Bidimensional Lattice Boltzmann Implementation using CUDA

Antonio Lucas N. de O. Barros
Cátia Souza do Nascimento
João Lima
CMP 557 - 2010/1
Programming Massively Parallel Processors using CUDA
Instituto de Informática
PPGC - UFRGS

Outline

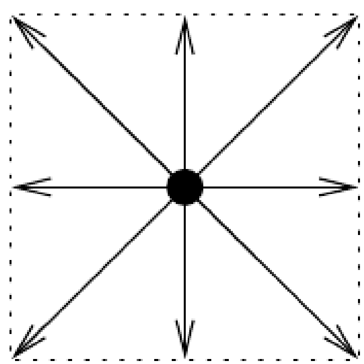
- Lattice Boltzmann
- Implementation
- Experimental Results
- Conclusion
- Future Work

Lattice Boltzmann

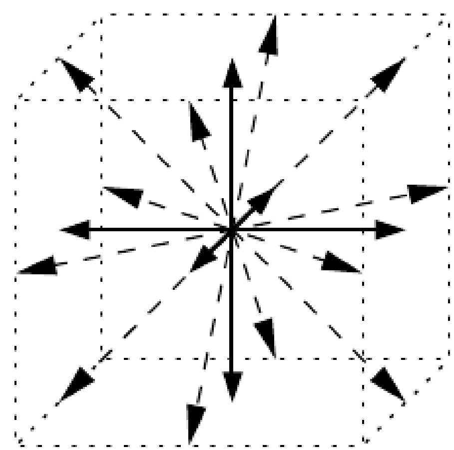
- Iterative Numeric Method
- Mesoscopic
- Relation with LGA method
  - Particle Representation:
    - **MLB** Uses real distribuitions
    - **LGA** uses boolean distributions

Lattice Boltzmann

- Lattice Structures
  - **D2Q9 – 2 Dimensions, 9 directions**
  - D3Q19- 3 Dimensions,19 directions



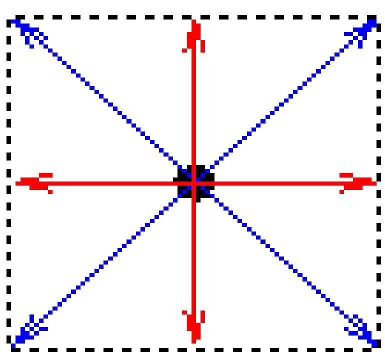
D2Q9

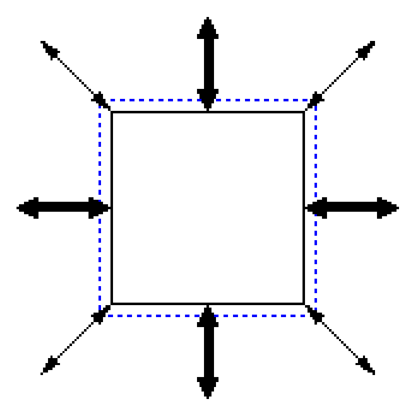


D3Q19

Implementation

- **D2Q9** – 2 Dimensions, 9 directions
- Based on: Schepke and Maillard (2009) sequential implementation





Implementation

- C++ code + CUDA kernels (4)
- Each direction is a thrust vector
  - thus, 9 thrust vectors
- Between kernel calls, data remains in GPU memory
  - CPU-GPU copies before/after the iteractions

Implementation

LBM(lb, input, output)

lb ← **read** obstacles from *input*
lb ← **read** parameters from *input*
for i ← 0 **to** lb.Maximum_iteractions() **do**
    lb.Redistribute_kernel()
    lb.Propagate_kernel()
    lb.Bounceback_kernel()
    lb.Relaxation_kernel()
lb.Write_Results(*output*)

Implementation - Redistribute

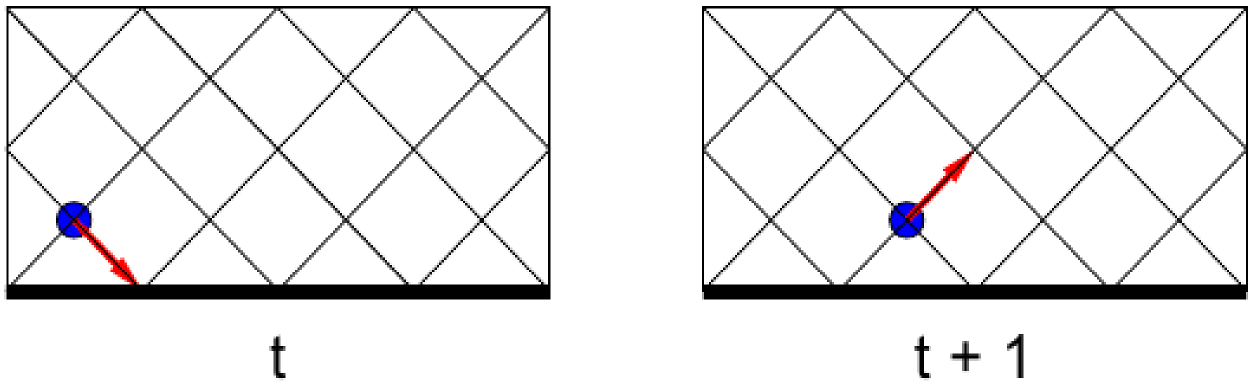
- Calculate the macroscopic density and speed from the values of each lattice point
- Partitioning by y axis
  - each thread process a line

Implementation - Propagate

- Propagate the particles distribution to all neighboring cells
- Blocked partitioning by x and y axis

Implementation - Bounceback

- Represents Boundary Conditions
- Invert the speed vector direction when collisions occur
- Blocked partitioning by x and y axis



Implementation - Relaxation

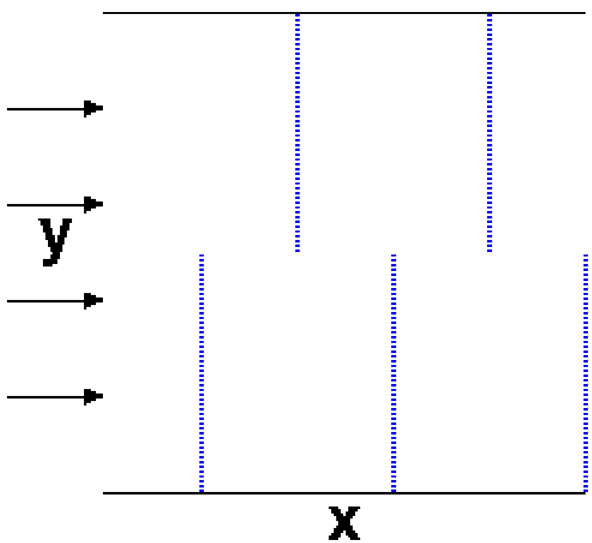
- Use the equilibrium value to apply in the distribution function of each lattice point
- Blocked partitioning by x and y axis

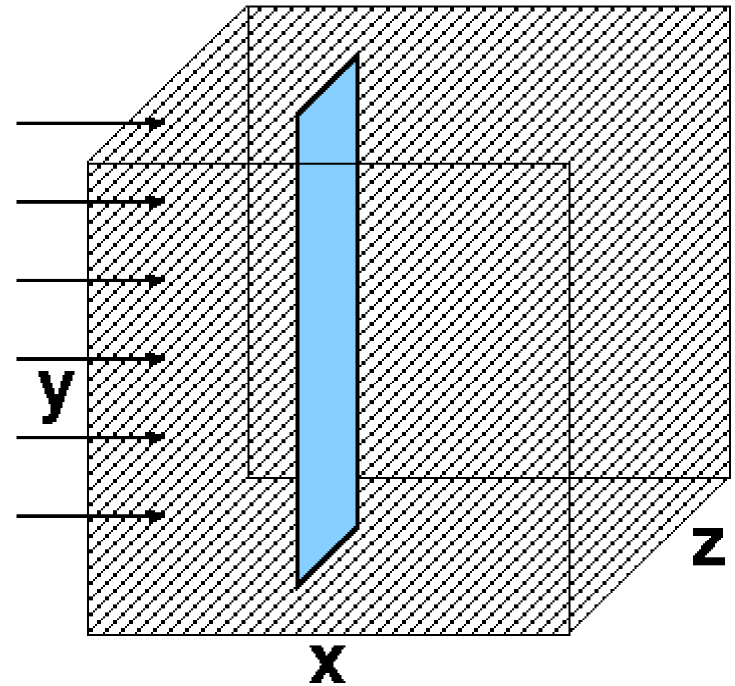
Experimental Results

- Hardware Used:
  - Core i7 @ 2.80 Ghz / GTX480 (Fermi)
- Graphics with execution time and speedup
- Three different inputs
  - lattice of 30x20
  - lattice of 200x50
  - lattice of 512x512

Experimental Results

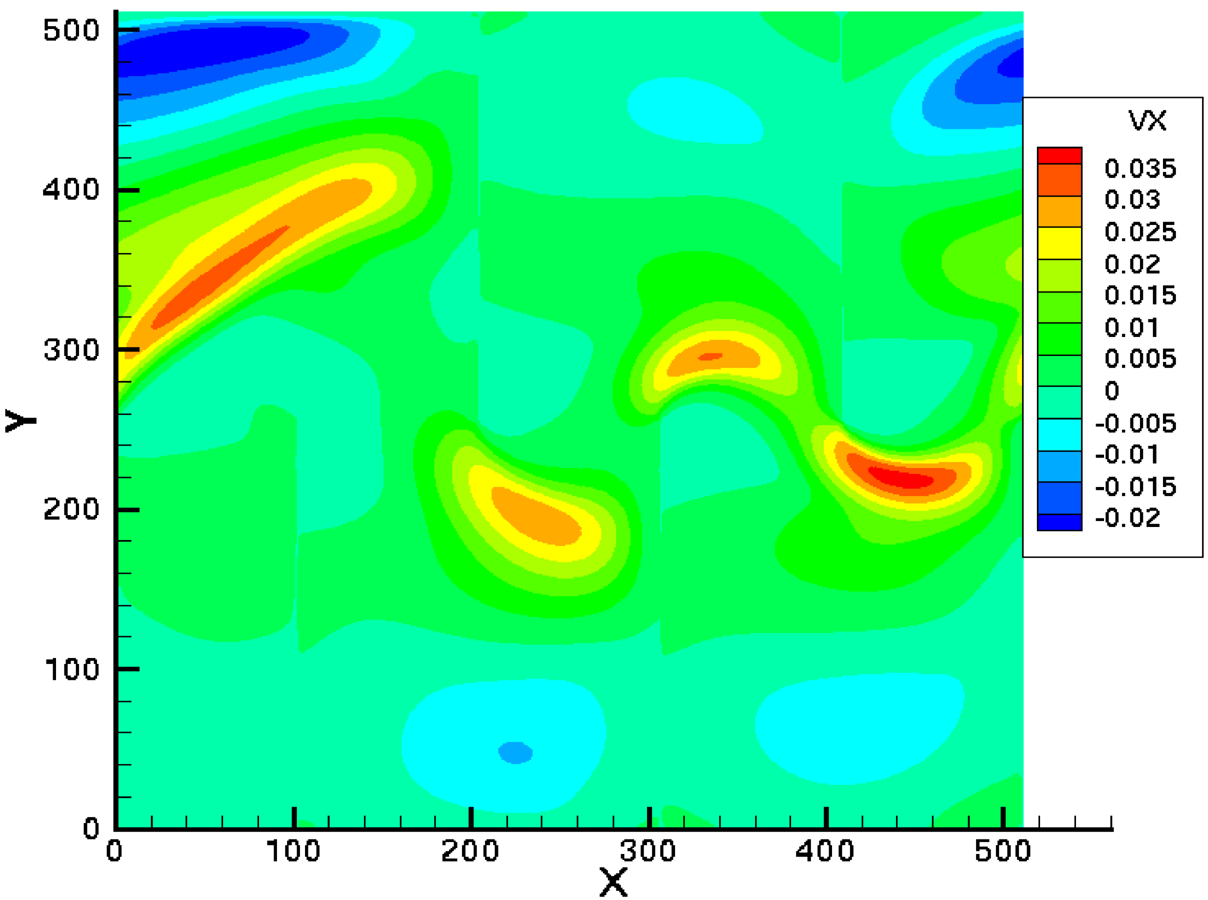
- The obstacle in our 2D tests, and 3D example



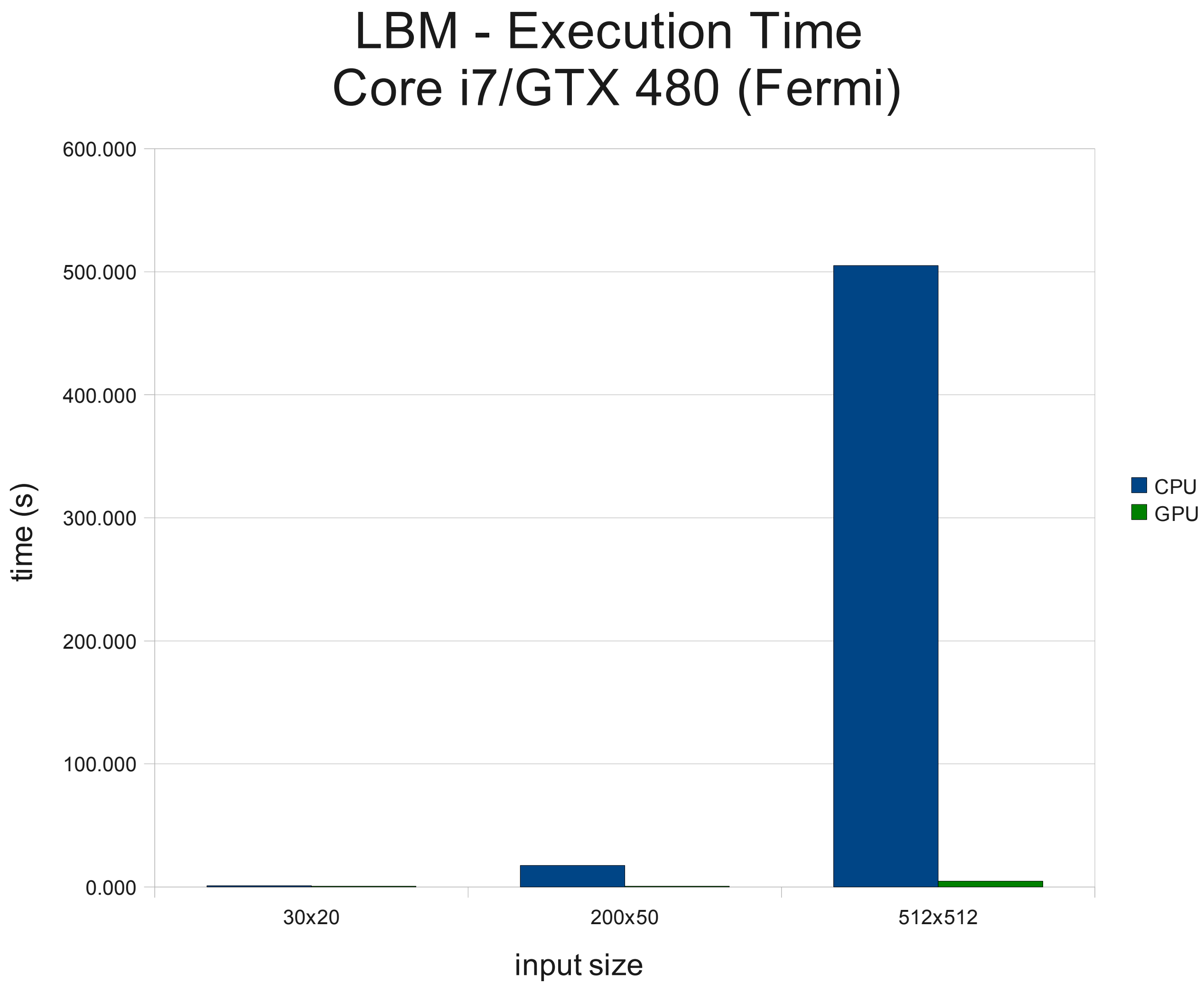


Experimental Results

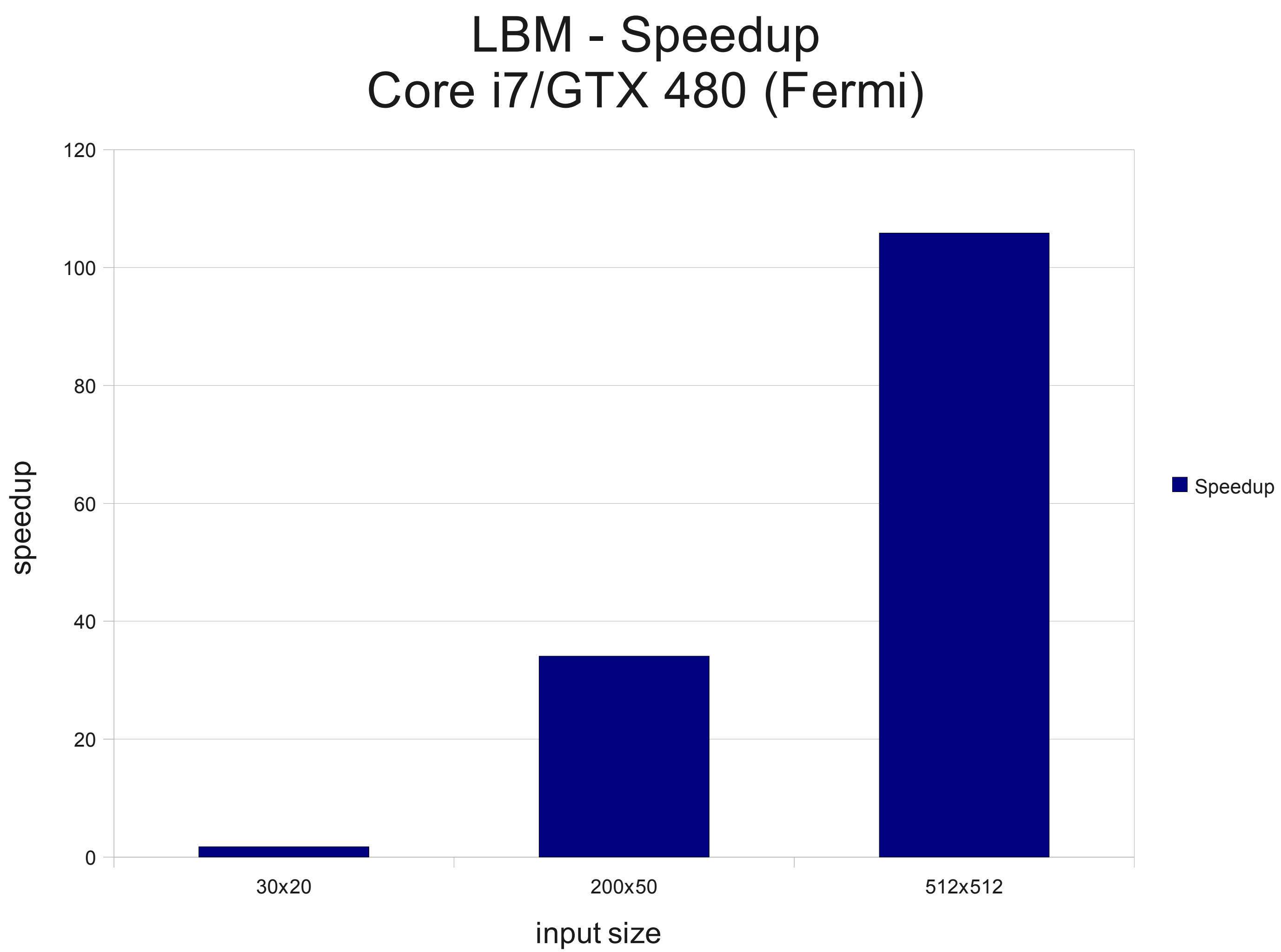
- Output example (512x512 with 90 iteractions)



Experimental Results



Experimental Results



Conclusion

- Lattice Boltzmann can be efficient in GPUs
- Different result values (error 10^{-4})
  - the precision of GPU was **float**, CPU **double**
- Coding in CUDA is difficult with many variables
  - in our case, dimensions

Future Work

- Optimize kernel functions
  - memory accesses
  - arguments
  - etc
- Use Fermi shared memory
- D3 version (D3Q15 or D3Q19)

References

- Schepke and Maillard (2009). **Parallel Lattice Boltzmann Method with Blocked Partitioning.** *International Journal of Parallel Programming,* **2009***, 37, 593-611*.

Bidimensional Lattice Boltzmann Implementation using CUDA

Antonio Lucas N. de O. Barros
Cátia Souza do Nascimento
João Lima

CMP 557 - 2010/1
Programming Massively Parallel Processors using CUDA

Instituto de Informática
PPGC - UFRGS