CCDSC 2016

On a Novel Method for High Performance Computational Fluid Dynamics

Christian Obrecht

Energy and Thermal Sciences Centre of Lyon (CETHIL)
Department of Civil Engineering and Urban Planning
National Institute of Applied Sciences of Lyon (INSA-Lyon)

October 6, 2016

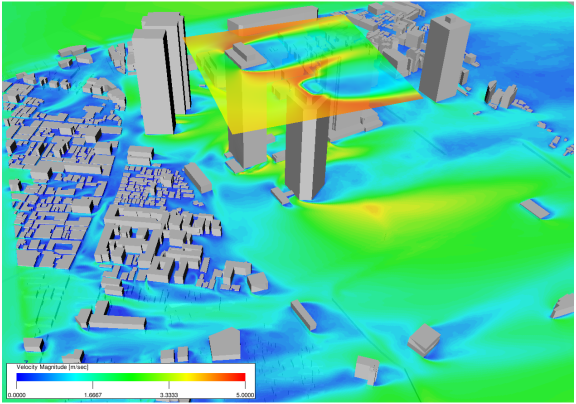
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

CETHIL UMR 5008

Outline

I – Motivation

Areas of interest: Urban physics



Margheri and Sagaut, 2014

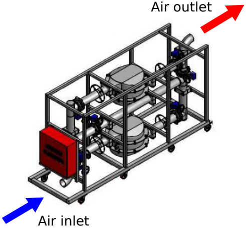
Urban micro-climate, pedestrian wind comfort, pollutant dispersion. . .

Areas of interest: Thermal energy storage

Shell and tube heat exchanger

Latent heat storage (phase change materials).

Zeolite beads



Sorption and/or chemical heat storage.

Computational Fluid Dynamics

The previous engineering applications rely heavily on CFD simulations.

- Multi-physics models.
- Complex geometries.
- $\mathcal{O}(10^9)$ fluid cells.
- Physically relevant simulation times.

Technical issues:

- Multi-physics commercial codes (e.g. Fluent) are expensive and do not scale over $\mathcal{O}(10^2)$ cores.
- Open CFD codes (e.g. code Saturne) are not designed for accelerators.

Unstructured vs Cartesian meshes

Unstructured

- Body fitting mesh.
- Time consuming generation process.
- Isotropy is an issue.
- Irregular data access pattern.

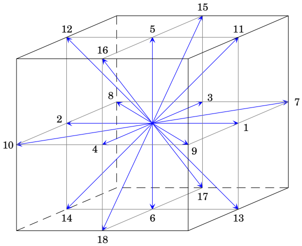
Cartesian

- Trivial meshing.
- GPU-friendly data layout.
- Hierarchical structure is often needed.

Lattice Boltzmann method

- Discretized version of the Boltzmann equation recovering the solutions of the Navier–Stokes equation.
- Regular Cartesian grid of mesh size δx with constant time step δt.
- Finite set of particular densities f_α associated to particular velocities $\boldsymbol{\xi}_\alpha$.
- Collision operator Ω (usually explicit).

$$\big|f_\alpha(\boldsymbol{x} + \delta t \boldsymbol{\xi}_\alpha, t + \delta t)\big\rangle - \big|f_\alpha(\boldsymbol{x},\, t)\big\rangle = \Omega \big|f_\alpha(\boldsymbol{x},\, t)\big\rangle$$



$$\rho = \sum_\alpha f_\alpha$$

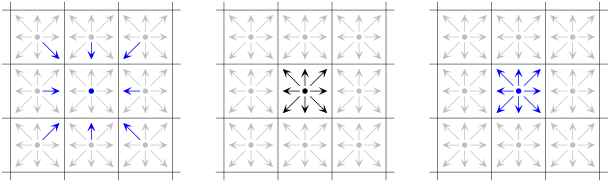
$$\rho \boldsymbol{u} = \sum_\alpha f_\alpha \boldsymbol{\xi}_\alpha$$

Pull formulation of the LBM

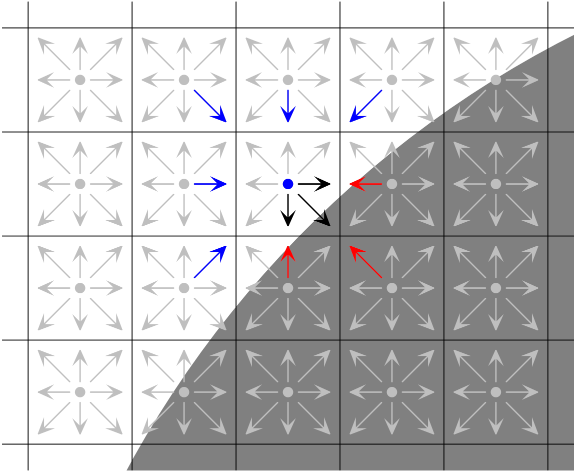
Two-step formulation of LBM: propagation (1) followed by collision (2).

$$\left|f_\alpha(\boldsymbol{x}, t+\delta t)\right\rangle = \left|f^*_\alpha(\boldsymbol{x}-\delta t\boldsymbol{\xi}_\alpha, t)\right\rangle \tag{1}$$

$$\left|f^*_\alpha(\boldsymbol{x}, t+\delta t)\right\rangle = \left|f_\alpha(\boldsymbol{x}, t+\delta t)\right\rangle + \Omega\left|f_\alpha(\boldsymbol{x}, t+\delta t)\right\rangle \tag{2}$$



Solid-fluid interface



Simple bounce-back boundary condition

LBM pros and cons

Pros

- Explicitness, algorithmic simplicity.
- Easy solid boundary processing.
- Well-suited to GPUs.

Cons

- Large memory consumption (19 scalars vs 4 hydrodynamic variables).
- Impact on performance in memory bound context.

II – Link-wise artificial compressibility method

Link-wise artificial compressibility method (LW-ACM)

- Novel formulation of the artificial compressibility method.
- Strong analogies with lattice Boltzmann schemes.

Updating rule:

$$f_\alpha(\boldsymbol{x},\, t+1) = f_\alpha^{(\mathrm{e})}(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,\, t) + 2\left(\frac{\omega-1}{\omega}\right)\left(f_\alpha^{(\mathrm{e,o})}(\boldsymbol{x},\, t) - f_\alpha^{(\mathrm{e,o})}(\boldsymbol{x}-\boldsymbol{\xi}_\alpha,\, t)\right)$$

where $f_\alpha^{(\mathrm{e})}$ are local equilibria which only depend on local ρ and $\boldsymbol{u}$, and $f_\alpha^{(\mathrm{e,o})}$ are the odd parts of the equilibrium functions:

$$f_\alpha^{(\mathrm{e,o})}(\rho,\, \boldsymbol{u}) = \frac{1}{2}\left(f_\alpha^{(\mathrm{e})}(\rho,\, \boldsymbol{u}) - f_\alpha^{(\mathrm{e})}(\rho,\, -\boldsymbol{u})\right).$$

P. Asinari, T. Ohwada, E. Chiavazzo, and A. F. Di Rienzo.
Link-wise artificial compressibility method.
Journal of Computational Physics, 231(15), 5109–5143, 2012.

First GPU implementation: TheLMA*

Two-step updating rule:

$$f_\alpha(\boldsymbol{x},\ t+1) = f_\alpha^*(\boldsymbol{x} - \boldsymbol{\xi}_\alpha,\ t) + 2\left(\frac{\omega - 1}{\omega}\right) f_\alpha^{(\mathrm{e,o})}(\boldsymbol{x},\ t)$$

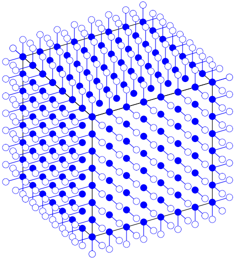
$$f_\alpha^*(\boldsymbol{x},\ t+1) = f_\alpha^{(\mathrm{e})}(\boldsymbol{x},\ t+1) - 2\left(\frac{\omega - 1}{\omega}\right) f_\alpha^{(\mathrm{e,o})}(\boldsymbol{x},\ t+1)$$

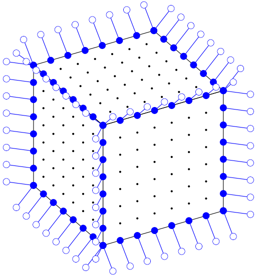
- LW-ACM very similar to LBM, with additional cost of loading and storing ρ and $\boldsymbol{u}$ at each time step.
- First GPU implementation of LW-ACM: slightly modified version of a TheLMA based single-GPU CUDA LBM solver.

C. Obrecht, F. Kuznik, B. Tourancheau, and J.-J. Roux.
The TheLMA project: Multi-GPU implementation of the lattice Boltzmann method.
International Journal of High Performance Computing Applications, 25(3):295–303, 2011.

Second GPU implementation: Louise

- Sufficient to have access to ρ and $\boldsymbol{u}$ at node $\boldsymbol{x}$ and its neighbours $\boldsymbol{x} - \boldsymbol{\xi}_\alpha$.
- Reduction of read redundancy: use CUDA blocks of $8 \times 8 \times 8$ threads, store ρ and $\boldsymbol{u}$ in an array of 10^3 `float4` structures in shared memory.





C. Obrecht, P. Asinari, F. Kuznik, and J.-J. Roux.
High-performance Implementations and Large-scale Validation of the Link-wise ACM.
Journal of Computational Physics, 275:143–153, 2014.

Data throughput and memory footprint

Louise data throughput per time step

- 992 `float4` structures read per CUDA block (41% of LBM).
- 512 written per block (21% of LBM).

Test hardware: GTX Titan Black (single precision)

- LBM: 38 million nodes (e.g. 320^3 cubic cavity).
- LW-ACM: 201 million nodes (e.g. 576^3 cubic cavity).

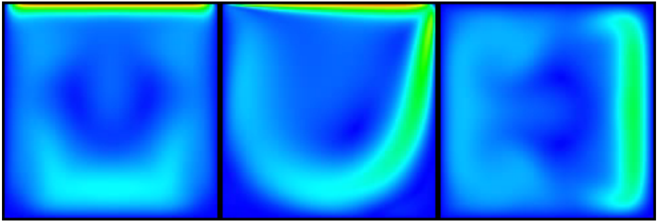
Local bounce-back boundary condition

- Bounce-back boundary condition: $f^*_\alpha(\boldsymbol{x} - \boldsymbol{\xi}_\alpha, t) = f_{\bar\alpha}(\boldsymbol{x}, t-1)$ where $\boldsymbol{x} - \boldsymbol{\xi}_\alpha$ is a wall node and $\bar\alpha$ is such that $\boldsymbol{\xi}_{\bar\alpha} = -\boldsymbol{\xi}_\alpha$.
- Louise does not keep f^*_α variables: finite difference boundary conditions (cumbersome for complex geometries).
- Louise* variant: *local* bounce-back $f^{(e)}_{\bar\alpha}(\boldsymbol{x},\ t) = f^{(e)}_\alpha(\rho, -\boldsymbol{u})$.

Updating rule at boundary node:

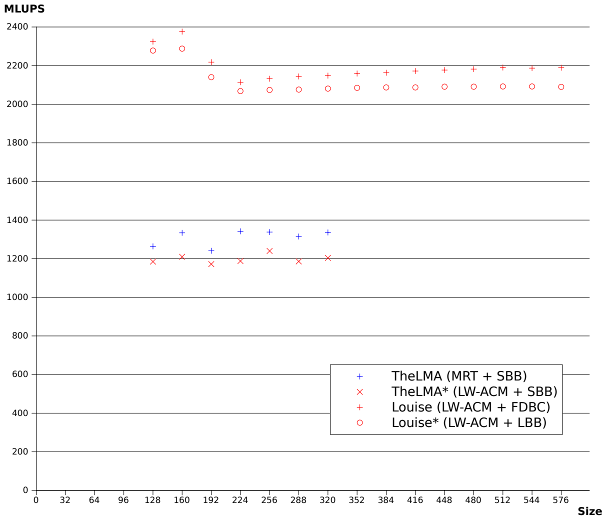
$$f_\alpha(\boldsymbol{x},\ t+1) = f^{(e)}_{\bar\alpha}(\boldsymbol{x},\ t) + 2\left(\frac{\omega-1}{\omega}\right)\left(f^{(e,o)}_\alpha(\boldsymbol{x},\ t) - f^{(e,o)}_{\bar\alpha}(\boldsymbol{x},\ t)\right).$$

Runtime video (Louise)



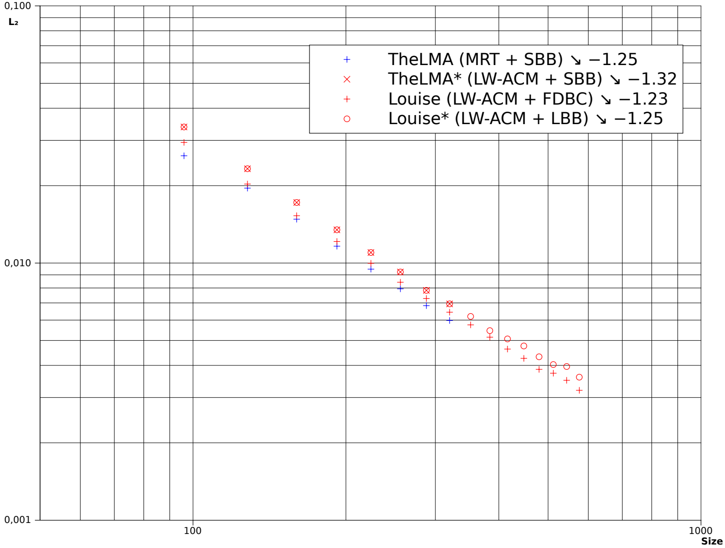
Lid-driven cubic cavity at $\mathrm{Re} = 1000$, $160^3 \approx 4.1$ million nodes, 20320 time steps, computation time 37.1 s on the GTX Titan, 2259 MLUPS.

Performance comparison: lid-driven cavity in single precision



GPU start temperature: 60 °C, runtime per resolution ≈ 30 s. For long term computations, performance is about 15% less.

Velocity discrepancy with respect to spectral element data



III – Work in progress

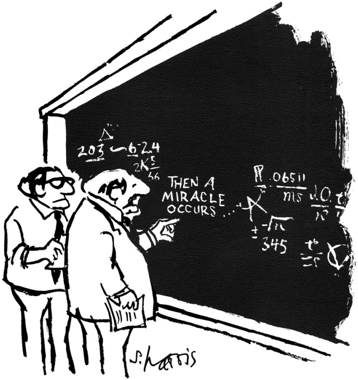
OpenCL Link-wise ACM on Many-core Processors (OpenCLAMP)

- OpenCLAMP: newly developed OpenCL program based on the same principles as Louise*.
- Performance portability: execution parameters specified in a JSON configuration file loaded at runtime.
- Performance on GTX Titan Black: similar than for Louise* code, i.e. higher than 2000 MLUPS, using $8 \times 8 \times 8$ work-groups.
- Performance on octocore Xeon (E5-2687W v2 at 3.40GHz): up to 40 MLUPS using $32 \times 1 \times 1$ work-groups.

Conclusions

- LW-ACM promising approach for CFD on GPUs.
- Device memory consumption divided by up to 5.25 with respect to LBM.
- Performance on Kepler GPUs increased by $1.8\times$.
- OpenCLAMP: to be released soon as a free software.
- Future work: extension to thermal flows, MPI-based multi-device.

Thank you for listening!



"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."