IRSTI 27.35.17

D.B. Zhakebayev, *K.K. Karzhaubayev, E.S. Moisseyeva, N.V. Tsoy

National Engineering Academy, Almaty, Kazakhstan
*e-mail: kairzhan.k@gmail.com

Simulation of thermal flows by lattice Boltzmann method on the CUDA computational platform

Abstract. Originating from lattice gas automata theory, the lattice Boltzmann method (LBM) is an interesting alternative to the solving of Navier-Stokes equations. In contrast to isothermal simulations, for a while thermal flow simulations were challenging for LBM. Thermal flow simulations are important task in various fields of research. Despite a large amount of work and research the dynamics of thermally induced flows are still highly demanded. Motivation of this work is development of computational tool for simulation of the dynamics of thermal flows. To this purpose, we developed LES-LBM solver accelerated by the Graphics Processing Unit (GPU) on the CUDA computational platform, integrating LBM with Large Eddy Simulation (LES). Simplicity of coding is usually an appealing feature of the LBM. Conventional implementations of LBM suffer from high memory consumption and poor computational performance. The main advantage of the solvers based on GPU is their ability to perform significantly more floating point operations per unit time (FLOPS) than a Central Processing Unit (CPU) and a good scalability of explicit parallel algorithms. LES-LBM code was tested on the NVIDIA GeForce GTX 1050 ti and NVIDIA TESLA K80 GPUs.

Key words: The lattice Boltzmann method, CUDA, thermal flow.

Introduction

In June 2007 NVIDIA released a new framework named CUDA for general parallel processing applications. This framework enables developers to implement GPU parallel programs in C, C++ languages and allows direct access to the GPU computing power without complicated graphics API. Special tools which are included in an official software development kit (SDK) allows to debug GPU programs in runtime. Since 2007 a lot of numerical libraries presented. They allow to create efficient programs with less effort, and cover such numerical algorithms like linear algebra operations, sparse matrix computations, Fourier transforms, image algorithms etc.

In a market NVIDIA has several separate products, GeForce is for gaming, Quadro is for professional OpenGL based rendering and Tesla for high performance computations. Tesla compute accelerators get strong positions in a such high performance areas as financial analysis and scientific computations. A lot of supercomputer providers use NVIDIA GPUs to create energy efficient computing clusters. One of the secret of the hight popularity among users is the support of most popular proprietary (CUDA) and open standards (OpenCL, DirectCompute) for GPU programming.

Effort to use GPU as a massively parallel processor computational fluid dynamics (CFD) started in the beginning of 2000. One of the first publications was a chapter in the GPU Gems Books by M. J. Harris [1]. In the Chapter 38 he described realization of simple fluid dynamics solver based on Stam's stable fluids [2]. Later several authors published results of implementing Marker and Cell method on a GPU [3].

The smoothed particle hydrodynamics (SPH) is another approach for simulation of the fluid dynamics, without direct use of the Navier Stokes equations. Originally developed in [4] SPH is a meshfree Lagrangian method which tracks position and movement of many fluid particles, which allows direct mass conservation. One drawback over grid-based techniques is the need for large numbers of particles to produce simulations of equivalent resolution. Explicit nature of the method allows it to run effectively on massive parallel processors as GPU [5-7].

The lattice Boltzmann method (LBM) is a relatively novel approach in computational fluid dynamics (CFD), which, unlike most other CFD methods, does not rely in directly solving the Navier-Stokes equations by a numerical algorithm.

One of the most interesting feature of LBM is that numerical procedure has the data locality property. Such a property is very well suited to be implemented in a massively parallel processors, like GPUs [8-11]. In [12] LBM Large Eddy Simulations for high Reynolds numbers were performed. Developed numerical implementation was able to run on four Fermi class GPUs simultaneously. Large GPU memory (24GB) allowed to perform simulations with relatively high spatial resolution (max grid size 10240x10240) with active double precision mode. However, authors mentioned that four GPUs were located on the same machine, and there is strong hardware limitation for further improvement of spatial resolution. To overcome this fact, authors recommended to extend their implementation to multinode GPU clusters.

On the other hand, while for describing hydrodynamic turbulence models based on the Navier-Stokes equations have been used almost exclusively for almost two centuries, a significant increase in interest in LBM methods has recently been explained by their computational efficiency. These methods, based on the Boltzmann equation, make it possible to predict the macroscopic magnitudes of continuum mechanics, such as velocity and pressure. Although it was proven several years ago that hydrodynamic turbulence can be accurately described using these methods, the development of LES within LBM is still at a very early stage [13]. For example, the LES-LBM methods are used in [14-19].

Lattice Boltzmann Method

The basic quantity of the LBM is the discrete velocity distribution function $f_i(\vec{x},t)$ which is also called particle populations. Particle populations represents the density of particles with velocity $\vec{c_i} = (c_{ix}, c_{iy}, c_{iz})$ at position-time $(\vec{x},t), \vec{x} = \{x,y,z\}$. The discrete velocities are chosen such as to link each lattice site to some of its neighbors. Fig. 1 shows the D3Q27 stencil, where each node is connected to 26 of its nearest neighbors, and position 0 assigned to resting particles.

$$\rho(\vec{x},t) = \sum_i f_i(\vec{x},t) \quad (1)$$

$$\rho u(\vec{x},t) = \sum_i c_i f_i(\vec{x},t) \quad (2)$$

By discretising the Boltzmann equation in velocity space, physical space, and time, we find the *lattice Boltzmann equation:*

$$f_i(\vec{x} + \vec{c}\Delta t, t+\Delta t) = f_i(\vec{x},t) + \Omega_i(\vec{x},t) + S_i \quad (3)$$

Where Ω is collision operator and S_i is force. One of simplest models for collision is Bhatnagar-Gros-Krook [20], which relaxes the populations towards an equilibrium f^{eq} at a rate determined by the relaxation time τ:

$$\Omega_i(f) = -\frac{f_i - f_i^{eq}}{\tau}\Delta t \quad (4)$$

$$f_i^{eq}(\vec{x},t) = w_i \rho\left(1 + \frac{\vec{u}.\vec{c_i}}{c_s^2} + \frac{(\vec{u}.\vec{c_i})^2}{2c_s^4} - \frac{\vec{u}.\vec{u}}{2c_s^2}\right) \quad (5)$$

Relation between physical kinematic viscosity and lattice relaxation time is given by [10], where c_s is the lattice speed of sound:

$$\nu = c_s^2\left(\tau - \frac{\Delta t}{2}\right) \quad (6)$$

Numerical algorithm for LBM consists of collision (7) and streaming (8) steps. During streaming particles propagate to neighbor nodes fig. 2a-b.

$$f_i^*(\vec{x},t) = f_i(\vec{x},t) - \frac{\Delta t}{\tau}\left(f_i(\vec{x},t) - f_i^{eq}(\vec{x},t)\right) \quad (7)$$

$$f_i(\vec{x} + \vec{c_i}\Delta t, t+\Delta t) = f_i^*(\vec{x},t) \quad (8)$$

$$S_i = \left(1 - \frac{\Delta t}{2\tau}\right) w_i \left(\frac{c_{i\alpha}}{c_s^2} + \frac{(c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta})u_\beta}{c_s^4}\right) F_\alpha \quad (9)$$

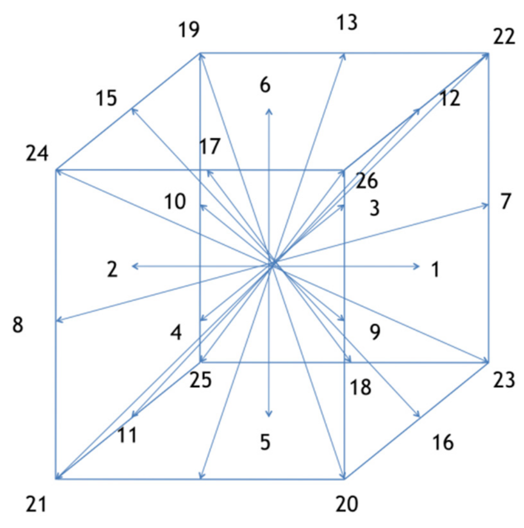

Figure 1 - D3Q27 lattice model used in the simulations

In many situations, the work from viscous dissipation and compression is so small that it does not significantly contribute to the heat balance. It is then sufficient to consider an advection-diffusion equation for temperature without heat source terms, together with the incompressible Navier-Stokes eq. For such simulations often two population model is applied, in which there is separate distribution function for temperature - g_i is introduced. For temperature distribution function same equations are applied for collision and streaming process (7-8).

Buoyancy is modeled through force density **F**. If density variations due temperature is small Boussinesq approximation could be used [10-11].

Numerical algorithm for LBM reads as:

```
for each time step t do
compute macroscopic velocity, temperature and density
compute equilibrium distribution for g
perform one step of collision for g
apply boundary conditions for temperature
perform propagation step
compute equilibrium distribution for f
perform one step of collision for f
apply boundary conditions for velocity
perform propagation step
endfor
```

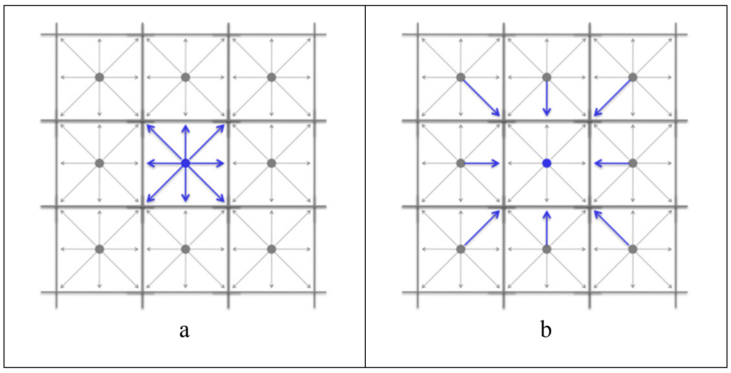

Figure 2 – Streaming from central node (a) and streaming into central node (b)

CUDA implementation

CUDA enabled simulation code is implemented in the CUDA C language which is an extension to the C language. Functions in a CUDA C are marked as host functions, device functions and kernels. Host functions are simple C functions which executed by host processor. Device functions are special functions which should be launched on a GPU. Kernel functions are used to launch GPU functions from the host code.

Kernel function runs in parallel on the GPU. During launch of the kernel a lot of kernel copies executed in parallel by GPU. The execution pattern is identified by a grid. The grid is the special configuration of the parallel threads, which are grouped into blocks. Grid and block layout could be one, two and three dimensional. Fig. 3 show example of the grid which has 6 blocks and two dimensional layout. Each block in fig. 3 also has two dimensional layout with 9 threads, total number of threads 54.

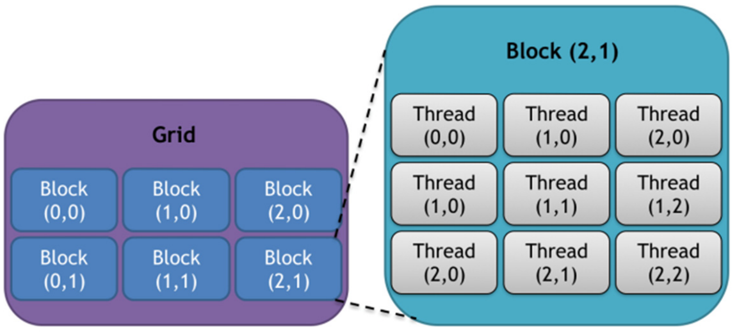

Figure 3 – CUDA execution model

Each thread is executed by Streaming multiprocessor (SM). SM has several scalar processors (SP) which actually runs code. Special scheduling algorithms and chips make use of large amount of SP on a GPU.

For convenience, "threadIdx" is a 3-component vector, so threads can be identified using a one-dimensional, two-dimensional, or three-dimensional thread index, forming a one-dimensional, two-dimensional, or three-dimensional block of threads. This provides a natural way to invoke calculations for elements in a domain, such as a vector, matrix, or volume.

The index of a threads and its identifier are directly related to each other: they are the same for a one-dimensional block; for a two-dimensional block of size (Dx, Dy), the thread ID of index(x, y) is (x + y Dx); for a three-dimensional block of size (Dx, Dy, Dz), the thread ID of index(x, y, z) is (x + y Dx + z Dx Dy).

There is a limit on the number of threads in a block, since it is expected that all threads of a block will be on the same processor core and must share the limited memory resources of this core. On modern GPUs, a block of threads can contain up to 1024 threads.

However, the kernel can be executed by several blocks of threads of the same shape, so that the total number of threads is equal to the number of threads in the block multiplied by the number of blocks.

Listing 1 provides example of GPU kernel function for streaming stage in LBM simulations. Variables i, j, k are used to assign memory location for each thread.

Listing 1 – GPU kernel function

```
__global__ void stream(float *f_dst, float *f_src)
{
    unsigned int i = threadIdx.x + blockIdx.x * blockDim.x;
    unsigned int j = threadIdx.y + blockIdx.y * blockDim.y;
    unsigned int k = threadIdx.z + blockIdx.z * blockDim.z;

    for (unsigned int l=0; l<NDIR; l++) {
        unsigned int i2 = (NX + i - dirx[l]) % NX;
        unsigned int j2 = (NY + j - diry[l]) % NY;
        unsigned int k2 = (NZ + k - dirz[l]) % NZ;

        f_dst[voffset(i, j, k, l)] = f_src[voffset(i2, j2, k2, l)];
    }
}
```

CUDA programming model introduces concept of memory types. CUDA enabled GPU has device, shared, texture, constant cache and register memory. Register memory is very fast memory, with on-chip implementation. Size of register memory may vary between GPU's, but it is usually small, and cannot store arrays. Shared memory is special memory which is shared between threads of the same block. Proper use of shared memory may decrease load to global GPU memory. Global GPU memory or device memory is large memory, which is slow comparably to other types of memory, but has advantage of big size, often several gigabytes. There is also special caches to store constant values and one-two-three dimensional textures. Use of texture cache may increase simulation speed if there exist special pattern while accessing data in cache.

To archive high performance CUDA implies that programmer uses right type of memory for various data. Results of the simulation should be uploaded to CPU or host memory using special CUDA API calls.

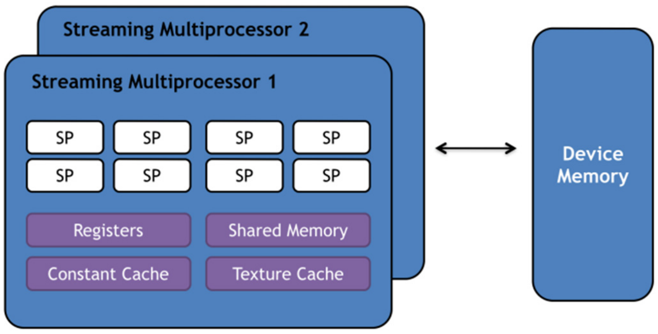

Figure 4 - CUDA hardware model

Results

The CUDA LBM solver implemented on the GPU is used to study the differentially heated cubic cavity outlined in Fig. 5. Two opposite vertical walls have imposed temperatures -□T0 and +T0, whereas the remaining walls are adiabatic. This configuration has been extensively studied in the two-dimensional configuration and various benchmark solution are available.

All simulation parameters were the same as in the benchmark solution [11]. In order to perform validation, the flow in the differentially heated cavity is computed for Rayleigh numbers equal to 10^4, 10^5, 10^6 and 10^7. The results are compared with available data. Table 1 gives the obtained Nusselt numbers as well as the values published in [11]. GPU simulations shows good results which are in accordance with the reference values.

Simulations were performed in a single precision mode on a NVIDIA 1050 and Tesla K80 GPUs. Their characteristics are presented in the table 2. Mesh size is 128x128x64. Additional comparison was made with multicore workstation with 64 available cores and 256 threads. Results of performance comparison is presented in fig. 6.

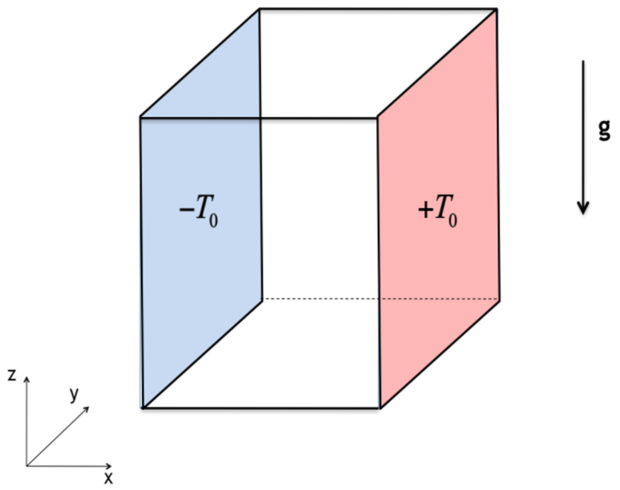

Figure 5 – Natural convection scheme

Table 1 – Comparison of Nusselt numbers

Rayleight number	$\mathbf{10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$	$\mathbf{10^7}$
Present	2.031	4.3302	8.6468	16.4193
Obrecht [11]	2.056	4.3382	8.6457	16.4202

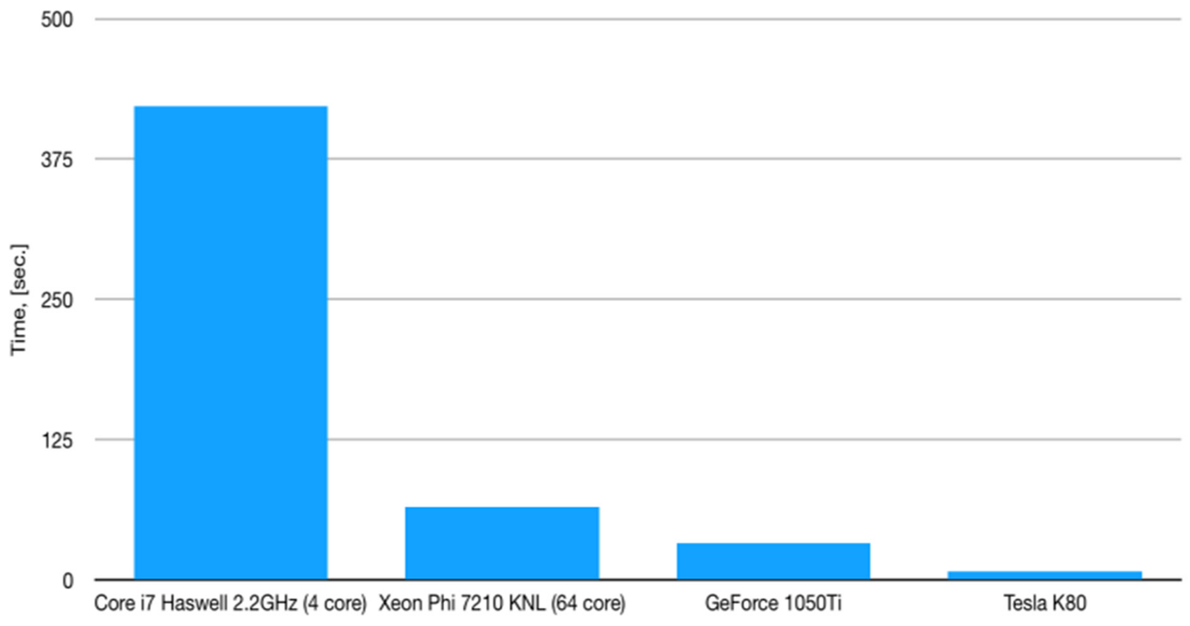

Figure 6 – Performance comparison

Table 2 – GPU parameters

Parameter/GPU	**1050 Ti**	**Tesla K80**
CUDA cores	768	4992
GPU Clock	1392 MHz	875 MHz
SP performance	2.1 TFlops	8.73 TFlops
DP performance	1/32 of SP	2.91 TFlops
Memory	4 GB GDDR5	24 GB GDDR5
Bandwidth	112 GB/s	480 GB/s
ECC support	No	Yes

Conclusion

In this paper, we presented in-house thermal LBM solver for CUDA enabled GPU workstations. Validity of the numerical algorithm was tested on a benchmark Natural convection problem. The code uses separate populations for velocity and temperature. According to the timings GPU based LBM gets higher performance in comparison with the single CPU code. Code runtime also compared with the results of OpenMP version of the code, which is executed on Xeon Phi KNL workstation with 64 available cores. Promising results shows that GPU accelerated LBM is a good alternative to CPU versions. Drawback of the CUDA code is the dependence on the CUDA platform and decisions made by Nvidia company. Also, most of the CUDA libraries are closed source, or proprietary standards, which is opposite to the role of the OpenMP.

Still a plenty of improvements are possible to the code. One is the extension of the algorithm to multi GPU and multi node configurations and more aggressive optimization for memory bandwidth.

Developed code could be used in various thermal fluid flow simulations for which Boussinesq approximation for density fluctuations is valid.

Acknowledgements

This work was supported by the MES Republic of Kazakhstan (grant number AP05132121).

References

1. Fernando R. *"GPU gems: programming techniques, tips and tricks for real-time graphics."* (Pearson Higher Education, 2004).
2. Stam J. "Stable fluids." *Proceedings of the 26th annual conference on Computer graphics and interactive techniques. – ACM Press/Addison-Wesley Publishing Co.* (1999): 121-128.
3. L. M. Itu, C. Suciu, F. Moldoveanu, A. Postelnicu and C. Suciu "Optimized GPU based simulation of the incompressible Navier-Stokes equations on a MAC grid." *RoEduNet International Conference 10th Edition: Networking in Education and Research, Iasi* (2011): 1-4.
4. R.A. Gingold, J.J. Monaghan "Smoothed particle hydrodynamics: theory and application to non-spherical stars." *Mon. Not. R. Astron. Soc*. 181 (1977): 375–89.
5. Alexis Hérault, Annamaria Vicari, and Ciro Del Negro "A SPH thermal model for the cooling of a lava lake." *Proceedings of the 3th SPHERIC Workshop* (Lausanne, 2008).
6. Robert A. Dalrymple and Alexis Hérault "Levee breaching with GPU-SPHysics code." *Proceedings of the 4th SPHERIC Workshop* (Nantes, 2009).
7. Robert A. Dalrymple, Annamaria Vicari, Ciro Del Negro, and Robert A. Dalrymple, "Modeling water waves in the surf zone with GPU-SPHysics." *Proceedings of the 4th SPHERIC Workshop* (Nantes, 2009).
8. Christian Obrecht, FrédéricKuznik, Bernard Tourancheau, Jean-Jacques Roux "Scalable lattice Boltzmann solvers for CUDA GPU clusters." *Parallel Computing* 139, no. 6-7 (2013): https://doi.org/10.1016/j.parco.2013.04.001.
9. Parmigiani A., Huber C., Chopard B. et al. Eur. *Phys. J. Spec. Top.* 171 (2009): 37. https://doi.org/10.1140/epjst/e2009-01009-7.
10. Krüger T., Kusumaatmaja H., Kuzmin A., Shardt O., Silva G., Viggen E. M. "The Lattice Boltzmann Method." *Springer* (2017).
11. Lars Moastuen *"Real-time simulation of the incompressible Navier-Stokes equations on the GPU."* (PhD diss., Oslo University, July 2007).
12. Li C., Maa J.P.Y., Kang H. *Sci. China Phys. Mech. Astron.* 55 (2012): 1894. https://doi.org/10.1007/s11433-012-4856-9.
13. Xu H., Malaspinas O., Sagaut P. "Sensitivity Analysis and Optimal Strategies of MRT-LBM for CAA-Determination of free relaxation parameters in MRT-LBM." *Eighth International Conference for Mesoscopic Methods in Engineering and Science* (2010).
14. Stiebler M. et al. "Lattice Boltzmann large eddy simulation of subcritical flows around a sphere on non-uniform grids." *Computers & Mathematics with Applications* 61, no. 12 (2011): 3475-3484.
15. Jacob J., Malaspinas O., Sagaut P. "A new hybrid recursive regularised Bhatnagar–Gross–Krook collision model for Lattice Boltzmann method-based large eddy simulation." *Journal of Turbulence* (2018): 1-26.
16. Pradhan A., Yadav S. "Large Eddy Simulation using Lattice Boltzmann Method based on Sigma Model." *Procedia Engineering* 127 (2015): 177-184.
17. Liou T. M., Wang C. S. "Large eddy simulation of rotating turbulent flows and heat transfer by the lattice Boltzmann method." *Physics of Fluids* 30, no. 1 (2018): 015106.
18. Hamane D., Guerri O., Larbi S. "Investigation of flow around a circular cylinder in laminar and turbulent flow using the Lattice Boltzmann method." *AIP Conference Proceedings* 1648, no. 1 (2015): 850094.
19. Cui X. et al. "A 2D DEM–LBM study on soil behaviour due to locally injected fluid." *Particuology* 10, no. 2 (2012): 242-252.
20. Z. Guo, C. Zheng, B. Shi "Discrete lattice effects on the forcing term in the lattice Boltzmann method." *Phys. Rev. E* 65 (4) (2002) 046308.
21. S.K. Kang, Y.A. Hassan "A direct-forcing immersed boundary method for the thermal lattice Boltzmann method." *Comput. Fluids* 49 (1) (2011): 36–45.
22. Obrecht C. et al. "The TheLMA project: A thermal lattice Boltzmann solver for the GPU." *Computers & Fluids* 54 (2012): 118-126