A CFD Heterogeneous Parallel Solver Based on Collaborating CPU and GPU

Jianqi Lai\textsuperscript{1,}\textsuperscript{*}, Zhengyu Tian\textsuperscript{1}, Hua Li\textsuperscript{1} and Sha Pan\textsuperscript{1}

\textsuperscript{1}College of Aerospace Science and Engineering, National University of Defense Technology, Deya Road, Kaifu District, Changsha, Hunan, China

*Jianqi.kd@nudt.edu.cn

Abstract. Since Graphic Processing Unit (GPU) has a strong ability of floating-point computation and memory bandwidth for data parallelism, it has been widely used in the areas of common computing such as molecular dynamics (MD), computational fluid dynamics (CFD) and so on. The emergence of compute unified device architecture (CUDA), which reduces the complexity of compiling program, brings the great opportunities to CFD. There are three different modes for parallel solution of NS equations: parallel solver based on CPU, parallel solver based on GPU and heterogeneous parallel solver based on collaborating CPU and GPU. As we can see, GPUs are relatively rich in compute capacity but poor in memory capacity and the CPUs do the opposite. We need to make full use of the GPUs and CPUs, so a CFD heterogeneous parallel solver based on collaborating CPU and GPU has been established. Three cases are presented to analyze the solver’s computational accuracy and heterogeneous parallel efficiency. The numerical results agree well with experiment results, which demonstrate that the heterogeneous parallel solver has high computational precision. The speedup on a single GPU is more than 40 for laminar flow, it decreases for turbulent flow, but it still can reach more than 20. What’s more, the speedup increases as the grid size becomes larger.

1. Introduction

With the rapid development of computer technology, computational fluid dynamics (CFD) plays an important role in aerodynamic performance analysis, optimal design and complex flow mechanism study. At the same time, the requirements of engineering calculation in CFD are also increasing. As for complete aircraft configuration, the large scale simulations on grids consisting of dozens of millions of elements, the huge amount of calculation could only be completed by parallel computing in computer cluster or supercomputer. Graphic Processing Unit (GPU) has a strong ability of floating-point computation and memory bandwidth for data parallelism. It has been widely used in the areas of common computing [1, 2] such as molecular dynamics (MD), computational fluid dynamics (CFD) and so on. The emergence of compute unified device architecture (CUDA) brings great opportunities to CFD, which could promote the development of CFD.

Traditionally, GPU was limited to the area of graphics rendering. Compute unified device architecture (CUDA) is the software and hardware architecture for GPU produced by NVIDIA in 2007, which reduce the complexity of compiling program [3, 4]. The emergence of CUDA as a programming model marked the beginning of widespread extension of GPUs to scientific computing. In the field of CFD, a series of important results have been achieved in the implementation of large-
scale parallel computation on GPU cluster [5-7]. Some scholars started to solve Euler equations and Navier-Stokes equations on GPU cluster, further research on Reynolds-averaged equations (RANS), Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS) were also widely carried out, and the computational efficiency had been greatly improved.

In this paper, a heterogeneous parallel solver based on collaborating CPU and GPU is established in parallel cluster. The reminder of this paper is organized as follows. Section 2 introduces the governing equations and numerical schemes. Section 3 discusses the GPU hardware and the detailed GPU implementation. Section 4 presents several numerical results to verify the solver’s computational precision in laminar and turbulent flow. Section 5 gives the performance analysis of the solver. Finally in section 6, we introduce some final conclusions and directions for future work.

2. Governing equations

Without regard to volume sources due to body forces and volumetric heating, Navier-Stokes equations for compressible flow can be written as follows [8]:

\[
\frac{\partial}{\partial t} \int_{\Omega} \rho d\Omega + \oint_{\partial \Omega} \left( \mathbf{F}_c - \mathbf{F}_v \right) d\mathbf{S} = 0
\]  

(1)

Where \( \mathbf{W} \) is the vector of conservative variables, \( \mathbf{F}_c \) is the vector of convective fluxes, \( \mathbf{F}_v \) is the vector of viscous fluxes, and \( \Omega \) is a control volume.

The spatial discretisation of these equations on structured grids is using a cell-centered finite volume scheme as shown in figure 1. Compared to the unstructured grids, structured grids can easily achieve better performance due to its body fitted property.

![Figure 1. Control volume of a cell-centered scheme.](image)

For a control volume \( \Omega_{i,j,k} \), the discrete form of governing equations separately for temporal and spatial can be written as:

\[
\Omega_{i,j,k} \frac{d\mathbf{W}}{dt} + \sum_{m=1}^{8} (\mathbf{F}_c - \mathbf{F}_v) \Delta S_m = 0
\]

(2)

The convective fluxes are computed by AUSM+UP scheme, and the viscous fluxes are computed by central scheme, the Van Leer’s limiter is used to prevent the generation of oscillations and spurious solutions in regions of large gradient. Here, explicit multi-stage Runge-Kutta scheme for the temporal discretization of governing equations are discussed. What’s more, the \( k-\omega \) SST two-equation model is used to solve the turbulence source term.

3. Heterogeneous parallel solver collaborating CPU and GPU
3.1. GPU hardware
The GPU used in this paper is NVIDIA GTX 1070 which has 8GB device memory. GTX 1070 contains 15 streaming multiprocessors (SMs) and each SM has 192 CUDA cores. The GTX 1070’s compute capability is 6.1, which could meet the demand of large-scale scientific computation. The large memory size and strong compute capability makes GTX 1070 attractive for the solution of incompressible or compressible Navier-Stokes equations. Table 1 shows the main performance parameters of GPU.

| GPU Parameters       | GTX 1070 |
|----------------------|----------|
| Device Memory        | 8GB      |
| Compute Capability   | 6.1      |
| Single-Precision Float| 6080 Gflop/s |
| Double-Precision Float| 194 Gflop/s |
| Bandwidth            | 256.3GB/s|

3.2. Parallel Programming Model of CUDA
The entire CUDA program contains both host (CPU) and device (GPU) as shown in figure 2, the part of host program executes on the CPU, while the other which calls kernel executes on the GPU. A kernel is a function running on the GPU, corresponds to a thread grid. A thread grid is executed by threads, which are grouped into batches of up to 1024 called thread block. Threads within the same thread block are guaranteed to run on the same SM at the same time [9, 10]. In CUDA, a thread grid can be divided into a number of blocks, while the number of blocks and the number of threads per block will have a great influence on the computational performance of GPU, we should do some tests to determine appropriate number of blocks in a thread grid and threads per block.

3.3. Heterogeneous parallel solver
Graphic Processing Unit (GPU) has a strong ability of floating-point computation and memory bandwidth for data parallelism. As we can see, GPU is relatively rich in compute capacity but poor in memory capacity and the CPU do the opposite. So the CPU is suitable for dealing with logical affairs such as selection, judgement and so on, while the GPU is good at handling with single instructions and other highly linearized tasks. In general, there are three different modes for parallel solution of NS equations: parallel solver based on CPU, parallel solver based on GPU and heterogeneous parallel solver based on collaborating CPU and GPU [11, 12].

The basic idea of multi-CPU parallel mode is shown in figure 3. It is easy to be understood. Firstly, the computational domain should be divided into several subregions of which the grid sizes are equal,
and then each CPU core load the computation of a subregion. However, the conventional parallel mode is hard to meet the requirements of large-scale computing in industry. Sometimes the task can’t be implemented because of the extensive calculation. As for GPU-only parallel mode, we can’t make the best of its strong computation ability because of the weak memory capacity of GPU. In order to make full use of the GPUs and CPUs, a CFD heterogeneous parallel solver based on collaborating CPU and GPU has been established.

![Figure 3. Multi-CPU parallel mode.](image)

Parallel mode based on collaborating CPU and GPU as shown in figure 4. According to the characteristics of NS equations in CFD, the computational task could be divided into branch instruction, single instruction and data transfer. The branch instruction task such as the treatment of boundary conditions and data transfer are implemented by CPU, while the single instruction task such as the calculation of fluxes in the inner face are implemented by GPU. It is a good choice for researchers to combine the characteristics of computing task with device.

![Figure 4. Parallel mode based on collaborating CPU and GPU.](image)

In this paper, we use CUDA of version 7.5, Intel icc11.1 for C code. A single node platform contains a GPU of NVIDIA GTX 1070 and a CPU of Intel Xeon 2670 with eight cores. It’s important to improve single node performance before realizing multi-node version. A node contains several GPUs or multi-node version will be discussed in future work.

4. Numerical results

In this section, three cases are presented to test the computational accuracy of heterogeneous parallel solver [13, 14]. For these three cases, the pressure distribution of symmetry plane is chosen to analyse the solver’s accuracy.

4.1. Double Ellipsoid

This case demonstrates the solver’s ability to simulate laminar, hypersonic flow over the double ellipsoid. A large number of wind tunnel experiments for the model were implemented with different Mach numbers and angle of attacks in Ref. 13, and a wind tunnel experiment result is used as a comparison.

Free flow conditions of this case are given in table 2.
Table 2. Free flow conditions for double ellipsoid.

| $Ma$ | $Re / m^{-1}$ | $P_0 / \text{KPa}$ | $T_0 / \text{K}$ | Angle of Attack(\text{deg}) |
|------|----------------|---------------------|------------------|----------------------------|
| 8.02 | 1.98×10$^7$    | 850                 | 720              | 0                          |

Pressure distribution of the upper and lower surfaces is shown in figure 5 and figure 6. The numerical results agree well with experiment results, which demonstrate that the heterogeneous parallel solver has high computational precision.

4.2. Aerospace plane

This case demonstrates the solver’s ability to simulate laminar, hypersonic flow over the aerospace plane which is significantly important to future space battle and transportation. The same as the first case, a wind tunnel experiment result from Ref. 13 is used as a comparison.

Free flow conditions of this case are given in table 3.

Table 3. Free flow conditions for aerospace plane.

| $Ma$ | $Re / m^{-1}$ | $P_0 / \text{KPa}$ | $T_0 / \text{K}$ | Angle of Attack(\text{deg}) |
|------|----------------|---------------------|------------------|----------------------------|
| 8.02 | 1.34×10$^7$    | 600                 | 740              | 0                          |

Pressure distribution of the upper and lower surfaces a shown in figure 7 and figure 8. The numerical results agree well with experiment results, showing the heterogeneous parallel solver has strong ability to deal with complex geometries.

4.3. Supersonic inlet

This case demonstrates the solver’s ability to simulate turbulent, supersonic flow over the supersonic inlet. The supersonic inlet is of great importance in the airbreathing propulsion system. The flow field...
of supersonic inlet is very complex that contains boundary layer, expansion wave, contour induced shocks, reflected and boundary layer induced shocks. Free flow conditions of this case are given in table 4.

| Ma | Re / m⁻¹ | P₀ / KPa | T₀ / K | Angle of Attack(deg) |
|----|-----------|----------|--------|----------------------|
| 2.41 | 5.07×10⁷ | 540 | 305 | -10 |

In this case, computing mesh was generated by the software of Gridgen, and structured grid was utilized here. Computational domain and grid detail for supersonic inlet is shown in figure 9.

Figure 9. Computational domain and grid detail for supersonic inlet

Figure 10 and figure 11 shows the comparison of a schlieren picture from Ref. 14 with corresponding Mach number contour lines of the computation for supersonic inlet, and it reveals a good agreement.

Figure 10. A schlieren picture for supersonic inlet

Figure 11. Mach number contour lines of the computation for supersonic inlet

Pressure distribution of the upper and lower surfaces is shown in figure 12 and figure 13. A quantitative comparison between numerical and experiment results demonstrate that the solver performs well in solving turbulence problems.

Figure 12. Pressure distribution of upper surface.  Figure 13. Pressure distribution of lower surface.
5. Performance analysis
In order to apply the solver to large-scale parallel computation, we must study the heterogeneous parallel efficiency of the solver. Speedup is an important performance parameter to evaluate parallel efficiency. It is defined as the CPU operation time divided by the heterogeneous parallel operation time while implemented in the same conditions, here the CPU operation time is used as a reference to compute heterogeneous parallel efficiency. To collect the operation time, we simulate 10 time steps, with each having 100 sub_iterations, and the average time per iteration step is chosen as the operation time.

The speedup varies with gridsize for double ellipsoid and aerospace plane can be shown in figure 14 and figure 15, and figure 16 shows that the speedup varies with gridsize for supersonic inlet. The solver achieves a speedup of more than 40 in laminar flow, it decreases when handling with turbulence flow, but it still can reach more than 20. The results demonstrate that the heterogeneous parallel solver has high computational efficiency. What’s more, for the three cases, the speedup increases as the grid size becomes larger. This is due to the proportion of single instruction task increases compared with branch instruction task. As we all know, GPUs specialize in dealing with single instruction task, while the CPUs are good at dealing with branch instruction task. So the heterogeneous parallel solver takes full advantages of the characteristics of hardware, and it provides a new direction for scientific computing.

![Figure 14. Speedup for double ellipsoid](image1)
![Figure 15. Speedup for aerospace plane](image2)
![Figure 16. Speedup for hypersonic inlet.](image3)

6. Conclusion
Heterogeneous parallel solver based on collaborating CPU and GPU was established on parallel computing cluster in this paper. The numerical results agree well with experiment results, which demonstrate that the solver has high computational precision in laminar and turbulent flow. The
speedup on a single GPU is more than 40 for laminar flow, it decreases for turbulent flow, but it still can reach more than 20. For the three cases, the speedup increases as the grid size becomes larger because the proportion of single instruction task increases compares with branch instruction task. A multi-GPU heterogeneous parallel solver will be discussed in future work.

References
[1] Esfahanian V, Baghapour B, Torabzadeh M, and Chizari H 2014 An efficient GPU implementation of cyclic reduction solver for high-order compressible viscous flow simulations Computers & Fluids 92 160-71
[2] Liu X, Zhong Z and Xu K 2016 A hybrid solution method for CFD applications on GPU-accelerated hybrid HPC platforms Future Generation Computer Systems 56 759-65
[3] NVIDIA Corporation 2013 CUDA C Programming Guide
[4] Dana A J, Julien C T, Thibault and Inanc S 2010 An MPI-CUDA Implementation for Massively Parallel Incompressible Flow Computations on Multi-GPU Clusters 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition p 522
[5] Brandvik T and Pullan G 2008 Acceleration of a 3D Euler solver using commodity graphics hardware 46th AIAA Aerospace Sciences Meeting and Exhibit p 607
[6] Watkins J E, Romero J and Jameson A 2016 Multi-GPU, implicit time stepping for high-order methods on unstructured grids 46th AIAA Fluid Dynamics Conf. p 3965
[7] Corrigan A, Camelli F F, Löhner R and Wallin J 2011 Running unstructured grid-based CFD solvers on modern graphics hardware International Journal for Numerical Methods in Fluids 66 221-29
[8] Blazek J 2005 Computational Fluid Dynamics: Principles and Applications (2nd Edition) (Amsterdam: Elsevier)
[9] Castonguay P, Williams D, Vincent P, Lopez M and Jameson A 2011 On the development of a high-order, multi-GPU enabled, compressible viscous flow solver for mixed unstructured grids 20th AIAA Computational Fluid Dynamics Conf. p 3229
[10] Kampolis I, Trompoukis X, Asouti V and Giannakoglou K C 2010 CFD-based analysis and two-level aerodynamic optimization on graphics units Computer Methods in Applied Mechanics and Engineering 199 712-22
[11] Xu T and Chen L 2016 GPU implementation of a viscous flow solver on unstructured grids 6th Int. Symp. on Physics of Fluids vol 42
[12] Xu C et al. 2014 Collaborating CPU and GPU for large-scale high-order CFD simulations with complex grids on the TianHe-1A supercomputer Journal of Computational Physics 278 275-97
[13] Li S 2007 Hypersonic Flow Characteristics of Typical Shape (Beijing: National Defense Industry Press)
[14] Reimartz B U, Herrmann C D and Ballmann J 2003 Aerodynamic performance analysis of a hypersonic inlet isolator using computation and experiment Journal of Propulsion & Power 19 868-75