The Simulation of Transient Process of Gas Discharge by Multi-GPU with CUDA Platform

Wu-Shung Fu, Wei-Hsiang Wang, Kun-Rong Huang

Department of Mechanical Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan, ROC

Abstract

For studying the phenomenon of fluid dynamics, Computational Fluid Dynamics (CFD) has been developed for twenty years. CFD bases on different numerical scheme to solve the no-linear partial differential equations. In this study, the finite volume and structured grid are adopted. Methods of the Roe scheme, preconditioning and dual time stepping matching the DPLR are simultaneously applied to solve the Navier-Stokes equations. The non-reflection boundary condition method is used to handle the open boundary situation to prevent the reflection waves from the boundary and the immersed boundary method is applied to treat the reaction force from the inserted complex moving geometry in the fluid. In order to reduce the computational time, the parallel computation device of CPU and GPU are used and compared in this study.

Governing equations

The governing equation of continuity, momentum, energy and ideal gas equations are listed as follows:

$$\frac{\partial U}{\partial t}+\frac{\partial F_1}{\partial x_1}+\frac{\partial F_2}{\partial x_2}+\frac{\partial F_3}{\partial x_3}=S$$

The source term S denotes the reaction force on the immersed boundary Ω.

$$F_j=\begin{pmatrix}\rho u_j\\ \rho u_1u_j+P\delta_{1j}-\tau_{j1}\\ \rho u_2u_j+P\delta_{2j}-\tau_{j2}\\ \rho u_3u_j+P\delta_{3j}-\tau_{j3}\\ (\rho e+P)u_j+q_j-u_i\tau_{ij}\end{pmatrix}\quad i=1,2,3$$

where

$P=\rho RT$ and $S=F\cdot\delta(X-\Omega)$ $\quad U=(\rho \quad \rho u_1 \quad \rho u_2 \quad \rho u_3 \quad \rho e)^T$

Immersed boundary method

In order to set the solid region in the fluid domain and retain the mesh in structure grid, the immersed boundary proposed by Peskin and refined by Mittal et al. are used. The mass conservation error of the structured and unstructured are 7.79x10^{-3}% (better) and -0.44% and the difference of the mesh are shown in Figure 1.

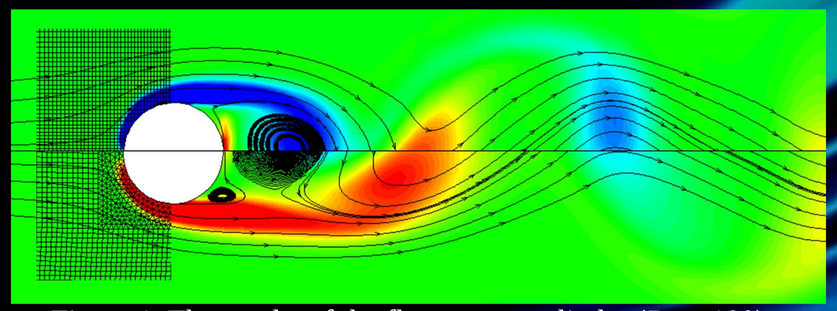

Figure 1. The results of the flow pass a cylinder (Re = 100), structured (up) and unstructured grid (down).

Comparison of OpenMP and CUDA

The OpenMP and CUDA are used to boost the computation and test in our previous study. The model of this study is three dimensional and the data of each finite volume are saved in a 3D array. The data in the 3D array can be read and moved easily by CPU in host but the procedure is much more complicated in the device by GPU. In order to simplify the process, the 3D array is transformed to 1D array and copied to device memory. The same project is handled by OpenMP and CUDA and the results are shown in figure 2. It shows that the speed up of the GPU is obviously higher than CPU with OpenMP method even the array needs to be converted to 1D and the coding needs more parallel optimization.

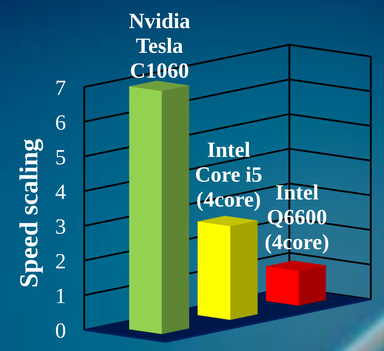

Figure 2. Comparison of computing efficiency

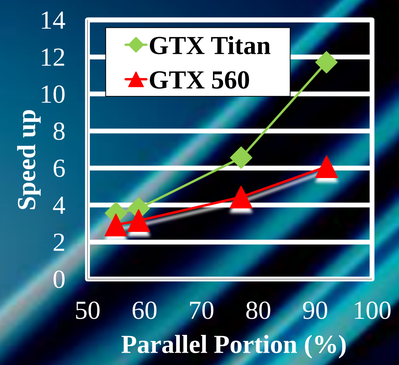

Figure 3. Comparison of speed up versus parallel percentage with different device

Optimization and Multi-GPU

The GPU loading of the original program is about 55%, it means that the project can be modified with more parallel optimization. The key point of increasing the speed is to reduce the data transfer between the host and device, and always keep the calculation loop in the device. Figure 3 shows the parallel percentage and the speed up results (3.75x10^4 mesh numbers). The program is improved from 54% parallel percentage to 92% and the comparison based on the OpenMP method with the Intel Core i7 X980. In former, only the solving scheme loop are handled by GPU (54%) while the latter put the boundary conditions loop into device(59%), and reduce the data transfer between the host and device (72%), and finally applying the Reduction method to parallelize the residual loop and reduce the frequency of data output (92%).

In order to increase the calculation speed and device memory, the multi-GPU technology are applied. The model is separated into several parts. Depending on the number of grids in host and the additional interfaces need to be defined and is written and read by another grid. Although using multi-GPU can improve the computational speed, the treatment of the interface will reduce the calculation speed due to data transform between different devices. Therefore, reducing the area of the interface is the better way to keep high efficiency. The results of multi-GPU are shown in figure 4. It reveals that the multi-GPU model exactly increase the speed of calculation than a single GPU .

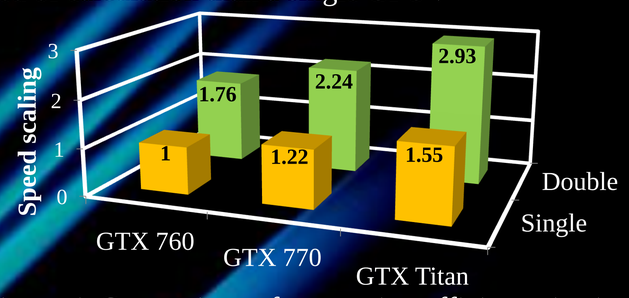

Figure 4. Comparison of computing efficiency in single and double GPU (2.2x10^6 mesh numbers)

Results

Utilizing the CUDA platform to reduce the computational time is efficient and economic, the calculation with OpenMP (i7 X980) method requires approximately a month to finish, while CUDA (GTX Titan) costs 2.5 days to make the same computation and the results are shown in figure 5. The most powerful feature of multi-GPU by CUDA platform is that the compute capability of super computing of individualization could be realize.

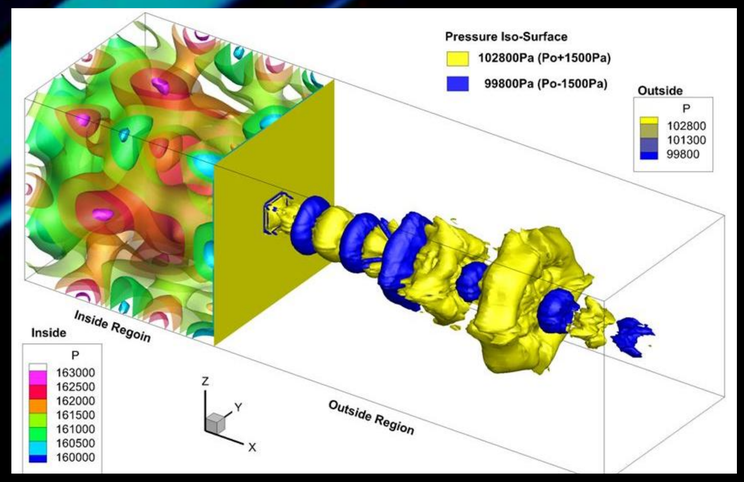

Figure 5. The results of the gas discharge into outside region (1.62x10^6 mesh numbers)