DEPARTMENT OF INFORMATION TECHNOLOGY, NITK SURATHKAL

Parallel Programming Lab 1

P Akshara - 181IT132

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

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*)

---

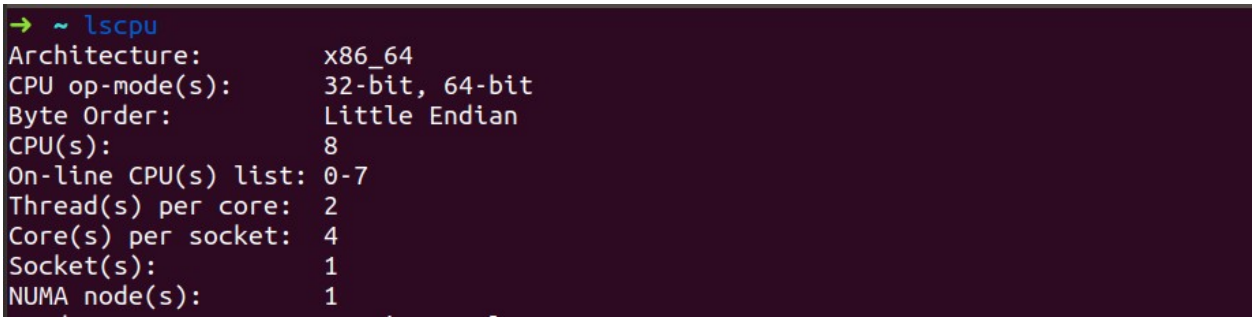
loop Specifies that the iterations of associated loops will be executed in parallel by threads in the team in the context of their implicit tasks.
#pragma omp for *[clause[[,]clause] ...]*
for-loops
clause:
private(*list*)
firstprivate(*list*)
lastprivate(*list*)
reduction(*reduction-identifier*: *list*)
schedule(*kind*[, *chunk_size*])
collapse(*n*)
ordered
nowait
kind:

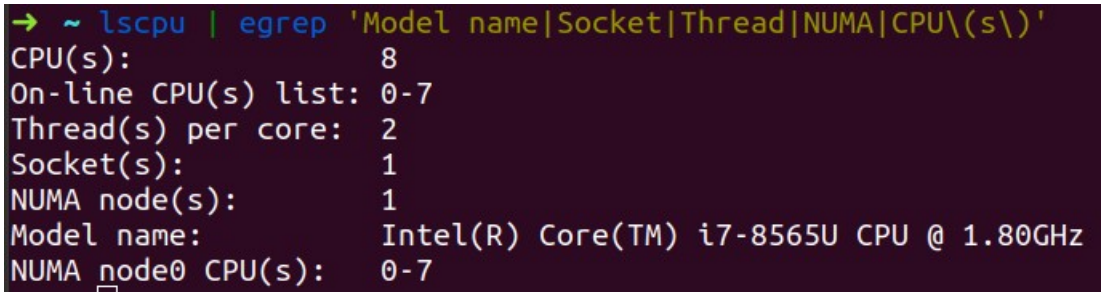
• **static:** Iterations are divided into chunks of size *chunk_size* and assigned to threads in the team in round-robin fashion in order of thread number.
• **dynamic:** Each thread executes a chunk of iterations then requests another chunk until none remain.
• **guided:** Each thread executes a chunk 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
```





```
➜ ~ lscpu -p
# The following is the parsable format, which can be fed to other
# programs. Each different item in every column has an unique ID
# starting from zero.
# CPU,Core,Socket,Node,,L1d,L1i,L2,L3
0,0,0,0,,0,0,0,0
1,1,0,0,,1,1,1,0
2,2,0,0,,2,2,2,0
3,3,0,0,,3,3,3,0
4,0,0,0,,0,0,0,0
5,1,0,0,,1,1,1,0
6,2,0,0,,2,2,2,0
7,3,0,0,,3,3,3,0
```

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

`$top`

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1882 akshara   20   0 4011552 327388 140000 S  10.2  4.1   0:19.97 gnome-shell
 1675 akshara   20   0  564252  97668  71816 S   1.0  1.2   0:12.18 Xorg
 1032 root      20   0 1674876  32168  14972 S   0.7  0.4   0:02.31 snapd
 2875 akshara   20   0  793508  37176  27856 S   0.7  0.5   0:00.45 gnome-terminal-
   49 root      20   0       0      0      0 I   0.3  0.0   0:00.06 kworker/6:0-eve
  289 root      20   0       0      0      0 I   0.3  0.0   0:00.53 kworker/u16:3-e
 1209 mysql     20   0 1358988 180896  15380 S   0.3  2.3   0:00.99 mysqld
 1393 gdm       20   0 3693228 162516 109504 S   0.3  2.0   0:02.96 gnome-shell
 1905 akshara    9 -11 1597092  18284  13832 S   0.3  0.2   0:00.28 pulseaudio
 1926 akshara   20   0  354512   7772   6344 S   0.3  0.1   0:00.64 ibus-daemon
 2603 akshara   20   0 11.859g 192648  92156 S   0.3  2.4   0:05.22 code
 2629 akshara   20   0  700556 145560  51816 S   0.3  1.8   0:01.75 code
```

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

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

```
➜ ~ bash -c 'echo "Threads/core: $(nproc --all)"'
Threads/core: 8
```

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);

}}}

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.

Output

Using if, omp_set_num_threads(5)

```
akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:~/Lab-Sem5/IT301 PC/Lab 1$ gcc -o q1 -fopenmp q1.c
akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:~/Lab-Sem5/IT301 PC/Lab 1$ ./q1
Hello world from thread =  0
Number of threads =  5
Hello world from thread =  3
Hello world from thread =  4
Hello world from thread =  1
Hello world from thread =  2
```

Using export OMP_NUM_THREADS=2

```
→  Lab-Sem5 git:(master) ✗ cd IT301\ PC/Lab\ 1/
→  Lab 1 git:(master) ✗ gcc -o q1 -fopenmp q1.c
→  Lab 1 git:(master) ✗ export OMP_NUM_THREADS=2
→  Lab 1 git:(master) ✗ ./q1
Hello world from thread =  0
Number of threads =  2
Hello world from thread =  1
```

Using *num_threads(3)*

```
→  Lab 1 git:(master) ✗ gcc -o q1 -fopenmp q1.c
→  Lab 1 git:(master) ✗ ./q1
Hello world from thread =  0
Number of threads =  3
Hello world from thread =  1
Hello world from thread =  2
```

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.

Output

```
(base) akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:~/Lab-Sem5/IT301 PC/Lab 1$ gcc -o q2 -fopenmp q2.c
(base) akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:~/Lab-Sem5/IT301 PC/Lab 1$ ./q2
Enter 0: for serial 1: for parallel
0
Serial val=0 id= 0
(base) akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:~/Lab-Sem5/IT301 PC/Lab 1$ ./q2
Enter 0: for serial 1: for parallel
1
Parallel val=1 id= 0
Parallel val=1 id= 3
Parallel val=1 id= 4
Parallel val=1 id= 5
Parallel val=1 id= 6
Parallel val=1 id= 2
Parallel val=1 id= 1
Parallel val=1 id= 7
```

3.Observe and record the output of following program

/*num_threads.c*/

#include<omp.h>

int main(){

#pragma omp parallel num_threads(4)

{

int i=omp_get_thread_num();

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

}

}

Output

```
(base) akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:~/Lab-Sem5/IT301 PC/Lab 1$ gcc -o q3 -fopenmp q3.c
(base) akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:~/Lab-Sem5/IT301 PC/Lab 1$ ./q3
Hello world from thread=0
Hello world from thread=3
Hello world from thread=2
Hello world from thread=1
```

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.

Output

Chunk size = 4, No of threads = 4

Thread 0: 0,1,2,3,16,17,18,19 **Thread 1: 4,5,6,7**

Thread 2: 8,9,10,11 **Thread 3: 12,13,14,15**

```
(base) akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:~/Lab-Sem5/IT301 PC/Lab 1$ ./q4
Thread id=0 i=0,c[0]=0
Thread id=0 i=1,c[1]=5
Thread id=0 i=2,c[2]=10
Thread id=0 i=3,c[3]=15
Thread id=0 i=16,c[16]=80
Thread id=0 i=17,c[17]=85
Thread id=0 i=18,c[18]=90
Thread id=0 i=19,c[19]=95
Thread id=3 i=12,c[12]=60
Thread id=3 i=13,c[13]=65
Thread id=3 i=14,c[14]=70
Thread id=3 i=15,c[15]=75
Thread id=1 i=4,c[4]=20
Thread id=1 i=5,c[5]=25
Thread id=1 i=6,c[6]=30
Thread id=1 i=7,c[7]=35
Thread id=2 i=8,c[8]=40
Thread id=2 i=9,c[9]=45
Thread id=2 i=10,c[10]=50
Thread id=2 i=11,c[11]=55
Execution 0(base) akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:~/Lab-Sem5/IT301 PC/Lab 1$
```

Chunk size = 2, No of threads = 4

Thread 0: 0,1,8,9,16,17 **Thread 1: 2,3,10,11,18,19**

Thread 2: 4,5,12,13 **Thread 3: 6,7,14,15**

```
(base) akshara@akshara-VivoBook-ASUSLaptop-X530FN-S530FN:~/Lab-Sem5/IT301 PC/Lab 1$ ./q4
Thread id=0 i=0,c[0]=0
Thread id=0 i=1,c[1]=5
Thread id=0 i=8,c[8]=40
Thread id=0 i=9,c[9]=45
Thread id=0 i=16,c[16]=80
Thread id=0 i=17,c[17]=85
Thread id=1 i=2,c[2]=10
Thread id=1 i=3,c[3]=15
Thread id=1 i=10,c[10]=50
Thread id=1 i=11,c[11]=55
Thread id=1 i=18,c[18]=90
Thread id=1 i=19,c[19]=95
Thread id=2 i=4,c[4]=20
Thread id=2 i=5,c[5]=25
Thread id=3 i=6,c[6]=30
Thread id=3 i=7,c[7]=35
Thread id=3 i=14,c[14]=70
Thread id=3 i=15,c[15]=75
Thread id=2 i=12,c[12]=60
Thread id=2 i=13,c[13]=65
```

Chunk size = 8, No of threads = 3

Thread 0: 0,1,2,3,4,5,6,7 **Thread 1: 8,9,10,11,12,13,14,15**

Thread 2: 16,17,18,19

```
→  Lab 1 git:(master) ✗ ./q4
Thread id=0 i=0,c[0]=0
Thread id=0 i=1,c[1]=5
Thread id=0 i=2,c[2]=10
Thread id=0 i=3,c[3]=15
Thread id=0 i=4,c[4]=20
Thread id=0 i=5,c[5]=25
Thread id=0 i=6,c[6]=30
Thread id=0 i=7,c[7]=35
Thread id=1 i=8,c[8]=40
Thread id=1 i=9,c[9]=45
Thread id=1 i=10,c[10]=50
Thread id=1 i=11,c[11]=55
Thread id=1 i=12,c[12]=60
Thread id=1 i=13,c[13]=65
Thread id=1 i=14,c[14]=70
Thread id=1 i=15,c[15]=75
Thread id=2 i=16,c[16]=80
Thread id=2 i=17,c[17]=85
Thread id=2 i=18,c[18]=90
Thread id=2 i=19,c[19]=95
Execution 0
```