**Implement Min, Max, Sum and Average operations using Parallel Reduction.**

**Code:**

#include <iostream>

#include <vector>

#include <omp.h>

#include <climits>

#include <random>

using namespace std;

void min\_reduction(vector<int>& arr) {

int min\_value = INT\_MAX;

#pragma omp parallel for reduction(min: min\_value)

for (int i = 0; i < arr.size(); i++) {

if (arr[i] < min\_value) {

min\_value = arr[i];

}

}

cout << "Minimum value: " << min\_value << endl;

}

void max\_reduction(vector<int>& arr) {

int max\_value = INT\_MIN;

#pragma omp parallel for reduction(max: max\_value)

for (int i = 0; i < arr.size(); i++) {

if (arr[i] > max\_value) {

max\_value = arr[i];

}

}

cout << "Maximum value: " << max\_value << endl;

}

void sum\_reduction(vector<int>& arr) {

int sum = 0;

#pragma omp parallel for reduction(+: sum)

for (int i = 0; i < arr.size(); i++) {

sum += arr[i];

}

cout << "Sum: " << sum << endl;

}

void average\_reduction(vector<int>& arr) {

int sum = 0;

#pragma omp parallel for reduction(+: sum)

for (int i = 0; i < arr.size(); i++) {

sum += arr[i];

}

cout << "Average: " << (double)sum / arr.size() << endl;

}

int main() {

const int size = 100;

vector<int> vec(size);

random\_device rd;

mt19937 gen(rd());

uniform\_int\_distribution<int> dis(1, 50); //Random values generated from 1 to 50

//uniform\_int\_distribution<int> dis(1, 1000); //Random values generated from 1 to 1000

cout<<"Size of Vector :"<<vec.size()<<endl;

cout<<"Vector values:";

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

vec[i] = dis(gen);

cout<<vec[i]<<" ";

}

cout<<endl;

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

min\_reduction(vec);

max\_reduction(vec);

sum\_reduction(vec);

average\_reduction(vec);

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

cout << "Runtime: " << end\_time - start\_time << " seconds" << endl;

}

**Output:**

**Case 1 : Random values between 1 to 1000**

hardik@hardik-Vector-GP66-12UGS:~/HPC\_Practical/Practical\_3$ ***g++ -fopenmp parallel\_reduction\_openmp.cpp -o parallel\_reduction***

hardik@hardik-Vector-GP66-12UGS:~/HPC\_Practical/Practical\_3$ ***./parallel\_reduction***

***Size of Vector :100***

***Vector values:770 8 38 20 905 362 694 771 419 978 832 323 39 82 18 125 260 618 959 421 82 518 591 580 254 262 755 475 866 604 491 114 442 206 886 891 47 854 744 608 63 764 889 36 586 265 656 754 902 606 177 817 696 415 373 827 830 244 772 841 685 657 459 753 315 706 503 923 13 150 386 99 705 665 920 379 242 545 75 308 582 983 469 452 492 635 586 695 331 670 428 745 301 743 238 643 224 600 766 91***

***Minimum value: 8***

***Maximum value: 983***

***Sum: 50587***

***Average: 505.87***

***Runtime: 0.113907 seconds***

**Case 2 : Random values between 1 to 50**

hardik@hardik-Vector-GP66-12UGS:~/HPC\_Practical/Practical\_3$ ***g++ -fopenmp parallel\_reduction\_openmp.cpp -o parallel\_reduction***

hardik@hardik-Vector-GP66-12UGS:~/HPC\_Practical/Practical\_3$ ***./parallel\_reduction***

***Size of Vector :100***

***Vector values:16 25 18 39 4 41 6 45 11 25 43 27 29 23 3 32 8 37 35 25 36 9 13 39 9 39 17 31 4 14 43 26 44 35 31 17 26 46 41 42 24 49 18 17 26 37 2 34 41 29 43 37 11 22 12 33 39 34 9 46 29 10 44 15 21 17 6 26 36 49 33 17 19 2 40 6 18 48 10 14 31 23 40 14 25 38 6 37 40 32 17 18 40 20 2 42 9 50 7 15***

***Minimum value: 2***

***Maximum value: 50***

***Sum: 2583***

***Average: 25.83***

***Runtime: 0.00251205 seconds***