**20BCE1550**

**Samridh Anand Paatni**

**CSE4001 Lab 07**

**Profiling**

**Matrix Multiplication**

Nested parallelism is not supported in ompP, so the ‘collapse’ keyword has been used.

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

typedef int \*\* matrix;

#define N 1100 // the size of the matrices

void multiply(matrix A, matrix B, matrix ans, int numThreads) {

int i = 0, j = 0, k = 0;

#pragma omp parallel shared(A, B, ans) private(i, j, k) num\_threads(numThreads)

{

#pragma omp for collapse(3)

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

for (k = 0; k < N; k++) {

ans[i][j] += A[i][k] \* B[k][j];

}

}

}

}

}

int main(int argc, char \*argv[]) {

int numThreads = atoi(argv[1]);

matrix A, B, C;

A = (int \*\*)calloc(N, sizeof(int\*));

B = (int \*\*)calloc(N, sizeof(int\*));

C = (int \*\*)calloc(N, sizeof(int\*));

for (int i = 0; i < N; i++) {

A[i] = (int \*) calloc(N, sizeof(int));

B[i] = (int \*) calloc(N, sizeof(int));

C[i] = (int \*) calloc(N, sizeof(int));

for (int j = 0; j < N; j++) {

A[i][j] = rand() % 10;

B[i][j] = rand() % 10;

C[i][j] = 0;

}

}

double t = omp\_get\_wtime();

multiply(A, B, C, numThreads);

t = omp\_get\_wtime() - t;

printf("took %f seconds\n", t);

for (int i = 0; i < N; i++) {

free(A[i]);

free(B[i]);

}

free(A);

free(B);

free(C);

printf("\n");

return 0;

}

**OmpP Profiler Output:**







