HPC-2B(Merge sort)

#include <iostream>

#include <vector>

#include <omp.h>

Using namespace std;

// Merge two halves

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

Vector<int> temp;

Int I = left, j = mid+1;

While (I <= mid && j <= right) {

If (arr[i] < arr[j])

Temp.push\_back(arr[i++]);

Else

Temp.push\_back(arr[j++]);

}

While (I <= mid) temp.push\_back(arr[i++]);

While (j <= right) temp.push\_back(arr[j++]);

For (int k = 0; k < temp.size(); ++k)

Arr[left + k] = temp[k];

}

// Sequential Merge Sort

Void sequentialMergeSort(vector<int>& arr, int left, int right) {

If (left >= right) return;

Int mid = (left + right) / 2;

sequentialMergeSort(arr, left, mid);

sequentialMergeSort(arr, mid+1, right);

merge(arr, left, mid, right);

}

// Parallel Merge Sort using OpenMP

Void parallelMergeSort(vector<int>& arr, int left, int right, int depth = 0) {

If (left >= right) return;

Int mid = (left + right) / 2;

If (depth < 4) {

#pragma omp parallel sections

{

#pragma omp section

parallelMergeSort(arr, left, mid, depth + 1);

#pragma omp section

parallelMergeSort(arr, mid+1, right, depth + 1);

}

} else {

sequentialMergeSort(arr, left, mid);

sequentialMergeSort(arr, mid+1, right);

}

Merge(arr, left, mid, right);

}

Int main() {

Int n;

Cout << “Enter number of elements: “;

Cin >> n;

Vector<int> original(n);

Cout << “Enter “ << n << “ elements:\n”;

For (int I = 0; I < n; ++i)

Cin >> original[i];

Vector<int> seqArr = original;

Vector<int> parArr = original;

Double startSeq = omp\_get\_wtime();

sequentialMergeSort(seqArr, 0, n-1);

double endSeq = omp\_get\_wtime();

double startPar = omp\_get\_wtime();

parallelMergeSort(parArr, 0, n-1);

double endPar = omp\_get\_wtime();

cout << “\nTime taken by Sequential Merge Sort: “ << (endSeq

• startSeq) << “ seconds\n”;

Cout << “Time taken by Parallel Merge Sort : “ << (endPar • startPar) << “ seconds\n”;

Return 0;

}