**DEPARTMENT OF INFORMATION TECHNOLOGY, NITK SURATHKAL**

**Parallel Programming**

**LAB 1 -5th August 2020**

**Note: Observe the results of each program, take the screenshot of the result and upload it in the Moodle.**

**Note:**

**parallel**

Forms a team of threads and starts parallel execution.

**#pragma omp parallel** *[clause[ [, ]clause] ...]*

*structured-block*

*clause*:

**if**(*scalar-expression*)

**num\_threads**(*integer-expression*)

**default**(**shared** | **none**)

**private**(*list*)

**firstprivate**(*list*)

**shared**(*list*)

**copyin**(*list*)

**reduction**(*reduction-identifier***:** *list*)

------------------------------------------------------------------------------------

**loop** Specifies that the iterations of associated loops will be

executed in parallel by threads in the team in the context

of their implicit tasks.

**#pragma omp for** *[clause[ [, ]clause] ...]*

*for-loops*

*clause*:

**private**(*list*)

**firstprivate**(*list*)

**lastprivate**(*list*)

**reduction**(*reduction-identifier***:** *list*)

**schedule**(*kind[***,** *chunk\_size]*)

**collapse**(*n*)

**ordered**

**nowait**

*kind***:**

• **static:** Iterations are divided into chunks of size *chunk\_size* and assigned to threads in the team in round-robin fashion in order of thread number.

• **dynamic:** Each thread executes a chunk of iterations then requests another chunk until none remain.

• **guided:** Each thread executes a chunk of iterations then requests another chunk until no chunks remain to be assigned.

• **auto:** The decision regarding scheduling is delegated to the compiler and/or runtime system.

• **runtime:** The schedule and chunk size are taken from the *run-sched-var* ICV.

**-------------------------------------------------------------------------------------------------------------**

I. Finding number of CPU s in system

a) lscpu command

$ lscpu  
$ lscpu | egrep 'Model name|Socket|Thread|NUMA|CPU\(s\)'  
$ lscpu -p

**b)Run top ot htop command to obtain the number of CPUs/cores in linux**

$top

**c) Execute nproc print the nu,ber of CPUs available on Linux**

$ nproc --all  
 $ echo "Threads/core: $(nproc --all)"

**1.Write a C/C++ simple parallel program to display the *thread\_id* and total number of threads.**

/\*simpleomp.c\*/

#include<omp.h>

int main(){

int nthreads,tid;

**#pragma omp parallel private(tid)**

{

**tid=omp\_get\_thread\_num();**

printf(“Hello world from thread=%d\n”,tid);

if(tid==0)

{

nthreads=omp\_get\_num\_threads();

printf(“Number of threads=%d\n”,nthreads);

}

}

}

**Execute the program as follows:**

$gcc –o simple –fopenmp simpleomp.c

$export OMP\_NUM\_THREADS=2

$./simple

**Note down the output in your observation book.**

Number of threads in a parallel region is determined by the *if* clause, *num\_threads(),omp\_set\_num\_threads(), OMP\_NUM\_THREADS.*

**Use these various methods to set number of threads and mention the method of setting the same.**

**2. Check the output of following program:**

/\*ifparallel.c\*/

#include<omp.h>

int main(){

int val;

printf(“Enter 0: for serial 1: for parallel\n”);

scanf(“%d”,&val);

#pragma omp parallel if(val)

{

**if(omp\_in\_parallel())**

printf(“Parallel val=%d id= %d\n”,val, omp\_get\_thread\_num());

else

printf(“Serial val=%d id= %d\n”,val, omp\_get\_thread\_num());

}

}

**Note down the output in your observation book.**

**3.Observe and record the output of following program**

/\*num\_threads.c\*/

#include<omp.h>

int main(){

#pragma omp parallel num\_threads(4)

{

int tid=omp\_get\_thread\_num();

printf(“Hello world from thread=%d\n”,tid);

}

}

**4.Write a C/C++ parallel program for adding corresponding elements of two arrays.**

/\*addarray.c\*/

#include<omp.h>

int main(){

int i,n,chunk;

int a[20],b[20],c[20];

n=20;

chunk=2;

/\*initializing array\*/

for(i=0;i<n;i++)

{ a[i]=i\*2;

b[i]=i\*3;

}

**#pragma omp parallel for default(shared) private(i) schedule(static,chunk)**

{

for(i=0;i<n;i++)

{

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

printf(“Thread id= %d i=%d,c[%d]=%d\n”, omp\_get\_thread\_num(),i,i,c[i]);

}

}

**Check the output by varying**

1. **Chunk size**
2. **Number of threads**

**Note down the allotment of i range for each thread.**