EE/CSCI 451
Spring 2019
Programming Homework 1
Assigned: January 30, 2019
Due: February 8, 2019, before 11:59 pm
Total Points: 50

1 Login to HPC

- The host is: hpc-login**3**.usc.edu

- Username and password are the same as your email account

- **Do not** run your program in the login node. After login, use the 'srun' command to run your program on a remote node. For example:
  srun -n1 ./run

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 $FLOPS$).

   Figure 1: Naive Matrix Multiplication

2. **Block Matrix Multiplication [15 points]**
   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.
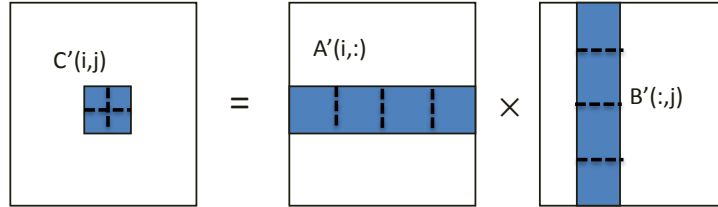

Figure 3: Block Matrix Multiplication

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 in hpc, type: '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".

2.2 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 on hpc. (20 pts)

3 Submission Instructions

You may discuss the algorithms. However, the programs have to be written individually. Submit the code and the report via ee451spring2019@gmail.com. Please make sure to include your name, student ID and the homework number in the PDF, and name your PDF file lastname_firstname_hw#.

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`