EE 451: Parallel and Distributed Computation

PA1 — Spring 2021

Due date: Sunday 31st January 2021 11:59 PM

1. **Examples**

   - A matrix-vector multiplication example code, "example.c", which multiplies a matrix with a vector is provided. To compile "example.c",

     ```
     1 gcc -O3 -o run example.c
     ```

   - 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",

     ```
     1 gcc -O3 -o run example.c
     ```

   - To run the compiled program in hpc,

     ```
     1 srun -n1 ./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. **Naive Matrix Multiplication**

   Implement the naive matrix multiplication to multiply two matrices of dimension $4K \times 4K$. Report the execution time (in ns) and performance (in $FLOPS$).

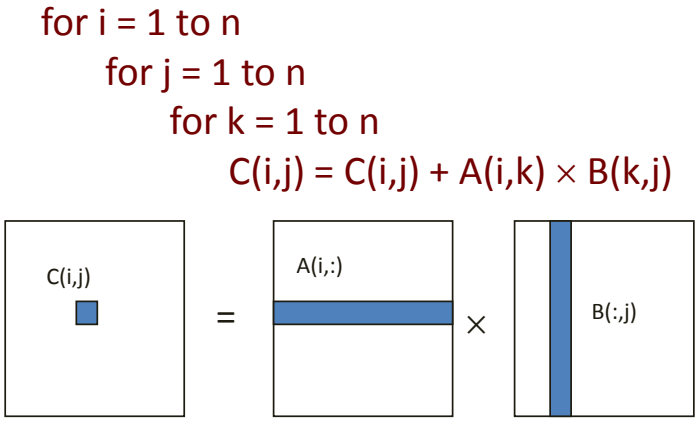

Figure 1: Naive Matrix Multiplication

3. **Block Matrix Multiplication**

   A matrix can be viewed to be constructed by bigger blocks. Assume an $n \times n$ matrix is

partitioned into $m \times m$ blocks and each block is a $b \times b$ matrix ,where $b = \frac{n}{m}$, b is called the block size. As shown in Figure 2, a 4×4 ($n = 4$) matrix can be viewed as a 2×2 ($m = 2$) block matrix; each block matrix is a 2×2 ($b = 2$) matrix.

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

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

Figure 2: Block Matrix

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

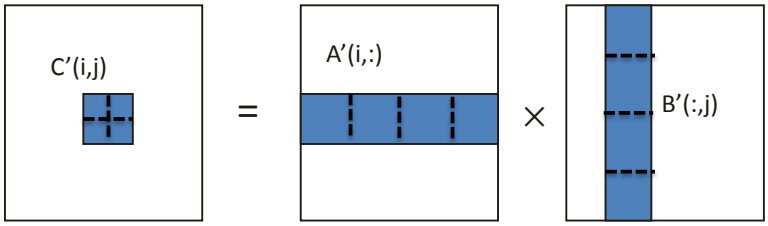

Figure 3: Block Matrix Multiplication

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.

4. **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)
- Screenshot of the execution time and performance on hpc. (20 pts)