Computer Science & Information Systems

**Systems for Data Analytics - Lab Sheet 8**

**Introduction to OpenMP**

1. Objective:

Students should be able to

1. Get familiarity with the shared memory model in Parallel computing environment using OpenMP
2. Get hands-on experience on different directives used in the programming exercises

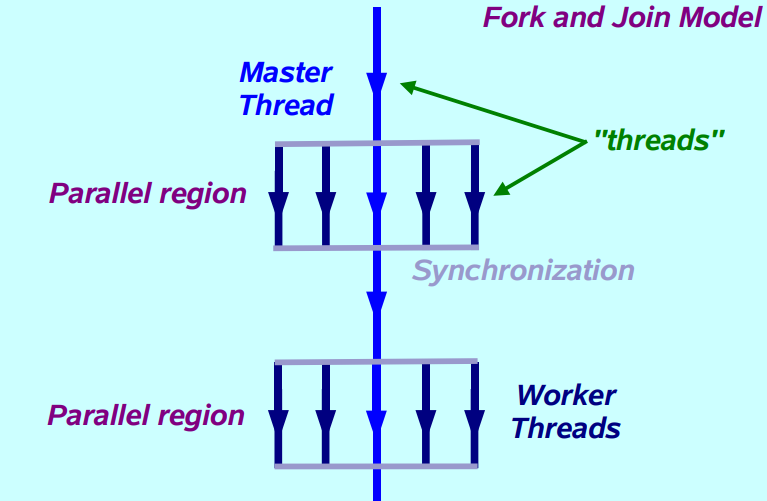
This lab sheet provides a quick introduction to using OpenMP.

OpenMP (Open MultiProcessing) is a parallel programming model based on compiler directives which allows application developers to incrementally add parallelism to their application codes.

OpenMP API specification for parallel programming provides an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++, and FORTRAN, on most platforms. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.

*Since OpenMP focuses on the parallelism within a node (shared memory multiprocessing) it can be combined with message-passing programming models, such as MPI, to execute on multiple nodes.*

OpenMP execution model



OpenMP core syntax

Function prototypes and types in the file: **#include<omp.h>**

Most of the constructs in OpenMP are compiler directives.

*#pragma omp construct [clause [clause]…]*

Example: **#pragma omp parallel num\_threads(4)**

#pragma omp parallel - Parallel region with default number of threads

omp\_set\_num\_threads(4) - Runtime function to request a certain number of threads

omp\_get\_thread\_num() - Runtime function returning a thread ID

#pragma omp critical - Threads wait their turn – only one at a time calls function

1. Steps to be performed:

Lab used – remote lab with Machines – MPI01 or MPI02

Language used – C

To compile: gcc -o <output-file> <sourcefile>.c -fopenmp

To run: ./<output-file>

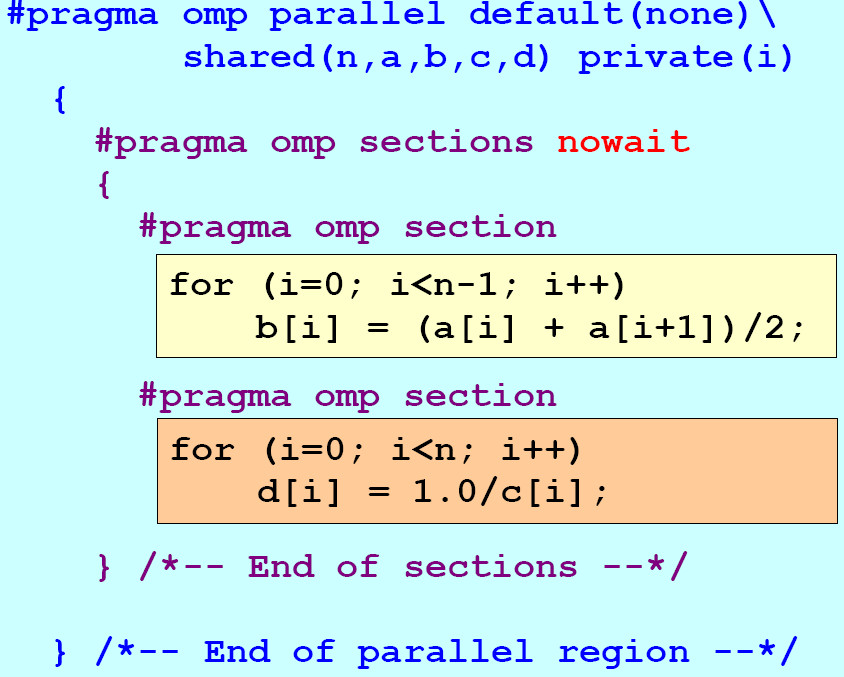
1. Outputs/Results:



1. Observations:

* Students to observe the output after applying the following changes in the program
  + Comment the pragma directive
  + Uncomment the pragma directive but do not use –fopenmp while compiling
  + Use the –fopenmp while compiling

Extra notes:

The omp section clause: 

For *Synchronization use barrier:*

Both loops are in parallel region

With no synchronization in between.

What is the problem?

Fix:

For(I=0; I<N; I++)

a[I] = b[I] + c[I];

#pragma omp barrier

For(I=0; I<N; I++)

d[I] = a[I] + b[I]

For(I=0; I<N; I++)

a[I] = b[I] + c[I];

For(I=0; I<N; I++)

d[I] = a[I] + b[I]

*Critical section:*

Cannot be parallelized if sum is shared.

Fix:

For(I=0; I<N; I++) {

……

sum += A[I];

……

}

For(I=0; I<N; I++) {

……

#pragma omp critical

{

sum += A[I];

}

……

}