Parallel Project

Hisham Alaa Ahmed Mohamed

CS Department

(Search Algorithm)

- Over all program using search algorithm to find some value

```
1   #include<iostream>
2   #include<omp.h>
3   #include<ctime>
4
5   using namespace std;
6
7   int* arr;//global array to be accessed by all methods
8
9   void sequentialSearch(int n,int value) { ... }
20
21  void parallelSearch(int n, int value) { ... }
38
39  int main() {
40      int n = 0, value = 0, index = 0;
41      cout << "Enter size of the array" << endl;
42      cin >> n;
43      arr = new int[n];
44      cout << "Enter value you want (Greater than or equal 100)" << endl;
45      cin >> value;
46      cout << "Enter index to put the value in it (less than the size)" << endl;
47      cin >> index;
```

```
48
49      for (int i = 0; i < n; i++)
50          arr[i] = (i==index)?(value):(rand() % 100);
51
52      clock_t end,start = clock();
53
54
55      sequentialSearch(n,value);
56      end = clock();
57      cout << "time of sequential algorithm is "
58           << (double)(end-start) / CLOCKS_PER_SEC << endl<<endl;
59
60      start = clock();
61      parallelSearch(n,value);
62      end = clock();
63      cout << "time of Parallel algorithm is "
64           << (double)(end-start) / CLOCKS_PER_SEC << endl<<endl;
65
66      return 0;
67  }
```

We have two functions one for the sequential search and the other one using the parallelism with OpenMP library.

- The first method is the sequential one

```
9   void sequentialSearch(int n,int value) {
10
11      for (int i = 0; i < n; i++) {
12          if (arr[i] == value)
13          {
14              cout << "the Searched num found at index " << i << endl;
15              break;
16          }
17      }
18
19  }
20
```

- The Second method is the parallel one

```
21  void parallelSearch(int n, int value){
22      int threadNum;
23      #pragma omp parallel private(threadNum) num_threads(10)
24      {
25          #pragma omp for schedule(static,n/omp_get_num_threads())
26          for (int i = 0; i < n; i++) {
27              if (arr[i] == value)
28              {
29                  #pragma omp critical
30                  {
31                      cout << "the Searched num found by thread " << omp_get_thread_num()
32                          <<" at index "<<i << endl;
33                  }
34              }
35          }
36      }
37  }
```

- The main method

```
39  int main() {
40      int n = 0, value = 0, index = 0;
41      cout << "Enter size of the array" << endl;
42      cin >> n;
43      arr = new int[n];
44      cout << "Enter value you want (Greater than or equal 100)" << endl;
45      cin >> value;
46      cout << "Enter index to put the value in it (less than the size)" << endl;
47      cin >> index;
48
49      for (int i = 0; i < n; i++)
50          arr[i] = (i==index)?(value):(rand() % 100);
51
52      clock_t end,start = clock();
53
54
55      sequentialSearch(n,value);
56      end = clock();
57      cout << "time of sequential algorithm is "
58           << (double)(end-start) / CLOCKS_PER_SEC << endl<<endl;
```

```
48
49      for (int i = 0; i < n; i++)
50          arr[i] = (i==index)?(value):(rand() % 100);
51
52      clock_t end,start = clock();
53
54
55      sequentialSearch(n,value);
56      end = clock();
57      cout << "time of sequential algorithm is "
58           << (double)(end-start) / CLOCKS_PER_SEC << endl<<endl;
59
60      start = clock();
61      parallelSearch(n,value);
62      end = clock();
63      cout << "time of Parallel algorithm is "
64           << (double)(end-start) / CLOCKS_PER_SEC << endl<<endl;
65
66      return 0;
67  }
```