UDC 621.38:629.735.015:004.93 (045)

S. S. Tovkach

CUDA-BASED TECHNOLOGY FOR IMPROVING THE EFFICIENCY OF THE AIRCRAFT MOTION

Automation and Energy Management Department, National Aviation University, Kyiv, Ukraine
E-mail: ss.tovkach@gmail.com

***Abstract*—*Considered the method of parallel computing based on CUDA-architecture with detecting large and small scale details of turbulence flow to adapt flight dynamics for motion control of the aircraft. Defined the acceleration value of the parallel implementation relatively to series and the integral effectiveness of parallel computing that allows to use the NVIDIA Tegra graphics processors to increase the processing power of massively parallel calculations.*

Index terms—Graphics processing unit; multi-threaded; flight dynamics; adaptive motion control; wavelet analysis; turbulence flow, parallel computing.

I. INTRODUCTION

The aircraft motion often deals with the control of flight dynamics. It can be considered a branch of system dynamics in which the system studies is a flight high-speed vehicle [1]. The field of flight dynamics is divided into following aspects:

– performance – the short time scales of response are ignored, and the forces are assumed to be in quasi-static equilibrium, as a result, flight speeds, rate of climb, maximum range, and time aloft is investigated;

– stability and control – the short- and intermediate-time response of the attitude and velocity of the vehicle is considered;

– navigation and guidance – the control inputs required to achieve a particular trajectory are considered.

In these notes can be focused on the issues of performance, stability and control. These aspects of the dynamics can be treated somewhat independently the types of responses are related, respectively, to the performance and stability of the vehicle and to the ability of the pilot to control its motion.

The most useful instrument to support these directions with high efficiency the motion control has been parallel computing. Massively parallel computers are routinely used to simulate large-scale engineering problems, including fluid flows [2].

The Message Passing Interface (MPI) has permitted communication among heterogeneous computers connected through a fast communication network using a grid or local interconnects. Advances in hardware such as multicore processors have further increased the speed of each compute node in the network. This has permitted multiplicative increases in the computing speed without the need to make similar gains in the individual chip speed.

Another paradigm in scientific computing that is just beginning to emerge is the use of multi-threaded Graphics Processing Units (GPUs), which act as co-processors for Central Processing Units (CPUs) (Fig. 1). In recent years, motivated by the need for fast graphics and games, Graphics Processing Units have become quite powerful, while also becoming significantly cheaper than CPUs of equivalent computing power. GPUs with the capability to conduct several dozen PFLOPS (1 PFLOPS = 1015 computing operations per second) have been developed in the last few years.

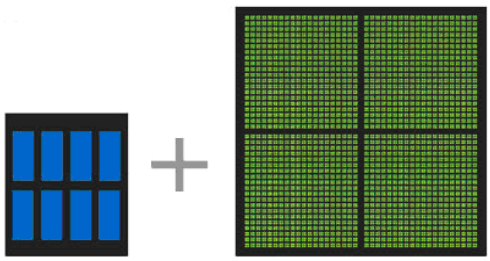


Fig. 1. Paradigm to use of multi-threaded GPU

Simultaneously a new programming language called CUDA (Compute Unified Device Architecture) created by NVIDIA has appeared. CUDA is a programming model based on the C programming language, making implementation of numerical algorithms on GPUs easier.

Brandvik and Pullan presented results for 2D and 3D Euler solvers implemented on the GPU. They used the Euler solvers to simulate turbine flows [3].

Elsen used a GPU to simulate the inviscid flow in simple and complex geometries by numerically solving the compressible Euler equations. Compared to the CPU, they achieved GPU speed-ups of over 40 times for simple geometries and 20 times for complex geometries [4].

So, the goal of the article is to analyze and find the way for improve the efficiency of the aircraft motion based on CUDA-technology.

©National Aviation University, 2016

II. WAVELET ANALYSIS FOR PARALLEL COMPUTING

Modern perspective method to detect where the turbulence will take place, separate the large and small scale details, as a result, reducing the drag component for improving the motion control of aircraft based on CUDA-technology is wavelet analysis [5].

The main equations for turbulent flow are given by:

$$\frac{\partial u}{\partial t}+(u\cdot\nabla)u=v\cdot\nabla^2 u-\frac{1}{\rho}\nabla\rho+f, \qquad (1)$$

where u, ρ, p is the velocity, density and pressure of the flow, ν is the viscosity, f is the resultant force of gravity and the external forces [5].

The turbulent velocity field $\hat{u}=(\hat{u},\hat{v},\hat{w})$ and the energy spectrum $\hat{e}=(1/s,k)$ at a scale s had been considered by the discrete wavelet transform:

$$\hat{u}_i=\frac{1}{\sqrt{s}\varphi_{\text{sum}}}\sum_j u_j\varphi\left(\frac{x_i-x_j}{s},\frac{y_i-y_j}{s},\frac{z_i-z_j}{s}\right), \qquad (2)$$

where φ is the mother wavelet, φ_{sum} is determined by

$$\varphi_{sum}=\sum_j\varphi \qquad (3)$$

The number of around cell is constant for use grid or pixel and the turbulent force from the energy $\hat{e}(k)$ and the wavelet function $y(x)$ is

$$f_i=W(\rho_i/\Delta t)\hat{e}_i(k)y(x_i), \qquad (4)$$

with the control parameter W to change the scale of turbulence.

III. CUDA ARCHITECTURE

Until recently, video accelerators were considered only as specialized devices designed only for processing graphics. In the early 21th century there was a use of graphic technology processor for general computing (GPGPU). This was made possible by the addition of programmable shader units and higher arithmetic precision raster containers that can be used stream processors to non visual computing.

Although this technology has a limited range of applications, GPU is the device which stepped ahead of the CPU in terms of increasing the total processing power, and the total number of kernels.

In fact, the main task of the GPU is reduced to mathematical calculations based on simple algorithms for receiving predictable input data, and close by ideology RISC CPU architecture.

Graphic technology processor for general computing technology is implemented by OpenCL and CUDA. OpenCL is the common standard programming interface of three-dimensional graphics. The basic principle of OpenGL is to provide a set of vector graphics primitives in the form of points, lines and polygons, followed by mathematical processing of the data and the construction of a raster image on the screen and/or in the memory. Vector transformation and rasterization executed by graphic pipeline (Fig. 2).

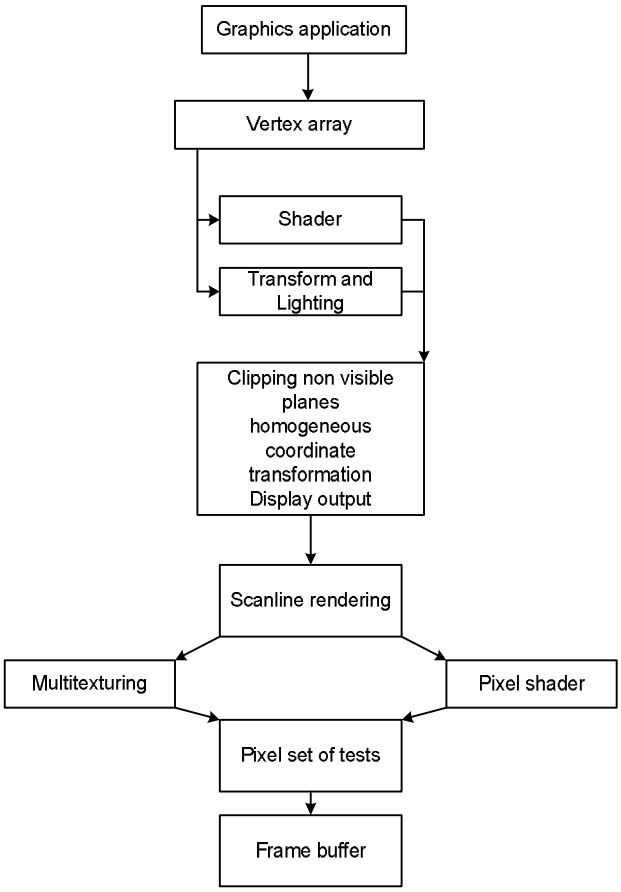


Fig. 2. Scheme of the graphic pipeline

CUDA is the architecture of parallel computing on the GPU company nVidia, supporting GPGPU technology for general-purpose computing (GeForce, Quadro, Tesla, Tegra, Fermi) [6]. Currently, there CUDA SDK [6], [7], which allows programmers to realize in the C programming language, C ++, Fortran and other algorithms feasible on the nVidia GPUs, and include special features in the text of the program in C. CUDA gives for developers the discretion to organize access to a set of instructions the graphics accelerator and control its memory, to organize on it difficult parallel computing.

The CUDA API is based on the C language [6], [7]. The graphics processor organizes hardware multithreading, which allows use of all GPU resources.

CUDA computing architecture also can be classified as the SIMT (Single Instruction Multiple Thread), where the input data divide into multiple

smaller sub-tasks each of which is handled by its thread. The threads are executed in parallel on the compute modules, which are used as a set of streaming multi-processors.

Multiprocessor is a SIMT-multicore processor, allowing in any given time to carry out all the nuclei of only one instructions, and each stream processor executes the instructions on his own data. To execute on the GPU task is divided programmer on multiple **threads**, which are combined into **blocks**, and the blocks in turn are combined into a **mesh.** The procedure performed by each of the streams is called the kernel (kernel). Computing resources are distributed between threads by the CUDA driver. Logically, the device can be represented as a set of multiprocessors with CUDA driver. Schematically CUDA performance of the program shown in Fig. 3.

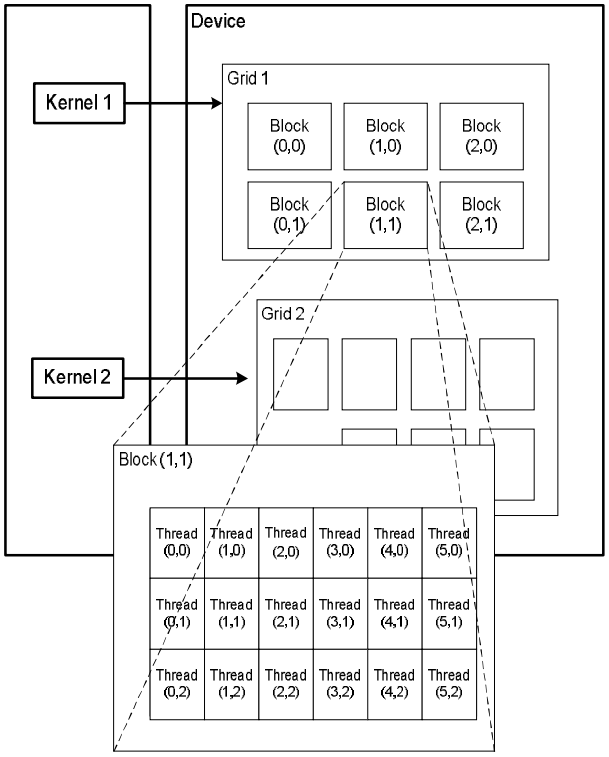


Fig. 3. CUDA architecture

The host is considered as part of the performance program based on the CPU, and device is the video card, or computing device that supports CUDA.

The flows and blocks are identified by codes, which determines the data part that performed the particular thread.

The maximum codes that can be used for the block flows are (512, 512, 64) the number of threads in the block should not exceed the maximum dimensions 512. The maximum grid dimensions are (65535, 65535, 1), is actually two-dimensional network of streams.

CUDA-performance unit is the warp. Warp size is 32 flow, due to the delay the 4 cycles (latency) of the performance one instruction on multiprocessor. Only in relation to warp can be talked about performance of the parallel threads, no other assumptions can not be made. However, this does not mean that warps performed on multiprocessor in series. The warps performance can be parallel, for example, in the case where a warp is waiting for data from the global memory other warps can be performed at this time.

The interaction between threads can be within block. Data exchange is carried out through a shared (shared) memory common to all threads in the block. Timing of threads can be done by calling the special synchronization functions.

Another key point architecture CUDA is the easy scalability. Once the code is written will run on all devices supporting CUDA. For development and debugging code to run on the GPU can use ordinary video cards that cost accessible to everyone. And when the product is ready – it has run on powerful Tesla or clusters built on the GPU architecture, the cost of which is lower than the cost of well known computing systems.

The most important issues are the investigation of acceleration depending on the total computational complexity and estimation the resulting acceleration in the organization of parallel computing [7]. These figures reflect the feasibility of using parallel computing. It can be considered that the sequential program runs on a single core of multi-core the CPU without special parallel computing tools between the kernels.

For systems the same type with technical parameters, the acceleration value of the parallel implementation relatively to series is usually defined as follows:

$$a = \frac{t_1}{t_k}, \tag{5}$$

where t_1 is the time performance on a single processor, t_k is the time performance of a parallel program on k computers.

The use of CUDA architecture for parallel computing can be considered the estimation of acceleration as relation the operating time of the parallel algorithm to the computing power (performance) of a particular model the GPU device.

The integral effectiveness of parallel computing defined as the relation the acceleration to the number of cores (CUDA cores):

$$E = \frac{a}{k}, \tag{6}$$

where k is the number of the GPU device cores.

The computational efficiency virtually independent of the dimension of the task; an increase the performance parallel computers is slightly reduced the integral efficiency.

Table I illustrates the main information for computing CPU and GPGPU CUDA devices [6], [7].

Analysis of this data shows that it is more advantageous to use the NVIDIA graphics processors to increase the processing power.

IV. Results

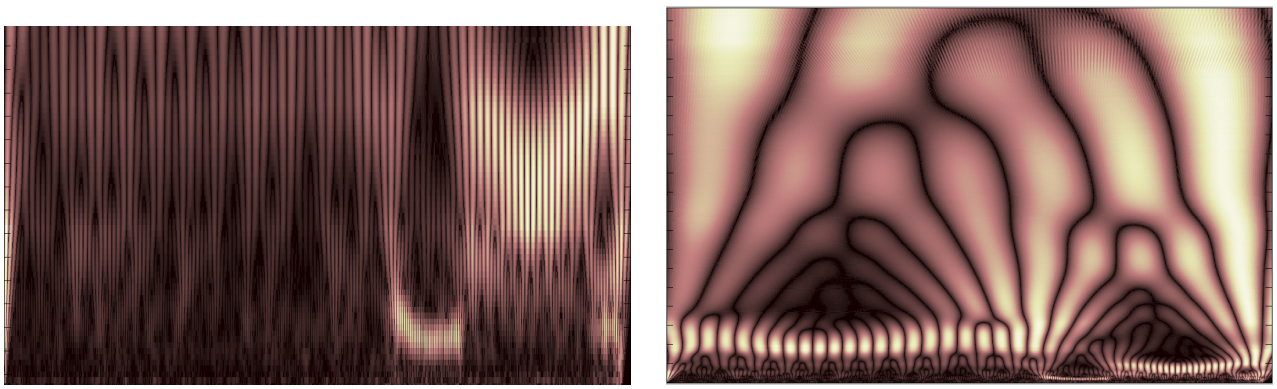
The efficiency of the aircraft motion based on CUDA-technology can be characterized by wavelet analysis turbulence modeling. The results were made on a computer equipped with the Intel Core i7 CPU and the NVIDIA Tegra GPU. The most of algorithm was implemented on GPU by using NVIDIA CUDA and used the Morlet as the mother wavelet.

Figure 4 show the result of turbulence flow. The computation time for the simulation was about 10 msec per frame. The calculation of the energy spectrum and the wavelet turbulence requires about 75% of the simulation.

TABLE I

MAIN CHARACTERISTICS OF COMPUTING DEVICES

Model	**Performance, GFLOPS**	**Energy consumption, W**	**Cost-effectiveness**
Intel Core 2 Duo	23.46	55	0.0073
Intel Core i7	99.20	95	0.0164
Tegra	1500	225	0.0247
Tesla	3520	225	0.0282



(a) (b)

Fig. 4. Flow without turbulence (a), and with turbulence (b)

V. Conclusion

CUDA-based method to simulate turbulent flow by wavelet analysis for improving the efficiency of the aircraft motion has been proposed. It makes better to reduce the drag component and optimize constituting of flight dynamics – performance, stability and control, navigation and guidance.

CUDA-architecture allows to use NVIDIA Tegra GPU embodiment in a real-time as a controlling part of adaptive control systems with high-resolution predictions. Moreover, the development of parallel computing with use the wavelet analysis can perform calculations on grids of large number of elements and small computation time.

In fact, this technology has become usefull for non-stationary processes analysis and extends the direction of investigation the object with the hybrid-morphology properties to act as a distributed sensitive system, which can reduce the power consumption and competiveness of the aircraft.

References

[1] David A. Caughey, "Introduction to Aircraft Stability and Control." *Sibley School of Mechanical & Aerospace Engineering Cornell University*. New York. 2011, 147 p.

[2] Nairita Pal, Prasad Perlekar, Anupam Gupta, and Rahul Pandit "Binary-Fluid Turbulence: Signatures of Multifractal Droplet Dynamics and Dissipation Reduction" *Indian Institute of Science*. Bangalore, India. 2016, pp. 1–11.

[3] T. Brandvik, and G. Pullan, "Acceleration of a 3D Euler solver using commodity graphics hardware," *46th AIAA Aerospace Sciences Meeting and Exhibit*, 2008.

[4] E. Elsen, P. LeGresley, and E. Darve, "Large calculation of the flow over a hypersonic vehicle using a GPU," *Journal of Computational Physics*, vol. 227, no. 24, 2008, pp. 10148–10161.

[5] M. Farge, N.K.R. Kevlahan, V. Perrier, “Turbulence analysis, modelling and computing using wavelets” *Laboratoire de Meteorologie Dynamique*, Paris Cedex 5, 2009, pp. 1–66.

[6] Parallel computing CUDA. [Online]. Available: http://www.nvidia.com.ua/object/cuda-parallel-computing-ru.html (in Russian).

[7] Andrew Kerr, Gregory Diamos, Sudhakar Yalamanchili, “A Characterization and Analysis of PTX Kernels.” *Georgia Institute of Technology*, Atlanta, Georgia, 2013, pp. 1–10.

Received October 19, 2016.

Tovkach Serhii. Candidate of Science (Engineering). Associate Professor.
Automation and Energy Management Department, National Aviation University, Kyiv, Ukraine.
Research interests: Automatic control systems and diagnostics systems of aircraft using wavelet analysis.
Publications: 40.
E-mail: ss.tovkach@gmail.com

С. С. Товкач. Технологія CUDA для підвищення ефективності руху повітряного судна

Розглянуто метод паралельних обчислень на основі CUDA-архітектури з визначенням великих і малих деталей турбулентного потоку для адаптації динаміки польоту під час керування рухом повітряного судна. Визначено значення прискорення паралельної реалізації відносно послідовної та інтегральну ефективність паралельних обчислень, що дозволяє використовувати графічні процесори NVIDIA Tegra для збільшення обчислювальної потужності масивно-паралельних розрахунків.

Ключові слова: графічний процесор; багатопоточність; динаміка польоту; адаптивна система керування рухом; вейвлет-аналіз; турбулентність потоку, паралельні обчислення.

Товкач Сергій Сергійович. Кандидат технічних наук. Доцент.
Кафедра автоматизації та енергоменеджменту, Національний авіаційний університет, Київ, Україна.
Направлення наукової діяльності: автоматизовані системи керування та діагностування систем повітряного судна з використанням вейвлет-аналізу.
Кількість публікацій: 40.
E-mail: ss.tovkach@gmail.com

С. С. Товкач. Технология CUDA для повышения ефективности движения воздушного судна

Рассмотрен метод параллельных вычислений, основанный на CUDA-архитектуре, с обнаружением больших и мелких деталей турбулентного потока для адаптации динамики полета при управлении движением самолета. Определено значение ускорения параллельной реализации относительно последовательной и интегральная эффективность параллельных вычислений, что позволяет использовать графические процессоры NVIDIA Tegra для увеличения вычислительной мощности массивно-параллельных расчетов.

Ключевые слова: графический процессор; многопоточность; динамика полета; адаптивное управление движением; вейвлет-анализ; поток турбулентности, параллельные вычисления.

Товкач Сергей Сергеевич. Кандидат технических наук. Доцент.
Кафедра автоматизации и энергоменеджмента, Национальный авиацыонный университет, Киев, Украина.
Направление научной деятельности: автоматизированные системы управления и диагностирования систем воздушного судна с использованием вейвлет-анализа.
Количество публикаций: 40.
E-mail: ss.tovkach@gmail.com