Parallelising d2q9-bgk.c with MPI

George Herbert
cj19328@bristol.ac.uk

11 April 2022

Abstract

d2q9-bgk.c implements the Lattice Boltzmann method (LBM) to simulate a fluid density on a lattice. This report analyses the techniques I utilised to parallelise d2q9-bgk.c with MPI, and port d2q9-bgk.c to a GPU with OpenCL.

1 Single Program, Multiple Data

Single program, multiple data (SPMD) is a form of parallelism in which independent processes run the same program. Message Passing Interface (MPI) is a specification for a library interface for passing messages between processes.

1.1 Hypothesis

In my OpenMP implementation, I achieved a substantial performance improvement with 28 threads. This was primarily because each thread ran on one core of a single BC4 compute node; therefore, 28 iterations of the inner loop in the `timestep` function executed in parallel. However, OpenMP was designed for shared-memory parallelism, and so was unable to utilise more than one node of BC4, which was a severe restriction considering BC4 contained hundreds of nodes. Therefore, I hypothesised an implementation of d2q9-bgk.c that used MPI to run on multiple processes across multiple nodes of BC4 in parallel would provide a more substantial performance improvement than the OpenMP implementation.

1.2 Implementation

I opted to use my final implementation of d2q9-bgk.c before single instruction, multiple data (SIMD) vectorization as a starting point.

1.3 Results

Table 1: Execution times with the 52 process MPI implementation and speedup over both the prior and 28 thread OpenMP implementation

Grid Size	Time (s)	Speedup	
		Prior	OpenMP
128×128			
128×256			
256×256			
1024×1024			

Each time was an average of five runs on a BlueCrystal Phase 4 (BC4) compute node—a Lenovo nx360 M5, which contained two 14-core 2.4 GHz Intel E5-2680 v4 (Broadwell) CPUs and 128 GiB of RAM [1].

2 Experiments

Having produced an SPMD implementation of d2q9-bgk.c with distributed memory parallelism, I ran several experiments to further optimise my program.

2.1 Vectorization

Table 2: Execution times with the first vectorization option and speedup over the prior implementation

Grid Size	Time (s)	Speedup
128×128		
128×256		
256×256		
1024×1024		

Table 3: Execution times with the second vectorization option and speedup over the prior implementation

Grid Size	Time (s)	Speedup
128×128		
128×256		
256×256		
1024×1024		

3 Hybrid MPI and OpenMP

Table 4: Execution times with the hybrid implementation and speedup over the prior implementation

Grid Size	Time (s)	Speedup
128×128		
128×256		
256×256		
1024×1024		

4 GPU Programming

5 Conclusion

References

[1] *BlueCrystal technical specifications.* URL: `https://www.bristol.ac.uk/acrc/high-performance-computing/hpc-systems-tech-specs/` (visited on 19/02/2022).