Graphics Processors as a Tool for Rotating Detonation Simulations

Michal Folusiak [1,2], Karol Swiderski [1], Arkadiusz Kobiera [2], Piotr Wolanski [1,2]
[1] Institute of Aviation
[2] Warsaw University of Technology
Warsaw, (Poland)

1 Introduction

In this work advantages of novel CUDA technology are utilized in order to develop fast and robust GPU versions of existing in-house codes used for detonation simulations. Speed of these codes is expected to outperform their classical counterparts. In order to quantitatively compare results of this work, numerical benchmarks of detonation problem were created. Computations of the same problem of cylindrical detonation were carried out on 4 similar domains and have shown that some procedures running on GPU can be faster even by the order of magnitude than the one running on CPU.
On the other hand, the use of GPU demands very careful memory management, especially if more complicated cases and chemical mechanism are involved. However, authors believe, that the compilers will soon evolve in a way to explicitly perform most critical optimizations of memory management, into models which are simpler, more efficient and easy to implement.

2 Motivation and problem description

The purpose of this work was to develop fast parallel library containing numerical procedures for rotating detonation CFD simulations. These procedures are suited for the existing in-house codes REFLOPS[1] and REFLOPS2 and were written in CUDA Fortran, which allows programmer to utilize the computational power of GPU in Fortran.
The aforementioned in-house codes enable simulation of compressible, inviscid, and reactive semi-ideal gas flows in complex computational domains on structured and unstructured grids respectively. In both codes, the time splitting method is used, such that each term of the Euler equation – advective, geometrical and chemical is integrated separately. In RELOPS the Euler equations are of the form:

$$\frac{\partial Q}{\partial t} = -\frac{\partial E}{\partial \xi} - \frac{\partial F}{\partial \eta} - \frac{\partial G}{\partial \zeta} - H - S$$

, where:

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \\ \rho_i \end{bmatrix} \quad E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E+p) \\ \rho_i u \end{bmatrix} \quad F = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ v(E+p) \\ \rho_i v \end{bmatrix} \quad G = \begin{bmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ w(E+p) \\ \rho_i w \end{bmatrix} \quad S = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \omega_i \end{bmatrix} \quad H = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

, or

$$H = \frac{1}{r} \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 \\ \rho wv \\ v(E+p) \\ \rho_i v \end{bmatrix}.$$

Correspondence to: michal.folusiak@ilot.edu.pl

Fluxes (E,F,G) in above equation are evaluated by designated numerical schemes and solvers. Among many methods implemented in our codes, the one most commonly used by authors for the detonation problem is second order HLLC-WAF–TVD. The advective term is then integrated using explicit Euler method, second (RK2) or fourth order (RK4) Runge-Kutta method. RK4 method is used also for integration of geometrical source terms in case of cylindrical grids in REFLOPS.
The chemical source term may be integrated using Chemeq2 or DVODE subroutines. The source term S is chemical composition change rate resulting from the chemical reactions. The production rate of chemical compounds ω_i is an overall sum of the production and destruction rates for a given chemical compound in all reactions taken into account in the chemistry model. The rate of the reactions is described by the Arrhenius equation. Chemeq2 is stiff ordinary differential equations (ODE) solver designed to integrate equations arising from reaction kinetics. The other solver is much more complex tool and is widely used to solve initial value problem of stiff or non-stiff ODEs. The stiffness of chemical kinetics ODEs system makes integration of these terms the most computationally costly part of most reactive flow simulations.
Results of benchmarks and detonation simulations have shown that the workload varies in range 20-60% and 40-80% of the program execution time for the CHEMEQ2 and DVODE respectively. About 80% of the remaining computational time is consumed by the scheme that computes numerical flux in REFLOPS2 on unstructured grids. This workload can be slightly reduced by lowering the level of spatial accuracy or by use of structured grids in REFLOPS.
Taking into consideration the above results of performance tests authors decided that the speedup of the code may be achieved by reducing the computational time of two most resource-consuming procedures: evaluation of numerical fluxes and chemistry integration. Since over 90% of the code was parallelized in the OpenMP, it was clear that the CPU version of the code has already reached its dead end. Therefore it was decided that the best way to gain a considerable performance benefit is by applying the novel technology – Computed Unified Device Architecture (CUDA).

3 Basis of CUDA technology

The cores of conventional processors (CPUs) are today very close to reach their speed limits. The speedup of computations is achieved mainly by introducing multicore architectures, advanced RAM chips or distributed computing. In the same time, graphics processing units (GPUs) which are specialized in highly parallel floating point operations on large datasets have reached the computational power of order of TFLOPS and exceeded the computational power of CPUs [2].
The development of these devices is driven by insatiable market of video games. The cause of the arising difference in performance between GPU and CPU technology is a result of the specificity of GPU architecture, graphics processing unit is massively parallel and scalable device that consists of hundreds of processing units. These processors are grouped into multiprocessors that can be devoted to different tasks.
On the other hand it is very hard to exploit computational power of GPUs for general purpose computations using very specialized libraries designed for image processing such as OpenGL. Fortunately, the vendors has noticed the potential of graphics processors and started to develop new tools which allow accessing these technology by non-specialists. In 2006, NVIDIA introduced Compute Unified Device Architecture (CUDA) – the new architecture of GPUs which comes with software environment that allows to use C, and recently Fortran, as a high level programming language.
Programming in CUDA C [2,3] or CUDA Fortran [4] is much more demanding and sensitive to the programmer's faults and with hundreds parallel threads hard to debug when comparing to its CPU counterparts. The programming model of GPU is very different from the CPU's programming model. Thus the programmer new to the CUDA technology may found the language to be uncomfortable and performance benefits to be unpredictable. This is why it is crucial to understand limitations of CUDA architecture in order to gain performance benefits.

4 The implementation

The development of CUDA technology is supported by the development of its two main official tools – CUDA C and CUDA Fortran compilers. However till today a few other compatible language extensions and tools is used. One of the most important is OpenCL, which is being developed as a cross-platform, "royalty-free" parallel programming language [5]. In authors' opinion the most interesting and perspective group of tools is based on compiler directives. Syntax of such a set of compiler directives is very similar to the one used in OpenMP. The most mature tool from this group is PGI Accelerator model [6]. It is developed for C and Fortran compilers, and

allows fast and implicit parallelization of simple codes on GPU. This model is still very limited and immature, but in authors' opinion, such simple tools will evolve into the most widely used and the fastest methods of parallel programming of massively parallel architectures. Comparison of variety of popular tools is presented in [7].

Till today, a few attempts have been made to employ the emerging CUDA technology in CFD. Most of works refers to the aerodynamics [8][9][10] or astrophysics [11]. The widest range of implementations concerns structured grids and simple low order numerical schemes. Number of unstructured grid solvers is very limited, because its random numbering of grid cells requires special treatment and makes coalesced memory access, which guarantees the best performance, impossible. Due to the complicated access pattern and limited size of low-latency shared memory, most of unstructured grid solvers use very simple numerical schemes (in order to reduce stencil size) or even do not take advantage of these on-chip memory.

Wide range of speedups is reported in papers concerning the applications of CUDA in CFD. These are ranging from 2.5 for Navier-Stokes equations on unstructured grids to 50 on structured grids. However, it is very hard to make any comparison of these results found in papers, because each speedup is referred to the unique CPU and GPU combination.

As it was mentioned above, two the most computationally expensive procedures in REFLOPS and REFLOPS2 codes are: integration of chemical source terms and evaluation of inter-cell flux. It was decided that the parallelization must be performed for one of two chemical solvers in the first place, because these were identified to be the largest performance bottleneck in the rotating detonation simulations. What is more chemical computations are performed independently for each cell, thus many problems related to the memory conflicts are automatically avoided. On the other hand, the procedure that evaluates numerical fluxes is much simpler than the one, but potential memory conflicts and the issue of the available shared memory must be carefully studied, particularly for memory consuming higher order solvers for unstructured grids.

The initial research showed that the process of moving DVODE, (which is very universal but also complex) to the GPU is due to the current limitations to the language and device architecture hardly possible and may not result in any performance benefits. Therefore, only the other, much simpler and a little faster procedure – Chemeq2 – was parallelized in CUDA. All the computationally demanding thermodynamic subroutines and functions referenced by this procedure must be also moved to the GPU.

Before the code, can be moved to the GPU, it must be carefully studied and reorganized in order to fulfill strict requirements of the language and architecture. The only way the performance benefit can be obtained in CUDA today is by following these requirements and understanding the advantages and disadvantages of the GPU technology. It is very similar to programming a low-level programming language – much more demanding to the programmers' faults and hard to debug.

The new kernel subroutine that can be called from CPU code is usually an equivalent of the large DO loop in CPU code. However, the code in such subroutine is no longer sequential and must be optimized in order to obtain maximum performance benefit resulting from using shared memory instead of global device memory. Because the amount of shared memory is limited, it is usual that some non-conflicting variables resides in the same memory space and are used alternately in subroutines. This makes the code hardly legible, but allows programmer to maximize the use of available fast memory.

The size of shared memory is being increased by vendor in each version of CUDA architecture, however the amount available in our GPU (version 1.3) is still only 16 kB per multiprocessor. This is far too low for most simulations of chemical kinetics, therefore for complex mechanisms the order of floating point operations accuracy must be lowered to single precision. It allows to save half the size of the memory, and also the speed of such codes is increased on devices of architecture version lower than 2.0.

Thermodynamic procedures are common for both REFLOPS and REFLOPS2 codes. Due to the differences in data structure of these codes separate kernel procedures have been implemented for each one. It was found that for procedure that integrates chemical kinetics it is generally easier to operate on unstructured grids in CUDA, particularly in the context of construction of computational grid of blocks.

All the thermodynamic functions translated to CUDA Fortran and the kernel routines are placed in the single module that is compiled into the static library that can be linked to any code and programming language. The call to the kernel functions is done by interface routines which are responsible for data initialization on GPU or multiple GPUs in OpenMP. These subroutines suited for REFLOPS and REFLOPS2 are also contained in the library.

5 Results

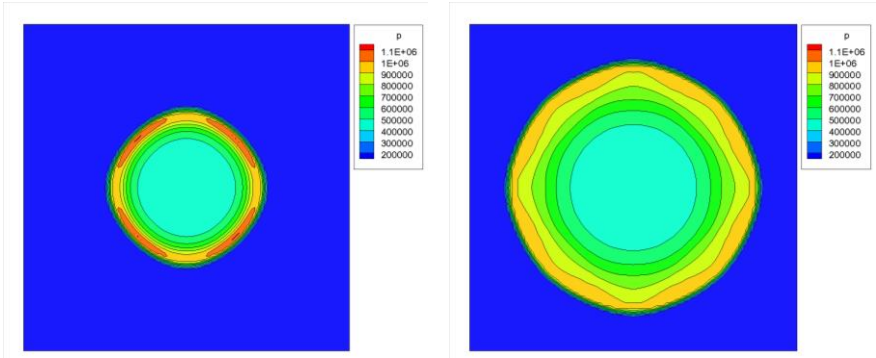
In order to quantitatively compare results of work, numerical benchmarks of detonation problem were created. Computations of the same problem of cylindrical explosion were carried on similar Cartesian domains (Table 1), the size of the domain was 2*2*0.02 m in each case. The detonation wave was initiated by energy source in the centre of the domain filled with stoichiometric hydrogen and air mixture.

Table 1 Computational domains. Number of divisions along z axis is a multiple of GPUs count used in benchmarks (2 and 3) for the highest performance.

No.	x divisions	y divisions	z divisions	cells
1	100	100	6	60 000
2	100	200	6	120 000
3	200	200	6	240 000
4	200	200	12	480 000

The hardware used in this comparison was of "middle class": Intel Core i7-930 QuadCore,2.8GHz(8 cores) and 3 NVIDIA GeForce GTX 285. Unless otherwise indicated, the reported speedups are calculated by division of computational time of program or procedure executed on 8 cores of the CPU in OpenMP by the execution time of the GPU version of the code.

CPU version of both chemistry solvers and GPU version of Chemeq2 code were implemented in REFLOPS, whereas there is no CPU version of Chemeq2 implemented in REFLOPS2. The results of the benchmark for two solvers are shown in Fig.1. It is noticeable that the two solvers give significantly different results. The detonation calculated by Chemeq2 is faster but the pressure peaks are lower. This is because the numerical method used in Chemeq2. These and previous calculations [1] indicate that in Chemeq2 the reactions faster approach the equilibrium conditions what leads to higher velocity and lower pressure of the leading shock. The DVODE calculates slower global reaction rates which causes reduction of the heat released in the initial phase of detonation and increases the pressure peaks as well. However, the analysis of the features of these two solvers is not in the scope of this work, only their features in terms of application on GPU architecture is studied. These computations showed also that for the considered case DVODE is 2.2 times slower than the other solver, what must be taken into account when comparing speedups of CPU and GPU versions of REFLOPS2 chemistry. Only tiny and acceptable differences were observed between single and double precision versions of the same solvers and no differences in results were observed between CPU and GPU versions of the Chemeq2. Results were compared at the same solution time T=0.0004 s.



a) b)

Figure 1 Pressure contours(in Pa) of cylindrical explosion for benchmark 1: a) DVODE, b) Chemeq2

In the Table 2 performance results of benchmark 1 are presented. It can be seen that the highest speedup of chemistry procedure is achieved for 3 parallel GPUs. It is 2.7 times faster than for single GPU and over 55 times faster than for single CPU core. In the Table 3 results of the benchmark 4 are presented. The performance benefit is the greater the larger is the computational domain. For very large datasets, the computational time of chemical procedures becomes negligible when comparing to the total time (Table 4).

Table 2 Benchmark results on domain 1 in REFLOPS.

Number of CPU cores / GPUs	Version	Time		% chemistry	Speedup
		total [s]	spent on CHEMEQ2 [s]		
1	CPU	349.8	227.7	65%	0.2
8	CPU	73.7	46.9	64%	1.0
3	CUDA, SP	34.0	4.2	12%	**11.1**
2	CUDA, SP	36.0	6.0	17%	7.8
1	CUDA, SP	41.5	11.6	28%	4.1
3	CUDA, DP	45.0	14.9	33%	3.2
2	CUDA, DP	51.2	21.5	42%	2.2
1	CUDA, DP	72.0	42.4	59%	1.1

Table 3 Benchmark results on domain 4 in REFLOPS.

Number of CPU cores / GPUs	Version	Time		% chemistry	Speedup
		total [s]	spent on CHEMEQ2 [s]		
8	CPU	1365.8	857.0	63%	1.0
3	CUDA, SP	531.2	39.8	7%	**21.5**
1	CUDA, SP	596.5	107.3	18%	8.0

Table 4 Performance comparison for benchmark 1-4 in REFLOPS on 3 GPUs, SP.

Benchmark	Time		% chemistry	Speedup
	total [s]	spent on CHEMEQ2 [s]		
1	34.0	4.2	12%	11.1
2	132.3	10.3	8%	19.9
3	269.7	20.8	8%	20.0
4	531.2	39.8	7%	21.5

In Figure 2 the pressure iso-surfaces and mixture composition contours in the chamber of Rotating Detonation Engine (RDE) are shown. These calculations were conducted in order to check the interaction of the detonation wave with external air. The structured grid was used in these calculations. The increase of calculation speed was about 20 times.

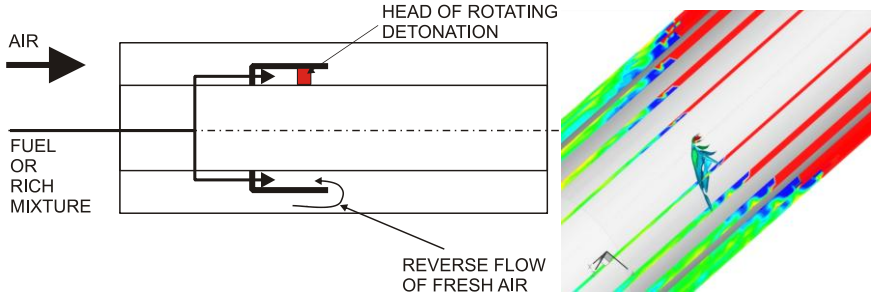


Figure 2 Simulations of rotating detonation with axiluary air supply a) scheme of the detonation chamber, b) pressure iso-surfaces and mixture composition contours.

In Figure 3 the contours of static pressure in the chamber of Rotating Detonation Engine(RDE) are shown. In this simulation of the detonation wave development in the domain consisting of 1 million cells of unstructured grid, it

was measured that CHEMEQ2 running on 3 GPUs is 26 times faster than DVODE running on 8 cores of CPU. The total execution time becomes two times lower in this case.

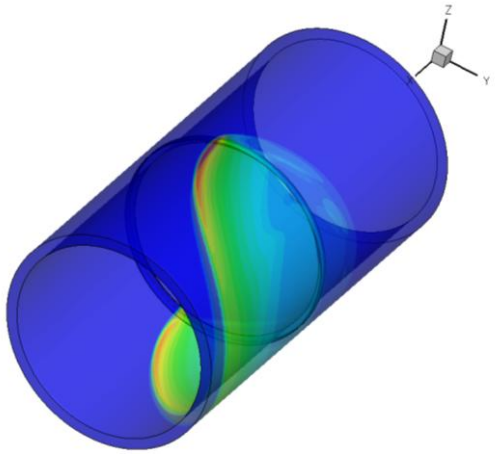


Figure 3 Pressure contours in the initial phase of rotating detonation case

6 Summary

In the paper, the implementation of the novel CUDA technology into two in-house CFD codes designed for detonation simulations, was presented. Both advantages and disadvantages resulting from the very different architecture and programming model of the GPU were shortly described. The presented results of detonation simulations showed that the emerging technology is very promising, speedups of order of magnitude can be achieved in case of chemistry kinetics solver which is quite a good result for an implementation of an iterative method on GPU. However, the presented results are preliminary and benchmark should be tested on the latest NVIDIA Fermi architecture where the DP is fully supported. Further optimizations of this procedure must be preceded by the development of an efficient, unstructured grid based, numerical scheme that will evaluate the intercell flux on GPU. GPUs are for sure the future of computational physics and it is expected that the architecture will become less restrictive. Authors believe that also programming tools will evolve into simple language extensions or sets of compiler directives similar to PGI Accelerator model.

Acknowledgements

This work was conducted in frame of project UDA-POIG.01.03.01-14-071 “Turbine engine with a detonation combustion chamber” supported by EU and Ministry of Regional Development.

References

[1] Kobiera A., Folusiak M. Swiderski K., Wolanski P., “REFLOPS” – a new parallel CFD code for reactive Euler flow simulation, Archivum Combuistionis, vol. 29(2009), No.3-4
[2] NVIDIA, NVIDIA CUDA C Programming guide, ver 3.2
[3] NVIDIA, NVIDIA CUDA C Best practices guide, ver. 3.2
[4] The Portland Group, CUDA Fortran Programming guide and reference, ver. 1.3
[5] http://www.khronos.org/opencl/
[6] The Portland Group, PGI Fortran & C Accelerator programming model, ver. 1.3
[7] Larkin,J., Cray Inc., A comparison of accelerator programming models, www.slideshare.net
[8] Martin M.J. et al, Solving the Euler equations for unstructured grids on graphical processor units, 2009
[9] Corrigan A. et al, Running unstructured grid based CFD solvers on modern graphics hardware, 19th AIAA CFD, 2009
[10] Kampolis I.C. et al, CFD-based analysis and two-level aerodynamic optimization on graphics processing units, Cumput. Mtehods Appl. Mech. Engrg. 199(2010)
[11] Schive H., Tsai Y., Chiueh T., GAMER: a GPU-accelerated adaptive mesh refinement code for astrophysics, Astrophysical Journal Supplement Series 186 (2010) 457-484