2022-09-20

Morton-Ordered GPU Lattice Boltzmann CFD Simulations with Application to Blood Flow

Gerald Gallagher
Technological University Dublin, gerald.gallagher@tudublin.ie

Fergal J. Boyle
Technological University Dublin, fergal.boyle@tudublin.ie

Funder: Technological University Dublin

Morton-Ordered GPU Lattice Boltzmann CFD Simulations with Application to Blood Flow

Gerald Gallagher[1, 2, a)], Fergal J. Boyle[1, 2]

[1] *School of Mechanical Engineering, Technological University Dublin, Bolton St., Dublin 1, D01 K822, Ireland.*
[2] *CFD Research Group, Environmental Sustainability and Health Institute, Technological University Dublin, Grangegorman, Dublin 7, D07 H6K8, Ireland.*

[a)] Corresponding author: gerald.gallagher@tudublin.ie

Abstract. Computational fluid dynamics (CFD) is routinely used for numerically predicting cardiovascular-system medical device fluid flows. Most CFD simulations ignore the suspended cellular phases of blood due to computational constraints, which negatively affects simulation accuracy. A graphics processing unit (GPU) lattice Boltzmann-immersed boundary (LB-IB) CFD software package capable of accurately modelling blood flow is in development by the authors, focusing on the behaviour of plasma and stomatocyte, discocyte and echinocyte red blood cells during flow. Optimised memory ordering and layout schemes yield significant efficiency improvements for LB GPU simulations. In this work, comparisons of row-major-ordered Structure of Arrays (SoA) and Collected Structure of Arrays (CSoA) memory layouts with a Morton-ordered SoA memory layout for the LB plasma solver are presented, with speedups of up to 20% achieved against the base row-major-ordered SoA model. Further investigation is recommended on whether these efficiency increases remain for larger mesh densities in comparison to CSoA layouts, and hybrid Morton ordering schemes could alleviate any limitations with dimension sizing. The current optimisations are deemed useful for future blood simulation validation work involving cubic LB domains, such as optical tweezers tests and in-plane and out-of-plane shear flow.

INTRODUCTION

The characteristics of blood flow must be taken into account when designing medical devices for implantation in the cardiovascular system. Computational fluid dynamics (CFD) has matured to the point where it is now routinely used for numerically predicting cardiovascular-system fluid flows.[1] Traditionally, blood has been treated as a continuum with the suspended cellular phases of blood wholly ignored. The omission of the cellular elements of blood in CFD analyses has been due to the lack of knowledge about the structure of these cells, the lack of a suitable fluid flow solver to handle a heterogeneous fluid, and the prohibitive computational resources required to predict the motion of a fluid containing an enormous number of suspended cells. Advancements in simulation accuracy are highly desirable to avoid issues with sub-optimal medical device design. Therefore, an efficient solver incorporating all aspects of blood flow - plasma and cellular phases - is required. The deformation and orientation of red blood cells (RBCs) is one of the primary cellular factors affecting blood viscosity at high shear rates or strains,[2] and previous works have examined RBCs in flow in both 2D and 3D using CFD and developed RBC structural models.[3,4] A 3D graphics processing unit (GPU) lattice Boltzmann-immersed boundary (LB-IB) CFD blood simulation software package is in development by the authors which incorporates an in-house state-of-the-art RBC spring particle structural model allowing for stomatocyte, discocyte and echinocyte RBCs.[5] The LB method was chosen due to its amenity to parallelisation and its ability to simulate mesoscopic physics and multiphase flows in complex geometries.[6]

The performance of the GPU LB component of the CFD package depends on memory access patterns, including memory ordering and memory layout. Comparisons of GPU data access optimisation techniques for the LB method have been previously completed with application to blood simulation in arterial geometries by Herschlag *et al*.,[7] with lexicographic and two-grid memory ordering discussed and Structure of Arrays (SoA) and Collected Structure of Arrays (CSoA) memory layouts classified. The study focused on complex geometries where lattice nodes in a domain can be either fluid or solid. Semi-direct addressing (where there is a reduction in stored solid node information) with

a CSoA layout and lexicographic ordering was deemed the most efficient setup. Two-grid ordering on GPUs (which involves Morton ordering where applicable[8,9]) was also examined, which has been shown to be effective.[10] Morton order space filling curves have even been used for IB memory ordering.[11] For the LB-IB solver in development by the authors, any solid nodes will be handled by the IB component. Therefore, this work aims to examine the optimisation techniques available for the LB-IB CFD blood simulation software package in development by first comparing the performance of SoA and CSoA memory layouts for row-major-ordered 3D LB simulations and then comparing the performance of row-major-ordered and Morton-ordered 3D LB simulations.

METHODOLOGY

The simulations involved laminar flow around a stationary sphere[12] for a Reynolds number of 21.1 and utilised the Bhatnagar–Gross–Krook (BGK) D3Q19 LB model.[6] The boundary conditions used at the back, bottom, front and top of the domain were virtual node periodic conditions. The boundary conditions used for the sphere surface were halfway bounce-back implementing no-slip velocity conditions. Zou-He boundary conditions were used to enforce the specified velocity in the x direction at all nodes at the inlet and were also used at the outlet to specify zero gradient pressure conditions.[13] The row-major SoA implementation was first validated with final drag coefficient errors of 2%. For comparison between the row-major SoA, row-major CSoA and Morton SoA implementations, the three codes used were identical apart from using Eqs. (1) and (2) to generate the indexes in the row-major SoA and CSoA codes respectively[7] and using bit interleaving of the 32-bit x, y and z integer coordinates to generate the indexes in the Morton code (with further details available in the Libmorton C++ library[14]):

$$f\_SoA(l,q) = l + q \times N \tag{1}$$

$$f\_CSoA(l,q) = \left(\left\lfloor \frac{l}{s} \right\rfloor \times Q + q\right) \times s + l \bmod s \tag{2}$$

where for 2D, $l = x + y \times N_x$, q is the given distribution element or lattice vector, Q is the total number of lattice vectors per node, $N = N_x \times N_y$ where N_x and N_y are the total number of nodes in x and y respectively, and s is the stride length. Note that for 3D simulations, $l = x + y \times N_x + z \times N_x \times N_y$ and $N = N_x \times N_y \times N_z$, with z representing the additional dimension. Figure 1 shows sample SoA and CSoA Stride 2 layouts for three adjacent 2D D2Q9 nodes (with the stride length of two showcasing the shift to the next vector direction after two elements):

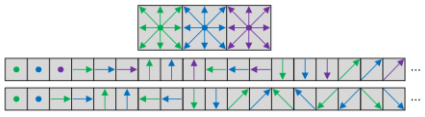


FIGURE 1. Three adjacent 2D D2Q9 nodes with SoA and CSoA Stride 2 layouts respectively underneath

For consistency, the codes were all compiled using the same compilation flags. Also, the results were only compared for the in-house solver without comparison to results from literature due to the general difficulty in establishing if other optimisations were present in different code implementations. Finally, the simulations were all stopped after 1000 iterations and executed in one session on one node on the Irish Centre for High-End Computing (ICHEC) supercomputer Kay with an NVIDIA Tesla V100 and CUDA 10.1.243. A sample 2D D2Q9 LB nodal layout[6] is shown in Fig. 2 and can be used to effectively explain the different memory orders implemented.

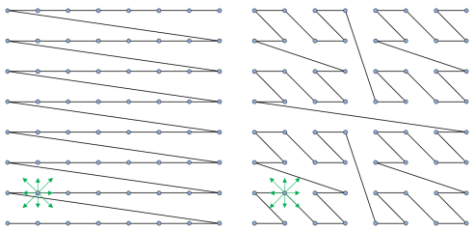
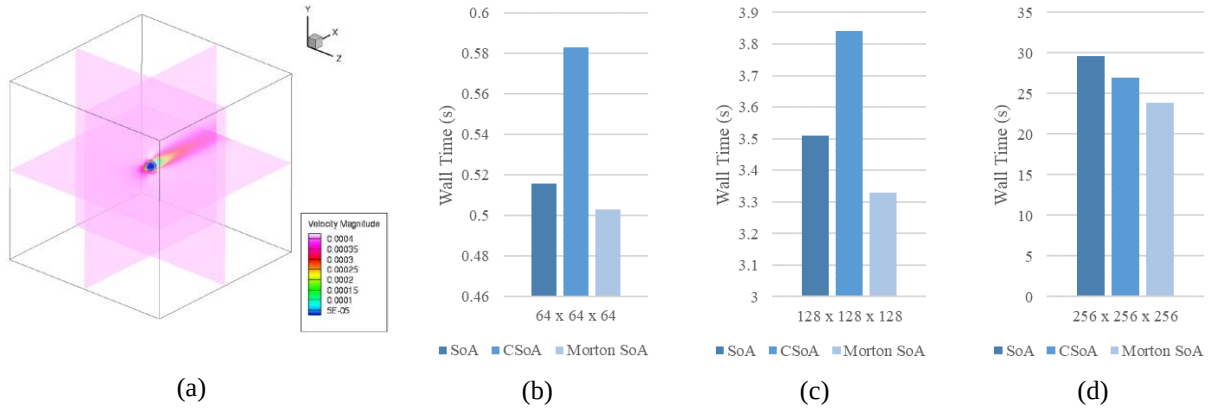


FIGURE 2. Row-major (left) and Morton (right) memory ordering for a sample 8 x 8 node 2D D2Q9 LB simulation

The node with green lattice vectors shown in Fig. 1 is representative of a single D2Q9 lattice node designated to have (x,y) position (1,1). Each node has nine (Q9) array elements associated with the relevant lattice vectors. Taking the LB streaming process as an example, lattice vectors from adjacent cells around node (1,1) are read and then written to the relevant array elements of node (1,1). For large mesh densities, the distance between the relevant elements in memory becomes large and there is a reduction in performance due to the increase in memory transactions. Memory access is coalesced when reads or writes can be combined into a single global memory transaction.[15] For example, taking a larger 128 x 128 nodal grid and taking the same node (1,1), the lattice vector data to be copied from nodes above and below the node of interest involves threads reading from and writing to elements separated by at least the row width of 128. In comparison, the maximum distance for the Morton-ordered thread access in this case is nine elements, which is a considerable improvement and reduces the number of memory transactions required.

RESULTS

The results obtained show that the Morton-ordered SoA simulations yielded better performance in all comparisons. An example visualisation of the velocity magnitudes for the converged test case and a graphical comparison of the wall times is shown in Fig. 3. A point of interest is the choice of stride length for the CSoA layout. The row-major SoA outperformed the row-major CSoA layout for the first two simulations; this was due to the chosen stride length of 64 not being small enough to offset the increased CSoA index calculation overhead in the kernels, and in the case of the first simulation, equalling the row length which resulted in the same layout as the SoA. The CSoA layout performed better than the SoA layout for the last simulation with a larger mesh density as expected, but the wall times always remained larger than the wall times for the Morton-ordered SoA simulations. This would indicate that the CSoA stride length should be optimised for smaller mesh densities in particular.



(a) (b) (c) (d)

FIGURE 3. (a) Example of final velocity magnitudes (m/s) for converged simulations and wall times after 1000 iterations for node sets (b) 64 x 64 x 64 (c) 128 x 128 x 128 and (d) 256 x 256 x 256

Table 1 shows all wall times that were collected for all simulations.

TABLE 1. Wall time comparison after 1000 iterations

Nodes in x, y & z	Row-Maj. SoA (s)	Row-Maj. CSoA (s)	Morton SoA (s)
64	0.52	0.58	0.50
128	3.51	3.84	3.33
256	29.67	26.94	23.84

Table 2 shows the percentage speedup of the Morton-ordered simulations, with the highlight being a speedup of nearly 20% over the original row-major-ordered SoA simulation for the final mesh density.

TABLE 2. Wall time speedup for Morton-ordered SoA simulations

Nodes in x, y & z	Morton SoA vs. Row-Maj. SoA (% Speedup)	Morton SoA vs. Row-Maj. CSoA (% Speedup)
64	2.5	13.8
128	5.2	13.4
256	19.7	11.5

There were some caveats. The Morton-ordered and comparison simulations required that each dimension node size be a power of two and that the node grid must be square in 2D or cubic in 3D. A hybrid Morton scheme would be a possible solution, where batches of nodes would be Morton ordered and the remaining nodes either row- or column-major ordered. Referring to column-major ordering, this could also be tested against Morton ordering as CUDA documentation mentions advantages in relation to memory coalescing over row-major ordering. Morton-ordered CSoA memory layouts could also be implemented and compared with the current results. Finally, extensive profiling should be carried out using the NVIDIA Visual Profiler to better analyse performance and detect non-coalesced global loads and stores.[15]

CONCLUSIONS

The main conclusion from this study was that 3D Morton-ordered LB simulations are efficient when used for cubic domains with significant relative decreases in wall time with increasing mesh density. Future work on whether the increase in efficiency remains for very large mesh densities is recommended, particularly in comparison to lexicographically-ordered simulations with CSoA layouts. Hybrid Morton ordering schemes could yield further performance increases and alleviate limitations. Also, CSoA memory layouts are efficient but depend on the choice of stride length, especially for smaller mesh densities. Future simulation validation test cases involving cubic LB domains can use the current optimisations, including RBC optical tweezers tests and in-plane (wheel configuration) and out-of-plane (tumbling, tank treading and swinging) shear flow.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial assistance provided by Technological University Dublin through the Fiosraigh scheme and advice and computing facilities provided by ICHEC.

REFERENCES

[1] P.D. Morris, A. Narracott, H. von Tengg-Kobligk, D.A. Silva Soto, S. Hsiao, A. Lungu, P. Evans, N.W. Bressloff, P. v Lawford, D.R. Hose, and J.P. Gunn, Heart **102**, 18 (2016).
[2] H.J. Meiselman and O.K. Baskurt, Semin Thromb Hemost **29**, 435 (2003).
[3] A. Moskal, R. Przekop, and I. Majewski, in *AIP Conf Proc* (American Institute of Physics Inc., 2018).
[4] T. Krüger, M. Gross, D. Raabe, and F. Varnik, Soft Matter **9**, 9008 (2013).
[5] M. Chen and F.J. Boyle, J Biomech Eng **139**, (2017).
[6] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E.M. Viggen, *The Lattice Boltzmann Method* (Springer International Publishing, Cham, 2017).
[7] G. Herschlag, S. Lee, J.S. Vetter, and A. Randles, in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (IEEE, 2018), pp. 825–834.
[8] G.M. Morton, *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing* (Ottawa, Canada, 1966).
[9] M.D. Mazzeo and P.V. Coveney, Comput Phys Commun **178**, 894 (2008).
[10] A.E. Nocentino and P.J. Rhodes, in *Proceedings of the 48th Annual Southeast Regional Conference on - ACM SE '10* (ACM Press, New York, New York, USA, 2010), p. 1.
[11] Y. Chen, W. Li, R. Fan, and X. Liu, IEEE Trans Vis Comput Graph **28**, 3235 (2022).
[12] F.W. Roos and W.W. Willmarth, AIAA Journal **9**, 285 (1971).
[13] Q. Zou and X. He, Physics of Fluids **9**, 1591 (1997).
[14] J. Baert, Libmorton: C++ Morton Encoding/Decoding Library (2018).
[15] NVIDIA, *CUDA C++ Best Practices Guide Design Guide* (2022).