Reducing Data Motion of Lattice Boltzmann Simulations through Application of Boundary Conditions on GPUs

Griffin Dube
gdube@g.clemson.edu
Clemson University
Clemson, South Carolina

John Gounley (Advisor)
gounleyjp@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, Tennessee

ABSTRACT

Lattice Boltzmann simulations are commonly used for solving many computational fluid dynamics problems. The main computation of many of these simulations are separated into two parts: collision and streaming of the lattice Boltzmann method particle distribution function, and application of boundary conditions. Most lattice Boltzmann performance studies on GPUs focus on the core collision and streaming kernels. In this study, we investigate how porting lattice Boltzmann Method boundary conditions to GPUs influences CPU-GPU data motion and overall application performance by porting the boundary conditions code of a lattice Boltzmann proxy application to CUDA. We complete all parts of the simulation on GPUs, reducing the number of data transfers required and improving time-to-solution by up to 49× by reducing the amount of time spent communicating data between CPU and GPU.

ACM Reference Format:
Griffin Dube and John Gounley (Advisor). 2020. Reducing Data Motion of Lattice Boltzmann Simulations through Application of Boundary Conditions on GPUs. In *Proceedings of Supercomputing '20 (SC '20).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Lattice Boltzmann simulations are often used to solve computational fluid dynamics problems across many disciplines. GPU acceleration is one common way to improve performance of fluid dynamics simulations. One caveat to this is the case where data is transferred frequently between CPU and GPU during computation. In these cases, the increase in data motion results in degraded performance. We use our lattice Boltzmann proxy application to show this performance degradation reduces the benefits of GPU acceleration and leads unpredictable performance. The methods used to reduce data motion in the general case of our proxy application can be applied to other lattice Boltzmann based applications. This poster makes the following contributions:

- Shows the impact that excessive data motion has on the performance of lattice Boltzmann simulations; and
- Proposes a method to improve single-node performance by completing all computation on the GPU

SC '20, November 15–18, 2020, Atlanta, GA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND

The lattice Boltzmann method uses the discretised Boltzmann equation to solve fluid dynamics problems numerically, and is broken up into several steps. We focus on the collision and streaming step and the application of boundary conditions step.

The collision and streaming step updates the density and macroscopic velocity in order to find the particle distribution function (PDF) for collision before streaming the resulting distribution to equidistant points in the domain lattice. Each point in the lattice breaks up velocity space using a velocity set, described by the number of dimensions and number of discrete velocities it represents [1]. We focus on the D3Q19 velocity set, as it is commonly used in three dimensional simulations [2]. The next step is application of the boundary conditions, where rules are enforced for handling points that lie on boundaries of the domain [3]. We focus on updating the boundary condition step in order to improve performance.

3 METHODS

The current version of our lattice Boltzmann proxy application completes collision and streaming on the GPU before transferring PDF data to the CPU for application of boundary conditions, and is referred to as the hybrid CPU-GPU version. Our new implementation performs the collision/streaming step and application of boundary conditions on the GPU, thus it is referred to as the GPU-only version of the application. We use CUDA C++ to rewrite the boundary conditions such that both steps are handled on the GPU, removing the need to transfer PDF information between CPU and GPU during the simulation.

We rewrite the boundary conditions of the domain using CUDA, dispersing boundary points across GPU threads for computation in

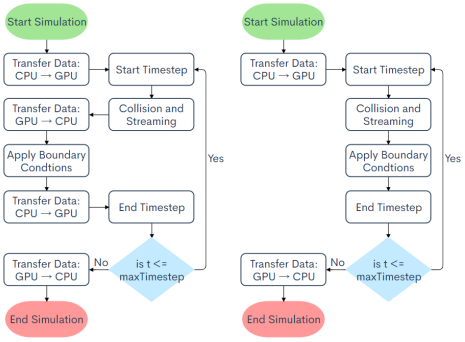

Figure 1: Simulation time-step flow for hybrid (left) and GPU-only (right) versions of the proxy application.

order to perform operations in parallel. This requires the simulation to transfer the PDF array twice: once at the beginning of the simulation to the GPU's device memory, then back to the CPU at the end of the simulation. This reduces the number of transfers significantly when compared to the hybrid CPU-GPU version, which transfers PDF data $2t + 2$ times where t is the number of simulation time-steps completed.

4 RESULTS

We measure performance on Oak Ridge National Laboratory's Summit supercomputer, focusing on single-node performance of our application using 2 IBM POWER9 CPUs and 6 NVIDIA V100 GPUs.

We compare this implementation to two others: A CPU-only version that performs all computation on the CPU, and the hybrid CPU-GPU version mentioned previously. For each implementation, two propagation patterns are used for computing PDFs to ensure that important values are not overwritten if they are still needed: two-grid, which stores data in two arrays reading from one and writing to another, and one-grid, which uses a single array to store values, performing different computation operations during even and odd time-steps [4].

4.1 Analysis of Performance

Performance is measured for our application at simulation sizes ranging from 21,504 to 29,816,640 fluid elements.

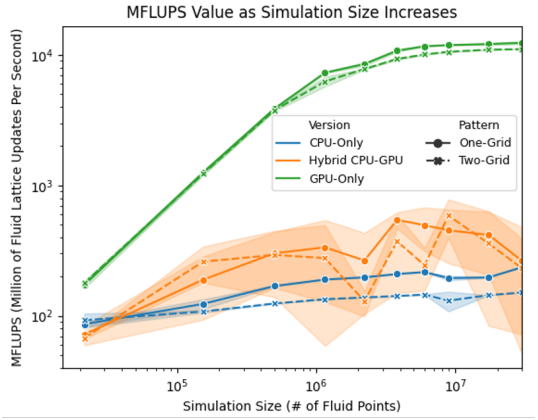

Figure 2: Average MFLUPS values as simulation size increases, plotted along with bands representing one standard deviation from the mean

Results show that a GPU-only implementation results in a speedup of 2× for small simulation sizes up to 49× for larger sizes. Additionally, the GPU-only implementation results in a much smaller standard deviation, shown by the error bars in Figure 2, which indicates reproducibility of results for a given simulation size. This confirms that moving all computation to the GPU and reducing the number of data transfers improves performance.

4.2 Analysis of Data Motion

The GPU-only version does not require additional transfers apart from the setup and final steps of the simulation, thus the percentage of runtime spent communicating PDF data for large simulation sizes is 2%, compared to 92% for the hybrid version, freeing up a higher percentage of runtime for computation, as displayed in Figure 3.

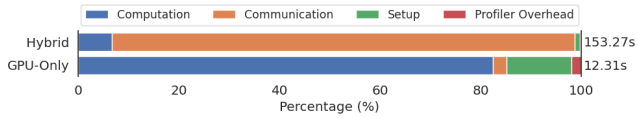

Figure 3: Percentage of runtime spent on simulation tasks.

The hybrid CPU-GPU version produces unpredictable results shown by the large standard deviation of the values in Figure 2. This is due to the overhead incurred in transferring data between CPU and GPU twice during each time-step, which causes contention between MPI ranks for NVLink bandwidth. We minimize data movement in the GPU-only version, preventing unpredictability and allowing the benefits of GPU acceleration to become apparent.

5 CONCLUSION

Our GPU-only version of the lattice Boltzmann proxy application results in speedup up to 49× that of current versions. This improvement is attributed to a reduction in data movement and increase in parallelism through performing GPU acceleration for all computation during the simulation. We achieve this by running the boundary conditions step on GPUs instead of CPUs. Further, reducing communication minimizes the possibility for resource contention, preventing unpredictable results. Reducing CPU-GPU data motion allows the application to take advantage of parallelism without the overhead or unpredictability of frequent transfers.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internship program. This research used resources of the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory. This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

REFERENCES

[1] T. Krüger et. al. 2016. *The Lattice Boltzmann Method - Principles and Practice.* Springer. Etc.
[2] J. Vetter A. Randles G. Herschlag, S. Lee. 2018. GPU data access on complex geometries for D3Q19 lattice Boltzmann method. *2018 IEEE International Parallel and Distributed Processing Symposium* (2018), 825–834.
[3] M. Hecht and J. Harting. 2010. Implementation of on-site velocity boundary conditions for D3Q19 lattice Boltzmann simulations. *J. Stat. Mech.* 2010, 01 (Jan 2010), P01018.
[4] Markus et al Wittmann. 2013. Comparison of Different Propagation Steps for Lattice Boltzmann Methods. *Comput. Math. Appl.* (2013), 924–935.