Coalesced Simulation of the Incompressible Navier-Stokes Equations over an Airfoil Using Graphics Processing Units

University of Tehran

S. M. Iman Gohari, Vahid Esfahanian, Hamed Moqtaderi, Hossein Mahmmodi Darian
Department of Mechanical Engineering, College of Engineering, University of Tehran, Iran.

مرکز تحقیقات خودرو، سوخت و محیط زیست
VFE Research Institute
Vehicle, Fuel & Environment Research Institute

Abstract

This work presents a Graphics Processing Unit (GPU) based implementation of the Finite Differencing Time Domain (FDTD) methods, for solving unsteady incompressible viscous flow over an airfoil using the Stream function-Vorticity formulation for the structured grid. For the large-scale simulations, FDTD methods can be computationally expensive and require considerable amount of time to solve on traditional CPUs. On the contrary, modern GPGPUs such GTX 480 are designed to accelerate lots of independent calculations due to advantage of their highly parallel architecture. In present work, the main purpose is to show a configuration for leveraging GPU processing power for the computationally expensive simulations based on explicit FDTD method and CUDA language. Our implemented calculation for the GPU FDTD simulation has efficient global memory coalescence with 66.67% of occupancy. Both GPU based versions of flow simulation and grid generation are over 28 and 150 times faster than a sequential CPU based versions, respectively. In Addition to time comparison, GPU performance analyzing is done through the NVIDIA Visual Profiler Software and it is demonstrated that memory throughput of present GPU calculation is about 60% of its theoretical value.

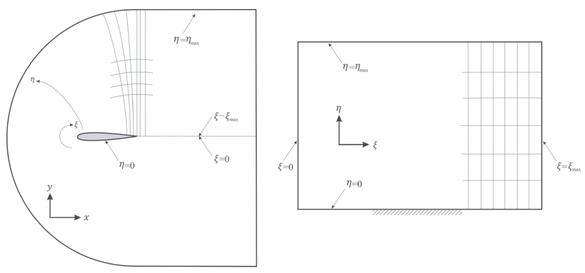
Numerical Implementation

The incompressible Navier-Stocks equations are solved numerically with the Stream Function and Vorticity formulation for the structured grid. It is clear that Every CFD simulation needs appropriate computational grid. Therefore, grid generation is one of most important part of CFD simulations. In addition, some CFD simulations need grid adaptation and refinement. This procedure needs grid regeneration and sometimes takes more time than the flow simulation itself. In the present simulation both flow simulation and grid generation is done through the GPU. The Stream function-Vorticity formulation needs the orthogonal grid points with the desirable grid spacing to capture the high velocity gradient in the boundary layer. Steger shown the iterative procedure for constructing the computational grid which is used in the present work. The numerical simulation of the unsteady incompressible Navier-Stokes equations for the laminar and turbulent flow about arbitrarily shaped two-dimensional airfoils is also considered. This solution is based on the technique of numerical generation of a curvilinear coordinate system which has coordinate lines coincident with the airfoil contour regardless of its shape. The explicit simulation utilizes the Vorticity-Stream function formulation with the direct satisfaction of the no-slip condition on the airfoil surface.

Grid Generation:

$$\alpha x_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma x_{\eta\eta} = -J^2(Px_{\xi} + Qx_{\eta}),$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = -J^2(Py_{\xi} + Qy_{\eta}).$$



Stream Function-Vorticity Formulation:

$$\nabla^2\psi = -\omega, \quad \omega_t + \frac{\partial(u\omega)}{\partial x} + \frac{\partial(v\omega)}{\partial y} = \nabla^2(\nu\omega).$$

$$C_p^*(\xi) = P(\xi) - P_{T.E.}(\xi) = \int_{\xi_{T.E.}}^{\xi} \frac{(\beta(\nu\omega)_{\xi} - \gamma(\nu\omega)_{\eta})}{J} d\xi'$$

GPU Based Solvers have been developed upon the following features:

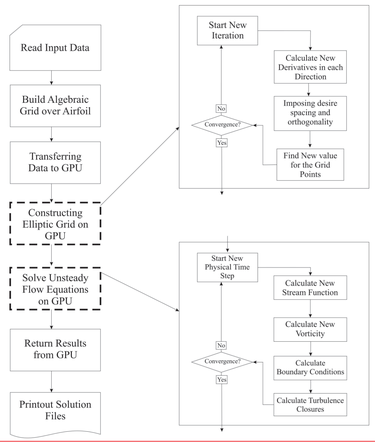
1) Finite difference approximations.
2) Unsteady Flow.
3) Algebraic Turbulence Model, Baldwin Lomax.
4) First or second order Upwind scheme for the convection terms.
5) Second order central scheme for the diffusion terms.
6) Over relaxed Jacobi algorithm.

GPU Implementation

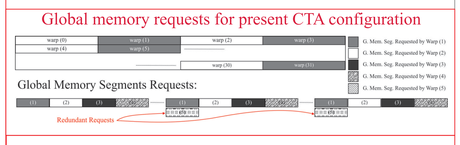
Cooperative Thread Array (CTA) is an array of simultaneous threads that cooperate and execute to computea desire result. In the GPU programming model, the thread block is the CTA. CTA conflguration i.e. size, shape, dimension are declared by programmer and has major effect on GPU performance and runtime. In the presence of significantly larger number of cores on the GPU in the contrast to CPU, in order to fully saturate the available computational resources, 1D CTA conflguration is considered. Present strategy of a 1D CTA shows higher speedup and performance than the regular 2D one. present work with the following consideration has higher performance:

Higher Memory Access Coalescing: Transferring data segments from global memory of graphics card is carried out by the global memory requests. Threads within the same warp have a coalesced memory requests for loading specific memory region when they can load the data segments by fewer requests. To have highly coalesced memory access pattern, it is required to have as fewer as possible redundancies in memory requests generated by each thread to eliminate the expense of associative search hardware and alleviate the schedule of each SM. The present GPU method has great reduction in global memory requests as illustrated in below figures.

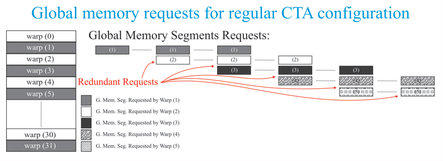
Flowchart of the GPU-enabled solver



Global memory requests for present CTA configuration



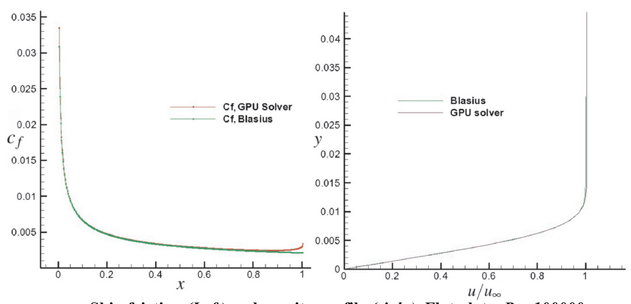
Global memory requests for regular CTA configuration



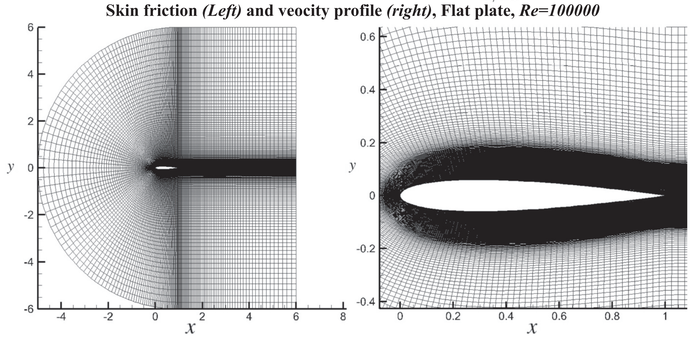
Results

Computational Implementation:

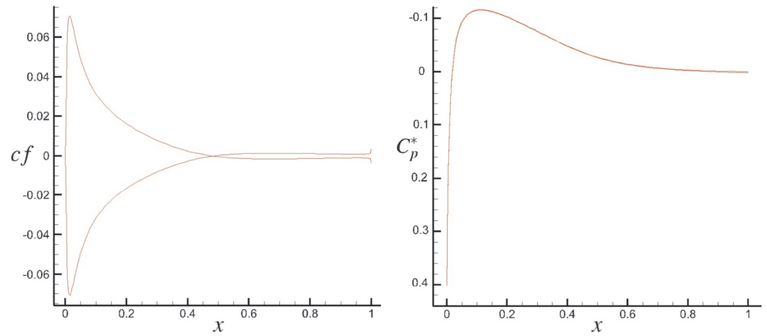
Simulation validated by Blasius Similarity solution of Laminar Flow:



Skin friction *(Left)* and veocity profile *(right)*, Flat plate, *Re=100000*



GPU structure grid over NACA 0012: over view *(left)*, close view *(right)*



Skin Friction *(Left)* and Pressure Coefficient *(right)*, NACA 0012, *Re=5000 AoA=0.0*

Comparison between GPU and CPU Time:

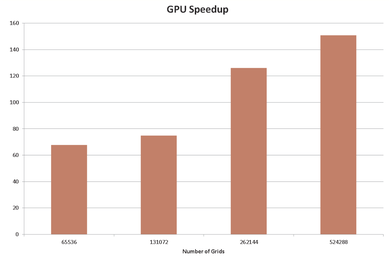
Grid Generation

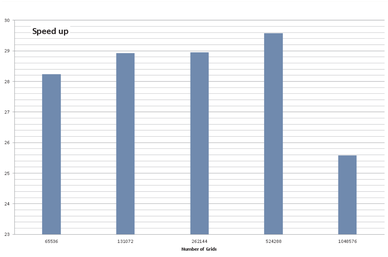
Grid Size	CPU time	GPU Time	Speedup
128*256	113.6	7720.25	67.95
256*256	200.00	13561.25	67.80
512*256	370.96	27757.42	74.83
512*512	724.76	91364.54	126.06
512*1024	1413.82	213113.21	150.73

Flow Solver

Grid Size	CPU time	GPU Time	Speedup
256*256	221.15	6241.85	28.24
512*256	434.07	12565.19	28.93
512*512	863.11	24976.37	28.95
1024*512	2297.50	48788.64	29.57
1024*1024	3267.00	83567.66	25.58

GPU Speedup





Here the **GPU Performance** is evaluated in the terms of occupancy and Global Memory throughput. By burning in mind that the total memory bandwidth of GTX 480 is 177.4 GB/s.

Performance Analyzing				
Memory throughput (Grid Generation)	Memory throughput (Flow Solver)	G. M. Performance (Grid Generation)	G. M. Performance (Flow Solver)	Occupancy
98.79 GB/s	105.21 GB/s	55.68%	59.30%	66.67%

Conclusions

We have investigated two dimensional unsteady flow solver for the incompressible Navier-Stokes equations based on Stream function-Vorticity formulation for graphics processing units. Both grid construction and flow simulation have been taken placed through GPU kernels with the Finite Differencing Time Domain (FDTD) method. We have shown that the one dimensional configuration of GPU CTAs can reliably provide higher coalesced memory access pattern. In addition, one dimensional CTA configuration has fewer redundancies in global memory requests in contrast to two dimensional one. Consequently, the present method has lower limit on GPU memory bandwidth. Both GPU based versions of flow simulation and grid generation are over 28 and 150 times faster than a sequential CPU based versions, respectively. In Addition to time comparison, GPU performance analyzing was done through the NVIDIA Visual Profiler Software and it was demonstrated that memory throughput of present GPU calculation was about 60% of its theoretical value.