School of Computer Science Engineering and Technology

Course- BTech
Course Code- ECSE302L

Type- Core
Course Name- HPC Lab (High-Performance Computing Lab)

Year- 2022
Date- 11-04-2022

Semester- Even
Batch- NA

Lab Assignment # 9

The goal of this assignment is to understand how memory access patterns affect performance, using matrix multiplication as an example. This program will run on a single core.

Given a matrix A of dimensions m×q and a matrix B of dimensions q×n, the product C = AB has dimensions m × n with the entries defined as

$$C_{ij} = \sum_{k=1}^{q} A_{ik} B_{kj}$$

To do:

1. Write a program called matmult that can be run using command line parameters as follows:
   matmult m q n filenameA filenameB filenameC
   where filenameA and filenameB are the names of two matrix input files, and filenameC is the name of a matrix output file to be created. The positive integers m, q, and n specify the dimensions of the input matrices. The format of the matrix files is a list of all the entries of the matrix (see the test examples below), in column-major order. Your program should compute the product of the two input matrices and write the result to the output file. Use double precision for all your matrices and computations.

2. Your code should also be able to operate by calling it with no files, i.e., matmult m q n
   In this case, your code should generate random entries for the two input matrices, and then perform the multiplication. This is useful when you are only interested in the performance of matrix multiplication for large matrices, and you don't want to bother with reading and writing files. Check that your code is correct by testing it with the files matrixA, matrixB, and matrixC, corresponding to AB = C. For this test data, m = 3, q = 4, n = 5. A set of larger test matrices and a test matrix generator are also provided.

3. Measure the execution time of matrix multiplication (not including reading and writing files) for different sizes of matrices. For simplicity, choose m = k = n and n = 500, 1000, 2000, . . . to as large a problem your computer can handle. Plot the timings as a function of n. This is your baseline performance.