NAME – GAUTAM CHANDRA SAHA

REG N0 – 201900099

DATE – 17/02/2022

**Implement merge sort (parallel) with openMP.**

#include<iostream>

#include<vector>

#include<algorithm>

#include<cstdio>

#include<omp.h>

using namespace std;

int merge\_sort\_parallel(vector<int> &arr, int lo, int hi){

if(lo<hi){

int mid = lo + (hi - lo )/2;

#pragma omp parallel sections

{

#pragma omp section

{

merge\_sort\_parallel(arr, lo, mid);

}

#pragma omp section

{

merge\_sort\_parallel(arr, mid+1, hi);

}

}

inplace\_merge(arr.begin()+lo, arr.begin()+mid+1, arr.begin()+hi+1);

}

return 0;

}

int merge\_sort\_serial(vector<int> &arr, int lo, int hi){

if(lo<hi){

int mid = lo + (hi - lo )/2;

merge\_sort\_parallel(arr, lo, mid);

merge\_sort\_parallel(arr, mid+1, hi);

inplace\_merge(arr.begin()+lo, arr.begin()+mid+1, arr.begin()+hi+1);

}

return 0;

}

vector<double> calc(int size){

vector<double> ans;

vector<int> arr(size);

for (int i = 0; i < size; i++)

arr[i]=rand()%size;

vector<int> arr2(arr);//copy the arr

//sort the array in serial

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

merge\_sort\_serial(arr,0,arr.size()-1);

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

ans.push\_back(end\_time-start\_time);

//sort the array in parallel

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

merge\_sort\_parallel(arr2,0,arr2.size()-1);

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

ans.push\_back(end\_time-start\_time);

return ans;

}

int main(){

cout<<"MERGE SORT IMPLEMENTATION USING OPEN MP"<<endl<<endl;

auto \_time = calc(500);

printf("%s%32s%32s\n\n","No. of Inputs","Exec time for parallel env","Exec time for serial env");

printf("%d%33lf%32lf\n",500,\_time[1],\_time[0]);

\_time = calc(1000);

printf("%d%32lf%32lf\n",1000,\_time[1],\_time[0]);

\_time = calc(1200);

printf("%d%32lf%32lf\n",1200,\_time[1],\_time[0]);

return 0;

}

**OUTPUT:**

Graphical user interface, text

Description automatically generated