Title: The Two-Dimensional Swept Rule Applied on Heterogeneous Architecture

Authors: Anthony Walker, Kyle E. Niemeyer

Journal: Mathematical and Computational Applications (ISSN 2297-8747)

Manuscript ID: mca-1187867

This paper presents the application of the Swept rule for two-dimensional grids on heterogeneous architectures. The Swept rule was introduced in 2016 (ref. [1]), extended to two-dimensional grids in 2016 (ref. [5]), ported to GPU in 2018 (ref. [6]) and applied on heterogeneous systems in 2020 (ref. [7]). Therefore, this work is a study of the combination of these four previous works, as stated by the authors: "*prior swept rule studies which involve multiple dimensions and architectures but not the combination of the two [1,5–7]*". Such a study could be interesting for the readers if the conclusions were clear, either showing a positive or negative impact in performance, or extrapolable to larger caseloads as referred in the introduction and abstract: "*These results can help make decisions to maximize these benefits and inform designs*".

The introduction and motivation of this work is, in general, clear and well written. **The objective is to contribute in the large-scale simulation of unsteady PDEs** (special mention to Wildfire phenomena in first paragraph) **on heterogeneous supercomputers** (there is a mention to exascale computing in line 44) **by overcoming the network latency barrier** (there are several mentions to this barrier throughout the paper). Specifically, the objectives of the study on the swept rule include understanding: "*1. its performance on distributed heterogeneous computing systems, 2. its performance with simple and complex numerical problems on heterogeneous systems, 3. the impact of different computing hardware on its performance, 4. the impact of input parameters on its performance*". Contrarily, the results are quite disconcerting and these questions are not sufficiently clarified. After reading the study and conclusions, one can hardly make decisions to benefit from the Swept solver as the authors conclude that the performance depends on virtually everything: "*the performance of two-dimensional swept rule depends on the implementation, problem, and computing hardware. Speedup can be achieved but care should be taken when designing and implementing a solver. Finally, any practical use of the swept rule requires tuning input parameters to achieve optimal performance, and in some cases speedup at all over a traditional decomposition*".

Network latency is described as the "*fixed cost of communicating information between devices in the system*", and **is presumed to be the key problem of distributed computing**: "*significantly restrict the solution's time to completion, especially when using distributed systems*". However, the actual importance of this barrier, the need to solve it, and the potential gain is not proven in this work. Namely, there is a lack of an explicit study of such bottleneck and it raises many doubts:

1. Given a problem, and a hardware, what is the problem size range for which the latency barrier is –theoretically– manifested (this is studied for instance in ref. [1])?

2. Then, given a latency-limited problem, how big is this barrier? What is the –theoretically– possible gain? How many communication stages have to be avoided (i.e., what is the optimal block size) to –theoretically– overcome it (the answer would lead to a theoretical optimal block size value)?

3. Finally, do the obtained results agree with the theoretical values?

In the end, the study somehow fails to demonstrate the impact of the method on the latency, even the impact of the latency on the performance. Indeed, the weak scaling study in Figures 9 and 10 indicates the opposite: it shows how the effect of the communications from single- to dual-node executions is very low in the Standard approach and higher in the Swept approach. From Figure 9, the time per timestep in the Standard approach increases by less than 10% due to the outbreak of communications. This suggests that any method specifically designed for reducing the communication overhead will reduce as much as this 10% of the time in this case. In other words, larger reductions mean that the method is helping to overcome other bottlenecks which are not studied or identified. In this regard, the authors conclude: "*[...] This suggests that something other than latency dominates the simulations so latency improvements are less noticeable*", which overrides the initial hypothesis.

Looking at other works (e.g., ref. [1] and [6]), the effects of the network latency are manifested at very small loads of the order of 10K elements, and both references agree that bigger loads generally penalize the Swept rule against the classical decomposition. This may suggest that the network latency barrier was not a big issue in large-scale simulations. There are several previous studies (not mentioned) on pretty similar approaches which are applied to different objectives (e.g., memory access optimization, memory locality, cache reuse), reaching very solid results (ref. [WOL89, SON99, WON00, LEV16]).

Apart from this, I have a list of issues that should also be properly addressed if the publication of the work is considered.

Introduction:

- The mention to Wildfire falls into oblivion, consider either removing or extending it.

Related work:

- The (multiple) mentions of network latency are very repetitive, appearing in every paragraph.

- The "latency barrier" concept is too abstract, there is a lack of magnitude. Consider, extending it in a more comprehensive way, detailing actual cases for which latency is manifested as a barrier: hardware features, maximum problem size, minimum number of nodes, potential gain when solved...

- The cases in which you can achieve significant speed-ups are in line with other state-of-the-art large-scale simulations?

- The previous studies on pretty similar approaches are not mentioned (ref. [WOL89, SON99, WON00, LEV16]).

Materials and methods:

- The beginning of section 3.2 (specifically Equation 1) is better understood after reading the `Up-Pyramid` phase of the Swept solution process in section 3.3. Consider changing the order of these subsections or some paragraphs.

- The paragraph starting in line 229 is better understood after reading the `Octahedron` phase in line 284. Consider changing the order of these subsections or some paragraphs.

- The specification of GPU share and block size in the paragraph starting in line 191 does not match the latter specifications in line 317, section 4.

- Figures 1 and 2 can be improved to help understanding: they are badly trimmed, the illustration is too small, the font in the legend is too big, and the axis lines are too thick.

Results:

- Prior to any study for overcoming a bottleneck, the network latency in this case, this bottleneck must be proven. In this regard, the weak scaling study is the key: it demonstrates that the Standard method is losing around 10% of the time per timestep due to the outbreak of the communications. This 10% is what any method specifically designed for reducing the communication overhead could reduce (larger reductions mean that the method is helping to overcome other bottlenecks which are not studied or identified).

- The results are not illustrative. There is not a clear trend, which suggests that not only network latency but many other factors are affecting the performance of the tests. This should be addressed by designing a better benchmark and different post-processing for, eventually, identifying the other bottlenecks and studying the impact of the method on them.

- The results are all relative speed-ups so there is a lack of a reference point. Consider plotting the real performance also, not only relative results (as in ref. [7]), and studying the maximum theoretical performance (in FLOPS) to compare with. The roofline model may be a practical tool in this case (ref. [WIL09]).

- The choice of parameters (GPU share, array and block sizes) seems arbitrary (this is related to my questions above) because there is a lack of theoretical estimations to compare with. At small loads (like this), the performance of the devices is usually unstable, affected by more factors such as memory locality on CPU's cache, or occupancy in GPU, among others. Consider estimating the –theoretically– optimal values prior to running the simulations.

Discussion and conclusions:

- The discussion and conclusions should be rewritten after obtaining new results which effectively evaluate the impact of the method on the latency.

References:

[WOL89] M. Wolf, "More iteration space tiling," in Proceedings of Supercomputing '89, 1989.

[SON99] Y. Song and Z. Li, "New tiling techniques to improve cache temporal locality," in Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation, 1999.

[WON00] D. Wonnacott, "Using time skewing to eliminate idle time due to memory bandwidth and network limitations," in Proceedings of International Parallel and Distributed Processing Symposium, 2000.

[WIL09] Williams, S.; Waterman, A.; Patterson, D. Roofline: An insightful visual performance model for multicore architectures. Commun. ACM 2009, 52, 65–76.

[LEV16] Levchenko, V., Perepelkina, A., & Zakirov, A. (2016). DiamondTorre algorithm for high-performance wave modeling. *Computation*, *4*(3), 1–15.