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:

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:

Code:

```
#include <iostream>
#include <omp.h>
#include <vector>
#include <chrono>

using namespace std;
using namespace std::chrono;

// Function for Matrix-Matrix Multiplication (sequential)
void matrixMatrixMultiplicationSeq(vector<vector<int>>& A,
vector<vector<int>>& B, vector<vector<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];
}
}
}
}
```

```
// Function for Matrix-Matrix Multiplication (parallel)
void matrixMatrixMultiplicationParallel(vector<vector<int>>& A,
vector<vector<int>>& B, vector<vector<int>>& C, int N) {
#pragma omp parallel for collapse(2)
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 = 100; // Size of the matrix

// Initialize matrices
vector<vector<int>> A(N, vector<int>(N, 1));
vector<vector<int>> B(N, vector<int>(N, 1));
vector<vector<int>> C(N, vector<int>(N, 0));

// Matrix-Matrix Multiplication - Sequential
auto start = high_resolution_clock::now();
matrixMatrixMultiplicationSeq(A, B, C, N);
auto end = high_resolution_clock::now();
auto durationSeqMatMat = duration_cast<milliseconds>(end - start);
cout << "Matrix-Matrix Multiplication (Sequential) Time: " <<
durationSeqMatMat.count() << " ms" << endl;

// Matrix-Matrix Multiplication - Parallel
start = high_resolution_clock::now();
matrixMatrixMultiplicationParallel(A, B, C, N);
end = high_resolution_clock::now();
auto durationParMatMat = duration_cast<milliseconds>(end - start);
cout << "Matrix-Matrix Multiplication (Parallel) Time: " <<
durationParMatMat.count() << " ms" << endl;

return 0;
}
```

Screenshots:

```
ubuntu@ubuntu-VirtualBox:~/Documents/HPC_LAB 2024/Assignment06$ g++ -fopenmp 06_01_a.cpp -o a
ubuntu@ubuntu-VirtualBox:~/Documents/HPC_LAB 2024/Assignment06$ ./a
Matrix-Matrix Multiplication (Sequential) Time: 11 ms
Matrix-Matrix Multiplication (Parallel) Time: 6 ms
ubuntu@ubuntu-VirtualBox:~/Documents/HPC_LAB 2024/Assignment06$
```

Information:
Analysis:
For larger size of N
Parallel Time < Sequential Time

Problem Statement 2:

```
#include <iostream>
#include <omp.h>
#include <vector>
#include <chrono>

using namespace std;
using namespace std::chrono;

// Function for Matrix-Vector Multiplication (sequential)
void matrixVectorMultiplicationSeq(vector<vector<int>>& A, vector<int>& B,
vector<int>& C, int N) {
for (int i = 0; i < N; ++i) {
C[i] = 0;
for (int j = 0; j < N; ++j) {
C[i] += A[i][j] * B[j];
}
}
}

// Function for Matrix-Vector Multiplication (parallel)
void matrixVectorMultiplicationParallel(vector<vector<int>>& A,
vector<int>& B, vector<int>& C, int N) {
#pragma omp parallel for
for (int i = 0; i < N; ++i) {
C[i] = 0;
for (int j = 0; j < N; ++j) {
C[i] += A[i][j] * B[j];
}
}
}
```

```
int main() {
int N = 500; // Size of the matrix and vector

// Initialize matrix and vector
vector<vector<int>> A(N, vector<int>(N, 1));
vector<int> B(N, 1);
vector<int> result(N, 0);

// Matrix-Vector Multiplication - Sequential
auto start = high_resolution_clock::now();
matrixVectorMultiplicationSeq(A, B, result, N);
auto end = high_resolution_clock::now();
auto durationSeqMatVec = duration_cast<milliseconds>(end - start);
cout << "Matrix-Vector Multiplication (Sequential) Time: " <<
durationSeqMatVec.count() << " ms" << endl;

// Matrix-Vector Multiplication - Parallel
start = high_resolution_clock::now();
matrixVectorMultiplicationParallel(A, B, result, N);
end = high_resolution_clock::now();
auto durationParMatVec = duration_cast<milliseconds>(end - start);
cout << "Matrix-Vector Multiplication (Parallel) Time: " <<
durationParMatVec.count() << " ms" << endl;

return 0;
}
```

Screenshots:

```
ubuntu@ubuntu-VirtualBox:~/Documents/HPC_LAB 2024/Assignment06$ g++ -fopenmp 06_02_a.cpp -o a
ubuntu@ubuntu-VirtualBox:~/Documents/HPC_LAB 2024/Assignment06$ ./a
Matrix-Vector Multiplication (Sequential) Time: 3 ms
Matrix-Vector Multiplication (Parallel) Time: 5 ms
ubuntu@ubuntu-VirtualBox:~/Documents/HPC_LAB 2024/Assignment06$
```

Information:
Analysis:
For larger size of N
Parallel Time > Sequential Time

Github Link:

https://github.com/omkarauti11/HPC_LAB