Fluid Dynamics Simulation on a GPU

Chris I. Juric, Damir Juric

Fluid Dynamics Simulation on a GPU

Chris I. Juric[1] and Damir Juric[2,3]

[1]Clonlara, Ann Arbor, Michigan, USA
[2]Université Paris Saclay, Centre National de la Recherche Scientifique (CNRS), Laboratoire Interdisciplinaire des Sciences du Numérique (LISN), 91400 Orsay, France
[3]Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Centre for Mathematical Sciences, Wilberforce Road, Cambridge CB3 0WA, UK

April 2024

Abstract

We solve the Navier-Stokes equations for incompressible fluid flow on modern GPU (Graphics Processing Unit) computing devices, using the DirectX11 API by Microsoft. We implement the well-known projection method on DirectX compute shaders in the languages HLSL (High Level Shader Language) and C++. We compute the flow inside of a lid-driven cavity and compare results to those from the standard benchmark. We also discuss future developments to enable the use of open software standards to achieve better results.

1 Introduction

The numerical simulation of the flow of fluids has immense importance across many scientific and engineering disciplines. Thus the ability to perform the simulations as quickly as technology allows has been pursued since the advent of computers. The term "shader" was introduced by Pixar in 1988 with their RenderMan Interface Specification (RISpec) Application Programming Interface (API). Early GPUs had a limited number of programmable shader stages but there are now many more that we can make use of, one of those being compute shaders. Compute shaders are special in the fact that they are able to be run separately from the graphics pipeline on the GPU and runs in the compute pipeline instead. In this paper we will describe the implementation and results of an incompressible fluid flow simulation using these compute shaders.

2 Method

The Navier-Stokes equations for incompressible fluid flow are:

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla P + \mu \nabla^2 \mathbf{u} \tag{2}$$

where Eqs. (1) and (2) express the conservation of mass and momentum respectively. Here $\mathbf{u}$ is the velocity vector field, P is the pressure, ρ is the density and μ is the viscosity of the fluid.

2.1 The Driven Cavity Problem

The driven cavity problem is a canonical test problem for the numerical solution of the incompressible Navier-Stokes equations. The problem consists of a square domain (a box of side length, L) filled with fluid, where the side walls of the box are solid stationary walls except for the top wall which is moving in the positive x-direction with a prescribed velocity, V, as depicted in figure 1. The non-dimensional Reynolds number is defined as $Re = \rho VL/\mu$. The Reynolds number is a measure of the ratio of inertial to viscous forces. For this test case we have chosen $Re = 10$.

To solve equations (1) and (2) we use Chorin's [1] projection method. Stam [2, 3] implemented a popular version of this method for the gaming community.

3 Computing on Graphics Processing Units

We implement the above method on GPU. In order to do this we program shaders which are the programs that run on GPUs. In our case we use HLSL. Inherently this is a language for parallel processing, that is, using many computing threads simultaneously to achieve faster computing speeds. The HLSL language was developed by Microsoft for the DirectX API. There are several types of shaders which are typically used in computer games to produce an image on the screen. However it was realized that shaders can also be used to do other computational work including the numerical solution of partial differential equations such as the Navier-Stokes equations which we described above. The advantage of using GPU architectures is that they are much faster at parallel (SIMD - Single Instruction Multiple Data)) computing than CPU architectures.

4 Overcoming CPU-GPU interfacing bottlenecks

An issue that occurs while interfacing a CPU with a GPU is saving results to storage. In our work we developed a method to resolve this issue by using a triple buffer solution. Triple Buffered Memory Access (similar to the Mailbox

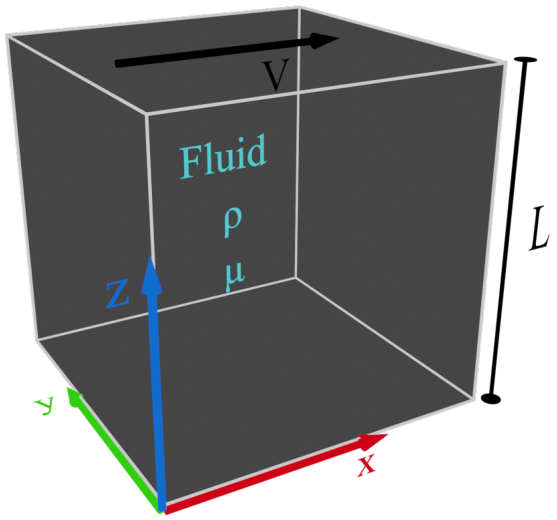

Figure 1: Schematic of the driven cavity problem

presentation mode in graphics API's) allows the GPU to continue computations while data is being transferred from VRAM (Video Random Access Memory) to RAM. Although this solves the issue of the CPU waiting for the GPU, because we use the main thread to start writing to a file, the GPU has to wait for the CPU to finish. The simple solution is to offload this work to a worker thread so the CPU can continue running Dispatch calls. (A Dispatch call as opposed to a Draw call is used to call the Compute Pipeline instead of the Graphics Pipeline.)

In the future it seems likely that this procedure could be improved by using Direct Storage once the technology becomes available.

5 Results and Discussion

The driven cavity case was run on a RTX 3070 mobile GPU for 10,000 timesteps and the result for the velocity field is shown in figures 2 and 4. Figure 2 represents the velocity vectors on a slice through the mid-plane of the computational box. One can see the vortical structure of the flow inside the cavity due to the motion of the top wall which drags the fluid to the right. Figure 4 shows a three-dimensional view of the streamlines in the flow highlighting the fact that this is a fully three-dimensional simulation of the Navier-Stokes equations. This result is in agreement with the standard benchmark solution of the driven-cavity problem using the well-known code BLUE [4] which is shown in figure 3.

We measure the computational time for this calculation on our GPU and compared it with a simulation on an 8-thread CPU. The GPU took 402 seconds for the 10,000 timesteps while the CPU took 2561 seconds. Thus the GPU was 6.37 times faster. While this is not an exact one to one comparison, it does show the speed-up capabilities of GPUs vs CPUs.

6 Conclusions

In this paper we demonstrated the advantage in speed of using a GPU to run a numerical simulation of incompressible fluid flow over the same computation run on a CPU. We observe that the GPU is about 6 times faster than the CPU on the simulation of the driven-cavity problem. The comparison of the results between the two shows nearly identical agreement. The solution method uses DirectX11 on GPUs written in HLSL and C++.

7 Future Vision

Khronos group consortium develops a variety of open standards for different compute applications, including Vulkan which is what we plan to use in our next implementation of GPU based fluid simulations instead of DirectX11. The reason for this choice is to allow multiplatform operability since DirectX11 and

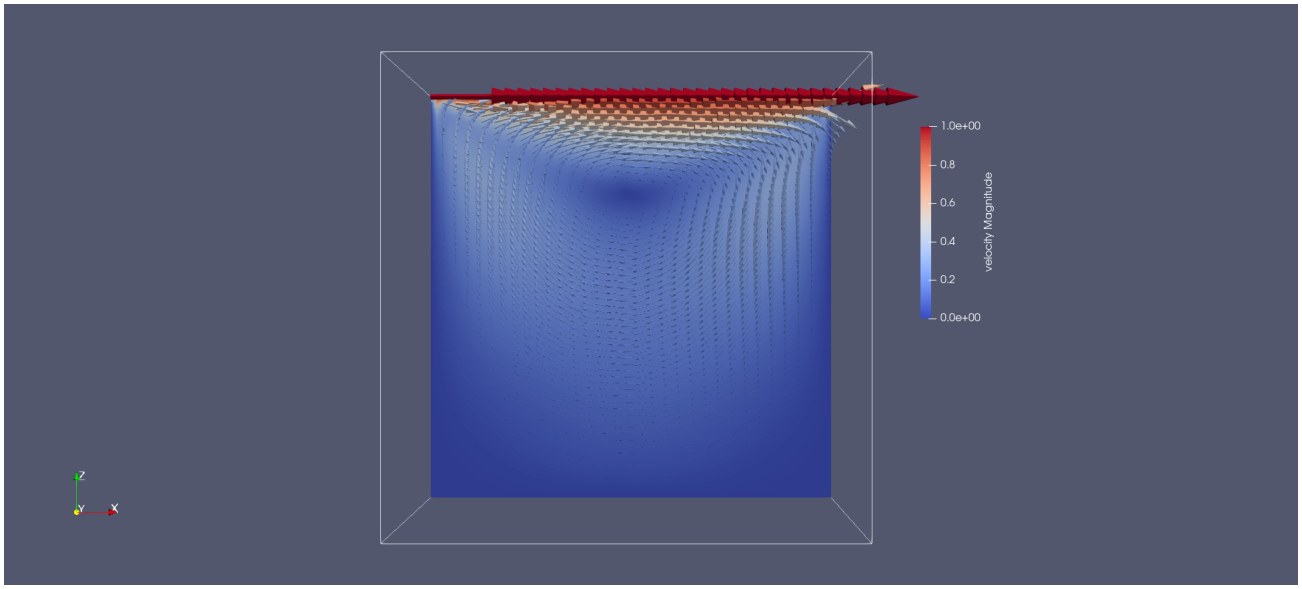

Figure 2: Our GPU results: Velocity vectors on a mid plane slice for the driven cavity.

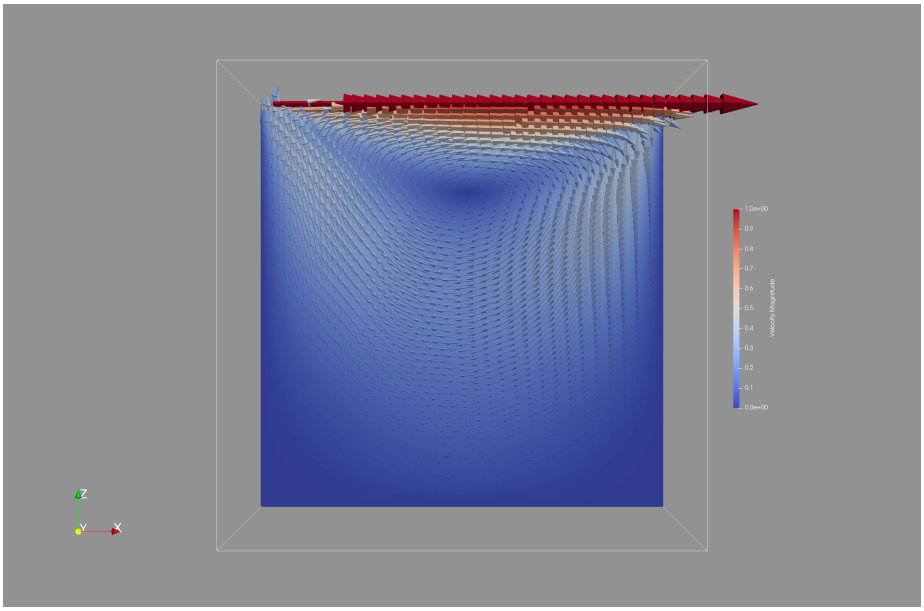

Figure 3: CPU results using Code BLUE [4]: Velocity vectors on a mid plane slice for the driven cavity.

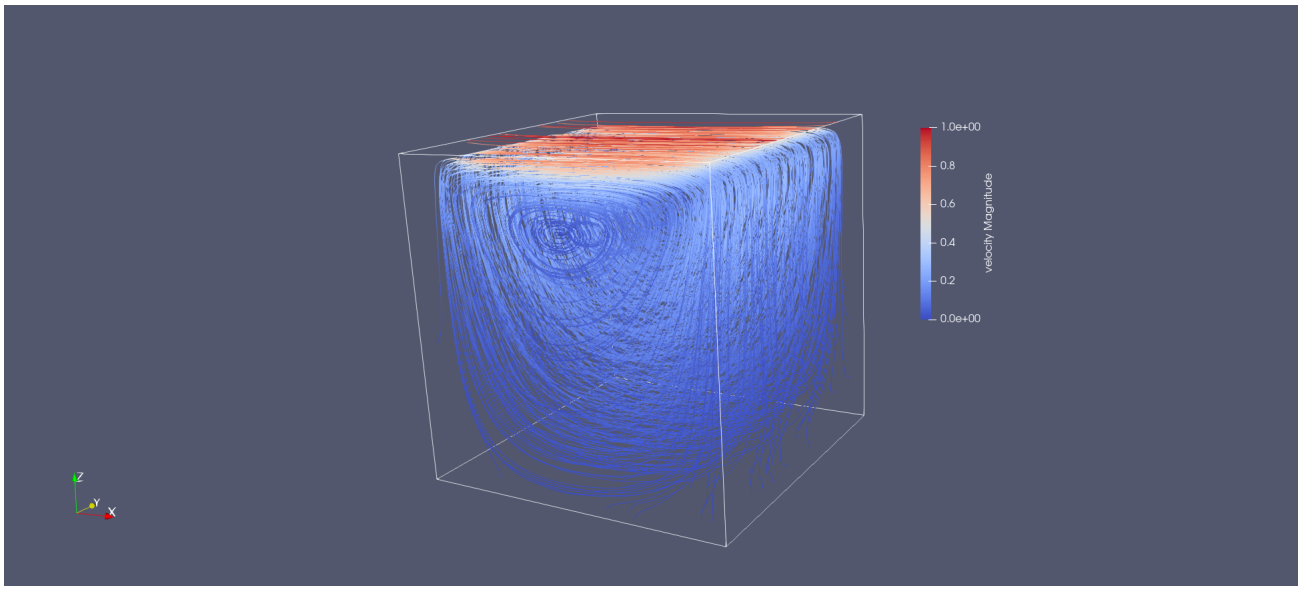

Figure 4: Our GPU results: Stream lines of the flow field in the driven cavity.

more recent DirectX versions by Microsoft are only compatible with the Windows operating system. This will however require the use of a different shader language namely Graphics Library Shader Language (GLSL) instead of High Level Shader Language (HLSL).

References

[1] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Maths. Comp.*, 22(104):745–745, 1968.

[2] J. Stam. Stable Fluids. *SIGGRAPH*, pages 121–128, 1999.

[3] J. Stam. Real-Time Fluid Dynamics for Games. *Proc. Game Developers Conference*, 2003.

[4] Seungwon Shin, Jalel Chergui, and Damir Juric. A solver for massively parallel direct numerical simulation of three-dimensional multiphase flows. *Journal of Mechanical Science and Technology*, 31(4):1739–1751, 2017.