited and and the addition of memory-sockets determines the real scaling factor. But again, this is hard to assess accurately without appropriate strong-scaling graphs.

* Units of time on y-axis missing for almost all wall time plots

* Section 7.2; line 467: This highlights one of the major shortcomings of the paper! While the single-core base case does not enter the MPI routine, it does represent an appropriate baseline for single-node/socket shared memory scaling. The authors not only neglect to account for this, but instead raise the baseline to an arbitrary 100-core baseline. To improve the quality of the paper, I recommend doing a specific shared-memory scaling analysis to determine if indeed scaling within a single memory socket follows the expected trends, and/or if the redundant use of ghost exchanges within a shared memory space affects performance. Then, once a single-socket/single-node baseline is established, scaling behaviour across multiple nodes/sockets can be analysed, thus giving a more grounded baseline for the speed-up plots in Fig. 15.

* Line 482 and onward: The technical term for "above perfect efficiency" is "super-linear speed-ups". The explicit explanation of equation (30) might also be superfluous; instead a more detailed explanation of how super-linear speed-ups are achieved would certainly elevate the paper.

* Section 8 outlines the lack of shared-memory parallelism; the question on line 522 already indicates that the authors are aware of this issue. Unfortunately, to my mind, this issue impacts the validity of the findings, since the demonstrated scaling behaviour (and the "hours to seconds" claim") can easily be put down to an insufficiently optimised baseline run. As no attempt is made to quantify the performance of the implementation with respect to single-node performance this casts doubt over the importance of the findings.

* Future consideration: Latency hiding by overlapping ghost communication with compute

work; possibly across time steps. Not for this work, but as a follow-on optimisation.