Project3: GPGPU Parallelization

Start Assignment

Due Apr 27 by 11:59pm **Points** 100 **Submitting** a file upload
Available Apr 13 at 12am - May 10 at 11:59pm

This project involves using CUDA to develop a parallel implementation of a computational fluid dynamics model that was parallelized using OpenMP in the previous project. In this project we will explore implementing parallelism using the CUDA programming model developed by NVIDIA for the programming of their GPU architectures. The model closely mimics the hardware structure of the GPU architecture and thus is often required to achieve optimal performance. The model is a fine grained threaded programming model where threads. Threads are grouped into thread blocks where there threads in different thread blocks generally run without synchronization, and threads within a thread block are assigned to the same streaming multiprocessor and thus can be synchronized efficiently. This is a co-processor model where the overall control flow of the algorithm is managed from the CPU with data being copied to and from the coprocessor using specialized library calls. The CPU can also instruct the co-processors to launch kernels which consists of large numbers of parallel threads. The details of this programming model will be discussed in class.

Note, this project is using the **scout** cluster which is an older cluster where NVIDIA Tesla K20 GPGPU cards are installed. To log into this cluster you will log into **scout-login**. The starter package for this project is provided on titan.hpc.msstate.edu under the file /scratch/CSE4163/Project3/project3.tar. Copy this file to the scout login and untar the directory as with previous projects to begin. The project description file from project3.tar is included below. The project source code and report documenting the results will be submitted as the deliverables of the project.

ProjectDescription.txt:

**** CUDA Programming for the fluid solver ****

In this project you are going to accelerate the fluid solver that we previously parallelized with OpenMP utilizing the CUDA programming model for NVIDIA GPGPUs. Note, for this project we will be utilizing the **scout** cluster instead of the shadow cluster. This cluster provides a few nodes with NVIDIA K20 GPGPU coprocessors. The provided runscripts will allow you to submit jobs to the cluster for GPGPU execution using the sbatch command with the provided job script files such as the file "runfluid32.js".

Since the code is utilizing a co-processor that has separate memory independent from the CPU memory, the execution of kernels will require copying data back and forth between the CPU and GPGPU. You should develop kernels in an incremental fashion so that you can debug them one by one. As with the OpenMP version, you should verify that the code produces the same results each step of the way. When the code is completely ported to GPGPU kernels, then it will be possible to generate the initial conditions and evolve the solution variables entirely on the GPGPU without copying data back and forth between the CPU and GPGPU. This will allow you to get the highest performance. When you have successfully implemented all of the fluid solver operations on the GPGPU the code should run significantly faster than the original serial version. You should be able to get run times where it only requires about 20 nanoseconds per cell per timestep to solve the fluid problem (at least on the larger meshes).

The GPGPU utilized in this project is the NVIDIA Tesla K20 GPGPU. The cores of this GPGPU run at a clock speed of 706 Mhz. The K20 consist of 13 Stream Multiprocessors (SMs), each with 192 CUDA cores (that execute individual threads) for a total of 192*13=2496 CUDA cores. Assuming each CUDA core can launch two single precision floating point operations in a single clock cycles, what is the peak floating point performance that can be achieved by the K20 GPGPU? The K20 also has a memory bandwidth of 205 Gigabytes per second. Based on memory bandwidth what is the peak performance that can be obtained from the fluid solver?

In your report you should document the strategies that you used in the development of your GPGPU kernel functions and also record the performance effects of changing grid sizes on the execution of the program. Do larger meshes exhibit higher performance? Does the program saturate the memory bandwidth of the GPGPU? Does it saturate the compute capacity of the GPGPU? Answer these questions in your report.