V Shree Vishnu

CS19B045

CS2610

Assignment 5

PERFORMANCE ANALYSIS:

Scenario 1 for row major and row major multiplication

Scenario #	1
Event	Value
CPU-Cycles	3,54,25,60,346
Instructions	7,68,13,87,755
Branches	1,11,50,11,913
Cache - references	19,69,32,139
Cache - misses	15,30,218
L1-dcache-loads	3,25,06,06,232
L1-dcache-load-misses	1,28,07,95,868
dTLB-loads	3,27,15,37,751
dTLB-load-misses	19,07,035
LLC-loads	7,58,64,930
LLC-load-misses	3,10,057
Run Time	8,52,362

Scenario 2 for row major and column major multiplication

Scenario #	2
Event	Value
CPU-Cycles	1,22,01,31,038
Instructions	4,21,75,73,372
Branches	31,02,54,606
Cache - references	15,31,79,245
Cache - misses	64,67,021
L1-dcache-loads	1,10,11,10,386
L1-dcache-load-misses	7,03,05,796
dTLB-loads	1,12,66,23,317
dTLB-load-misses	21,79,285
LLC-loads	57,12,296
LLC-load-misses	1,32,096
Run Time	2,84,249

REASONING:

The L1 d-cache misses in Scenario 1 is much higher than Scenario 2 because for calculating an element in matrix C, we multiply one row of matrix A with one column of matrix B. These elements of B that belong to the same column are not adjacent to each other in cache. But in Scenario 2, we multiply one row of A with one row of B and adjacent elements of row are also adjacent in the cache memory. Hence the number of cache misses is much lesser in Scenario 2.

SYSTEM CONFIGURATION:

Processor	Intel Core i5-9300H
Processor Frequency	2.40 GHz(base), 4.10 GHz(Turbo boosted)
Number of Cores	4

Cache Block Size = 64 Bytes

Hierarchy:

L1 Cache (Write-back): 256KB (I - 128KB, D - 128KB)

4x32KB for each of i-cache and d-cache

8-way set associative

L2 Cache (Write-back): 1MB

4x256KB

4-way set associative

L3 Cache (Write-back): 8MB

4x2MB

16-way set associative

Operating System: Ubuntu 20.04.2 LTS

The C code for row major multiplication is file *Row.c* and column major multiplication is *Column.c.*

The programs were compiled using O3 optimization and the bash code for compiling that way is given as *compile.sh.*

The bash script for analysing using perf is given as *analyse.sh.* The corresponding terminal outputs for each Scenario is in *terminal-out-row.txt* and *terminal-out-column.txt.*

PROGRAM:

Both the programs have O(n^3) complexity.

Compiled using ./compile.sh.

Analyse with perf using ./analysis.

The tag -r <n> is used to get the average report for n runs of perf tool.