Realtime Computational Fluid Dynamics Simulations Using the Lattice Boltzmann Method

Thomas Georgiou

Thomas Jefferson High School for Science and Technology Computer Systems Lab

January 28, 2010

Uses for Fluid Dynamics

- Computer Graphics
- Aerodynamics and Engineering
- Meteorology
- Oceanography
- Plasma Physics
- National Security
- and more

The Boltzmann Equation

$$f(x + vt, v, t) = f(x, v, t) + \Omega(x, v, t)$$

Conists of:

- Streaming
- Collisions

Discretization of Phase Space

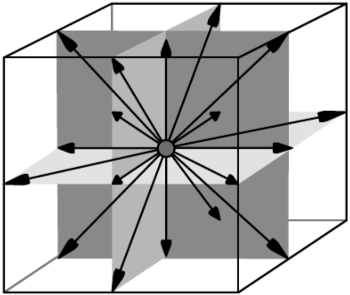
In order to solve the Boltzmann equation numerically, the domain must be split up into discrete components. This includes space, velocity, and time.

Naming Scheme

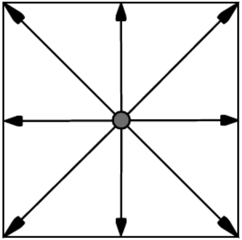
DnQm

- n is the number of space dimensions
- m is the number of velocities

Lattice and Velocity Configurations



D3Q19



D2Q9

The Stream Step

$$f(x + e_i, e_i, t + dt) = f(x, e_i, t)$$

Boundary:

$$f(x, e_{\bar{i}}, t + dt) = f(x, e_i, t)$$

The Collision Step

The BGK Collision Operator

$$\Omega_{BGK} = \frac{f - f_{eq}}{\tau}$$

Collisions tend to push the system towards local equilibrium.

f_{eq} is the equilibrium distribution function

Low Mach number expansion of the Maxwell Boltzmann distribution:

$$\sqrt{\frac{m}{2\pi kT}} e^{\frac{-mv^2}{2kT}} \approx w_i(\rho + 3e_i \cdot u - \frac{3}{2}u^2 + \frac{9}{2}(e_i \cdot u)^2)$$

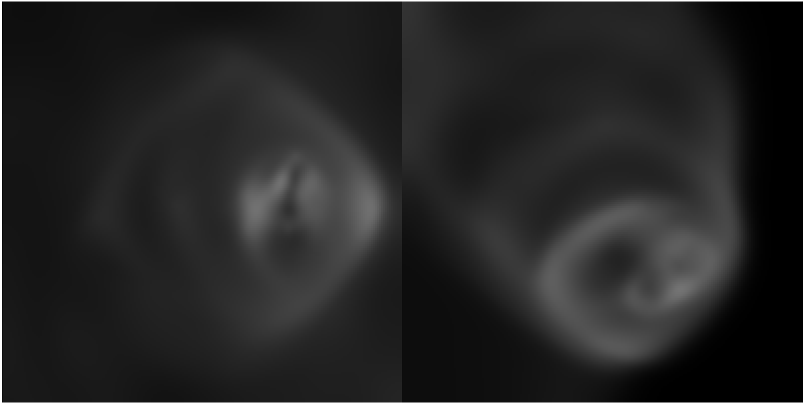
Relaxed towards equilibirum with:

$$f(x, e_i, t + dt) = (1 - \omega)f(x, e_i, t + dt) + \omega f_i^{eq}$$

Software Used for Implementation

- C
- OpenGL
- OpenMP
- MPI
- Qt4

Visualization - Density Plot

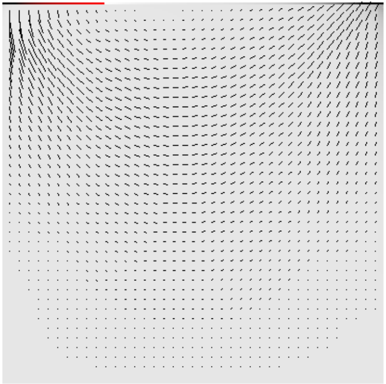


Visualization - Velocity Vector Field

- Compute velocity field from fluid distribution functions

$$v = \frac{u}{\rho}$$

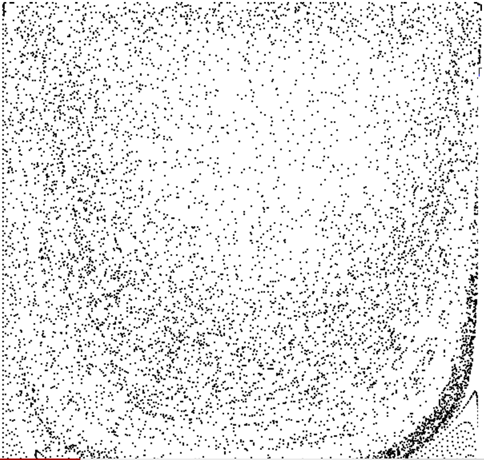
- Draw grid of vectors along velocity



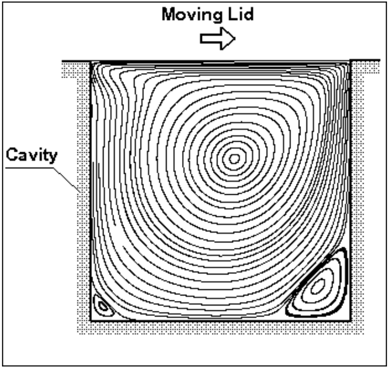
Visualization - Tracer Particles

- Particles placed in the fluid
- Advected using Euler's method

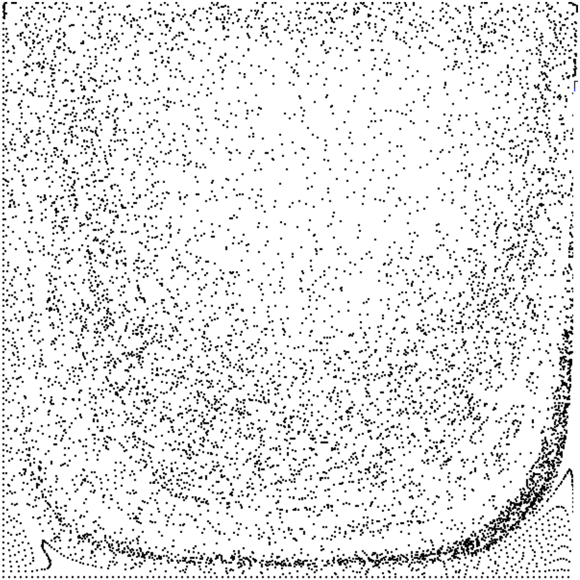
$$x(t + dt) = x(t) + v(x, y, t)$$



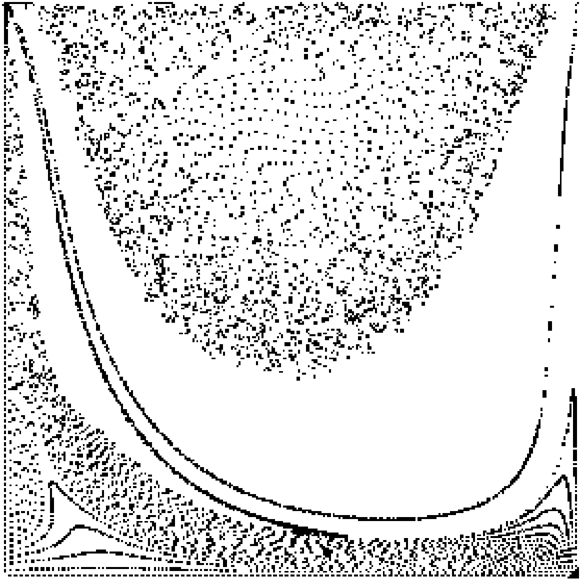
Current Results - Lid Driven Cavity



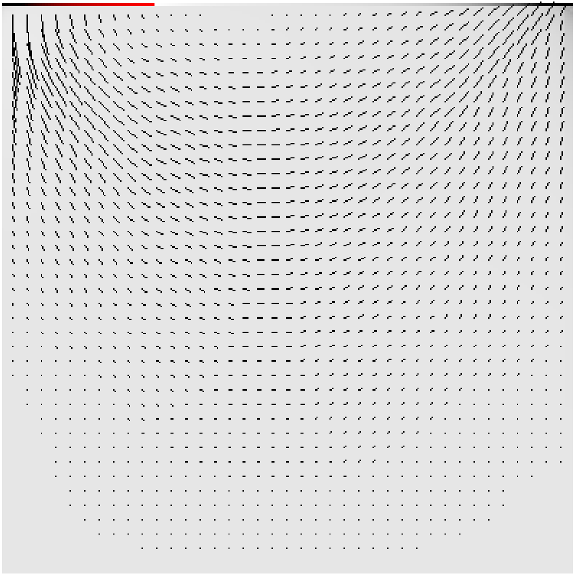
Lid Driven Cavity



Lid Driven Cavity



Lid Driven Cavity



Results - Performance

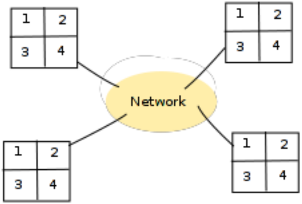
Performance metric = MLUPS (Mega Lattice Updates per Second)
Single threaded performance:

Core 2 X9650	4.64 MLUPS
Xeon E5520	3.84 MLUPS

Multi threaded performance scales almost linearly under shared memory systems using OpenMP. Using 4 threads on a Core 2 Quad X9650, 16.26 MLUPS are acheived.

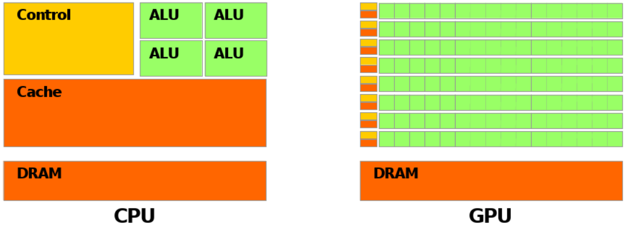
Next Steps - Performance

- The next step in improving performance is making the program parallelize across a network of nodes using MPI.
- MPI will be used with OpenMP.
- OpenMP - intra node parallelism
- MPI - inter node parallelism
- Initial results exceed 66 MLUPS using two nodes.



Next Steps - CUDA

- GPUs are massively data parallel - SIMD
- Problem is very data parellel
- Each lattice update can be performed simultaneously
- CPU and GPU version will be connected together via MPI for improved performance



Next Steps - Simulation

- More verification
- Lid Driven Cavity - quantitative results
- Reynolds number
- Flow Past a Cylinder
- Free Surfaces