Page :
Date : / /

Assignment NO: 3

- TITLE : Parallel sorting algorithm
- Problem Statement : Parallel sorting algorithm : For bubble sort and Merge sort based on existing sequential algorithm, design and implement parallel algorithm utilizing all resources available.
- Objective : i) To understand parallel bubble sort
  ii) To understand parallel merge sort
- Outcomes : i) Understand and implement parallel bubble sort and Merge sort using OpenMP
- Software and Hardware Requirement : A system with configuration : 4GB RAM, 500 GB HDD, i3 above CPU, GPU functionalities, C++ framework, Google colab, OpenMP.
- Date of completion :
- Theory :

OpenMP :

The application programming interface (API) OpenMP (Open Multi Processing) supports Multi-platform shared memory multiprocessing programming in C, C++, Fortran. On many platforms, instruction-set architectures and operating systems, including Solaris, AIX, HP-UX,

linux, MacOS, and Windows. It consist of a set of compiler directives, library routines and environment variables that influences run-time behaviour.

OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms raging for the standard desktop computer to the supercomputer. An application built with the hybrid model of parallel programming can run on a computer cluster using both OpenMP and Message Passing interface (MPI) such that OpenMP is used for parallelism within a node with MPI is used for parallelism between nodes. There have also been efforts to run OpenMP on software distribute Shared Memory systems to translate OpenMP into MPI and to extend OpenMP for non-shared Memory systems.

Compilers with an implementation of OpenMP 3.0:

- GCC 4.3.1
- Mercurium compiler
- Intel fortran & C/C++ version 11.0
- IBM XL compiler
- Sun Studio 12
- GCC 4.7
- GCC, 4.9, 4.9.1
- ROSE (compiler framework)
- VAMPIR

A] Bubble Sort :

Using openMP :

Algorithm :

```
start = open.get.wtime();
        for i to n-1:
            # Pragma omp Parrallel for.
            for j to n-1 :
                if a[j] > a[j+1] :
                    temp = a[j+1]
                    a[j+1] = a[j]
                    a[j] = temp
                endfor
            end for
        end = omp-get-wtime();
```

for running bubble sort Parallely in C++ #include <omp.h> header file is used and the number of threads are set using:

```
omp_set_num_threads (2);
```

Bubbel sort sometime referred to as sinking sort is simple sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in wrong order.

B] Merge Sort : using OpenMP

Merge sort is efficient general-purpose comparision-based sorting algorithm. Merge sort is a divide & conquer algorithm.

Algorithm : Merge sort (int *arr, int start, int end)

```
if (start < end):
    mid = (start + end) / 2
    #Pragma omp parallel sections
    {
        #Pragma omp section
        mergesort (arr, start, mid)
        #Pragma omp section
        mergesort (arr, mid+1, end)
    }
    Merge (arr, start, end, mid)
end if
```

Merge (arr, start, end, mid) :

```
while i<= mid && j<= end :
    if (arr[i] < arr[j]) :
        temp[k] = arr[i]
        k++
        i++
    end if
    else
        temp[k] = arr[j]
        k++
        j++
    end else
end while
while (i<=mid) :
    temp[k] = arr[i]
    i++
    k++
```

```
end while
    while (j<= end) :
        temp[k] = arr[i]
            k++
            j++
    end while
```

- Test cases :

	Algorithm	Sequential Time	Parallel Time	Size
i]	Merge sort Sequential/ Parallel	0.04493	0.233914	n = 100000
ii]	bubble sort Sequential/ Parallel	44.5374	33.0307	n = 100000

- Conclusion : Thus we successfully implemented bubble and merge sort parallely using OpenMP.