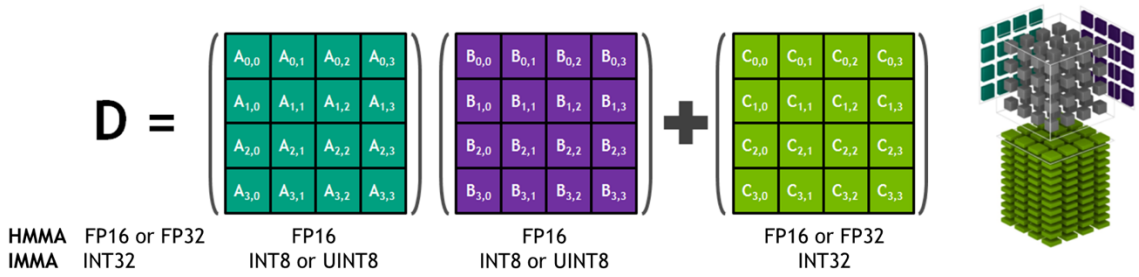
Kennesaw State University

HPC & Parallel Computing

Project – GPU programming

Instructor: Kun Suo
Points Possible: 100
Difficulty: ★★★★☆



General Matrix Multiplication (GEMM) is typically defined as:

$$C = AB$$

$$C_{m,n} = \sum_{n=1}^{N} A_{m,n} B_{n,k}$$

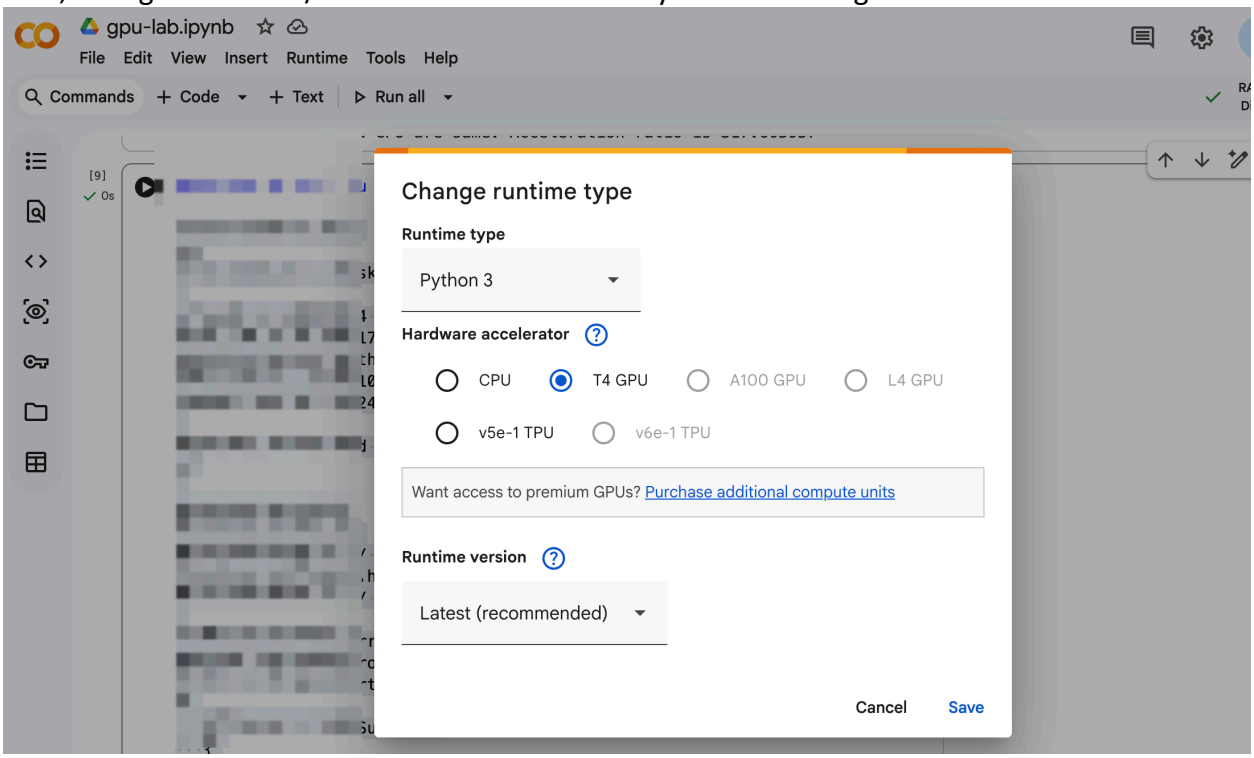
where A is an m*n matrix and B is an n*k matrix.

Task 1 (10 points):

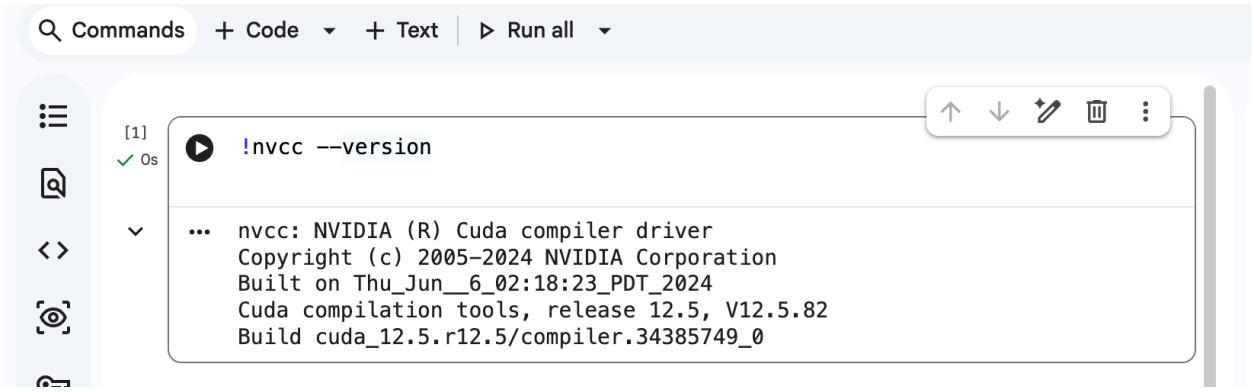
Write a program running on CPU that the user inputs 3 integers M, N, K (512 ~ 8192) and it randomly generates two matrices A (M*N) and B (N*K), and performs matrix multiplication to obtain matrix C. The output is the time taken for the matrix calculation.

Task 2 (40 points):

First, configure CUDA C/C++ and the environment yourself in Google Colab with GPU runtime.



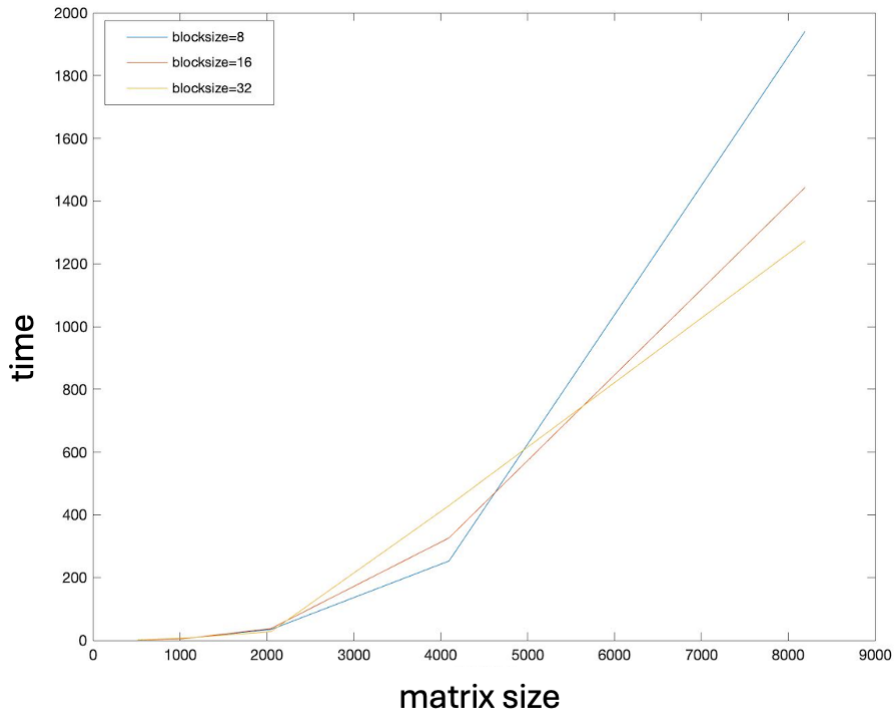
Test the CUDA environment as follows:



Modify the code in Task 1 and implementing a parallel version of general matrix multiplication using CUDA. Inputs of the program are matrix dimensions m, n, k, and CUDA block_size. In the code, make sure it has 2 functions: One function uses the CPU for GEMM (General Matrix Multiplication), and the other uses CUDA. After both functions finish running, the output matrices C from both functions are compared to see if they are identical. If they are the same, the speedup

ratio of CPU vs. GPU is printed on the screen; otherwise, an error message indicating inconsistent results is printed. (Note that if your matrix elements are floating-point numbers, elements within a certain tolerance can be considered the same. You can set a very small threshold (Epsilon), such as 10^{-9}. The equality check formula is: $|a - b| < \varepsilon$. If the difference between a and b is less than this small value, we consider them "equal".) The time of verification function should not be included in the parallel solution.

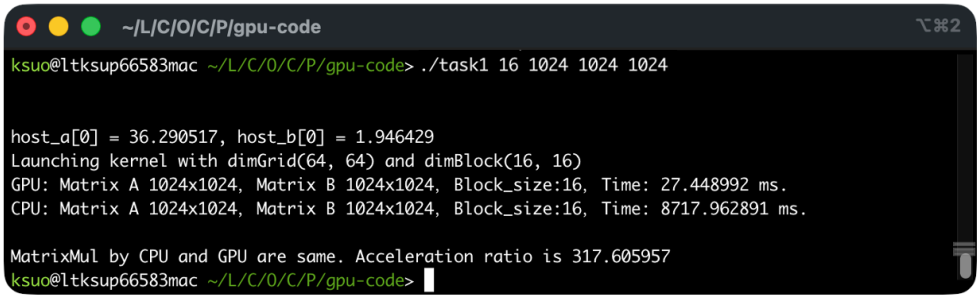
For GPU programs, measure and plot the execution time for different block sizes and matrix sizes:



For a given block size (e.g., block size = 8), compare the speedup ratio of the CPU version and the GPU version.

Order of Matrix	1024	2048	4096
CPU Time			
GPU Time			
Speedup			

Expected Output



Task 3 (50 points):

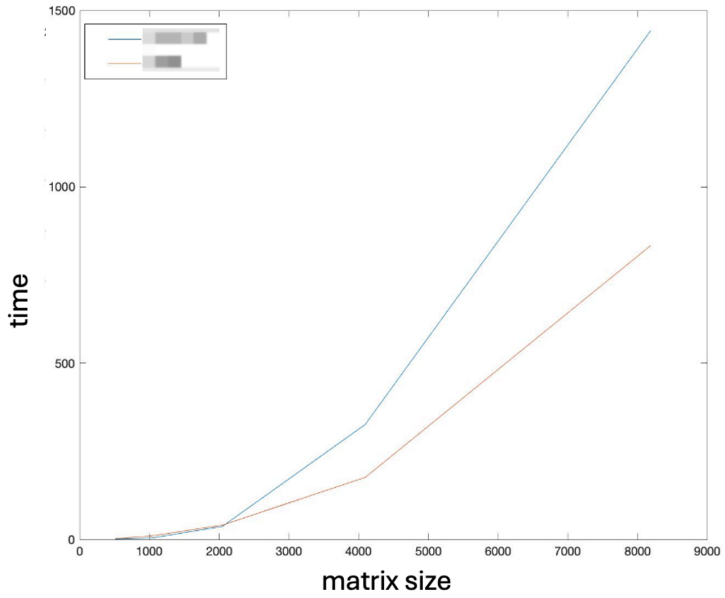
cuBLAS is a CUDA-X math library containing highly optimized basic linear algebra subroutines for matrix and vector operations, specifically tuned for superior performance on NVIDIA hardware. Using NVIDIA's matrix computation library cuBLAS, we will perform matrix multiplication with matrix sizes increasing from 512 in increments of 2 up to 8192, and compare and analyze its performance against the matrix multiplication performed in Task 2.

In your implementation, you must include a verification function that checks whether the output matrix C produced by both GPU (Task 2) and CUBLAS (Task 3) are identical to the original matrix C. (Note that if your matrix elements are floating-point numbers, elements within a certain tolerance can be considered the same. You can set a very small threshold (Epsilon), such as 10^{-9}. The equality check formula is: $|a - b| < \varepsilon$. If the difference between a and b is less than this small value, we consider them "equal".) The time of verification function should not be included in the parallel solution.

CUBLAS Reference Documentation:
https://docs.nvidia.com/cuda/pdf/CUBLAS_Library.pdf

CUBLAS matrix multiplication example:
https://github.com/kevinsuo/CS4522/blob/main/matrixMulCUBLAS.pdf

For GPU vs. cuBLAS results, measure and plot the execution time for different matrix sizes:



For a given block size (e.g., block size = 32), compare the speedup ratio of the GPU version and the cuBLAS version.

Order of Matrix	512	1024	2048	4096	8192
GPU Time					
cuBLAS Time					
Speedup					

Expected Output

gpu-lab.ipynb - Colab

```
!./task2 32 2048 2048 2048

host_a[0] = 80.119367, host_b[0] = 56.831910
Launching kernel with dimGrid(64, 64) and dimBlock(32, 32)
GPU: Matrix A 2048x2048, Matrix B 2048x2048, Block_size:32, Time: 172.929657 ms.
CUBLAS: Matrix A 2048x2048, Matrix B 2048x2048, Time: 128.310303 ms.

MatrixMul results by GPU and CUBLAS are same. Acceleration ratio is 1.347746
```

```
!./task2 32 4096 4096 4096

host_a[0] = 95.407001, host_b[0] = 9.680780
Launching kernel with dimGrid(128, 128) and dimBlock(32, 32)
GPU: Matrix A 4096x4096, Matrix B 4096x4096, Block_size:32, Time: 765.797424 ms.
CUBLAS: Matrix A 4096x4096, Matrix B 4096x4096, Time: 551.941956 ms.

MatrixMul results by GPU and CUBLAS are same. Acceleration ratio is 1.387460
```

```
!./task2 32 8192 8192 8192

host_a[0] = 10.147438, host_b[0] = 18.723300
Launching kernel with dimGrid(256, 256) and dimBlock(32, 32)
GPU: Matrix A 8192x8192, Matrix B 8192x8192, Block_size:32, Time: 6107.385742 ms.
CUBLAS: Matrix A 8192x8192, Matrix B 8192x8192, Time: 4339.895020 ms.

MatrixMul results by GPU and CUBLAS are same. Acceleration ratio is 1.407266
```

Submitting Assignment

Submit your assignment file through D2L using the appropriate link.
The submission must include the ***source code***, and ***a report describe your code logic***. ***Output screenshot of your code*** and ***dot-line-chart (e.g., Chart 1) & Tables*** should be included in the report.