Parallel Computational Fluid Dynamics Conference (ParCFD2013)

Parallel 3D Numerical Simulation of Continuous Detonation Engine on Graphics Processing Units

Liu Meng*, Zhang Shuangb, Wang Jianpinga—

*aState Key Laboratory of Turbulence and Complex System, Department of Mechanics and Aerospace Engineering, Peking University, 100871, China
bSchool of Electronics Engineering and Computer Science, Peking University, 100871, China

Abstract

Continuous Detonation Engine is a new concept of a supersonic combustion engine for aerospace. 3D numerical simulation of this engine determines the chemical combustion process and kinetic properties of reactive flow within the combustion chamber. To capture the Chapman-Jouguet detonation phenomenon, the grid size needs to be small than 250. The utilization of this fine grid requires a significant amount of computation workload to depict this combustion. With the help of NVIDIA’s CUDA, the computation is discretized and distributed to multiple graphics processing units, which makes the computation time to be reasonable and tolerable. This paper introduces the architecture of NVIDIA’s Tesla C1060 card and CUDA, showing the way to map our model into a CUDA programming and the performance of acceleration. It explicates the details of the movement within the engine. The result justifies that continuous detonation engine is a prominent propeller which releases massive impulse. Some problems of CUDA programming and future plan will also be discussed.

© 2013 The Authors. Published by Elsevier Ltd. Open access under CC BY-NC-ND license. Selection and peer-review under responsibility of the Hunan University and National Supercomputing Center in Changsha (NSCC)

Keywords: Continuous Detonation Engine; 3D Simulation; CUDA

1. ENGINE CONCEPT AND THE NEED OF PARALLEL SIMULATION

1.1 CDE Concept

Continuous detonation engine (CDE) is an engine using detonation combustion to release the fuel power. Compared to pulsed detonation engine (PDE), CDE is capable to provide strong thrust without any intervals. This feature, combined with the inherent high efficiency of detonation, makes CDE become a viable and practical engine for high speed aircrafts in the future.

The concept and structure of CDE are simple, as showed in Fig. 1. This is the combustion chamber. It is formed as a toroidal room between two coaxial cylinder walls. The mixture of fuel and oxidizer is injected into the chamber at the head wall (at the left end of the two cylinders in the figure). The detonation wave rotates continuously, compressing and igniting the fresh mixture. The burnt gas goes downstream, and is then emitted out of the chamber at the right end. The detonation will be quenched when the injection of fresh mixture stops.

1.2 Relative works and the need for 3D simulation

* Corresponding author. Tel.: +86-10-8252038
E-mail address: wangip@pku.edu.cn

1877-7058 © 2013 The Authors. Published by Elsevier Ltd. Open access under CC BY-NC-ND license. Selection and peer-review under responsibility of the Hunan University and National Supercomputing Center in Changsha (NSCC)
doi:10.1016/j.proeng.2013.08.005
There are already some simulations analyzing the continuous detonation. Most of them are 2D simulations, like a fundamental work done by M. Hishida et al. [1]. This paper assumes the cylinder walls will be very flat, so that the curvature of the walls can be ignored. This assumption decreases the workload of computation, but it will not be fully helpful to design the exact size and shape of a practical CDE. Particularly, the diameter ratio between the inner and outer cylinders should be finely studied. It is not clear how the ratio will influence the detonation wave.

A fine grid is essentially needed for CDE simulation. To capture the Chapman-Jouguet detonation, the grid size needs to be smaller than 250 μm, so that the total number of all the nodes in the grid is at the order of $10^9$ to $10^{12}$, for a model as large as a cola can. And every step of time integration in the loop is very tiny. Both of the facts mean that 3D simulation has significant amount of workload, so that it has to be applied in a parallel way.

![Scheme of CDE combustion chamber](image)

Fig. 1: Scheme of CDE combustion chamber. 1. detonation wave, 2. burnt gas, 3. fresh mixture, 4. contact surface, 5. oblique shock, and 6. rotation direction.

Graphics processing units (GPU) can be an effective and cheap way to provide powerful parallel computation. They are called as general purpose GPU, or GPGPU. This paper shows the method we do with coarse grids on NVIDIA’s Tesla C1060 card, and gives the result of the detonation wave. Experience and future work are also discussed.

2. NUMERICAL MODEL

2.1 Grid

The grid is a little different from the physical shape of the combustion chamber. To promote the thrust of CDE, a Laval-shape nozzle is added at the emission end of the chamber.

The diameters of the inner walls and the outer walls are 60mm and 80mm. The length of combustion chamber is 70mm. The Laval nozzle is 40mm long, and the area ratio of throat and exit is about 1:16. The number of nodes in directions of radius, toroid, and axis are 32, 256, and 128, respectively.

2.2 Boundary conditions

The ignition is set as reading a 2D Chapman-Jouguet result at the head end. The initial status is identical from the inner wall to the outer wall at the same toroidal position. The rest of the chamber will be set as fresh mixture of fuel and oxidizer.

At the head end, the boundary nodes, or the ghost nodes, are set as the injection of fresh mixture. Because the detonation wave travels around the head end, the pressure varies dramatically. If the wall pressure is too high, the mixture will not go into the chamber. The movement of the injection flow depends on how the wall pressure changes.

At the inner wall and the outer wall, the boundaries are set as noncatalytic, adiabatic and slipping walls.

In the toroidal direction, the boundary condition is set as periodic boundary.

At the exit end, the flow is set as an isentropic expansion.

2.3 Governing equations
The fuel is set as \( H_2 \), and the oxidizer is set as air. The stoichiometric air mixture is premixed. 3D Euler equations, arbitrary coordinates, and two-step chemical kinetic model are used. In the following equations, \( \alpha \) denotes the induction reaction, and \( \beta \) denotes the exothermic reaction:

\[
\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{E}}{\partial \xi} + \frac{\partial \mathbf{F}}{\partial \eta} + \frac{\partial \mathbf{G}}{\partial \zeta} = \mathbf{S}, \quad (1)
\]

where

\[
\mathbf{U} = \begin{pmatrix}
\rho \\
\rho u \\
\rho v \\
\rho w \\
e \\
\rho \beta \\
\rho \alpha
\end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
\rho \alpha \\
\rho \alpha
\end{pmatrix}, \quad \mathbf{E} = \begin{pmatrix}
\rho \bar{U} \\
\rho \bar{U} u + p \bar{\xi}_x \\
\rho \bar{U} v + p \bar{\xi}_y \\
\rho \bar{U} w + p \bar{\xi}_z \\
\bar{U} (p + \epsilon) \\
\rho \bar{\alpha} \\
\rho \bar{\alpha}
\end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix}
\rho \bar{V} \\
\rho \bar{V} u + \rho \bar{m}_x \\
\rho \bar{V} v + \rho \bar{m}_y \\
\rho \bar{V} w + \rho \bar{m}_z \\
\bar{V} (p + \epsilon) \\
\rho \bar{\beta} \\
\rho \bar{\beta}
\end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix}
\rho \bar{W} \\
\rho \bar{W} u + \rho \bar{p}_x \\
\rho \bar{W} v + \rho \bar{p}_y \\
\rho \bar{W} w + \rho \bar{p}_z \\
\bar{W} (p + \epsilon) \\
\rho \bar{\alpha} \\
\rho \bar{\alpha}
\end{pmatrix}, \quad (2)
\]

\[
\bar{U} = u \xi_x + v \xi_y + w \xi_z
\]

\[
\bar{V} = u \bar{m}_x + v \bar{m}_y + w \bar{m}_z
\]

\[
\bar{W} = u \bar{p}_x + v \bar{p}_y + w \bar{p}_z
\]

The mixture is set as perfect gas, so that the pressure and energy will be calculated as

\[
p = \rho RT, \quad e = \frac{p}{\gamma - 1} + \beta pq + \frac{1}{2} \rho u^2 + \frac{1}{2} \rho v^2 + \frac{1}{2} \rho w^2, \quad (4)
\]

where \( \gamma \) is the heat ratio.

In the two-step chemistry model, the two source terms are determined as

\[
\frac{d \alpha}{dt} = -k_1 \rho \exp \left( - \frac{E_1}{RT} \right), \quad (5)
\]

\[
\frac{d \beta}{dt} = \begin{cases}
0 & \quad (\alpha > 0) \\
-k_2 \rho^2 \left\{ \beta^2 \exp \left( - \frac{E_2}{RT} \right) - (1 - \beta)^2 \exp \left( - \frac{E_2 + q}{RT} \right) \right\} & \quad (\alpha \leq 0)
\end{cases} \quad (6)
\]

The flux terms in (1) are processed by using MPWENO scheme. The three-step Runge-Kutta method is applied for time integration.

3. PARALLEL METHOD

3.1 CUDA

CUDA is a language for NVIDIA’s GPU parallel programming, and it can also represent the hardware architecture. CUDA is an extension to C, and it can be easily applied to transfer CPU serial program to a parallel one.

The hardware of this GPU device keeps updating. Currently, the core unit of the device is a combination of stream multiprocessors (SM), memories, and special calculation units. SM can provide high speed calculation for single precision problems. Double precision problems will be dealt on special calculation units, which will be much slower. Fermi, the next generation of NVIDIA’s CUDA architecture, will promote the double precision function greatly [4]. GPU device is coprocessors, so that the calculation progress will be controlled by CPU.

The task carried on GPU is called as kernel. Kernels are run by many threads on the device, and threads are arranged in blocks. Blocks can also be arranged in grids. The arrangement can be 1D, 2D or 3D. All the threads in one block have a same memory to share. This shared memory will not be accessed by threads from other blocks.
3.2 Features for CUDA

Acceleration is surely the objective. To realise this goal by CUDA, there are some features should be particularly noticed. One is coalescence. There is a large memory on the device, and the coprocessors can access the memory very quickly. But in order to keep this high speed, the threads should access serial memory positions in warps. Warp basically means a group of 32 units. Some users who want to obtain better performance will also pay attention to memory bank conflicts. The program may need to change the algorithm or data structure to maintain the high access speed.

Another important feature is scarce resources. The maximum number of threads and blocks on one device card is limited. The registers and shared memories, which have very fast access speed, are also limited on the device. If the kernel needs to use too many registers, the program will turn to use other device memories that will obviously become very slow. If the kernel needs to use too many shared memory, the max number of threads running at the same time will be low. To find a balance point to utilize all the resources is very important.

And another thing which needs to be noticed is control instructions, as if and switch. Within one block, all the threads had better to execute same branch of the control instructions. If threads go to different branches, threads have to wait when other threads run the different branches, making it like a serial process.

3.3 Major procedures in this paper

E. Elsen et al. have summarized some typical data categories in CUDA programming [2]. We have met some of the categories in this paper, and they are dealt differently.

Pointwise. It means that the kernel almost uses all the variables on the same node of a structured grid. In the serial program, these calculations are always included in a loop. CUDA program can simply use one thread to calculate one node, so that the loop is no longer needed. Because there is no communication between threads, it is also called as embarrassed parallel program.

Stencil. It means that the kernel needs to access some serial position in the memories. Data which will be accessed by many threads can be stored in shared memory within one block. The difficulties emerge when kernels deal with boundary conditions. In some directions, the variables on grids may be fragments on the memories [3]. An elaborate design of data reading, processing and relocation is needed. Shared memories can make most of the data access become coalescent, and reduce the times for reading and writing.

Reduction. In this case we treat the relevant array as a binary tree, using minimum operation to get the result. Because the shared memories can not be used between different blocks, some extra memories and operations are inevitably needed for communication.

4. RESULT AND DISCUSSIONS

![Fig. 2: Streamlines and temperature profile of CDE, at 1004 μs after ignition.](image-url)
In Fig. 2, the streamlines and temperature profile are showed. The basic structure of detonation wave is clearly identified. The average speed of the flow at the exit is over 1900m/s. And the speed of the detonation wave basically matches the theoretical speed, which means this calculation is successful.

The parallel algorithm is adjusted to cope with CUDA features, so that the speed can not be simply compared with serial CPU program. Approximately this CUDA program is in the order of 20 times quicker than CPU serial program. All the kernels run in this program are not all accelerated extensively. The pointwise kernel runs very quickly. But the kernel which deals with MPWENO occupies the most of the time. The reason for slow speed is too many registers are required in this kernel, so that the kernel has to invoke the device memory instead with slow speed.

This experiment of CUDA shows that this new parallel device has powerful abilities. A cluster formed by many CPUs and GPU cards can surely change the traditional idea of parallel computation. Because CPU control the process of parallel computation, debugging can be much easier than common parallel languages. But learning and utilizing CUDA is not that easy. Users have to realize the details of hardware architecture, and have to arrange the memories manually. CUDA is not stable. It is updated quickly. The old version of CUDA may not calculate correctly in some cases. Users should use the latest version of CUDA, and compare GPU results with the corresponding CPU programs as testing some benchmark cases.

REFERENCES