Practical No 6

PRN: 22520005

Name: Aftab Imtiyaj Bhadgaonkar

Batch: B6

Course: High Performance Computing Lab

Q1) Implementation of Matrix-Matrix Multiplication.

Code(Sequential):

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void matrixMultiply(int **A, int **B, int **C, int N) {
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
C[i][j] = 0;
for (int k = 0; k < N; k++) {
C[i][j] += A[i][k] * B[k][j];
}
}
}
}
```

```
int main() {
int N;
printf("Enter the size of the matrix: ");
scanf("%d", &N);
int **A = (int **)malloc(N * sizeof(int *));
int **B = (int **)malloc(N * sizeof(int *));
int **C = (int **)malloc(N * sizeof(int *));
for (int i = 0; i < N; i++) {
A[i] = (int *)malloc(N * sizeof(int));
B[i] = (int *)malloc(N * sizeof(int));
C[i] = (int *)malloc(N * sizeof(int));
}
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
A[i][j] = 1;
B[i][j] = 1;
}
}
clock_t start = clock();
matrixMultiply(A, B, C, N);
clock_t end = clock();
double time_taken = (double)(end - start) / CLOCKS_PER_SEC;
printf("Time taken for sequential matrix multiplication: %f seconds\n",
time_taken);
for (int i = 0; i < N; i++) {
free(A[i]);
free(B[i]);
free(C[i]);
```

```
}
free(A);
free(B);
free(C);
return 0;
}
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ gcc -o seq1 seq1.c
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./seq1
Enter the size of the matrix: 100
Time taken for sequential matrix multiplication: 0.003332 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./seq1
Enter the size of the matrix: 500
Time taken for sequential matrix multiplication: 0.515632 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./seq1
Enter the size of the matrix: 1000
Time taken for sequential matrix multiplication: 4.366357 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$
```

Code(Parallel):

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
void matrixMultiplyParallel(int **A, int **B, int **C, int N) {
#pragma omp parallel for
for (int i = 0; i < N; ++i) {
for (int j = 0; j < N; ++j) {
C[i][j] = 0;
for (int k = 0; k < N; ++k) {
C[i][j] += A[i][k] * B[k][j];
}
}
}
}
int main() {
int N, num_threads;
printf("Enter the size of the matrix: ");
scanf("%d", &N);

printf("Enter the number of threads: ");
scanf("%d", &num_threads);

omp_set_num_threads(num_threads);

int **A = (int **)malloc(N * sizeof(int *));
int **B = (int **)malloc(N * sizeof(int *));
```

```
int **C = (int **)malloc(N * sizeof(int *));
for (int i = 0; i < N; i++) {
A[i] = (int *)malloc(N * sizeof(int));
B[i] = (int *)malloc(N * sizeof(int));
C[i] = (int *)malloc(N * sizeof(int));
}
for (int i = 0; i < N; ++i) {
for (int j = 0; j < N; ++j) {
A[i][j] = 1;
B[i][j] = 1;
}
}
double start_time = omp_get_wtime();
matrixMultiplyParallel(A, B, C, N);
double end_time = omp_get_wtime();
double time_taken = end_time - start_time;
printf("Time taken for parallel matrix multiplication: %f seconds\n",
time_taken);
for (int i = 0; i < N; i++) {
free(A[i]);
free(B[i]);
free(C[i]);
}
free(A);
free(B);
free(C);
return 0;
}
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ gcc -o para1 -fopenmp para1.c
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para1
Enter the size of the matrix: 100
Enter the number of threads: 4
Time taken for parallel matrix multiplication: 0.001061 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para1
Enter the size of the matrix: 500
Enter the number of threads: 4
Time taken for parallel matrix multiplication: 0.133618 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para1
Enter the size of the matrix: 1000
Enter the number of threads: 4
Time taken for parallel matrix multiplication: 1.150448 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para1
Enter the size of the matrix: 100
Enter the number of threads: 8
Time taken for parallel matrix multiplication: 0.001081 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para1
Enter the size of the matrix: 500
Enter the number of threads: 8
Time taken for parallel matrix multiplication: 0.127864 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para1
Enter the size of the matrix: 1000
Enter the number of threads: 8
Time taken for parallel matrix multiplication: 0.950593 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$
```

Analysis:

Time difference:

	Parallel	Sequential (1 thread)
Threads : 4		
Size : 100	0.001061	0.003332
Size : 500	0.133618	0.515632
Size : 1000	1.150448	4.366357
Threads : 8		
Size : 100	0.001081	
Size : 500	0.127864	
Size : 1000	0.950593	

Q2) Implementation of Matrix-vector Multiplication.

Code(Sequential):

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void matrixVectorMultiply(int **matrix, int *vector, int *result, int N) {
for (int i = 0; i < N; ++i) {
result[i] = 0;
for (int j = 0; j < N; ++j) {
result[i] += matrix[i][j] * vector[j];
}
}
}

int main() {
int N;
printf("Enter the size of the matrix and vector: ");
scanf("%d", &N);

int **matrix = (int **)malloc(N * sizeof(int *));
for (int i = 0; i < N; i++) {
matrix[i] = (int *)malloc(N * sizeof(int));
}

int *vector = (int *)malloc(N * sizeof(int));
int *result = (int *)malloc(N * sizeof(int));
```

```
for (int i = 0; i < N; ++i) {
for (int j = 0; j < N; ++j) {
matrix[i][j] = 1;
}
vector[i] = 1;
}

clock_t start = clock();

matrixVectorMultiply(matrix, vector, result, N);

clock_t end = clock();
double time_taken = (double)(end - start) / CLOCKS_PER_SEC;

printf("Time taken for sequential matrix-vector multiplication: %f seconds\n",
time_taken);

for (int i = 0; i < N; i++) {
free(matrix[i]);
}
free(matrix);
free(vector);
free(result);
return 0;
}
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ gcc -o seq2 seq2.c
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./seq2
Enter the size of the matrix and vector: 1000
Time taken for sequential matrix-vector multiplication: 0.002192 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./seq2
Enter the size of the matrix and vector: 2000
Time taken for sequential matrix-vector multiplication: 0.010761 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./seq2
Enter the size of the matrix and vector: 5000
Time taken for sequential matrix-vector multiplication: 0.057830 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$
```

Code(Parallel):

```
#include <stdio.h>

#include <stdlib.h>

#include <omp.h>


void matrixVectorMultiplyParallel(int **matrix, int *vector, int *result, int N) {

#pragma omp parallel for

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

result[i] = 0;

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

result[i] += matrix[i][j] * vector[j];

}

}

}


int main() {

int N, num_threads;

printf("Enter the size of the matrix and vector: ");
```

```
scanf("%d", &N);

printf("Enter the number of threads: ");
scanf("%d", &num_threads);

omp_set_num_threads(num_threads);

int **matrix = (int **)malloc(N * sizeof(int *));
for (int i = 0; i < N; i++) {
matrix[i] = (int *)malloc(N * sizeof(int));
}

int *vector = (int *)malloc(N * sizeof(int));
int *result = (int *)malloc(N * sizeof(int));

for (int i = 0; i < N; ++i) {
for (int j = 0; j < N; ++j) {
matrix[i][j] = 1;
}
vector[i] = 1;
}

double start_time = omp_get_wtime();

matrixVectorMultiplyParallel(matrix, vector, result, N);

double end_time = omp_get_wtime();
```

```
double time_taken = end_time - start_time;

printf("Time taken for parallel matrix-vector multiplication: %f seconds\n",
time_taken);

for (int i = 0; i < N; i++) {
free(matrix[i]);
}
free(matrix);
free(vector);
free(result);
return 0;
}
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ gcc -o para2 -fopenmp para2.c
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para2
Enter the size of the matrix and vector: 1000
Enter the number of threads: 4
Time taken for parallel matrix-vector multiplication: 0.000722 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para2
Enter the size of the matrix and vector: 2000
Enter the number of threads: 4
Time taken for parallel matrix-vector multiplication: 0.003028 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para2
Enter the size of the matrix and vector: 5000
Enter the number of threads: 4
Time taken for parallel matrix-vector multiplication: 0.015286 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para2
Enter the size of the matrix and vector: 1000
Enter the number of threads: 8
Time taken for parallel matrix-vector multiplication: 0.000833 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para2
Enter the size of the matrix and vector: 2000
Enter the number of threads: 8
Time taken for parallel matrix-vector multiplication: 0.002546 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$ ./para2
Enter the size of the matrix and vector: 5000
Enter the number of threads: 8
Time taken for parallel matrix-vector multiplication: 0.014830 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assg6$
```

Analysis:

Time difference:

	Parallel	Sequential (1 thread)
Threads : 4		
Size : 1000	0.000722	0.002192
Size : 2000	0.003028	0.010761
Size : 5000	1.015286	0.057830
Threads : 8		
Size : 1000	0.000833	
Size : 2000	0.002546	
Size : 5000	0.014830	