EE/CSCI 451
Spring 2015
Programming Homework 1

Assigned: January 30, 2015
Due: February 8, 2015, before 11:59 pm, submit via blackboard
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 'qsub' command to reserve a node. For example:
qsub -I -X -d. -l nodes=1:ppn=16:gpu, walltime=02:00:00

2 Matrix Multiplication [25 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)
```

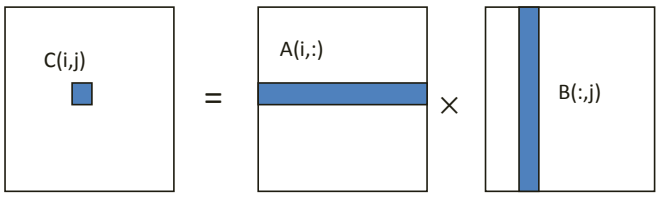

Figure 1: Naive Matrix Multiplication

2. **Block Matrix Multiplication [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.

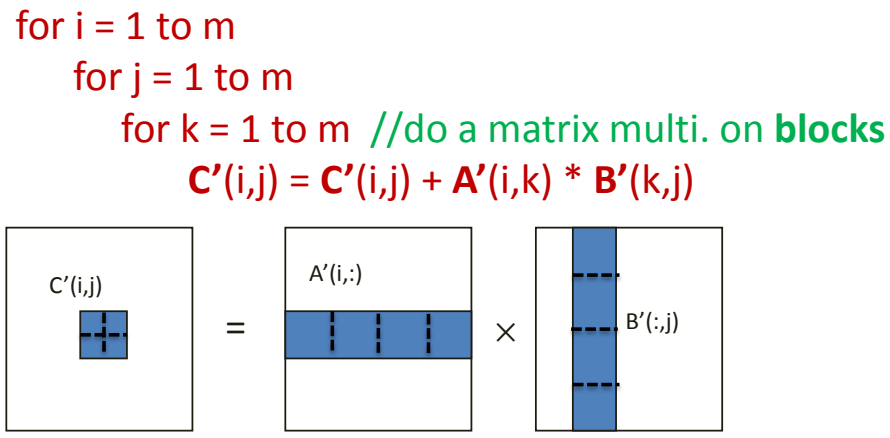

Figure 3: Block Matrix Multiplication

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.

3. Submission

- An example program, "example.c", which multiplies a matrix with a vector is provided. You can refer to it for some useful functions.
- A frame, "problem1.c" is also given; you can either insert your codes into it or write the whole program by yourself. To compile "problem1.c", type: 'gcc -lrt -o run problem1.c'; to run it, type: './run'.
- Initialize the matrices in this way: $A[i][j] = i, B[i][j] = i + j, C[i][j] = 0$, for any $0 \leq i, j < n$. After the computation, print out the value of $C[100][100]$. Refer to "problem1.c".
- For block matrix multiplication, parse the block size b as a command line parameter. Hence to run the $b = 8$ case, we can type: './run 8'. To realize so, see [1].
- Submit two .c/.cpp files: 'p1a.c' for naive matrix multiplication; 'p1b.c' for block matrix multiplication.

3 K-Means algorithm [25 points]

K-means [2] is a clustering algorithm which is widely used in signal processing, machine learning and data mining. In this problem, you will implement the K-Means algorithm to cluster a matrix into K clusters. K-Means algorithm has the following steps:

1. Initialize a mean value for each cluster.
2. For each data element, compute its 'distance' to the mean value of each cluster. Assign it to the 'closest' cluster.
3. After each data element is assigned to a cluster, recompute the mean value of each cluster.
4. Check convergence; if the algorithm converges, replace the value of each data with the mean value of the cluster which the data belongs to, then terminate the algorithm; otherwise, go to Step 2.

In this problem, the input data is a 800×800 matrix which is stored in the 'input.raw'; the value of each matrix element ranges from 0 to 255. Thus, the matrix can be displayed as an image shown in Figure 4. We will have 4 clusters (K=4). The initial mean values for the clusters are 0, 85, 170 and 255, respectively. To simplify the implementation, you do not need check the convergence; run 30 iterations (Step 2 and 3) serially then terminate the algorithm and output the matrix into the file named 'output.raw'. You can refer to the given program 'problem2.c' to read input file and write output file. It copies the input matrix to the out file directly without any modification.

Submit your 'C/C++' program and the report. In the report, you need include the execution time for the 50 iterations (excluding the read/write time) and the corresponding image of the output matrix. You can display the output image, 'output.raw', using the imageJ [4] software or the given Matlab script file, 'show_raw.m'.

Figure 4: Input matrix

4 Submission

You may discuss the algorithms. However, the programs have to be written individually. Submit the code and the report via **Blackboard**. Your program should be written in C or C++. Make sure your program is runnable. Write clearly how to compile and run your code in the report as well. It is recommended to include a Makefile [3] along with your submission.

If your program has error when we compile or run your program, you will lose at least 50% of credits.

References

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

[2] "k-means clustering,"
`http://en.wikipedia.org/wiki/K-means_clustering`

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

[4] "imageJ,"
`http://rsb.info.nih.gov/ij/download.html`