LAB 1

DUE DATE: Fri Sep 29 5pm (upload to polylearn) Individual Project

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Write a program in C to perform Matrix Matrix multiplication. The requirements for the program:

1. Modify the program lab1.c which contains an implementation of matrix multiplication in C so it executes with multiple threads.

Note : compile: gcc lab1.c -o mm -lpthread

run: ./mm 2

Note: Matrix multiplication can be done very simple as shown in Fig 1, this algorithm is O(n3), and this algorithm is enough for this lab. NOTE: MAKE SURE YOUR ALGORITHM COMPILES AND RUNS ON THE LAB COMPUTERS.

If you are curious and want 10 extra credit points you can also try to implement Strassen algorithm O(n2.807) , although to really see the speed up using Strassen your matrix must be at least n>100, these algorithm appears in several libraries (BLAS, etc).

Matrix multiplication is still an area of research, most important problems do require multiplication -linear transformations for image/video processing, solving systems of linear equations, rank algorithms,…- of huge matrices (don’t even fit in memory) and getting an algorithm that perform faster is paramount.

|  |  |
| --- | --- |
|  | int row, col, k;  float Pvalue=0;  for (row=0; row<Hight; row++){  for(col=0; col<Width; col++) {  Pvalue=0;  for(k=0; k<Width; k++){  Pvalue+=fa[row\*Width+k]\*fb[k\*Width+col];  }  fc[row\*Width+col]=Pvalue;  }  } |

{\displaystyle \mathbf {A} \mathbf {B} ={\begin{pmatrix}a&b&c\\x&y&z\end{pmatrix}}{\begin{pmatrix}\alpha &\rho \\\beta &\sigma \\\gamma &\tau \\\end{pmatrix}}={\begin{pmatrix}a\alpha +b\beta +c\gamma &a\rho +b\sigma +c\tau \\x\alpha +y\beta +z\gamma &x\rho +y\sigma +z\tau \\\end{pmatrix}}\,,}

Fig 1. Matrix Multiplication

**What to turn in**

Upload a report in PDF with the following:

1. Your name
2. Explanation of the matrix multiplication code used if different than the one provided in Fig 1
3. Table with execution times (use same time functions as provided for correctness and consistency)

|  |  |
| --- | --- |
| Matrix A and B Size | Execution Time |
| A.512, 512 |  |
| B. 1024\*1024 |  |
| C. 8192\*8192 |  |
| D. Extra Credit: Rectangular Matrix  first(900\*100), second (100,9000) |  |

1. Table with the execution time for the multithreaded matrix multiplication algorithm

|  |  |  |  |
| --- | --- | --- | --- |
| Matrix A and B Size | Execution Time 2 Threads | Execution Time 4 Threads | Execution Time 8 Threads |
| A. |  |  |  |
| B. |  |  |  |
| C. |  |  |  |
| D. |  |  |  |

1. Appendix with your code, clean and properly commented (I may ask for a demo of your code working on the computer labs if I don’t understand how your code is able to run).