

Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Class: Final Year (Computer Science and Engineering)

Year: 2024-25

Semester: 1

Course: High Performance Computing Lab

Practical No. 6

Exam Seat No: 21510111

Title of practical: Implementation of OpenMP programs.

Implement following Programs using OpenMP with C:

1. Implementation of Matrix-Matrix Multiplication.
2. Implementation of Matrix-vector Multiplication.

Problem Statement 1: Implementation of Matrix-Matrix Multiplication.

Screenshots:

```
1.c x 2.c x
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 // Function to multiply two matrices A and B serially, store result in C
6 void serialMatrixMultiply(int **A, int **B, int **C, int rowA, int colA, int colB) {
7     for (int i = 0; i < rowA; i++) {
8         for (int j = 0; j < colB; j++) {
9             C[i][j] = 0; // Initialize C[i][j]
10            for (int k = 0; k < colA; k++) {
11                C[i][j] += A[i][k] * B[k][j];
12            }
13        }
14    }
15 }
16
17 // Function to multiply two matrices A and B in parallel using OpenMP, store result in C
18 void parallelMatrixMultiply(int **A, int **B, int **C, int rowA, int colA, int colB) {
19     #pragma omp parallel for collapse(2)
20     for (int i = 0; i < rowA; i++) {
21         for (int j = 0; j < colB; j++) {
22             C[i][j] = 0; // Initialize C[i][j]
23             for (int k = 0; k < colA; k++) {
24                 C[i][j] += A[i][k] * B[k][j];
25             }
26         }
27     }
28 }
29
30 // Function to allocate a matrix dynamically
31 int **allocateMatrix(int rows, int cols) {
32     int **matrix = (int **)malloc(rows * sizeof(int *));
33     for (int i = 0; i < rows; i++) {
34         matrix[i] = (int *)malloc(cols * sizeof(int));
35     }
36     return matrix;
37 }
38
39 // Function to initialize a matrix with random integers
40 void initializeMatrix(int **matrix, int rows, int cols) {
41     for (int i = 0; i < rows; i++) {
42         for (int j = 0; j < cols; j++) {
43             matrix[i][j] = rand() % 10; // Random number between 0 and 9
44         }
45     }
46 }
47
```

```
48 // Function to print a matrix
49 void printMatrix(int **matrix, int rows, int cols) {
50     for (int i = 0; i < rows; i++) {
51         for (int j = 0; j < cols; j++) {
52             printf("%d ", matrix[i][j]);
53         }
54         printf("\n");
55     }
56 }
57
58 // Main function
59 int main() {
60     int rowA, colA, rowB, colB;
61     double start, end;
62
63     // Input matrix dimensions
64     printf("Enter rows and columns of matrix A: ");
65     scanf("%d %d", &rowA, &colA);
66
67     printf("Enter rows and columns of matrix B: ");
68     scanf("%d %d", &rowB, &colB);
69
70     if (colA != rowB) {
71         printf("Matrix multiplication not possible with given dimensions!\n");
72         return -1;
73     }
74
75     // Allocate and initialize matrices
76     int **A = allocateMatrix(rowA, colA);
77     int **B = allocateMatrix(rowB, colB);
78     int **C_serial = allocateMatrix(rowA, colB); // For serial result
79     int **C_parallel = allocateMatrix(rowA, colB); // For parallel result
80
81     // Initialize A and B with random values
82     initializeMatrix(A, rowA, colA);
83     initializeMatrix(B, rowB, colB);
84
85     // Perform serial matrix multiplication
86     start = omp_get_wtime();
87     serialMatrixMultiply(A, B, C_serial, rowA, colA, colB);
88     end = omp_get_wtime();
89     printf("Serial Time: %f seconds\n", end - start);
90
91     // Perform parallel matrix multiplication using OpenMP
92     start = omp_get_wtime();
93     parallelMatrixMultiply(A, B, C_parallel, rowA, colA, colB);
94     end = omp_get_wtime();
95     printf("Parallel Time: %f seconds\n", end - start);
96 }
```

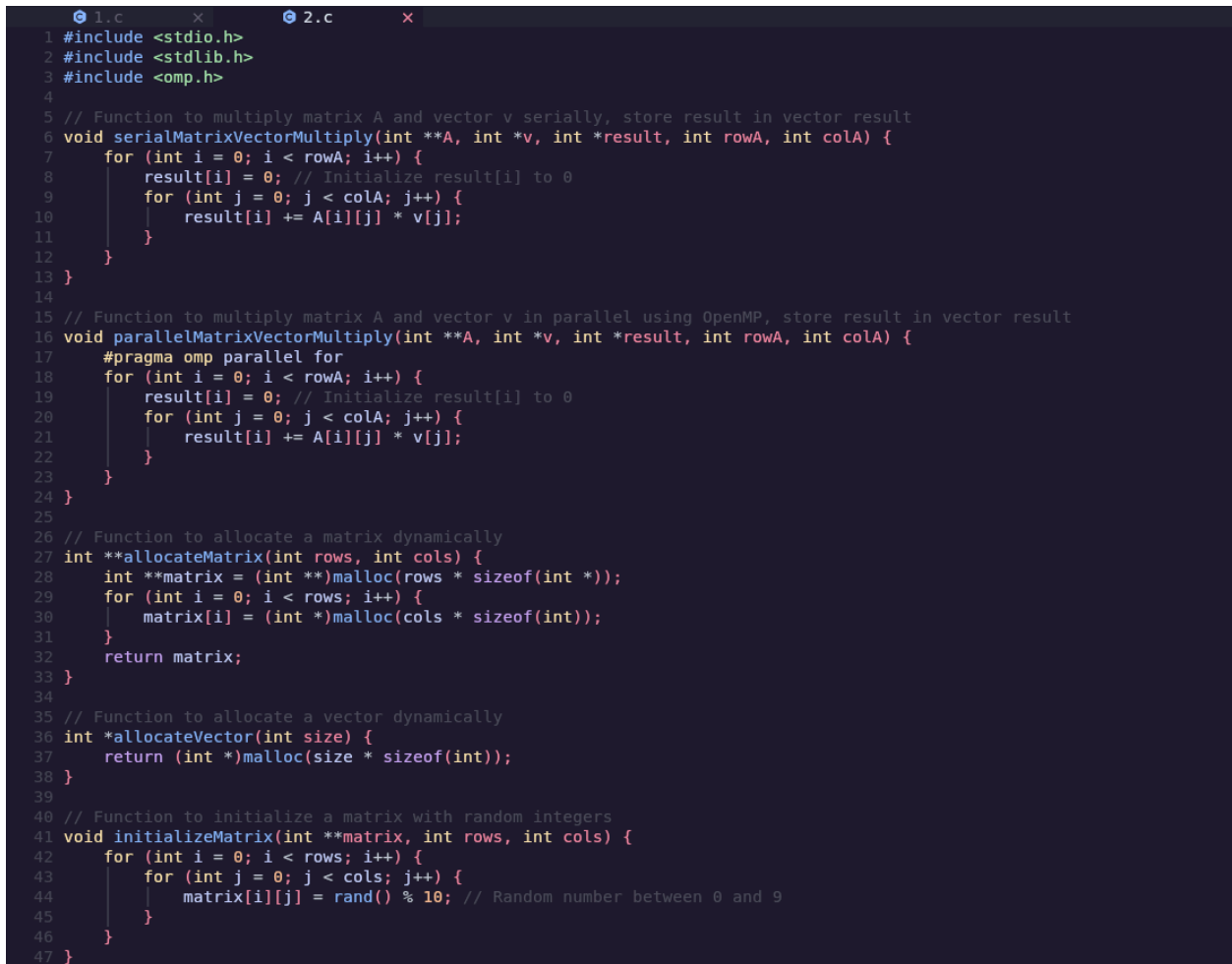
Information:

Analysis:

```
[main][~/acad/hpc_lab/as6]$ gcc -fopenmp 1.c -o 1 && ./1
Enter rows and columns of matrix A: 500 500
Enter rows and columns of matrix B: 500 500
Serial Time: 0.977917 seconds
Parallel Time: 0.261746 seconds
Results are identical.
[main][~/acad/hpc_lab/as6]$
```

Problem Statement 2:

Screenshots:



```
1.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 // Function to multiply matrix A and vector v serially, store result in vector result
6 void serialMatrixVectorMultiply(int **A, int *v, int *result, int rowA, int colA) {
7     for (int i = 0; i < rowA; i++) {
8         result[i] = 0; // Initialize result[i] to 0
9         for (int j = 0; j < colA; j++) {
10             result[i] += A[i][j] * v[j];
11         }
12     }
13 }
14
15 // Function to multiply matrix A and vector v in parallel using OpenMP, store result in vector result
16 void parallelMatrixVectorMultiply(int **A, int *v, int *result, int rowA, int colA) {
17     #pragma omp parallel for
18     for (int i = 0; i < rowA; i++) {
19         result[i] = 0; // Initialize result[i] to 0
20         for (int j = 0; j < colA; j++) {
21             result[i] += A[i][j] * v[j];
22         }
23     }
24 }
25
26 // Function to allocate a matrix dynamically
27 int **allocateMatrix(int rows, int cols) {
28     int **matrix = (int **)malloc(rows * sizeof(int *));
29     for (int i = 0; i < rows; i++) {
30         matrix[i] = (int *)malloc(cols * sizeof(int));
31     }
32     return matrix;
33 }
34
35 // Function to allocate a vector dynamically
36 int *allocateVector(int size) {
37     return (int *)malloc(size * sizeof(int));
38 }
39
40 // Function to initialize a matrix with random integers
41 void initializeMatrix(int **matrix, int rows, int cols) {
42     for (int i = 0; i < rows; i++) {
43         for (int j = 0; j < cols; j++) {
44             matrix[i][j] = rand() % 10; // Random number between 0 and 9
45         }
46     }
47 }
```

```
49 // Function to initialize a vector with random integers
50 void initializeVector(int *vector, int size) {
51     for (int i = 0; i < size; i++) {
52         vector[i] = rand() % 10; // Random number between 0 and 9
53     }
54 }
55
56 // Function to print a vector
57 void printVector(int *vector, int size) {
58     for (int i = 0; i < size; i++) {
59         printf("%d ", vector[i]);
60     }
61     printf("\n");
62 }
63
64 // Main function
65 int main() {
66     int rowA, colA;
67     double start, end;
68
69     // Input matrix dimensions
70     printf("Enter rows and columns of matrix A: ");
71     scanf("%d %d", &rowA, &colA);
72
73     // Allocate and initialize matrix A and vector v
74     int **A = allocateMatrix(rowA, colA);
75     int *v = allocateVector(colA);
76     int *result_serial = allocateVector(rowA); // For serial result
77     int *result_parallel = allocateVector(rowA); // For parallel result
78
79     // Initialize matrix A and vector v with random values
80     initializeMatrix(A, rowA, colA);
81     initializeVector(v, colA);
82
83     // Perform serial matrix-vector multiplication
84     start = omp_get_wtime();
85     serialMatrixVectorMultiply(A, v, result_serial, rowA, colA);
86     end = omp_get_wtime();
87     printf("Serial Time: %f seconds\n", end - start);
88
89     // Perform parallel matrix-vector multiplication using OpenMP
90     start = omp_get_wtime();
91     parallelMatrixVectorMultiply(A, v, result_parallel, rowA, colA);
92     end = omp_get_wtime();
93     printf("Parallel Time: %f seconds\n", end - start);
94 }
```

Information:

Analysis:

```
[main][~/acad/hpc_lab/as6]$ gcc -fopenmp 2.c -o 2 && ./2
Enter rows and columns of matrix A: 500 500
Serial Time: 0.001119 seconds
Parallel Time: 0.021906 seconds
Results are identical.
[main][~/acad/hpc_lab/as6]$ gcc -fopenmp 2.c -o 2 && ./2
Enter rows and columns of matrix A: 1000 1000
Serial Time: 0.004044 seconds
Parallel Time: 0.001282 seconds
Results are identical.
[main][~/acad/hpc_lab/as6]$ █
```

Github Link: <https://github.com/Sidd-77/hpc-lab/tree/main/as6>