SOFT354 Report

# Introduction

In this report I will be investigating the use of parallel programming in Matrix-Matrix multiplication, how the efficient it is to do so and how scalable it is.

## Matrix-Matrix multiplication explained

A matrix is a rectangular arrangement of numbers into rows and columns, where each number inside it is referred to an element or entry. Each entry can be referenced with a coordinate-like system, for example in figure 1, the first element in the matrix P is P[0,0] the second is P[1,0] and so on. When multiplying two matrices M and N, each element of the output matrix P is the dot product of a row in matrix M and a column of matrix N.

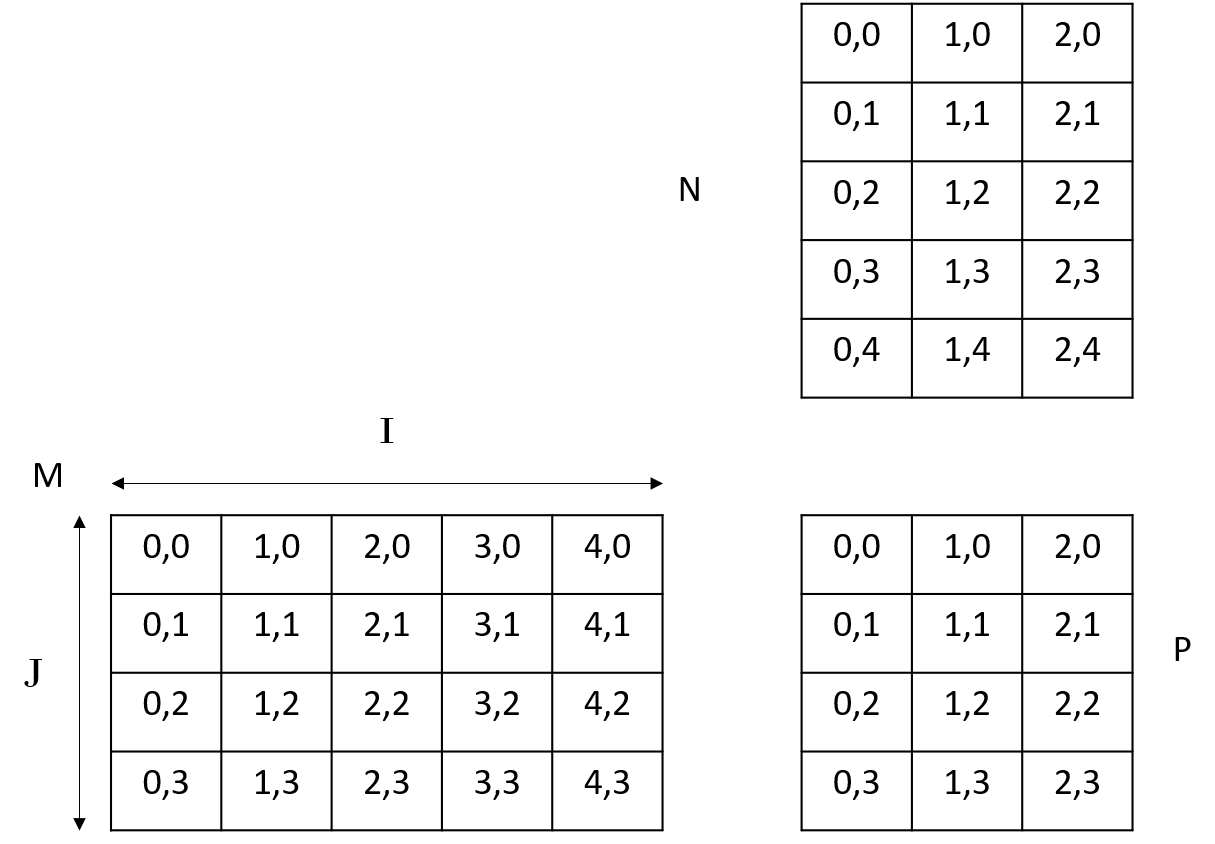


Figure 1 Matrices represented to show how a multiplication is formed

To work out the value of the element P[0,0], we use the formula P[0,0] = M[:,0]  https://whatis.techtarget.com/WhatIs/images/dot-prod.jpg N[0,:], where : implies all values. Expanding this we get P[0,0] = M[0,0] \* N[0,0] + M[1,0] \* N[0,1] + … M[k,0] \* N[0,h]. leading to the general formula of the figure below.

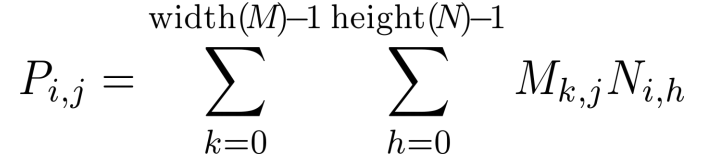


Figure 2 formula for matrix multiplication (Dr Mario Gianni, 2019)

## Matrix multiplication uses

Matrices are an essential tool for any mathematical program, and are used frequently to represent graphical data, this means implementing matrix operations like matrix multiplication is essential to image processing and other similar programs. Another more current use of matrices is in machine learning, Neural networks often use matrices to represent the data and this makes it easier to perform back propagation on the set of weights for a given node. As this is such a basic tool for programmers to use its essential that matrix multiplication can be done efficiently and quickly, which is where the use of parallel computation becomes increasingly more valuable.

## CUDA and parallel computation

A CPU deals with instructions in sequential order, the concept of parallel computation is the instructions that do not need to be completed in sequential order can be done at the same time to improve performance.

CUDA is a parallel computational architecture developed by NVIDIA which makes it possible to use the many computing cores in a graphics processor to perform general-purpose mathematical calculations, achieving dramatic speedups in computing performance. In these GPU-accelerated applications the essential sequential parts of the workload is done by the CPU, while compute intensive portions are run on the thousands of GPU cores in parallel. (NVIDIA Developer, 2020).

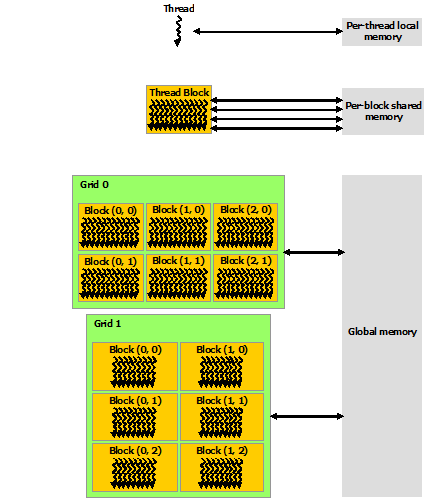


Figure 3 CUDA program architecture (Docs.nvidia.com, 2020)

A CUDA program uses several GPU cores ordered into Grids, these grids are then split up into blocks, and blocks are split into threads, where each thread executes one part of the parallel program individually. The general program flow of a CUDA program involves a serial portion of the program which sets up memory for the devices, the execution of a kernel, To allow communication between grids blocks and threads, there are several layers of memory, a thread has its own local memory, above that is the shared memory which is shared between all threads in a single block, and above that is the global memory which is visible to all blocks and all grids.

In matrix-matrix multiplication each of the elements in an output matrix (P) can be calculated without the knowledge of any other elements in P. This means that we can apply parallel techniques as a similar process is being applied to each of the elements in M and N to achieve significant speedup as the program size increases.

# Implementation

## CPU

## GPU

# Evaluation

# Conclusion

# Bibliography

NVIDIA Developer. (2020). *CUDA Zone*. [online] Available at: https://developer.nvidia.com/cuda-zone [Accessed 12 Jan. 2020].

Docs.nvidia.com. (2020). *Programming Guide :: CUDA Toolkit Documentation*. [online] Available at: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html [Accessed 12 Jan. 2020].

SOFT 356 lecture (week 4), 2019, Dr Mario Gianni. https://dle.plymouth.ac.uk/pluginfile.php/1781554/mod\_resource/content/2/SOFT354%20-%20Data%20Parallelism%20in%20CUDA%20-%20Week%204.pdf