Homework 12

Juan Camilo Velasquez and Joaquin Rodriguez

Assignment 12: In the present assignment, we used Cupy to create and solve a dense system of equations. The main idea was to create a CuPy raw kernel function that initialized a banded matrix and solved the system $Ku = f$. A schematic of the system is shown below:

$$\mathbf{K} = \begin{bmatrix} 4 & -2 & 0 & \cdots & 0 & 0 & 0 \\ -2 & 4 & -2 & \cdots & 0 & 0 & 0 \\ 0 & -2 & 4 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & -2 & 0 \\ 0 & 0 & 0 & \cdots & -2 & 4 & -2 \\ 0 & 0 & 0 & \cdots & 0 & -2 & 2 \end{bmatrix}, \quad f = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1/N \end{Bmatrix}$$

where, N is the number of equations in the linear system of equations. One of the GPU nodes (gpu1v100) from Arc HPC was used to run the program using CUDA capabilities. During runtime, the execution was subdivided using blocks of 16x16 threads and a total number of grids enough to cover the whole execution. The performance of the implementation was evaluated with an extensive system ($N = 30000$). For that, the CPU time was measured from the creation of the matrix and vector until after printing the solution vector. As a way to compare the improvements gained when using fine-grained parallelization with CUDA, the system was solved using only the numpy function, which uses CPU-only resources. The results are shown in the table below.

	Numpy	**CUDA-CuPy**	**× Factor**
Run Time	13.0575 s	4.4485 s	2.93

Table 1. Runtime comparison for numpy and CUDA-CuPy codes

Conclusions:

The results clearly demonstrate that the CUDA implementation achieves a significant speedup, approximately tripling the performance of the original numpy implementation. This is a notable enhancement, especially considering that numpy leverages highly efficient versions of LAPACK and BLAS from the Intel Math Kernel Library (MKL). This example underscores the potential of CUDA for enhancing high-performance applications, mainly through fine-grained parallelization, as demonstrated in the matrix creation scenario.