Magallanes, John

Professor McDanel

CPS 376

17 December 2021

Final Project: Ocean Propagation

For my final project, I decided to focus on simulating ocean waves. To simulate ocean waves, we need to dive into the topic of wave propagation: “Progression and transformation of waves in time and space.” I choose this topic because trying to simulate waves may be computationally heavy as the waves you try to simulate scale in size. My goal for this project is to use my knowledge of parallel computing and integrate using CUDA threads to possibly speed up the computational performance of water waves.

Before diving into the code, I needed to understand the idea of wave propagation. One of my goals for my project was to simulate waves on a two-dimensional scale. As I began to research, I stumbled upon the Naiver-Stokes equations (NSE). The Naiver-Stokes equations describe the motion of fluids using velocity, pressure, density, and viscosity. However, these equations were too general and I needed to find equations that would help me simulate ocean waves. Thus, I found equations that were based on the NSE which are called the Shallow-Water equations (SWE). The Shallow-Water equations describe the motion of shallow water/fluid. These equations would prove useful in understanding how waves can be simulated using computers. One difficult challenge of this project was understanding these equations. The NSE and SWE are very physics and math-heavy which required me to research to have a better understanding of these equations.

After researching my project, I decided that the goals of my project changed. My current goal for the project changes from simulating ocean waves to water waves. Specifically, I wanted to parallelize simulating water waves that are created from a point of origin since the SWE allowed for this. I was able to find a SWE Github repository that displayed a 3-dimensional visualization of waves and a 2-dimensional vector velocity versus time graph using Python. Looking at this repository, I decided that displaying a vector velocity graph instead of a 3-dimensional representation would be the new goal of my project since plotting vectors would be easier than displaying complex visual graphics. Therefore, I decided that I only wanted to simulate waves on a grid plane and display their velocity vectors over a period of time. Using this GitHub repository, I would base this existing code as a road map to translate the necessary equations/computations from Python to CUDA.

One challenge arose from how calculations were produced in the SWE repo. The organization of code in the repository made it difficult to understand what aspects of the code were doing the main computation. Thus, I changed the structure of the original SWE repo and made a file titled “custom_swe.py” to save my changes. This was an important step because I wanted to isolate what part of the code is responsible for doing the majority of the computation. In other words, what code from the SWE repo should I put into the CUDA kernel? Another challenge that arose was the use of Python libraries that are not available in CUDA. For example, the SWE repo uses the scientific computing package NumPy to perform calculations for the velocity vectors of the waves. When programming in CUDA, the C++ standard library is the only library accessible to use. An initial idea that I had was trying to install a C++ NumPy package onto my CUDA code, but this is not possible. Another idea I had was trying to manually recreate the functions used in the SWE repo. I was able to go online and reference some

functions that resembled NumPy functions such as linspace. However, other functions such as transpose, became very difficult to replicate onto CUDA or even C++. One reason for this was because, in the SWE repo, the matrices used to calculate the water waves result in large NumPy arrays. This is important because NumPy arrays differ from standard arrays in Python. For example, using NumPy arrays allows for special indexing for accessing information and for matrix multiplication. Replicating these NumPy functions onto CUDA without being able to see how it works makes it difficult to replicate them onto CUDA.

My intended goals for this project were to use multiple threads to help increase performance in the calculation of water waves and project the results onto a graph that displays velocity over time. Some challenges that occurred in this project were understanding the advanced level of math required to comprehend equations such as Shallow-Water, being able to organize existing Python code and lay out what to translate onto CUDA, and trying to replicate functions from libraries such as NumPy onto CUDA. Once these challenges are resolved, then I could have sent the necessary information onto the CUDA kernel and begun the challenge of using threads to do parallel computations of the water waves.

Works Cited

- http://www.coastalwiki.org/wiki/Wave_propagation
- https://www.britannica.com/science/Navier-Stokes-equation
- https://github.com/jostbr/shallow-water