EE 451
Spring 2020
Programming Homework 1
Assigned: January 23, 2020
Due: January 29, 2020, before 11:59 pm
Total Points: 50

1 Environment Setup

- Any Linux distribution with C/C++ compiler.

2 Matrix Multiplication [50 points]

1. **Naive Matrix Multiplication [10 points]**
   Implement the naive matrix multiplication to multiply two matrices of dimension $4K \times 4K$. Report the execution time (in ns) and performance (in $FLOPS$).

```
for i = 1 to n
    for j = 1 to n
        for k = 1 to n
            C(i,j) = C(i,j) + A(i,k) × B(k,j)
```

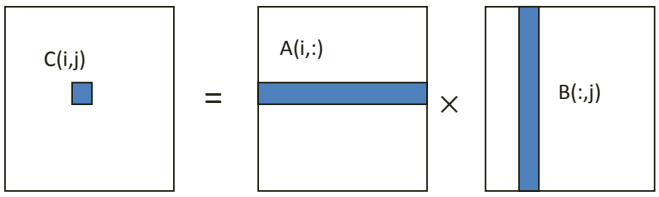

Figure 1: Naive Matrix Multiplication

2. **Block Matrix Multiplication [20 points]**
   A matrix can be viewed to be constructed by blocks. Assume an $n \times n$ matrix is partitioned into $m \times m$ blocks and each block is a $b \times b$ matrix ,where $b = \frac{n}{m}$ is called the block size. For example, in Figure 2, a 4×4 ($n = 4$) matrix can be viewed as a 2×2 ($m = 2$) block matrix; each block is a 2×2 ($b = 2$) matrix.

   Block matrix multiplication computes the output block by block. Figure 3 depicts the principle of Block Matrix Multiplication. Here, the algorithm has m^3

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{bmatrix} \quad \Rightarrow \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{bmatrix}.$$

$$\mathbf{P}_{11} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \mathbf{P}_{12} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}, \mathbf{P}_{21} = \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}, \mathbf{P}_{22} = \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix}.$$

Figure 2: Block Matrix

iterations as oppose to n^3 iterations for Naive Matrix Multiplication; the computation of each iteration is based on blocks rather than a single element for Naive Matrix Multiplication.

```
for i = 1 to m
    for j = 1 to m
        for k = 1 to m  //do a matrix multi. on blocks
            C'(i,j) = C'(i,j) + A'(i,k) * B'(k,j)
```

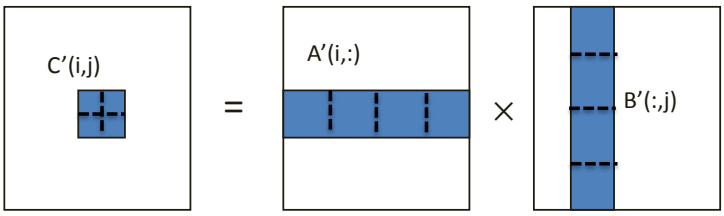

Figure 3: Block Matrix Multiplication

Use block matrix multiplication to solve the previous problem with block size $b = 4, 8, 16$ respectively. Report the execution time (in ns) and performance (in $FLOPS$) respectively. Compare the result against the result obtained by using naive matrix multiplication. Report your observation.

2.1 Examples

- A matrix-vector multiplication example code, "example.c", which multiplies a matrix with a vector, is provided. To compile "example.c", type: 'gcc -O3 -o run example.c'. You can refer to it for some useful functions.
- A matrix-matrix multiplication example code framework "problem1.c"; you can either insert your codes into it or write the whole program by yourself. To compile "problem1.c", type: 'gcc -O3 -o run problem1.c'.
- To run the compiled program, type: './run'.
- Initialize the matrices: $A[i][j] = i, B[i][j] = i + j, C[i][j] = 0$, for any $0 \leq i, j < n$. After the computation, print out the value of $C[100][100]$. Refer to "problem1.c".

2.2 Submission

- Two .c/.cpp files: 'problem1a.c' for naive matrix multiplication; 'problem1b.c' for block matrix multiplication. Make sure your program is runnable. (10+20 pts)
- Report. Write clearly how to compile and run your code. Screenshot of the execution time and performance. (20 pts)

3 Submission Instructions

You may discuss the algorithms. However, the programs have to be written individually. Submit the code and the report to Blackboard. Please make sure to include your name, student ID and the homework number.

References

[1] "Command Line Parameter Parsing,"
`http://www.codingunit.com/c-tutorial-command-line-parameter-parsing`

[2] "Using make and writing Makefiles,"
`http://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html`