IT 300 Parallel Computing
Lab1 : 9th Aug 2017

Lab1 Objective:

- To understand execution of openMP parallel program.
- To study on compiler directive: #pragma openmp parallel
- To set number of threads, get number of threads

1. Consider the following openMP program

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
int nthreads, tid;

/* Fork a team of threads giving them their own copies of variables */
#pragma omp parallel private(nthreads, tid)
  {

  /* Obtain thread number */
  tid = omp_get_thread_num();
  printf("Hello World from thread = %d\n", tid);

  /* Only master thread does this */
  if (tid == 0)
    {
    nthreads = omp_get_num_threads();
    printf("Number of threads = %d\n", nthreads);
    }

  }  /* All threads join master thread and disband */

}
```

Here, the lines in bold are directive or runtime library routins of OpenMP.
The program prints the thread number and total number of threads.

How to execute program?

```
$ gcc -fopenmp simple_omp.c
$ ./a.out
```

Note: Observe the output

omp_get_thread_num(): Used to get id of the thread.
omp_get_num_threads(): Used to find number of threads.
To set number of threads use following :
#pragma omp parallel private(nthreads, tid) **num_threads(4)**

The number of threads can be checked at command prompt using
echo $OMP_NUM_THREADS

The number of threads can be set at command prompt using
export OMP_NUM_THREADS=4

Run the program by setting number of threads as 4, 8, 16 and checkthe output.

2. Compiler directive:

```
#pragma omp parallel [clause ...]  newline
                    if (scalar_expression)
                    private (list)
                    shared (list)
                    default (shared | none)
                    firstprivate (list)
                    reduction (operator: list)
                    copyin (list)
                    num_threads (integer-expression)


   structured_block
```

Consider following program:

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
int nthreads, tid;
int i=5;

/* Fork a team of threads giving them their own copies of variables */
#pragma omp parallel private(nthreads, tid) firstprivate(i)
  {

  /* Obtain thread number */
  tid = omp_get_thread_num();
  printf("Hello World from thread = %d, i=%d\n", tid,i);
   i=i++;
printf("Hello World from thread = %d, i=%d after incr of i\n", tid,i);
  /* Only master thread does this */
  if (tid == 0)
    {
    nthreads = omp_get_num_threads();
    printf("Number of threads = %d\n", nthreads);
    }

  }  /* All threads join master thread and disband */

}
```

Observation:

a. check value of i by defining it as firstprivate(i), private(i) and shared(i). Analyse the output.

Write an openmp program to find sum of elements in series 1,2,3.... upto id of thread and execute the same.