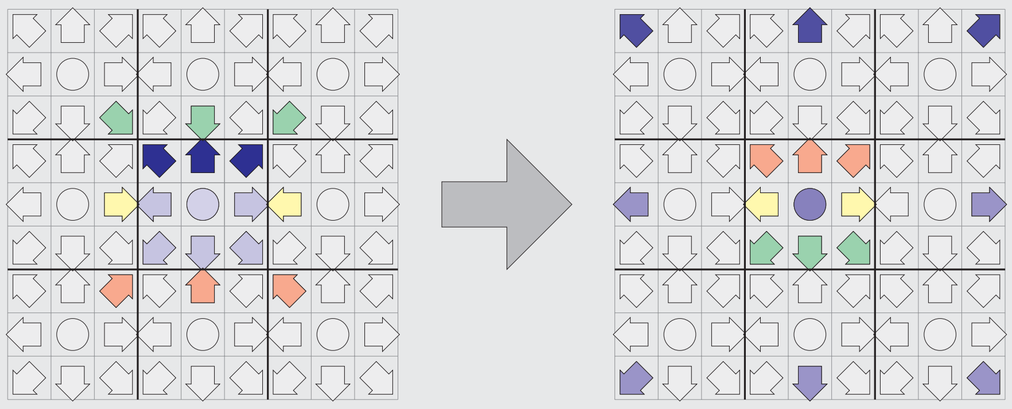
Soft matter lattice Boltzmann simulations on Graphics Processing Units

Michał Januszewski, Marcin Kostur

Institute of Physics, Dept. of Mathematics, Physics and Chemistry, University of Silesia, Katowice, Poland

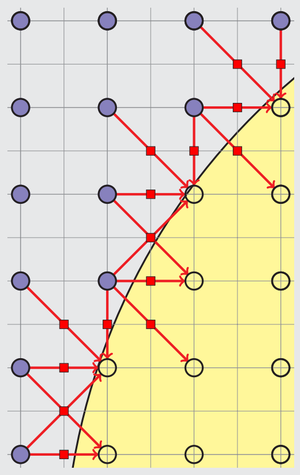
The Lattice Boltzmann Method



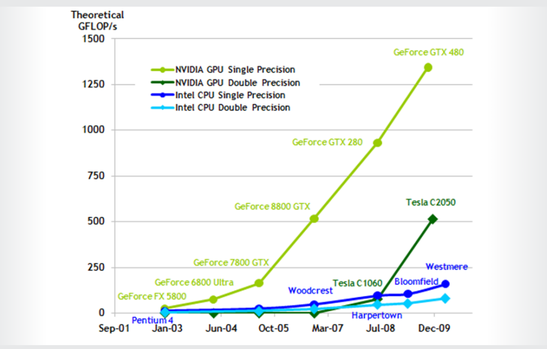
- Fluid described at the mesoscopic level by a set of velocity distributions f_α
- Density $\rho_i = \sum_\alpha f_\alpha(\vec{x}_i, t)$ and momentum $\rho_i \vec{v}_i = \sum_\alpha \vec{c}_\alpha f_\alpha(\vec{x}_i, t)$
- Basic algorithm (BGK approximation):

$$f_\alpha(\vec{x}_i + \vec{c}_\alpha, t+1) = f_\alpha(\vec{x}_i, t) - \frac{f_\alpha(\vec{x}_i, t) - f_\alpha^{(\mathrm{eq})}(\rho_i, \vec{v}_i)}{\tau}$$

- Particle-fluid coupling:
  - Moving mid-link bounce-back rules
  - Solid particles (no internal fluid)



The GPU Revolution



Courtesy of NVIDIA Inc.

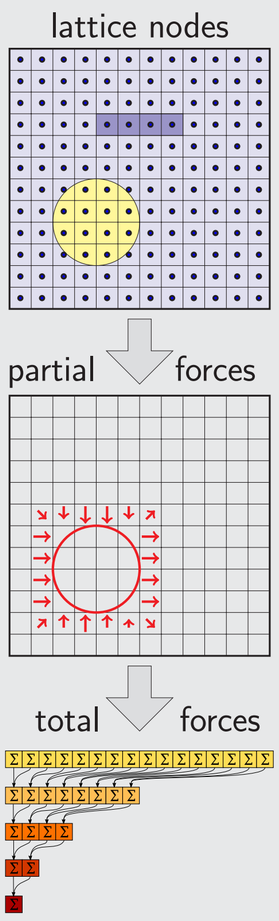
A single modern (NVIDIA) Graphics Processing Unit (GPU):

- Hundreds of computational cores programmable using CUDA C/OpenCL
- Computational power > 1 **TFLOPS**
- Several gigabytes of main memory (global memory)
- Tens of kilobytes of fast on-chip memory (shared memory)
- Programming model based on **kernels**, executed in **threads** grouped into **blocks**
- Inter-thread communication possible **within a single block** only

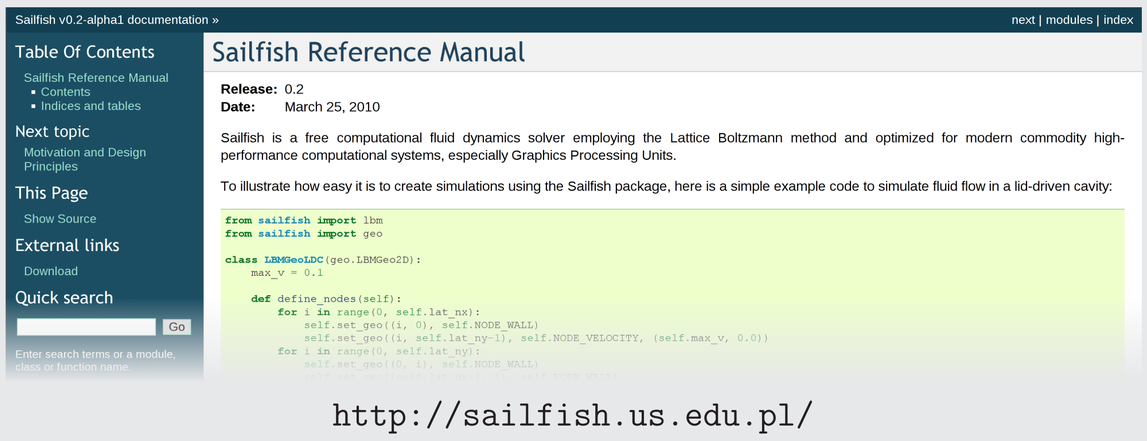
LB-Particle Models on GPUs

The A-B access pattern (two copies of the simulation domain) is used.

1. Collide and stream (1D blocks of several hundred nodes, 1 thread – 1 node).
   - 1.1 Apply boundary conditions if necessary.
   - 1.2 Apply the collision rule (BGK/MRT).
   - 1.3 For particle boundary nodes, save force and torque due to momentum transfer to a vector field in global memory.
   - 1.4 Use shared memory for the propagation step in the X dimension, propagate the remaining dimensions directly in global memory.
2. For every particle (in parallel, different kernels for different particles, 1 thread – 1 node):
   - 2.1 Parallel reductions to calculate the partial force and torque on a particle, coming from a smaller area of the simulation domain.
   - 2.2 Parallel reduction to sum these partial results and obtain the total force and torque.
3. Move the particles (1 thread – 1 particle).
4. For every particle (as in step 2):
   - 4.1 Update the geometry if necessary due to particle movement.
   - 4.2 Parallel reductions to obtain theh total force like in step 2.



The Sailfish Framework



Sailfish v0.2-alpha1 documentation » next | modules | index

Table Of Contents
- Sailfish Reference Manual
  - Contents
  - Indices and tables

Next topic: Motivation and Design Principles

This Page: Show Source

External links: Download

Quick search: Go

Enter search terms or a module, class or function name.

Sailfish Reference Manual

Release: 0.2
Date: March 25, 2010

Sailfish is a free computational fluid dynamics solver employing the Lattice Boltzmann method and optimized for modern commodity high-performance computational systems, especially Graphics Processing Units.

To illustrate how easy it is to create simulations using the Sailfish package, here is a simple example code to simulate fluid flow in a lid-driven cavity:

```
from sailfish import lbm
from sailfish import geo

class LBMGeoLDC(geo.LBMGeo2D):
    max_v = 0.1

    def define_nodes(self):
        for i in range(0, self.lat_nx):
            self.set_geo((i, 0), self.NODE_WALL)
            self.set_geo((i, self.lat_ny-1), self.NODE_VELOCITY, (self.max_v, 0.0))
        for i in range(0, self.lat_ny):
            self.set_geo((0, i), self.NODE_WALL)
```

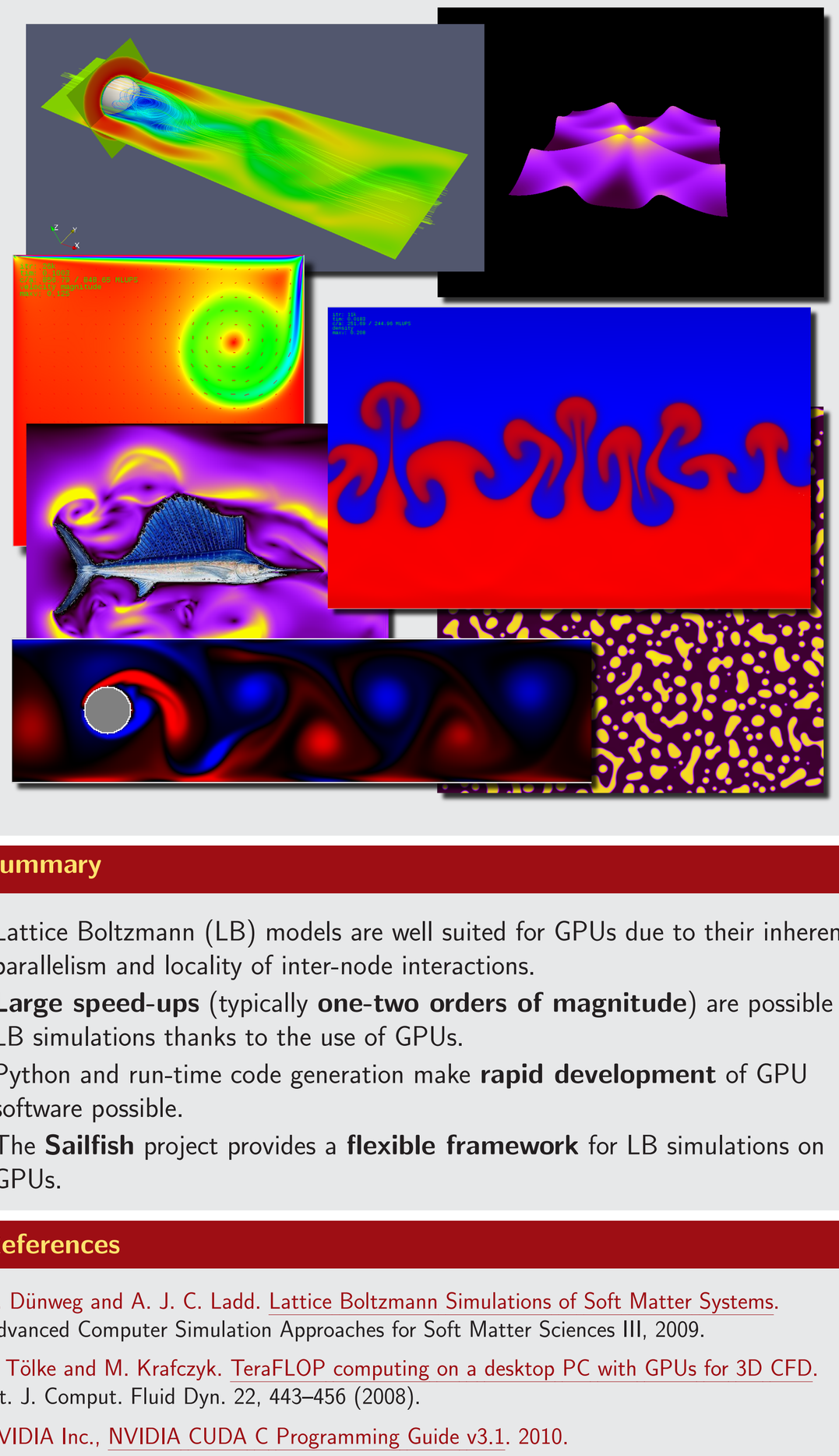
http://sailfish.us.edu.pl/

Sailfish is an open source lattice Boltzmann simulation framework written in Python. It employs **run-time code generation** and **symbolic calculation** techniques to automatically generate code for the GPU. The code is generated from Mako templates, and dynamically compiled using PyCUDA/PyOpenCL.

Key features:

- Generated code is automatically optimized
- CUDA and OpenCL backends
- BGK and MRT models in 2D and 3D
- Single and double precision calculations
- Single and binary fluid models
- Built-in on-line visualization modules
- Fluid-structure interactions using the moving boundary method

Sample Simulations



Summary

- Lattice Boltzmann (LB) models are well suited for GPUs due to their inherent parallelism and locality of inter-node interactions.
- **Large speed-ups** (typically **one-two orders of magnitude**) are possible in LB simulations thanks to the use of GPUs.
- Python and run-time code generation make **rapid development** of GPU software possible.
- The **Sailfish** project provides a **flexible framework** for LB simulations on GPUs.

References

- B. Dünweg and A. J. C. Ladd. Lattice Boltzmann Simulations of Soft Matter Systems. Advanced Computer Simulation Approaches for Soft Matter Sciences III, 2009.
- J. Tölke and M. Krafczyk. TeraFLOP computing on a desktop PC with GPUs for 3D CFD. Int. J. Comput. Fluid Dyn. 22, 443–456 (2008).
- NVIDIA Inc., NVIDIA CUDA C Programming Guide v3.1. 2010. http://www.nvidia.com/cuda