# Reducing data motion of lattice Boltzmann simulations through application of boundary conditions on GPUs

Griffin Dube[1,2], John Gounley[2]

[1]Clemson University, Clemson, SC, USA 29631; [2]Oak Ridge National Lab, Oak Ridge, TN, USA 37830

## Introduction

- Lattice Boltzmann simulations are commonly used for solving many computational fluid dynamics problems.
- Main computation of many of these simulations can be broken into two parts: collision and streaming of points, and application of boundary conditions [1].
- Data motion caused by excessive communication between CPU and GPU leads to degraded performance.
- We propose a solution by performing all simulation computation steps on the GPU, applying it to a lattice Boltzmann proxy application to evaluate the impact on performance.

800-Word Summary

## Methods

- Our GPU-only version removes the need to transfer data between CPU and GPU during time-steps by performing collision and streaming and application of boundary conditions on the GPU.
- We port boundary conditions to the GPU using CUDA, dividing fluid points among GPU threads for computation.

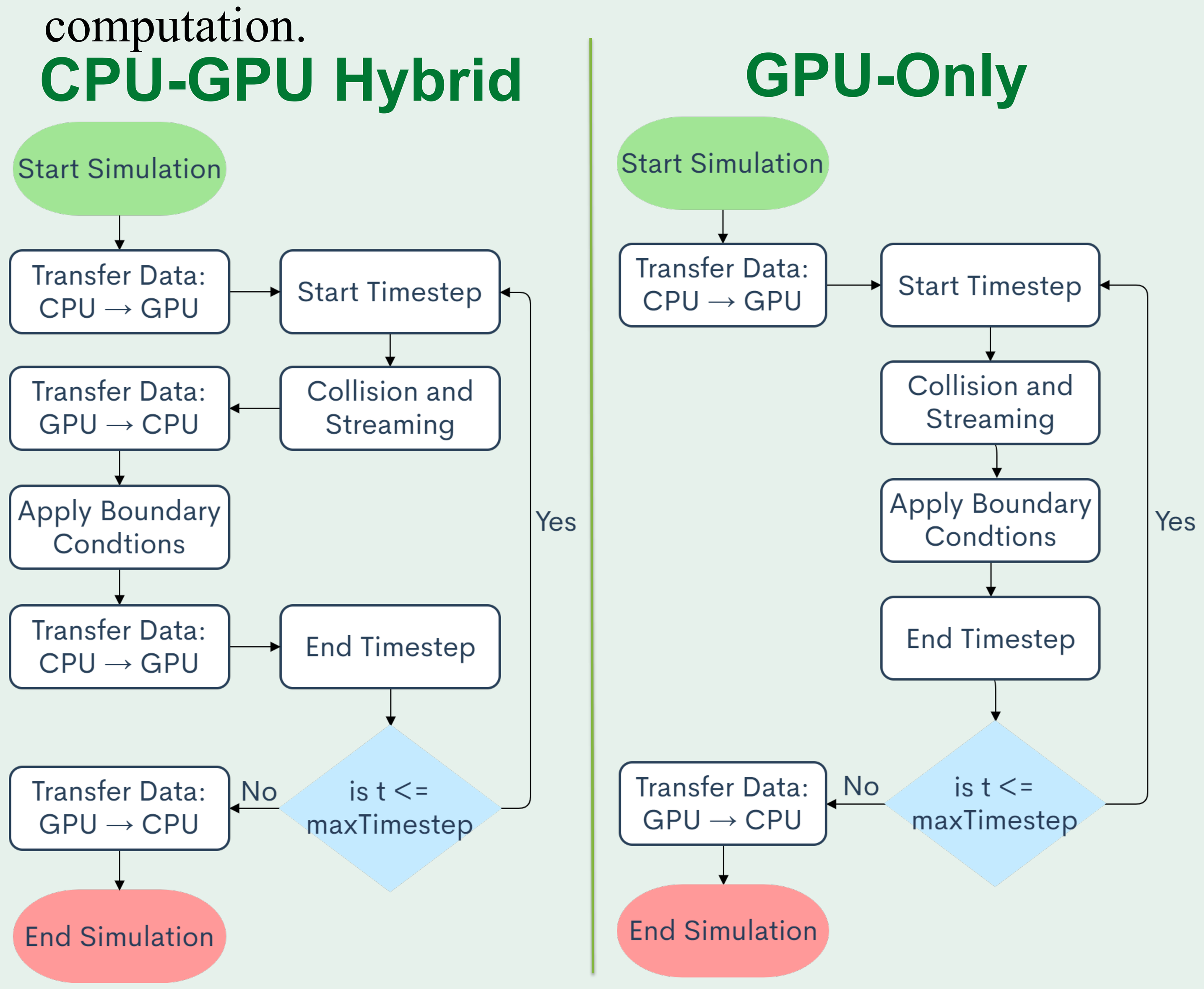


Figure 1. Data flow during simulation time-steps.

## Results

- We measure performance on Oak Ridge National Laboratory's Summit; focusing on the D3Q19 velocity set and using a common lattice Boltzmann simulation performance metric: millions of fluid lattice updates per second (MFLUPS).
- Two methods were used for computing particle distribution function information: two-grid, which reads from one array and writes to another, and one-grid, which writes to a single array but uses alternating operations in even and odd time-steps [2].

| | |
|---|---|
| Node Architecture | 2 IBM POWER9 CPUs<br>6 NVIDIA V100 GPUs |
| Simulation Sizes | 21,504 to 29,816,640 fluid points |
| Density Flow | From:<br>$\rho = 1.01$ to $\rho = 1.00$ |

Table 1: Experimental Configurations.

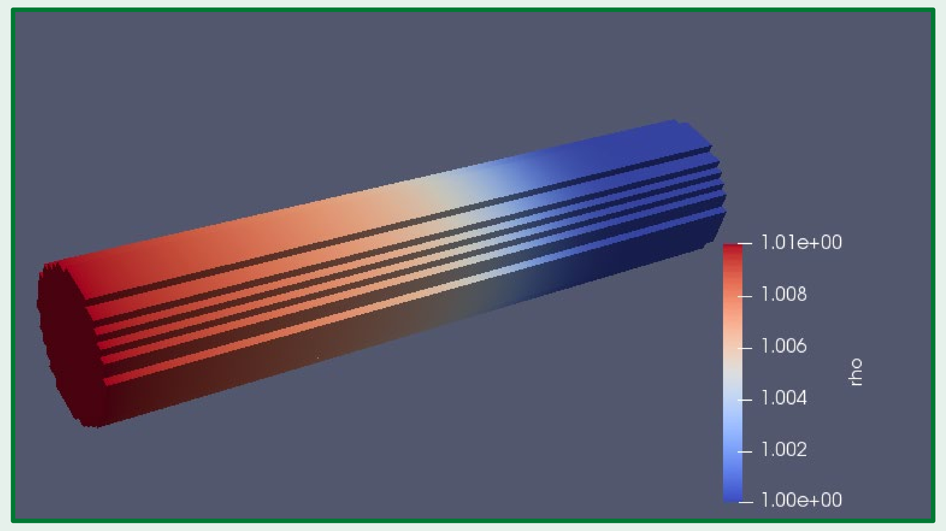


Figure 2. Density field after 100 time-steps.

- GPU-only version of the application improves as the simulation size grows because the benefit of increased parallelism provided by the GPU is not negated by excessive communication between CPU and GPU, providing speedup between 2x and 49x.

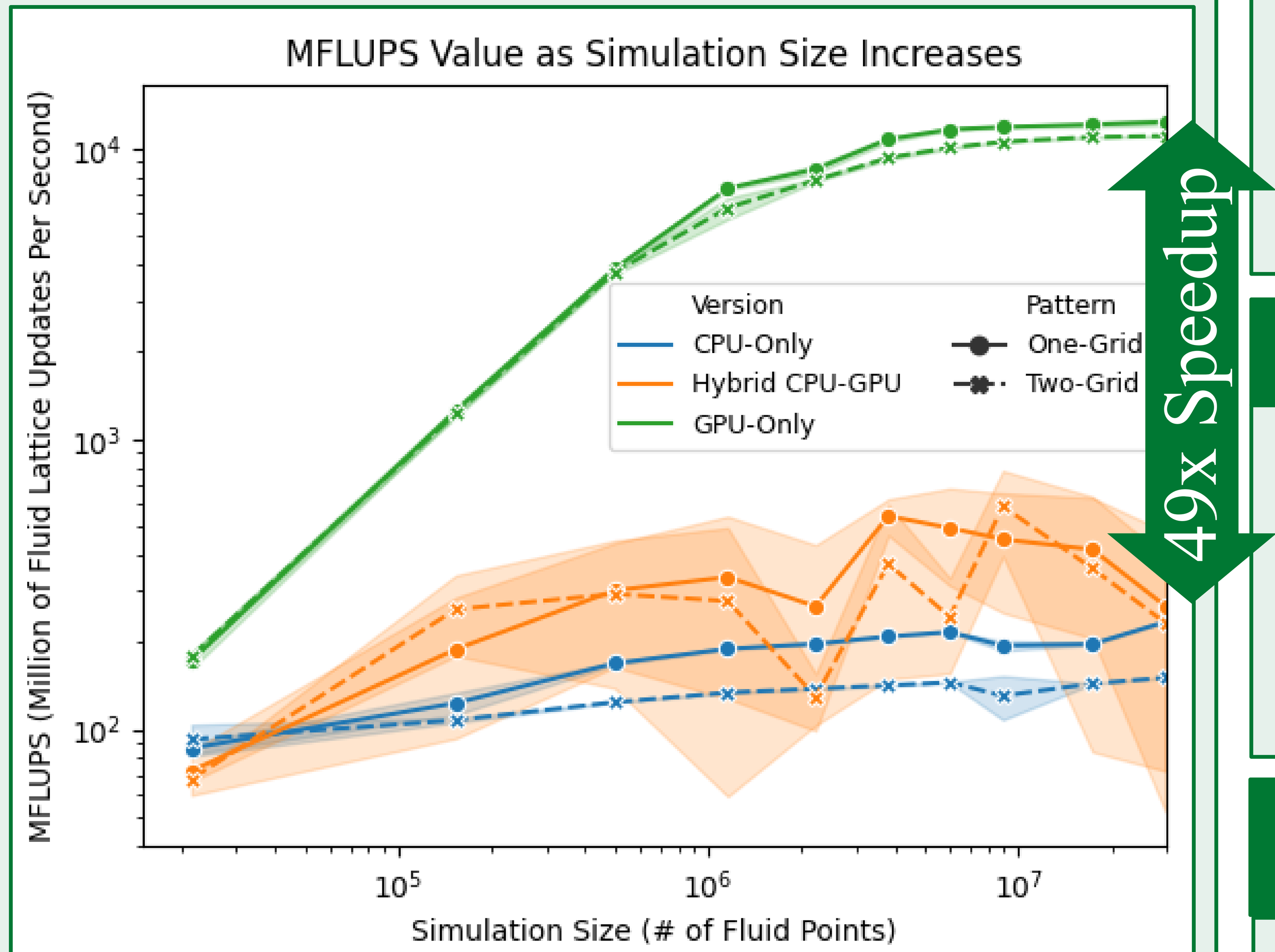


Figure 3. MFLUPS values as simulation size increases for one-grid and two-grid propagation patterns with std. deviations.

- Increased data motion results in unpredictable results during multiple runs, shown by the standard deviation of the CPU-GPU hybrid version's MFLUPS values.
- By applying boundary conditions and collision and streaming in a single location we minimize communication between CPU and GPU, resulting in improved reproducibility.
- GPU-only version reduces percentage of runtime spent on communication from 92% to 2%

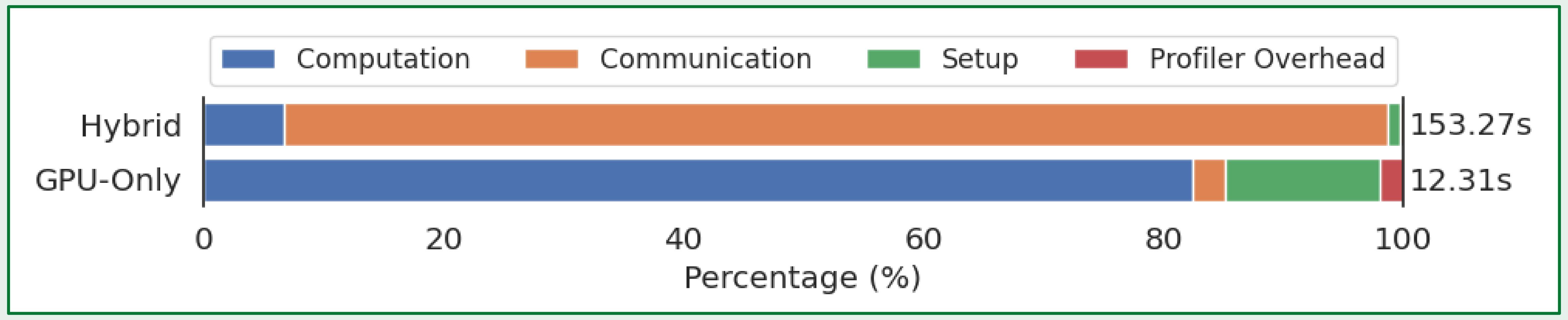


Figure 4. Percentages of runtime spent on simulation operations.

## Conclusions

- Communication overhead is an issue in the CPU-GPU hybrid version of our application leading to poor, unpredictable performance.
- Our version of the application offers speedup between 2x and 49x greater than other versions of the application.
- We reduce the runtime spent on communication from 92% to 2%, reducing the possibility of contention for communication resources.
- Reduced communication and increased parallelism from the GPUs greatly improve performance and prevent unpredictable results.

## References

[1] T. Krüger et. al. 2016.The Lattice Boltzmann Method - Principles and Practice.Springer. Etc.

[2] Markus et al Wittmann. 2013. Comparison of Different Propagation Steps forLattice Boltzmann Methods.Comput. Math. Appl.(2013), 924–935.

## Acknowledgements

Contact Information:
Griffin Dube
gdube@g.clemson.edu

Personal Webpage

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.