TOWARDS URBAN-SCALE FLOW SIMULATIONS USING THE LATTICE BOLTZMANN METHOD

Christian Obrecht∗,1,2, Frédéric Kuznik2, Bernard Tourancheau3,4 and Jean-Jacques Roux2
1 EDF R&D, Département EnerB&AT, F-77818 Moret-sur-Loing Cedex, France
2 INSA-Lyon, CETHIL, UMR5008, F-69621 Villeurbanne Cedex, France
3 INSA-Lyon, CITI, INRIA, F-69621 Villeurbanne Cedex, France
4 Université Lyon 1, F-69622 Villeurbanne Cedex, France
∗ Corresponding author: <christian.obrecht@insa-lyon.fr>

ABSTRACT
The lattice Boltzmann method (LBM) is an innovative approach in computational fluid dynamics (CFD). Due to the underlying lattice structure, the LBM is inherently parallel and therefore well suited for high performance computing. Emerging many-core devices, such as graphic processing units (GPUs), nowadays allow to run very large scale simulations on rather inexpensive hardware. In this contribution, we present some simulation results obtained using our multi-GPU LBM solver. For validation purpose, we study the flow around a wall-mounted cube and show good agreement with previously published results. Furthermore, we discuss larger scale flow simulations involving nine cubes which demonstrate the practicability of CFD simulations in building aeraulics.

INTRODUCTION
Because of the computational cost of flow simulations, building aeraulics is generally taken into account using simplified models. However, this approach is not satisfactory in terms of accuracy when modelling energy efficient buildings. Recent advances, in both computational fluid dynamics (CFD) and high performance computing allow to consider the practical use of explicit flow simulations in building models.

In this contribution, we shall present simulation results obtained using the lattice Boltzmann method (LBM). Being based on a mesoscopic point of view, this novel CFD approach has numerous advantages over classic macroscopic methods such as the solving of the Navier-Stokes equations. Among other benefits, it is worth mentioning the high numerical stability, the ability to deal with complex geometries and the straightforwardness of various physical couplings.

Although parallel implementations of the LBM may be rather efficient, performing large scale simulations on mainstream architectures still requires the use of expensive clusters. The present simulations were carried out using several graphic processing units (GPUs) in parallel within a single server. Performance afforded by such hardware configuration is comparable to the one obtained using large clusters at a fairly lower cost.

The remaining of the paper is organised as follows. The first section is a summary of the LBM, presenting the specific model we retained together with the subgrid-scale model we added in order to enable simulations at high Reynolds number. Then, we give a short description of state-of-the-art GPU implementations of LBM solvers and of our multi-GPU LBM framework. In the third section, for validation purpose, we present the simulation of a fully developed flow over a wall-mounted cube in a flat channel. The simulation results are compared to experimental data. The last section reports the simulation of the flow over nine identical wall-mounted cubes at high Reynolds numbers.

LATTICE BOLTZMANN METHOD
Lattice Boltzmann equation
Although originating from the lattice gas automata theory (Frisch et al., 1986), the lattice Boltzmann method is nowadays usually interpreted as a discrete version of the Boltzmann transport equation (McNamara and Zanetti, 1988):

$$\partial_t f + \mathbf{v} \cdot \nabla_x f + \frac{\mathbf{F}}{m} \cdot \nabla_x f = \Omega (f),$$  (1)

where $f(x, \mathbf{v}, t)$ describes the evolution in time of the distribution of one particle in phase space, $\mathbf{F}$ is the external force field, $m$ the mass of the particle, and $\Omega$ the collision operator.

Discretisation occurs both in time, with constant time steps $\delta t$, and in phase space, generally using a regular orthogonal grid of mesh size $\delta x$ and a finite set of $N+1$ particle velocities $\{\mathbf{v}_\alpha\}$, with $\mathbf{v}_0 = 0$. The later is commonly a subset of the velocities linking a node to its nearest neighbours as the D3Q19 stencil we used for our simulations (see Fig. 1).

The discrete counterpart of the distribution function $f$ is a finite set of functions $f_\alpha (x, t)$ associated to the particle velocities $\mathbf{v}_\alpha$. Let us denote:

$$[a_\alpha] = (a_0, \ldots, a_N)^T,$$

$T$ being the transpose operator. The lattice Boltzmann equation (LBE) writes:

$$[f_\alpha (x + \delta \mathbf{v}, t + \delta t)] - [f_\alpha (x, t)] = \Omega ([f_\alpha (x, t)]).$$  (2)
The mass density $\rho$ and the velocity $u$ of the fluid are given by:

$$\rho = \sum_{\alpha} f_{\alpha}, \quad u = \frac{1}{\rho} \sum_{\alpha} f_{\alpha} \xi_\alpha. \quad (3)$$

**Multiple-relaxation-time LBM**

The simplest (and most commonly used) way to express the collision operator is the LBKG approach (Qian et al., 1992), which uses the Bhatnagar-Gross-Krook approximation (Bhatnagar et al., 1954). We instead chose to use the multiple-relaxation-time (MRT) approach (d’Humières, 1994). Although of higher computational cost, MRT was shown of better accuracy and numerical stability than LBKG (Lallemand and Luo, 2000).

In the MRT approach, collision is performed in moment space. The particle distribution is mapped to a set of moments $\{m_\alpha | i = 0, \ldots, N\}$ by an orthogonal matrix $M$:

$$|f_\alpha(x, t)\rangle = M^{-1}|m_\alpha(x, t)\rangle \quad (4)$$

where $|m(x, t)\rangle$ is the moment vector. For the D3Q19 stencil, the orthogonal matrix $M$ can be found in appendix A of d’Humières et al. (2002). The corresponding moment vector is:

$$|m_\alpha(x, t)\rangle = (\rho, e, e, j_x, j_y, j_z, q_x, q_y, q_z, 3p_{xx}, 3\pi_{xx}, p_{uw}, \pi_{uw}, p_{xu}, p_{yz}, p_{zx}, m_x, m_y, m_z)^T. \quad (5)$$

where $e$ is energy, $e$ is energy square, $j = (j_x, j_y, j_z)$ is the momentum, $q = (q_x, q_y, q_z)$ is the heat flux, $p_{xx}, p_{yx}, p_{yz}, p_{zx}, p_{uw}$ are related to the strain rate tensor $S$, $\pi_{xx}, \pi_{uw}$ are fourth-order moments and $m_x, m_y, m_z$ are third-order moments with respect to the particle velocities. The mass density and the momentum are the conserved moments.

The LBE thus writes:

$$[f_\alpha(x + \delta \xi_\alpha, t + \Delta t) - f_\alpha(x, t)] = -M^{-1}\Lambda [m_\alpha(x, t) - m_\alpha^{\text{eq}}(x, t)] \quad (6)$$

where $\Lambda$ is a diagonal collision matrix and the $m_\alpha^{\text{eq}}$ are the equilibrium values of the moments. For the sake of isotropy, $\Lambda$ obeys:

$$\Lambda = \text{diag}(0, s_1, s_2, 0, s_4, 0, s_4, 0, s_4, \ldots)$$

The other rates are set according to Lallemand and Luo (2000), i.e. $s_1 = 1.19, s_2 = s_{10} = 1.4, s_4 = 1.2$, and $s_{16} = 1.98$.

**Large-eddy simulation**

For large-eddy simulation (LES), the kinematic viscosity $\nu$ is linked to the turbulent viscosity $\nu_t$ in the Smagorinsky model (Smagorinsky, 1963), the turbulent viscosity is given by:

$$\nu_t = |S|(C_S \delta x)^2, \quad |S| = \sqrt{2S:S}, \quad (10)$$

where $C_S$ is the Smagorinsky constant. Adding eddy viscosity to the MRT model is achieved by replacing the relaxation rate $s_9$ with:

$$s_9^* = \frac{1}{\tau_0 + \tau_t}, \quad (11)$$

where $\tau_0$ and $\tau_t$ are the molecular and turbulent relaxation times:

$$\tau_0 = \frac{1}{c_s^2} \nu_0 + \frac{1}{2}, \quad \tau_t = \frac{1}{c_s^2} \nu_t. \quad (12)$$

Following Krafczyk et al. (2003), the second order moments obey:

$$P_{ij} = \sum_\alpha \xi_{\alpha i} \xi_{\alpha j} f_\alpha = e^2 \rho \delta_{ij} + \rho u_i u_j - Q_{ij}, \quad (13)$$

with:

$$Q = \frac{2c_s^2 \rho}{s_9^*} S. \quad (14)$$

Thus, the strain rate tensor may be computed from the moment vector. For the D3Q19 stencil, we obtain:

$$P_{xx} = \frac{1}{\tau_t} (30 \rho + e) + p_{xx}, \quad (15)$$
\[ P_{yy} = \frac{1}{3\delta}(30\rho + e) + \frac{1}{2}(p_{ww} - p_{xx}), \quad (16) \]
\[ P_{zz} = \frac{1}{3\delta}(30\rho + e) - \frac{1}{2}(p_{xx} + p_{ww}), \quad (17) \]
\[ P_{xy} = p_{xy}, \quad P_{yz} = p_{yz}, \quad P_{xz} = p_{xx}. \quad (18) \]

Finally, assuming that \( \nu_t \) depends on \( S \) at current time, we have:
\[ \tau_t = \frac{1}{2} \left( \sqrt{\tau_0^2 + 18|Q|(C_S\delta x)^2} - \tau_0 \right). \quad (19) \]

**MULTI-GPU SOLVER**

**Algorithmic aspect**

From an algorithmic standpoint, the LBE (Eq. 6) naturally breaks in two elementary step:

\[ |\tilde{f}_\alpha(x, t)\rangle = |f_\alpha(x, t)\rangle + \Omega \left[ |f_\alpha(x, t)\rangle \right] \quad (20) \]
\[ |f_\alpha(x + \delta t \xi_\alpha, t + \delta t)\rangle = |\tilde{f}_\alpha(x, t)\rangle \quad (21) \]

Equation 20 describes the collision step in which an updated particle distribution is computed. Equation 21 describes the propagation step in which the updated particle populations are transferred to the neighbouring nodes as outlined by Fig. 2 (in two dimensions for the sake of simplicity).

**Figure 2: Propagation step**

It is worth mentioning that in the first step, computations only involve informations local to each node. Moreover, in the second step, data transfers only require proper synchronisation with the nearest neighbours. As a matter of consequence, the LBM is fairly well suited for parallel implementations.

**CUDA implementations**

During the last decade, the computational power of GPUs has grown exponentially, reaching 1.35 Tflop/s single precision peak performance with the latest generation of NVIDIA processors (NVIDIA, 2010). Early attempts to implement the LBM on such hardware (Fan et al., 2004) were quite promising. With the introduction of the CUDA technology by NVIDIA in 2007, general purpose programming on GPUs became more practicable. Several successful CUDA implementations of the three-dimensional LBM (Tölke and Krafczyk, 2008; Obrecht et al., 2010) were reported since.

On recent hardware, single GPU implementations are able to handle up to about \( 7.7 \times 10^6 \) nodes per second, whereas multithreaded CPU implementations handle at most about \( 8.5 \times 10^7 \) nodes per second using a single quad core processor (Lee et al., 2010). It is also worth mentioning that performance of GPU implementations is communication bound (Obrecht et al., 2011a), while performance of CPU implementations is computation bound. Thus, making use of a model of higher algorithmic complexity (e.g. MRT instead of LBGK) has in general little impact on performance.

**TheLMA framework**

GPUs provide large computational power at fairly low cost. Yet, although growing more versatile at each generation, CUDA enabled GPUs still have numerous drawbacks. The CUDA tool chain for instance, due to hardware limitations, is unable to link several GPU binaries. In cases like LBM, this forbids the use of library oriented development techniques. The limited amount of on board memory may also be problematic. Using the latest computation devices, a single GPU implementation of the D3Q19 scheme may handle at most about \( 4.2 \times 10^7 \) nodes in single precision.

To address both issues, we created the TheLMA framework (TheLMA, 2010). TheLMA stands for *Thermal LBM on Many-core Architectures*, thermal simulations being our main topic of interest. The TheLMA framework is designed to improve code reusability. Setting up a new simulation usually only requires minor code modifications. Moreover, TheLMA provides native multi-GPU support (Obrecht et al., 2011b). For now, this support is limited to single servers, but extension to hybrid clusters is under active development.

**FLOW AROUND A SINGLE CUBE**

In order to validate our MRT-LBM solver, we chose to simulate a fully developed flow around a wall-mounted cube in a channel. The simulation results are compared to experimental data from Meinders et al. (1999). Figure 3 outlines the simulation setup.

The channel is represented by a cavity containing \( 1,024 \times 768 \times 192 \approx 1.51 \times 10^8 \) nodes. Solid walls are simulated using half-way bounce-back boundary condition (see for instance Pan et al., 2006). The inlet velocity is imposed by adding the corresponding equilibrium values to the distribution functions and the outlet condition is obtained by imposing null velocity gradient. The size of the cube is set to \( H = 58 \delta x \) in order to have \( h/H \approx 3.3 \) as in our reference, and the position of the cube is such as \( x_0 = 4H \). It should be mentioned that, in order to save memory, \( y_0 \) is less in our setup than in the experimental one. This allows to improve the resolution of the obstacle, with little impact on the flow since we have \( y_0 > 6H \).
In their work, Meinders et al. give the time-averaged streamwise velocity of the flow in the vertical symmetry plane, obtained through laser Doppler anemometry (LDA). The measurements were conducted at Reynolds number $Re = 4,440$, where:

$$Re = \frac{u_B H}{\nu}$$

(22)

and $u_B$ is the bulk velocity of the inflow. In our simulation, we averaged the streamwise velocity over time from $50T_0$ to $200T_0$, where $T_0 = H/u_0$ is the turnover time and $u_0$ is the maximum inlet velocity. The overall computation time was less than six hours using a Tyan B7015 server with eight Tesla C1060 computing devices.

Figures 4 and 5 show upstream and downstream normalised velocity profiles with respect of $x/H$ for both simulation and measurements.

Agreement of simulation data with experimental data is rather satisfying since uncertainties on both position and value of measurements should be taken into account. Unfortunately, our reference does not provide such informations, nor does it give detailed data regarding the inlet velocity profile. Although not perfect, agreement with measurements is by far better than in previously published work (see Fig. 9 in Yakhot et al., 2006). The most important flow features are also well predicted by our model.

FLOW AROUND NINE CUBES

To illustrate the possible use of multi-GPU LBM solvers in building aeraulics, we chose to simulate the flow around nine identical wall-mounted cubes. Figure 6 outlines the simulation setup.

The simulation domain is again represented using a $1,024 \times 768 \times 192$ mesh. The size of the cubes is set to $H = 48\delta x$, and the position is such as $x_0 = 3H$ and $x_1 = H/2$. Thus, we have $y_0 = 6H$ and $h = 4H$.

We impose logarithmic velocity profile for the inflow and constant streamwise velocity on the top lid. In this configuration, we define the Reynolds number as:

$$Re = \frac{u_1 H}{\nu}$$

(23)

where $u_1$ is the inflow velocity at obstacle height. Furthermore, to reduce the impact of lateral faces on the flow, we apply the same boundary condition as for the outlet, i.e. null velocity gradient in the direction normal to the face.

We chose to run simulations at Reynolds numbers $Re_1 = 40,000$ and $Re_2 = 1,000,000$. Smagorinsky subgrid-scale models were reported satisfactory in similar situations, for Reynolds numbers up to at least $Re_1$ with LBM flow solvers (Krafczyk et al., 2003), and at least $Re_2$ for Navier-Stockes solvers (Šarić et al., 2006). Although the LBM part in our implementation differs from the former, we may be rather confident in the results at $Re = Re_1$. The simulation at $Re = Re_2$ is more relevant at building scale, however the results should be considered with greater care.

As in the single cube simulation, we averaged density and velocity over time from $50T_0$ to $200T_0$, the turnover time being set to $T_0 = 4H/u_0$. The overall computation time was about 17 h 19 min for $10^8$ time steps. The corresponding performance is $2.4 \times 10^9$ node updates per second, which is at least a $28 \times$ speedup over
Figure 4: Upstream normalised velocity profiles with respect of \( x/H \) for several values of \( z/H \)

Figure 5: Downstream normalised velocity profiles with respect of \( x/H \) for several values of \( z/H \)
optimised multithreaded CPU implementations. Figures 7 and 8 display the averaged pressure relative variation \( r \) and velocity streamlines in the vertical symmetry plane near the obstacle, with:

\[
    r = \frac{p - p_\infty}{p_\infty}
\]

and \( p_\infty \) is the freestream pressure near the inflow.

Both simulations lead to rather similar mean flow patterns. The flow in the gaps between the cubes is structured in two independent cells. This specific flow feature is due to the three-dimensional effects of the flow. The simulations also show similar pressure fields. At the walls, \( r \) equals to the pressure coefficient. The obtained values are within the range of coefficients used in practice and seem therefore reasonable.

**CONCLUSION**

In the present work, we provide building scale flow simulation results obtained using our multi-GPU implementation of the LBM. We show that the required computation times remain below reasonable limits. Thus, we believe this contribution is a significant step towards the use of effective CFD simulations in building models. Moreover, it is worth mentioning that the LBM applies to a wide range of situations and therefore may be useful in other fields of building simulation than external aerodynamics.

Several improvements to our approach, regarding both performance and accuracy are within reach. From a physical standpoint, the use of more elaborate subgrid-scale models than the static Smagorinsky model we implemented would be desirable. On-going research founded on the same mesoscopic point of view as the LBM might provide advances on this issue (Sagaut, 2010). From a computational standpoint, porting grid refinement techniques to the GPU would be of highest practical interest and we plan to add such a feature to our framework in near future.

**REFERENCES**


Figure 7: Time averaged streamlines and pressure relative variation at $Re = 40,000$

Figure 8: Time averaged streamlines and pressure relative variation at $Re = 1,000,000$


