Acta Astronautica xx (xxxx) xxxx–xxxx

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/aa

Analysis of impact of general-purpose graphics processor units in supersonic flow modeling

V.N. Emelyanov[a], A.G. Karpenko[b], A.S. Kozelkov[c], I.V. Teterina[a], K.N. Volkov[d,*], A.V. Yalozo[c]

[a] Faculty of Rocket and Space Engineering, Baltic State Technical University, St. Petersburg 190005, Russia
[b] Faculty of Mathematics and Mechanics, St Petersburg State University, Old Petergof, St. Petersburg 198504, Russia
[c] Russian Research Institute of Experimental Physics, Russian Federal Nuclear Center, Sarov 607188, Russia
[d] Faculty of Science, Engineering and Computing, Kingston University, London SW15 3DW, United Kingdom

ARTICLE INFO

Keywords:
Supersonic flow
Shock tube
Boundary layer
CFD
High-performance computing
Parallel algorithm
Speedup

ABSTRACT

Computational methods are widely used in prediction of complex flowfields associated with off-normal situations in aerospace engineering. Modern graphics processing units (GPU) provide architectures and new programming models that enable to harness their large processing power and to design computational fluid dynamics (CFD) simulations at both high performance and low cost. Possibilities of the use of GPUs for the simulation of external and internal flows on unstructured meshes are discussed. The finite volume method is applied to solve three-dimensional unsteady compressible Euler and Navier–Stokes equations on unstructured meshes with high resolution numerical schemes. CUDA technology is used for programming implementation of parallel computational algorithms. Solutions of some benchmark test cases on GPUs are reported, and the results computed are compared with experimental and computational data. Approaches to optimization of the CFD code related to the use of different types of memory are considered. Speedup of solution on GPUs with respect to the solution on central processor unit (CPU) is compared. Performance measurements show that numerical schemes developed achieve 20–50 speedup on GPU hardware compared to CPU reference implementation. The results obtained provide promising perspective for designing a GPU-based software framework for applications in CFD.

1. Introduction

Propulsion power engines play an important role in determining space flight safety issues [1]. Modeling of fluid chemically reacting flows and heat transfer in rocket engines is necessary for adequate prediction of the functional efficiency and reliability of rocket engines [2–5] and nozzles [6]. It was demonstrated tha graphic processor units (GPU) could accelerate solution of these problems [7,8]. New generation of propulsion engines wjould also, definitely, need effective mathematical simulations [9,10]. The present paper discusses the effectiveness of GPU for fluid dynamics simulations relevant to space flight safety.

The methods of computational fluid dynamics (CFD) are extensively applied in design and optimization of rocket techniques to get more insight into 3D unsteady flows through fluid or gas passages. Accurate prediction of compressible flows still remains a challenging task despite a lot of work in this area. The quality of CFD calculations of the flows strongly depends on the proper prediction of flow physics (shock waves, rarefaction waves, recirculation regions). Investigations of heat transfer, skin friction, secondary flows, flow separation and reattachment effects demand reliable numerical methods, accurate programming, and robust working practices.

The stagnation in the clock-speed of central processing units (CPU) has led to significant interest in parallel architectures that offer increasing computational power by using many separate processing units. Modern graphics hardware contains such an architecture in the form of the graphics processing units (GPU). GPU platforms including GPU clusters make it possible to achieve speedups of an order of magnitude over a standard CPU in many CFD applications and are growing in popularity [11].

Fig. 1 shows that a recent GPU is significantly more powerful than its CPU contemporary, and that the computing power of GPUs are increasing at a greater rate than that of CPUs. The GPU employs a parallel architecture so each generation improves on the speed of previous ones by adding more cores, subject to the limits of space, heat and cost. CPUs, on the other hand, have traditionally used a serial design with a single core, relying instead on greater clock speeds and shrinking transistors to drive more powerful processors. While this

* Corresponding author.
E-mail address: k.volkov@kingston.ac.uk (K.N. Volkov).

http://dx.doi.org/10.1016/j.actaastro.2016.10.039
Received 18 August 2016; Received in revised form 22 October 2016; Accepted 25 October 2016
Available online xxxx

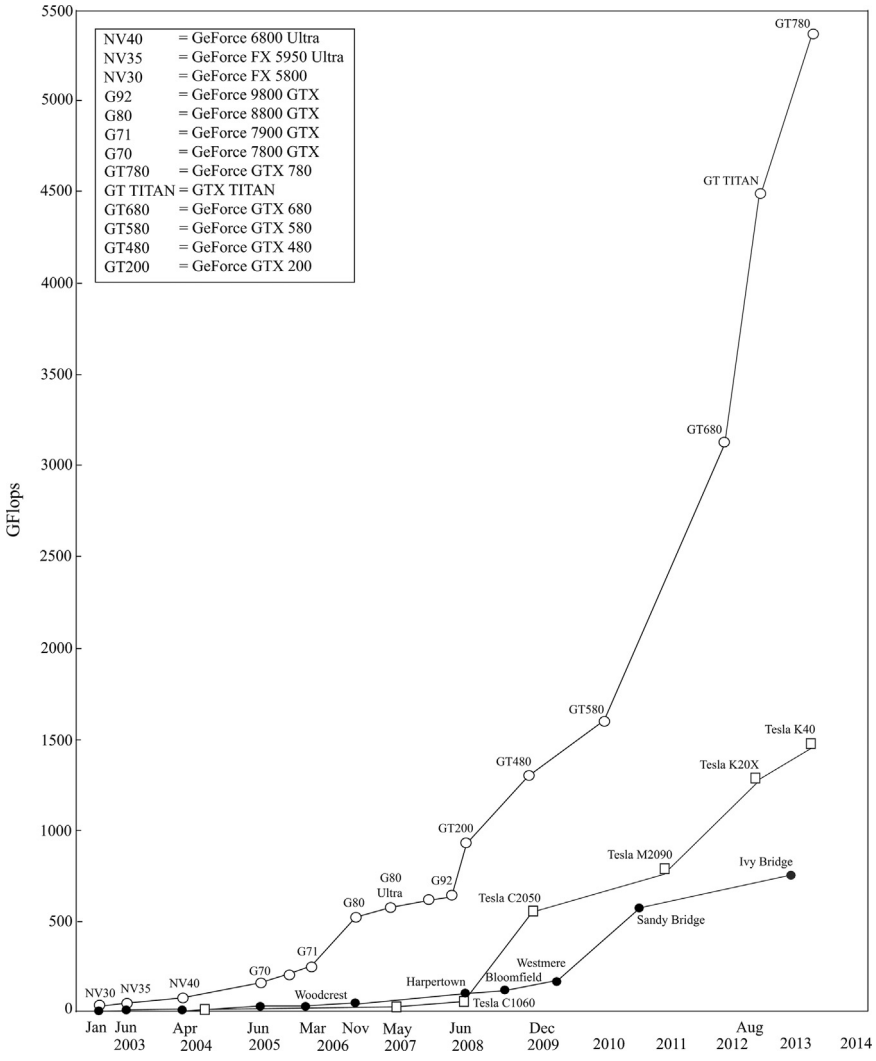

Fig. 1. Floating point operations per second for the CPUs and GPUs.

approach has been reliable in the past, it is now showing signs of stagnation as the limit of current manufacturing technology is being reached. Recent CPUs, therefore, tend to feature two or more cores, but GPUs still enjoy a significant advantage in this area [12].

Speed and accuracy are key factors in the evaluation of CFD solver performance. In CFD applications, the increasing demands for accuracy and simulation capabilities produce an exponential growth of the required computational resources. High performance computing (HPC) resources are widely used in engineering applications.

The use of GPUs is a cost effective way of improving substantially the performance in CFD applications [13]. Taking advantage of any multi-core architecture requires programs to be written for parallel execution. For CFD, this has traditionally meant splitting the flow domain into several parts (domain decomposition) that are solved independently on each processor node in a cluster, with the flow properties at boundaries being communicated between the nodes after each time step (processor balancing). This is also the process adopted for GPUs, but the GPU introduces several additional constraints that make the stream programming paradigm particularly useful [14].

Although GPU has attractive characteristics for massively parallel computations, it has not been implemented in CFD for a long time due to the complex programming techniques. Developers must have special knowledge about computer graphics which is unfamiliar for general CFD researchers. But thanks to the CUDA (Compute Unified Device Architecture) library provided by NVIDIA, researchers are free from the restrictions of computer hardware knowledge and need to concentrate on CFD algorithms and CUDA programming language.

Depending on the complexity of the CFD problem to represent and solve, structured or unstructured meshes are used. Computational algorithms are more efficiently implemented on structured meshes, and data structures to handle the mesh are easy to implement [15,16]. However, structured meshes present poor accuracy if the problem to be solved has complex internal or external boundaries. On the other hand, unstructured meshes present more flexibility and higher accuracy to represent problems that have complex geometries and boundaries [17]. However, the data structures to handle it are not easy to implement, and also explicit neighboring information should be stored.

Much of the efforts in running CFD codes on GPUs has been directed toward the case of CFD solvers based on structured and block-structured meshes [14,18–23]. These solvers are easily to implement on GPUs due to their regular memory access pattern. There are various examples of implementation of CFD solvers on structured meshes for simulation of flows of viscous incompressible fluid [24–26].

Unstructured mesh based analysis methods on HPC systems with shared memory and distributed memory have been largely studied. However, shared and distributed memory systems are fundamentally different from GPUs. A GPU is a SIMT (Single Instruction Multiple Thread) engine, whereas shared and distributed memory systems are MPMD (Multiple Program Multiple Data) engines. However, the common aspect of these parallel engines is that in both of them the mesh application is limited by memory latency. Achieving good performance for unstructured mesh based CFD solvers on GPUs is