Members

Jasmine Brewer, Samuel Elliott, Matthew Gross, Aniq Shahid, & Li-yin Young

Optimizing Hydrodynamic Simulations of Quantum Fluids

CSCI 4576/5576 Project Proposal, Fall 2014

**Introduction**

The lattice Boltzmann method was first popularized for its success in mainstream computational fluid dynamics, but has recently also emerged as an important tool in the fields of physics, biology, and chemistry for understanding complex fluids [1]. Unfortunately, the range of physical problems which can be solved with the algorithm is currently limited by its high memory and computational requirements [2]. The capacity of the lattice Boltzmann method to solve scientific problems has therefore become an important problem in high performance scientific computing. The proposed project intends to tackle this problem directly by optimizing an existing scientific code that simulates the evolution of a quantum fluid for more efficient and scalable overall performance.

**Problem**

The naïve implementation of the lattice Boltzmann method which has been previously implemented is overly restrictive on the simulation sizes achievable with available resources, due at least in part to lack of parallelism and inefficient data and memory handling [3]. We propose to remedy these problems by implementing the following:

1. **Single-Core Optimization:** We will vectorize computations and adopt blocking wherever these result in improved performance and reevaluate data structures to improve memory-usage.
2. **OpenMP:** We will parallelize data-independent sections of the single-core optimized code using OpenMP.
3. **MPI:** We will implement MPI for running OpenMP-implemented functions in parallel and handling intercommunication. The final code will combine hybrid MPI and OpenMP model.
4. **Data Handling:** We will port IO to HDF5 to improve data write times, implement data compression, and improve data processing.
5. **Intel Phi:** We will extend the hybrid OpenMP/MPI implementation to take advantage of an architecture using the Xeon Phi coprocessors.
6. **Performance Comparison between Phi vs Xeon:** We intend to perform architecture-based comparative analysis in terms of speed, scalability and power usage between optimized implementations for Intel Phi and Xeon X5660 (Janus).

**Program Inputs/Outputs**

The input data for the program are the grid size, lattice resolution, time duration, and time resolution of the simulation, as well as a prescribed set of initial conditions which depend on the problem to be solved. The output data are the macroscopic variables (for example, local density and fluid velocity) at each time step.

**Problem Size**

The physical case for the 2D benchmark will be a lattice size of 1001 by 1001 with a standard deviation of 200 and 1000 time steps. The physical case for the 3D benchmark will be a lattice size of 251 by 251 by 251 with a standard deviation of 50 and 500 time steps. We will run the following accuracy and performance tests of weak scaling for the 2D case:

|  |  |  |  |
| --- | --- | --- | --- |
| Nx | Ny | Standard Deviation | Number of Time steps |
| 251 | 251 | 50 | 250 |
| 501 | 501 | 100 | 250 |
| 1001 | 1001 | 200 | 250 |

The accuracy and performance tests for the 3D case will be

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nx | Ny | Nz | Standard Deviation | Number of Time steps |
| 125 | 125 | 125 | 25 | 250 |
| 251 | 251 | 251 | 50 | 250 |
| 501 | 501 | 501 | 100 | 250 |

**Benchmark**

We will gauge the scaling performance by changing the lattice size and standard deviation by the same factor and leaving the time steps the same. We will then benchmark the physical case, for which the number of time steps must be increased by the same factor as the lattice size and standard deviation, and analyze how the code performs in terms of strong and weak scaling.

**Analysis**

We implement these techniques and methods with the goal of producing an output code which can solve scientific problems that the input code could not. The performance of each code version will be compared to the original code performance so as to evaluate exactly how effective our changes are and provide us with insight as to what could still be improved. By adopting this incremental development methodology, we aim to be able to correct mistakes and gain foresight into future changes in order to maximize both efficiency and code development. We intend to demonstrate our achievement by producing the following:

1. **Benchmarks** indicating the performance enhancement due to solving each significant sub-problem. We plan to do benchmarking at the beginning of the project and after each new implementation. Performance improvement will be gauged by running the code on INTEL Phi.
2. **Detailed profiling analysis** will be done at the beginning of the project and after the completion of each major project task. Profiling will be done using PerfSuite and Tau.
3. **Scientific Results.** We will finally demonstrate our code’s scalability and scientific capacity by simulating decaying turbulence [4], a simple-to-implement but computationally-intensive and interesting physical system.

**References**

1. Shiyi, Chen and Gary D. Doolen. “Lattice Boltzmann method for fluid flows.” *Annual Review of Fluid Mechanics*. 1998, 30.
2. Mattila, K, et. al. “Comparison of implementations of the lattice-Boltzmann method.” *Computers and Mathematics with Applications*. Vol. 55 Issue 7, April 2008, 1514-1524.
3. Latt, Jonas. “Hydrodynamic Limit of the Lattice Boltzmann Equations.” *Thesis, University of Geneva*. 2007.
4. Succi, Sauro. “The Lattice Boltzmann Equation for Fluid Dynamics and Beyond.” *Oxford University Press .* 2001.