Operating Systems–2: Spring 2024
Programming Assignment1: Efficient Matrix Squaring
Submission Deadline: 29th January 2024, 9:00 pm

Goal:- This assignment aims to perform parallel matrix multiplication in C++.

Details:- Consider a square matrix A. The goal of the problem is to find the square of matrix A in parallel. Assume that the matrices are given in row-major order.

The main program will read **N, K,** and the **matrix A** from an input file. Here, the variable N represents the number of rows of A; K represents the number of threads.

To multiply two matrices, C = A * B, the result entry $C_{(i,j)}$ is computed by taking the dot product of the ith row of A and the jth column of B: $C_{(i,j)} = \sum_{k=0}^{N-1} A_{(i,k)} B_{(k,j)}$. In our case, the matrix A and B are the same.

We can divide the work by creating one thread to compute a row or a collection of rows in the C matrix and then executing those threads on different processors in parallel.

One can design (at least) two techniques for computing the C matrix in parallel:

- Chunk: The C matrix is divided into chunks of size $p = N/K$. Then thread1 will compute the values of the rows 1 to p of C; thread2 will compute the values of the rows p + 1 to 2 * p; thread3 will compute the values of the rows 2*p + 1 to 3*p and so on. Thus, the thread th_i will be responsible for computing the rows corresponding to chunk i.

- Mixed: Here, the rows of the C matrix are evenly distributed among the threads. Thread1 will be responsible for the following rows of the C matrix: 1, k+1, 2*k+1 …..,. Similarly, thread2 will be responsible for the following rows of the C matrix: 2, k+2, 2*k+2 …..,. This pattern continues for all the threads.

One can develop other designs for sharing the computation of the C matrix among the various threads, such as mixed chunks, etc.

Input File:- As mentioned above, the input will consist of three parameters: **N, K, and the matrix A**. Please name the input file as *inp.txt*

Output File:- Your program should output the following in an outfile file named *out.txt: (a) The resulting square matrix* (b) *Time taken to compute the square matrix*.

Report Details:- As part of this assignment, you have to prepare a report describing your program's low-level design. Further, you have to measure the performance of the algorithms. You will have to create two plots to measure the time taken:

1. Time vs. Size, N: In this graph, the y-axis will show the time taken to compute the square matrix using both the approaches mentioned above: chunks and mixed. The x-axis will be the

values of N (size on input matrix) varying from 16 to 2048 (size of the matrix will vary as 16*16, 32*32, 64*64, 128*128….) in the power of 2. Please have K fixed at 8 for all these experiments.

2. Time vs. Number of threads, K: Like the previous graph, the y-axis will show the time taken to do the matrix squaring using both approaches mentioned above: chunks and mixed (similar to experiment 1). The x-axis will be the values of K, the number of threads varying from 2 to 32 (in powers of 2, *i.e.*, 1,2,4,8,16,32). All experiments must have N fixed at 1024.

To handle temporary outliers, ensure that each point in the above plots is averaged over 5 times. Thus you will have run your experiment 5 times for each point on the x-axis.

Note that both the plots will have two curves, as mentioned above: chunks and mixed.

You have to write the observation you gained based on the plots described above and mention any anomalies in the report.

Extra Credit:- As mentioned above, one can develop another approach for the parallel computation of the C matrix based on the idea of mixed chunks or some other technique. Please work on developing such a different technique.

Note that it is important that your solution avoids any forms of locks or synchronization techniques. You will study these techniques in later chapters.

To show the effectiveness of this technique, you will have to perform the experiments and generate the graph plots 1 and 2 mentioned above. In this case, the plots will contain three curves: chunks, mixed, and mixed-chunks (or any other appropriate technique developed by you).

This extra-credit option will be evaluated only if you have implemented it. You will not be given extra credit if you describe this idea without implementing it.

Deliverables:-

You will have to submit the following as a part of this assignment:

1. A report describing the following:
(a) Low-level design of your program. You must explain the implementation of the techniques chunks and mixed. And in case, you implemented the extra-credit option, then you will have to explain the details of the working of that technique as well.

(b) The graph plots are described above, as well as its analysis. Please name this report file as Assgn1_Report-<Roll No>.pdf

2. Prepare a README file that contains the instructions on how to execute your submitted file. The file should be named Assgn1_ReadMe-<Roll No>.txt

3. Name the source code file in the following format: Assgn1_Src-<Roll No>.cpp

4. Combine the source code, report, input file, and README as a zip archive file and name it Assgn1-<Roll No>.zip. Upload this zip file.

Please see the instructions given above before uploading your file. **Please follow this naming convention.** Your assignment will **NOT be evaluated** if there is any deviation from the instructions posted there, especially with regard to the naming convention.

Marking Scheme:

The evaluation scheme for this assignment will be as follows:

S. No	Section	Marks
1.	Program Design & Report	50
2.	Program Execution	40
3.	Code Indentation and Documentation	10
4.	Extra Credit	30
Total		**130/100**

It can be seen that with the extra credit, one can get 130 marks out of 100. Please note that, as mentioned above, the extra-credit portion will be evaluated only if the results are also shown. Otherwise, no marks for extra credit will be awarded.

Please keep in mind that all the submissions are subjected to plagiarism checks.