CRANFIELD UNIVERSITY

JOSÉ OLIVEIRA

DEVELOPMENT OF PHYSICS OR HPC OPTIMISATION
OF A PARALLEL 2D LATTICE BOLTZMANN SOLVER
USING GPUS/CUDA

SCHOOL OF AEROSPACE, TRANSPORT AND
MANUFACTURING
Computational & Software Techniques In Engineering

MSc
Academic Year: 2016–2017

Supervisor: Dr Irene Moulitsas
August 2017

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND
MANUFACTURING
Computational & Software Techniques In Engineering

MSc

Academic Year: 2016–2017

JOSÉ OLIVEIRA

Development of Physics or HPC Optimisation of a parallel
2D lattice Boltzmann solver using GPUs/CUDA

Supervisor: Dr Irene Moulitsas
August 2017

This thesis is submitted in partial fulfilment of the
requirements for the degree of MSc.

Abstract

Type your abstract here.

Keywords

Keyword 1; keyword 2; keyword 3.

Contents

List of Figures

List of Tables

List of Abbreviations

CUDA   Compute Unified Device Architecture

CFD   Computational Fluid Dynamics

GPGPU   General-purpose computing on graphics processing units

Acknowledgements

The author would like to thank . . .

Chapter 1

Introduction

Chapter 2

Literature review

In this chapter, we will be taken into the broad field of the Lattice Boltzmann method using the CUDA platform. The topic of this thesis is associated with a number of different study areas, such as Computational Fluid Dynamics, Lattice Boltzmann method, High performance computing and GPGPU. We will then present an extensive literature review on these themes in the remaining of this chapter. Firstly, we will look into Computational Fluid Dynamics and the most commonly used approaches to it. Then we will be giving an overview on the Lattice Boltzmann method. Afterwards we will discuss how it is possible to employ parallelisation techniques in scientific computing. Finally we will review some previous works in this area.

2.1 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) comes from the need to model fluid flows and associated processes. A wide range of applications come from studying CFD, notably:

- Aircraft design [2]

- Solid particle erosion [10]

- Wind flow simulation [1]

- Combustion chamber simulations [11]

- Environmental and weather prediction [12]

- Automotive and motor sports [13, 3]

CFD can be used as a design and troubleshooting tool, as well as making the process dynamics easier to understand. It is used extensively by scientists and researchers, but it also has innumerable applications in the industry. CFD simulations is a viable tool for manufacturing because it eliminates expensive simulations.

As such, it is the science of determining a solution to fluid flow through space and time [5]. The models needed to calculate the fluid computations include:

- Flow geometry

- Differential (Governing) equations – These describe the physics and chemistry of the flow

- Boundary and initial conditions

- Discretization of the domain

2.1.1   Macroscopic scale

In this approach, the fluid can be seen as a collection of a huge number of particles. To solve these governing equations, one needs to apply conservation of energy, mass and momentum[9]. But since these equations are difficult, or even impossible to solve analytically, discrete schemes, boundary and initial conditions are used to convert these equations into a system of algebraic equations. These equations can then be solved until an appropriate solution is produced. NAVIERSTOKES

2.1.2 Microscopic scale

If we consider the fluid to be represented by individual particles then we will fall under the microscopic approach. In this approach, there is no definition of temperature or viscosity and collision between particles needs to be considered. Thus one needs to solve the differential equation of Newton's second law [9]. Hence, the location and velocity of each particle needs to be taken into account.

We can easily see that this approach becomes unfeasible for normal fluid sizes as the number of equations needed to be solved grows to the order of billions (consider that one mole of water contains more than 6×10^{23} molecules.

2.2 Lattice Boltzmann Method

The Lattice Boltzmann method (LBM) is a mesoscopic scale approach to CFD. It is used to describe a fluid based on probabilities using the Maxwell-Boltzmann equation in the fluid's equilibrium state [9].

In this method, we do not consider the individual characteristics of each particle. By grouping particles together in a D2Q9 (nodes containing 9 particles for 2D problems - Fig 2.1) or in a D3Q19 (nodes containing 19 particles for 3D problems - Fig 2.2) model, we can analyse the behaviour of the particles collectively [9].
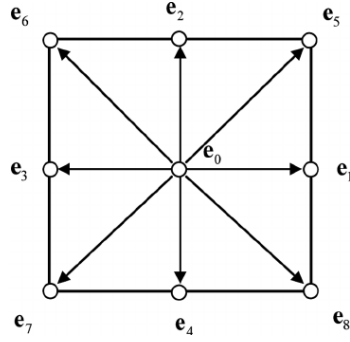
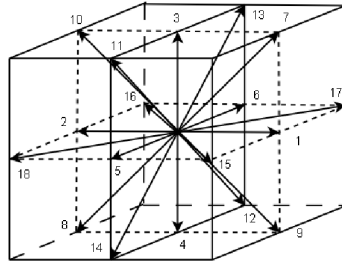Figure 2.1: 2D LBM model using 9 particles. (D2Q9)

Figure 2.2: 3D LBM model using 19 particles. (D3Q19)

This way we can reap the advantages of both the macro and microscale approaches without the need of high end computers [9]. Since communications between nodes are very limited, LBM also offers the possibility of employing parallel computing to achieve the solution in even faster times.

2.3 High Performance Computing

As humanity evolves, so too does our desire for expanding previous unobtainable goals. As computer technology kept progressing further and further, we soon realised that some problems simply took too many resources to be completed. But what if instead of focusing on computational power we focused our efforts in splitting the workload? MOORE's LAW

High performance computing (HPC) comes from the harnessing of computer power to deliver a much higher performance that one could not obtain from a typical computer. To this end, we can talk of HPC as being a collection of computer resources, all of them working simulateously to achieve a solution of the same problem. DIFFERENT PARALLELISATION STRATEGIES FOR SPLITTING THE WORKLOAD EMERGE, DEPENDING ON THE UNDERLYING HARDWARE. Problems that could otherwise take weeks, months or even years can now be solved in minutes, hours or days under these powerful devices.

2.3.1 Distributed memory

2.3.2 Shared memory

2.3.3 GPGPU

General Purpose computing on Graphical Processing Units stands for the use of Graphics Processing unit (GPU) to perform computations on applications normally performed by the CPU. One of the main advantages of using this approach is the amount of cores that a single GPU has. While a typical desktop CPU has up to 4 cores, a GPU can have thousands of cores, allowing users to take advantage of its massively parallel architecture. GPUS CAN NOW SOLVE PROBLEMS THAT WERE TRADITIONAL FOR CPU. GPU ARE CHEAPER AND ITS NOT POSSIBLE TO GET CPUS AS POWERFULL

CUDA was then developed by NVIDIA SWITCH THE ORDER. CUDA code allows programmers to take advantage of GPUs by employing a unified shader pipeline under the familiar C language [6]. Users were no longer required to have specific knowledge of OpenGL or DirectX and could now perform general computations (rather than graphic-specific computations) whilst benefiting from the massive computational powered offered

by GPUs. OPENCL

2.4   Previous parallelisation works

This thesis is a continuation of work that started some years ago in Cranfield University. As such, the work from the previous students needs to be analysed.

In 2014, Tamás Józsa and Máté Szőke, adapted two different inhouse C and C++ codes into one single C code unifying the advantages of each one of the two original codes [7, 14]. Józsa then parallelised the critical parts of the C code using CUDA and ran tests on the Fermi GPU Cluster from Cranfield, achieving a three times speedup in general, with a peak of 15 times speedup [7].

Szőke proposes a CPU APPROACH different parallelisation approach using Unified Parallel C, which is a Partitioned Global Address Space language[14]. This means that it's possible to use shared memory to compute the solution. However, the author verified that a local memory-based approach (like MPI) provided the best results. He also compared the results obtained with the ones obtained by Józsa on the CUDA approach. They found that to achieve the same speed-up as that of a single GPU card, one needs an entire workstation (16 threads in the case of Astral) [14].

In 2015, Ádám Koleszár continued the work and further optimised the parallel version of the LBM method using CUDA [4]. He did an excellent job, resulting in a 10 times faster execution than the previous 2D parallel solver, which means that his new optimised code was 30 times faster than the original, in-house, serial solver.

Finally, in 2016, Maciej Kubat proposed a new version of the LBM solver. Firstly he converts the 2D parallel solver to a 3D parallel solver which entailed a major reengineering of the code, from data containers to logic cycles [8]. After first trying for a direct adaptation, he found that his code was too slow to produce meaningful solutions. After

optimising his own code he was able to reach an almost one hundred times speedup. However, Kubat states that a lot can be done to improve his code, from boundary conditions to code readability and maintainability.

Chapter 3

Conclusions

References

[1] B. Blocken, A. van der Hout, J. Dekker, and O. Weiler. Cfd simulation of wind flow over natural complex terrain: Case study with validation by field measurements for ria de ferrol, galicia, spain. *Journal of Wind Engineering and Industrial Aerodynamics*, 147:43–57, 2015.

[2] R. Czyba, M. Hecel, K. Jablonski, M. Lemanowicz, and K. Platek. Application of computer aided tools and methods for unmanned cargo aircraft design. pages 1068–1073, 2015.

[3] S. Desai, E. Leylek, C.-M.B. Lo, P. Doddegowda, A. Bychkovsky, and A.R. George. Experimental and cfd comparative case studies of aerodynamics of race car wings, underbodies with wheels, and motorcycle flows. *SAE Technical Papers*, 2008.

[4] Ádám Koleszár. Optimisation of 2d lattice boltzmann method using cuda, 2015.

[5] Joel Ducoste. An overview of computational fluid dynamics. Ghent University, 2008.

[6] E. Kandrot J. Sanders. *CUDA by Example: An Introduction to Generalpurpose GPU Programming*. Addison-Wesley Professional, 2010.

[7] Tamás I. Józsa. Parallelization of lattice boltzmann method using cuda platform, 2014.

[8] Maciej Kubat. Development of physics or hpc optimisation of a parallel 2d lattice boltzmann solver using gpus cuda, 2016.

[9] A. A. Mohamad. *Lattice Boltzmann Method.* 2011.

[10] D.A. Pandya, B.H. Dennis, and R.D. Russell. A computational fluid dynamics based artificial neural network model to predict solid particle erosion. *Wear*, 378-379:198–210, 2017.

[11] A. Penkner and P. Jeschke. Analytic rayleigh pressure loss model for high-swirl combustion in a rotating combustion chamber. *CEAS Aeronautical Journal*, 6(4):613–625, 2015.

[12] S. Reichrath and T.W. Davies. Using cfd to model the internal climate of greenhouses: Past, present and future. *Agronomie*, 22(1):3–19, 2002.

[13] B.S. Rosen, J.P. Laiosa, and W.H. Davis Jr. Cfd design studies for america's cup 2000. 2000.

[14] Máté Tibor Szőke. Efficient implementation of a 2d lattice boltzmann solver using modern parallelisation techniques, 2014.