# Project 7 Report

Yuhao Ye

UIN: 529006730

CSCE 312 – 599

## The expectation of different implementation before the running the code.

1. Jik (matrixMult1)(mix of column-major and row-major)
   This implementation of jik is basically the combination of row-wise and column-wise.
   This function access the data in A by Row-wise and data in B by Column-wise. Since traverse by column will spent more time. Hence, my assumption is that the runtime of this Function will be a little slower than pure row-major traverse and faster than pure column-major traverse

2. Ikj (matrixMult2)(row-major)
   This function traverse every elements in A and B by row-major. For the reason that the value stored in the vector by row-major order. Hence traverse through column in one row will take advantage of spatial locality which will increase the possibility of write and read hit in cache. Hence, I assumed that matrixMult2 function would run faster than matrixMult1 and matrixMult3(column-major).

3. Jki (matrixMult3)column-major
   Since this function traverse the element by column-major. Hence, there is high possibility that The read and write miss will occur when CPU is accessing the data from cache. Hence this function suppose to be the slowest among other function.

4. Blocking Cache
   This function will take advantage of temporal locality o inner loops.
   Since the data in the block will be repeated traversing for several times hence. This is why accessing data in A and B take the advantage of temporal locality. What's more the data will be write in matrix C by row traverse. The result of calculation will written in succession. Hence, writing data in C take advantage of spatial locality.
   So my assumption is Blocking Cache will me the fastest among these 4 functions.

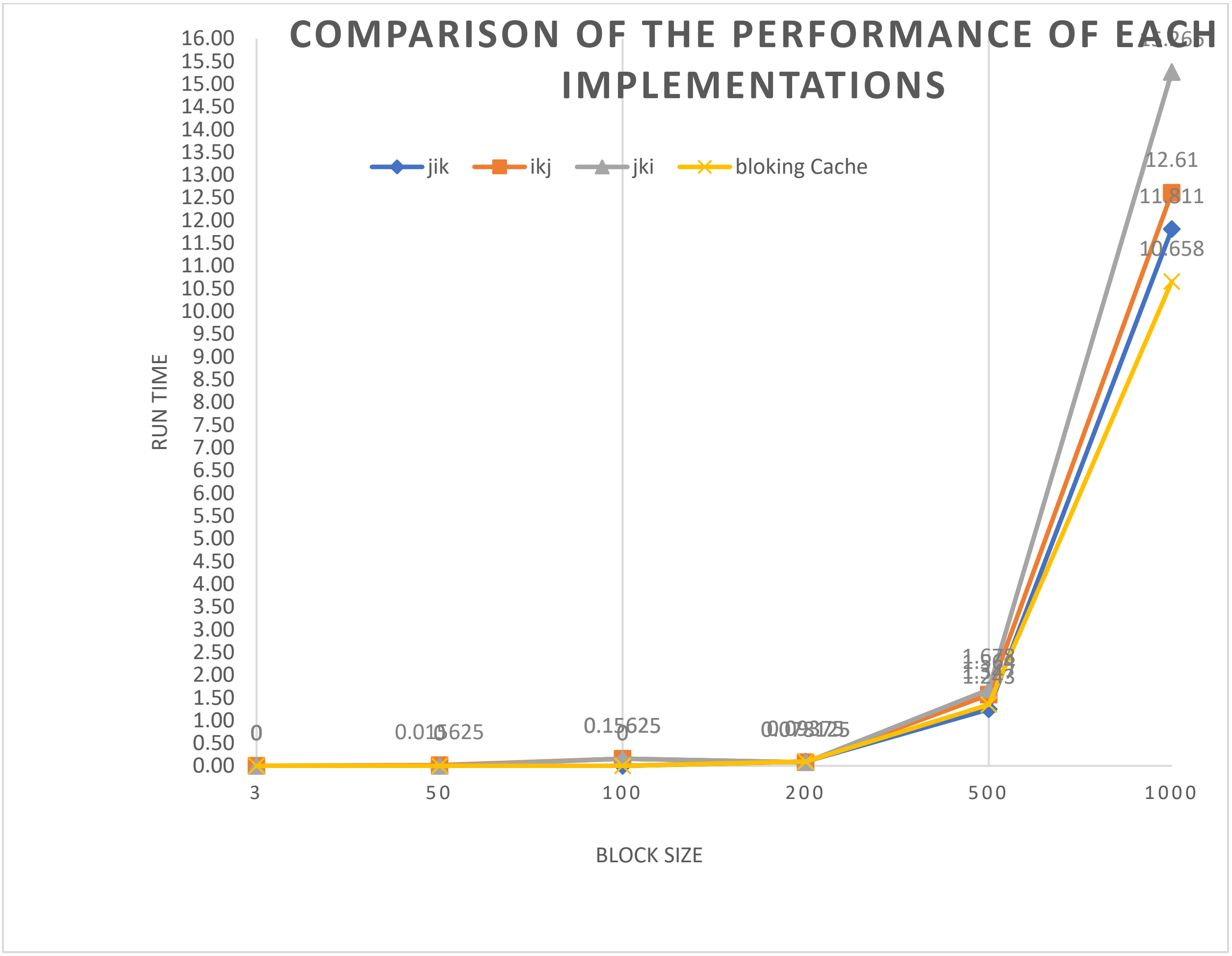
## Analysis of the result:

**In this picture, jik represent mix of column and row Matrix Multiplication**

**Ikj represents implementation of row-major Matrix Multiplication**

**Jki represents implementation of column-major Matrix Multiplication:**

**Running code on  gcc compiler on Windows10 terminal.**



Generally , the performance of the function matrixMult2, matrixMult3, Blocking Cache are in line with expectations. From the graph above, we found that when block size = 1000. The runtime of jki > ikj > blocking Cache. However, matrixMult1 is faster than matrixMult2, which is supposed to be slower than matrixMult2.