HPC -2 (OpenMP)

#include <algorithm>

#include <iostream>

#include <omp.h>

#include <vector>

using namespace std;

// Function to perform Bubble Sort

void parallelBubbleSort(vector<int> &arr) {

int n = arr.size();

bool swapped;

do {

swapped = false;

#pragma omp parallel for shared(swapped)

for (int i = 0; i < n - 1; ++i) {

if (arr[i] > arr[i + 1]) {

swap(arr[i], arr[i + 1]);

swapped = true;

}

}

} while (swapped);

}

// Function to perform Merge Sort

void merge(vector<int> &arr, int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

vector<int> L(n1), R(n2);

for (int i = 0; i < n1; ++i) {

L[i] = arr[left + i];

}

for (int j = 0; j < n2; ++j) {

R[j] = arr[mid + 1 + j];

}

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

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k++] = L[i++];

} else {

arr[k++] = R[j++];

}

}

while (i < n1) {

arr[k++] = L[i++];

}

while (j < n2) {

arr[k++] = R[j++];

}

}

void parallelMergeSort(vector<int> &arr, int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

#pragma omp parallel sections

{

#pragma omp section

{ parallelMergeSort(arr, left, mid); }

#pragma omp section

{ parallelMergeSort(arr, mid + 1, right); }

}

merge(arr, left, mid, right);

}

}

int main() {

int n;

cout << "Enter the number of elements: ";

cin >> n;

// Input elements from the user

vector<int> data(n);

cout << "Enter the elements:\n";

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

cin >> data[i];

}

// Make a copy of data for parallel sorting

vector<int> data\_copy(data);

// Parallel Bubble Sort

cout << "\nParallel Bubble Sort:\n";

vector<int> par\_data(data\_copy);

double start\_time = omp\_get\_wtime();

parallelBubbleSort(par\_data);

double end\_time = omp\_get\_wtime();

cout << "Sorted list: ";

for (int num : par\_data) {

cout << num << " ";

}

cout << "\nTime taken: " << end\_time - start\_time << " seconds\n";

// Parallel Merge Sort

cout << "\nParallel Merge Sort:\n";

start\_time = omp\_get\_wtime();

parallelMergeSort(data\_copy, 0, n - 1);

end\_time = omp\_get\_wtime();

cout << "Sorted list: ";

for (int num : data\_copy) {

cout << num << " ";

}

cout << "\nTime taken: " << end\_time - start\_time << " seconds\n";

return 0;

}

OUTPUT:-

~/HPC$ g++ -o hpc2 -fopenmp hpc2.cpp

~/HPC$ ./hpc2

Enter the number of elements: 5

Enter the elements:

3 4 7 1 9

Parallel Bubble Sort:

Sorted list: 1 3 4 7 9

Time taken: 0.265723 seconds

Parallel Merge Sort:

Sorted list: 1 3 4 7 9

Time taken: 0.102292 seconds