HIGH PERFORMANCE COMPUTING PROJECT REPORT

Parallelization of Lattice Boltzmann Method Solver and 3D Mean Blur Stencil

# 1. Problem Statement

The project focuses on the parallelization of a Lattice Boltzmann Method (LBM) solver combined with a 3D mean blur stencil operation.   
The LBM simulates fluid flow through discrete lattice structures, while the 3D mean blur stencil performs spatial averaging on 3D data grids.   
Both involve intensive grid-based computations with local data dependencies, making them excellent candidates for parallelization.

# 2. Sequential Implementation

A sequential C++ implementation was developed to serve as the baseline for performance comparison.   
It executes correctly but suffers from high computational intensity due to nested loop structures.  
  
Key characteristics:  
- Written in standard C++ for portability  
- Implements complete LBM simulation (collision and streaming)  
- Includes 3D mean blur stencil with configurable kernel sizes  
- Provides accurate baseline results for validation

# 3. Performance Analysis (Sequential Profiling)

Profiling with gprof identified computational hotspots:  
  
- LBMSolver::simulate(...) → 88.3% runtime  
 • Dominant bottleneck with nested grid updates  
 • Excellent parallelization potential due to independent cell operations  
  
- meanBlurStencil(...) → 9.06% runtime  
 • Performs local neighborhood averaging  
 • Parallelizable across grid cells  
  
- Other functions (<3% runtime) → Initialization, I/O, utilities, not worth parallelizing.  
  
Conclusion: LBMSolver::simulate is the critical bottleneck and primary target for optimization.

# 4. Chosen Parallelization Platforms

Two parallelization approaches were selected:  
  
OpenMP (Shared-memory CPU parallelization)  
- Integrates with C++ using pragmas  
- Ideal for loop-based parallelization  
- Utilizes multi-core CPUs efficiently  
  
CUDA (GPU acceleration)  
- Provides massive parallelism for stencil operations  
- Effective for LBM and blur operations  
- Requires optimized memory hierarchy and thread block organization

# 5. Conclusion

This project applied sequential implementation, profiling, and parallelization on OpenMP and CUDA.   
LBMSolver::simulate was identified as the dominant bottleneck, with both CPU and GPU strategies offering substantial performance improvements.   
The combination of OpenMP and CUDA ensures scalability and high performance across architectures.