ISSN: 2290-8887
Volume 01, Issue 01, April,2019

Simulation of thermal flows with the aid of lattice Boltzmann method at the CUDA computational platform

D. Bukeikhanov[1], N. Abdildin[2]
Dept. of National Engineering Academy, Almaty, Kazakhstan, India[1, 2.]

Abstract— Originating from the lattice fuel automata principle, the lattice Boltzmann approach (LBM) is an interesting opportunity to the fixing of Navier-Stokes equations. In assessment to isothermal simulations, for a while, thermal glide simulations were challenging for LBM. Thermal float simulations are a vital mission in diverse fields of research. no matter a large amount of labor and studies the dynamics of thermally brought on flows are nevertheless relatively demanded. the motivation of this work is the improvement of a computational tool for simulation of the dynamics of thermal flows. To this purpose, we developed a LES-LBM solver multiplied with the aid of the pix Processing Unit (GPU) at the CUDA computational platform, integrating LBM with Large Eddy Simulation (LES). The simplicity of coding is typically an appealing function of the LBM. conventional implementations of LBM be afflicted by high memory consumption and bad computational performance. the main advantage of the solvers based on GPU is their potential to carry out notably extra floating factor operations per unit time (FLOPS) than a significant Processing Unit (CPU) and the right scalability of specific parallel algorithms. LES-LBM code changed into examined at the NVIDIA GeForce GTX 1050 ti and NVIDIA TESLA K80 GPUs.

Keywords— The lattice Boltzmann method, CUDA, thermal go with the flow.

1. Introduction

In June 2007 NVIDIA launched a brand new framework named CUDA for preferred parallel processing programs. This framework enables builders to implement GPU parallel packages in C, C++ languages and permits direct access to the GPU computing electricity without complex pics API. unique gear which can be included in an official software improvement package (SDK) permits debugging GPU programs in runtime. seeing that 2007 a whole lot of numerical libraries supplied. They allow to create efficient packages with much less attempt and cover such numerical algorithms like linear algebra operations, sparse matrix computations, Fourier transforms, photograph algorithms, and so on. In a marketplace NVIDIA has numerous separate merchandise, GeForce is for gaming, Quadro is for expert OpenGL based rendering and Tesla for excessive-performance computations. Tesla compute accelerators get sturdy positions in such high-performance regions as monetary evaluation and scientific computations. a lot of supercomputer carriers use NVIDIA GPUs to create electricity efficient computing clusters. one of the secrets and techniques of the excessive popularity amongst users is the help of maximum popular proprietary (CUDA) and open requirements (OpenCL, DirectCompute) for GPU programming.

The attempt to use GPU as a vastly parallel processor computational fluid dynamics (CFD) commenced at the start of 2000. one of the first guides become a bankruptcy within the GPU gems Books via M. J. Harris [1]. In bankruptcy 38 he described the belief of a simple fluid dynamics solver based totally on Stam's stable fluids [2]. Later numerous authors published the consequences of imposing the Marker and mobile approach on a GPU [three]. The smoothed particle hydrodynamics (SPH) is any other technique for simulation of the fluid dynamics, without direct use of the Navier Stokes equations. in the beginning, advanced in [four] SPH is a mesh free Lagrangian method that tracks the location and movement of many fluid particles, which allows direct mass conservation. One downside over grid-based techniques is the want for large numbers of debris to provide simulations of equal decision. The specific nature of the approach allows it to run successfully on huge parallel processors as GPU [5-7].

The lattice Boltzmann method (LBM) is a highly novel approach in computational fluid dynamics (CFD), which, not like most other CFD methods, does not rely not directly on fixing the Navier-Stokes equations by way of a numerical algorithm.

One of the most interesting features of LBM is that numerical technique has the data locality assets. such belongings may be very properly applied to be carried out in a vastly parallel processor, like GPUs [8-11]. In [12] LBM big Eddy Simulations for high Reynolds numbers had been done. advanced numerical implementation becomes capable of run on 4 Fermi-elegance GPUs concurrently. massive GPU memory (24GB) allowed to carry out simulations with an exceptionally high spatial decision (max grid length 10240x10240) with lively double precision mode. however, the authors stated that four GPUs have been placed on the same device, and there is a sturdy hardware problem for similar improvement of spatial resolution. to triumph over this fact, the authors encouraged extending their implementation to multinode GPU clusters.

Then again, whilst for describing hydrodynamic turbulence fashions based totally at the Navier-Stokes equations had been used almost exclusively for nearly two centuries, a significant increase in the hobby in LBM methods has lately been defined by using their computational efficiency. these strategies, primarily based on the Boltzmann equation, make it feasible to predict the macroscopic magnitudes of continuum mechanics, together with velocity and stress. although it becomes verified several years in the past that hydrodynamic turbulence can be accurately defined the use of those methods, the development of LES within LBM remains at a totally early level [13]. as an example, the LES-LBM techniques are utilized in [14-19].

2. Lattice Boltzmann Method

The simple quantity of the LBM is the discrete velocity distribution characteristic $f_i(\vec{x},t)$ which is likewise known as particle populations. Particle populations constitute the density of particles with velocity $\vec{c}=(c_{ix},c_{iy},c_{iz})$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 of discretizing the Boltzmann equation in speed area, bodily space, and time, we find the lattice Boltzmann equation:

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

Where is Ω collision operator and S_i is force. certainly one of handiest fashions for collision is Bhatnagar-Gros-Krook [20].

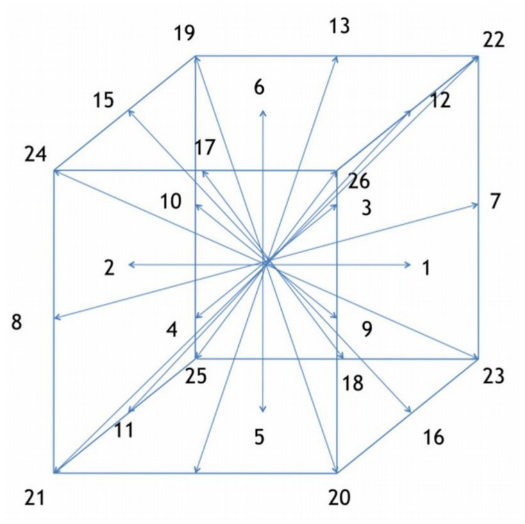

Figure 1 - D3Q27 lattice model used in the simulations

In many conditions, the work from viscous dissipation and compression is so small that it does no longer appreciably contribute to the warmth balance. it's miles then sufficient to consider an advection-diffusion

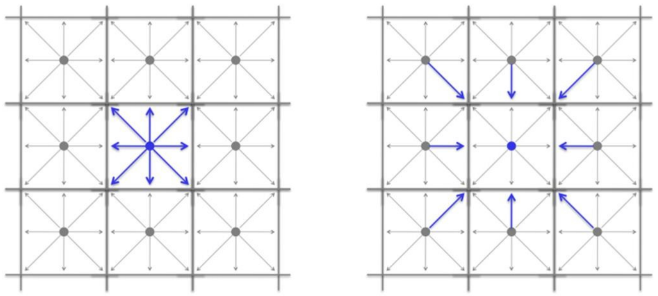

Figure 2 – Streaming from central node (a) and streaming into central node (b)

In many conditions, the work from viscous dissipation and compression is so small that it does no longer appreciably contribute to the warmth balance. it's miles then sufficient to consider an advection-diffusion

3. CUDA implementation

CUDA enabled simulation code is applied within the CUDA c program language period that's an extension to the c language. features in a CUDA C are marked as host features, tool functions, and kernels. Host functions are easy C functions accomplished with the aid of the host processor. tool functions are unique functions that have to be launched on a GPU. Kernel capabilities are used to release GPU features from the host code. Kernel characteristic runs in parallel at the GPU. all through the launch of the kernel loads of kernel copies executed in parallel with the aid of GPU. The execution sample is diagnosed through a grid. The grid is the unique configuration of the parallel threads, which can be grouped into blocks. Grid and block format might be one, two and 3 dimensional. Fig. three suggests an instance of the grid which has 6 blocks and two-dimensional layout. each block in fig. three also has a two-dimensional layout with nine threads, the overall quantity of threads 54.

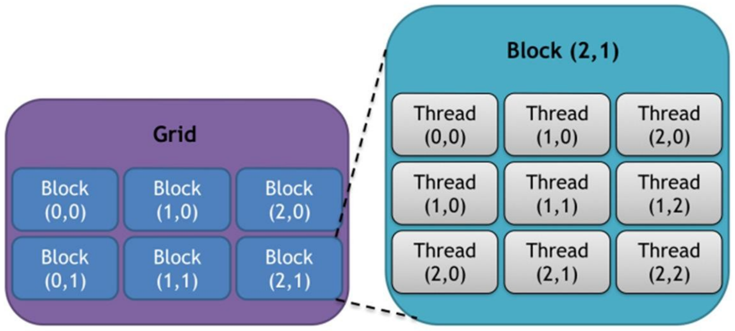

Figure 3 – CUDA execution model

Every thread is achieved via a Streaming Multiprocessor (SM). SM has several scalar processors (SP) which simply runs code. special scheduling algorithms and chips make use of the big amount of SP on a GPU.

For convenience, "threadIdx" is a 3-aspect vector, so threads may be identified the usage of a one-dimensional, -dimensional, or 3-dimensional thread index, forming a one-dimensional, two-dimensional, or 3-dimensional block of threads. This affords a herbal manner to invoke calculations for elements in a site, consisting of a vector, matrix, or quantity. The index of a threads and its identifier are without delay related to each different: they're the same for a one-dimensional block; for a - dimensional block of size (Dx, Dy), the thread identification of index(x, y) is (x +y Dx); for a three-dimensional block of length (Dx, Dy, Dz), the thread identity of index(x, y, z) is (x + y Dx + z Dx Dy). There is a restriction on the wide variety of threads in a block in view that it's miles expected that each one thread of a block may be on the identical processor core and have to proportion the confined memory sources of this middle. On current GPUs, a block of threads can contain as much as 1024 threads.

However, the kernel may be carried out by using numerous blocks of threads of the equal form, so that the whole wide variety of threads is equal to the number of threads within the block multiplied by way of the wide variety of blocks. list 1 offers the instance of GPU kernel function for the streaming degree in LBM simulations. Variables i, j, ok are used to assigning reminiscence vicinity 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 version introduces the concept of memory kinds. CUDA enabled GPU has a device, shared, texture, constant cache and sign up reminiscence. sign in memory is a very fast memory, with on-chip implementation. size of the sign in memory might also range between GPU's, but it also includes small, and cannot save arrays. Shared memory is a unique memory that is shared among threads of the equal block. proper use of shared reminiscence may also lower the burden to international GPU memory. international GPU memory or device reminiscence is a large reminiscence, which is sluggish comparably to different kinds of reminiscence, however, it has the advantage of huge size, often numerous gigabytes. There also are special caches to shop steady values and one-two-three dimensional textures. the use of texture cache may also boom the simulation pace if there exists a special pattern while getting access to facts in the cache.

To archive, excessive-performance CUDA implies that the program uses the proper kind of memory for diverse facts. The effects of the simulation should be uploaded to CPU or host memory using unique CUDA API calls.

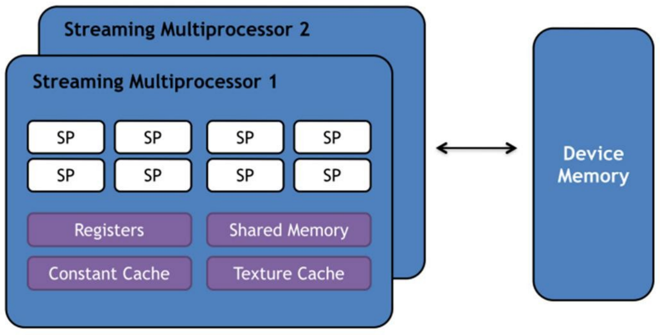

Figure 4 - CUDA hardware model

4. Results

The CUDA LBM solver applied on the GPU is used to have a look at the differential heated cubic hollow space outlined in Fig. 5. two contrary vertical partitions have imposed temperatures - T0 and +T0, while the remaining partitions are adiabatic. This configuration has been significantly studied inside the -dimensional configuration and diverse benchmark solutions are available.

All simulation parameters were similar to within the benchmark solution [11]. in an effort to carry out validation, the drift in the deferentially heated cavity is computed for Rayleigh numbers identical to 10^4, 10^5, 10^6 and 10^7. The results are in comparison with available records. table 1 offers the received Nusselt numbers as well as the values posted in [11]. GPU simulations display properly outcomes that are in accordance with the reference values. Simulations have been executed in a single precision mode on an NVIDIA 1050 and Tesla K80 GPUs. Their characteristics are presented in desk 2. Mesh size is 128x128x64. an additional comparison becomes made with a multicore workstation with 64 to be had cores and 256 threads. The outcomes of the performance evaluation are provided in fig. 6.

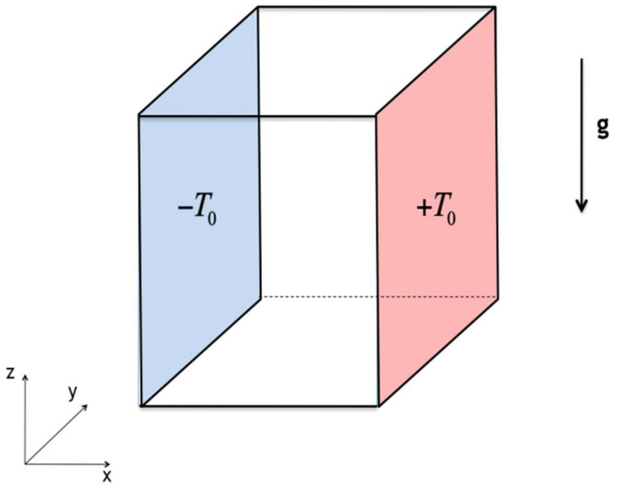

Figure 5 – Natural convection scheme

Table 1 – Comparison of Nusselt numbers

Rayleight number	**10^4**	**10^5**	**10^6**	**10^7**
Present	2.031	4.3302	8.6468	16.4193
Obrecht [11]	2.056	4.3382	8.6457	16.4202

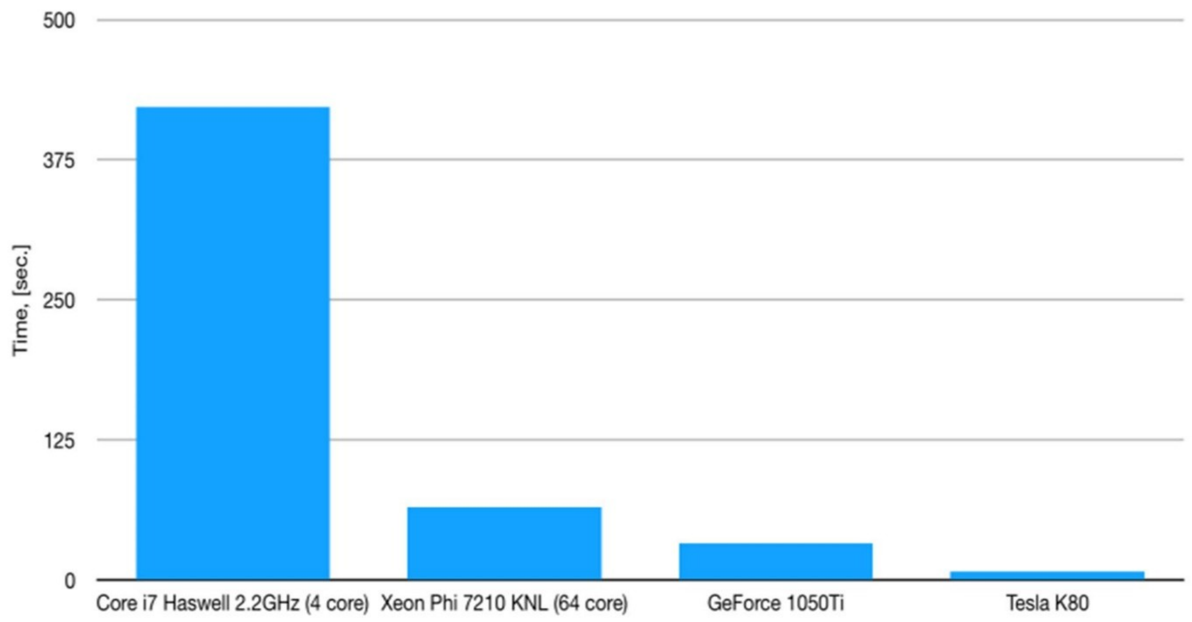

Figure 6 – Performance comparison

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

5. Conclusion

In this paper, we presented an in-house thermal LBM solver for CUDA enabled GPU workstations. The validity of the numerical algorithm turned into tested on a benchmark herbal convection hassle. The code uses separate populations for pace and temperature. in line with the timings GPU based totally LBM receives better overall performance in comparison with the unmarried CPU code. Code runtime also compared with the consequences of the OpenMP version of the code, which is completed on a Xeon Phi KNL computing device with 64 available cores. Promising results suggest that GPU elevated LBM is a superb alternative to CPU versions. The drawback of the CUDA code is the dependence at the CUDA platform and selections made by means of the Nvidia corporation. additionally, most of the CUDA libraries are closed source, or proprietary requirements, which is contrary to the function of the OpenMP. nevertheless, lots of enhancements are viable to the code. One is the extension of the algorithm to multi GPU and multi-node configurations and extra aggressive optimization for memory bandwidth. The developed code will be used in various thermal fluid waft simulations for which Boussinesq approximation for density fluctuations is legitimate.

6. 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*
10. *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
11. 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."
12. *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