DEPARTMENT OF INFORMATION TECHNOLOGY, NITK SURATHKAL

Parallel Programming

LAB 1 -3rd August 2016

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
```

```
jacky@jacky-Strix-G531GT-G531GT:~$ lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              8
On-line CPU(s) list: 0-7
Thread(s) per core:  2
Core(s) per socket:  4
Socket(s):           1
NUMA node(s):        1
Vendor ID:           GenuineIntel
CPU family:          6
Model:               158
Model name:          Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz
Stepping:            10
CPU MHz:             867.624
CPU max MHz:         4100.0000
CPU min MHz:         800.0000
BogoMIPS:            4800.00
Virtualization:      VT-x
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            8192K
NUMA node0 CPU(s):   0-7
Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pa
t pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aper
fmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm
pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand l
ahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_
shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms
 invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dther
m ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_l1d
```

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

```
jacky@jacky-Strix-G531GT-G531GT:~$  lscpu | egrep 'Model name|Socket|Thread|NUMA|CPU\(s\)'
CPU(s):                  8
On-line CPU(s) list:     0-7
Thread(s) per core:      2
Socket(s):               1
NUMA node(s):            1
Model name:              Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz
NUMA node0 CPU(s):       0-7
```

$ lscpu -p

```
jacky@jacky-Strix-G531GT-G531GT:~$ 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

```
jacky@jacky-Strix-G531GT-G531GT: ~
File  Edit  View  Search  Terminal  Help
jacky@jacky-Strix-G531GT-G531GT:~$ top

top - 21:53:52 up  1:47,  1 user,  load average: 0.51, 0.70, 0.89
Tasks: 316 total,   1 running, 245 sleeping,   1 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.5 sy,  0.0 ni, 98.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  8000920 total,  2009176 free,  3863760 used,  2127984 buff/cache
KiB Swap:  2097148 total,  2097148 free,        0 used.  3376264 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 5031 jacky     20   0 3350052 468444 195756 S   4.6  5.9   2:19.67 Web Content
 2569 jacky     20   0 4189896 577108 217636 S   3.3  7.2  12:29.35 firefox
 1672 jacky     20   0  618020 113860  92204 S   2.6  1.4   4:56.05 Xorg
 1809 jacky     20   0 3966816 258172 132336 S   1.7  3.2   4:21.63 gnome-shell
 5460 jacky     20   0  799964  37128  27352 S   0.7  0.5   0:02.94 gnome-terminal-
    1 root      20   0  225544   9336   6688 S   0.3  0.1   0:17.20 systemd
 2742 jacky     20   0 3303172 470612 120028 S   0.3  5.9   3:36.51 Web Content
 5526 jacky     20   0   51420   4288   3420 R   0.3  0.1   0:00.09 top
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kthreadd
    3 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 rcu_par_gp
    5 root      20   0       0      0      0 I   0.0  0.0   0:00.58 kworker/0:0-eve
    6 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/0:0H-kb
    7 root      20   0       0      0      0 I   0.0  0.0   0:00.01 kworker/0:1-cgr
    9 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 mm_percpu_wq
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.11 ksoftirqd/0
   11 root      20   0       0      0      0 I   0.0  0.0   0:04.06 rcu_sched
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.02 migration/0
   13 root     -51   0       0      0      0 S   0.0  0.0   0:00.00 idle_inject/0
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/1
   16 root     -51   0       0      0      0 S   0.0  0.0   0:00.00 idle_inject/1
   17 root      rt   0       0      0      0 S   0.0  0.0   0:00.02 migration/1
   18 root      20   0       0      0      0 S   0.0  0.0   0:00.04 ksoftirqd/1
   20 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/1:0H-kb
   21 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/2
   22 root     -51   0       0      0      0 S   0.0  0.0   0:00.00 idle_inject/2
   23 root      rt   0       0      0      0 S   0.0  0.0   0:00.01 migration/2
```

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

```
$ nproc --all
```

```
jacky@jacky-Strix-G531GT-G531GT:~$ nproc --all
8
```

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

```
jacky@jacky-Strix-G531GT-G531GT:~$ 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);

}

}

}
```

Execute the program as follows:

```
$gcc –o simple –fopenmp simpleomp.c

$export OMP_NUM_THREADS=2

$./simple
```

```
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o simple -fopenmp simpleomp.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./a.out
bash: ./a.out: No such file or directory
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./simple
Hello world from thread=0
Number of threads=4
Hello world from thread=3
Hello world from thread=1
Hello world from thread=2
```

Here I have used the omp_set_num_threads() function to set the number of threads.And assigned the number of threads =4.

```
int nthreads,tid;
omp_set_num_threads(4);
```

```
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o simple -fopenmp simpleomp.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ set OMP_NUM_THREADS=8
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./simple
Hello world from thread=0
Number of threads=4
Hello world from thread=2
Hello world from thread=1
Hello world from thread=3
```

Number of threads are still 4 after using set OMP_NUM_THREADS=8

So, let’s change in code and then apply:

```
int nthreads,tid;
//omp_set_num_threads(4);
```

Here I have used OMP_NUM_THREADS to set the number of threads.And assigned the number of threads =9 and then 8.

```
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o simple -fopenmp simpleomp.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ set OMP_NUM_THREADS=9
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./simple
Hello world from thread=1
Hello world from thread=5
Hello world from thread=4
Hello world from thread=3
Hello world from thread=7
Hello world from thread=0
Number of threads=8
Hello world from thread=2
Hello world from thread=6
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o simple -fopenmp simpleomp.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ set OMP_NUM_THREADS=8
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./simple
Hello world from thread=5
Hello world from thread=3
Hello world from thread=4
Hello world from thread=0
Number of threads=8
Hello world from thread=2
Hello world from thread=6
Hello world from thread=7
Hello world from thread=1
```

<u>Note down the output in your observation book.</u>

Number of threads in a parallel region is determined by the *if* clause, *num_threads(),omp_set_num_threads(), OMP_NUM_THREADS.*

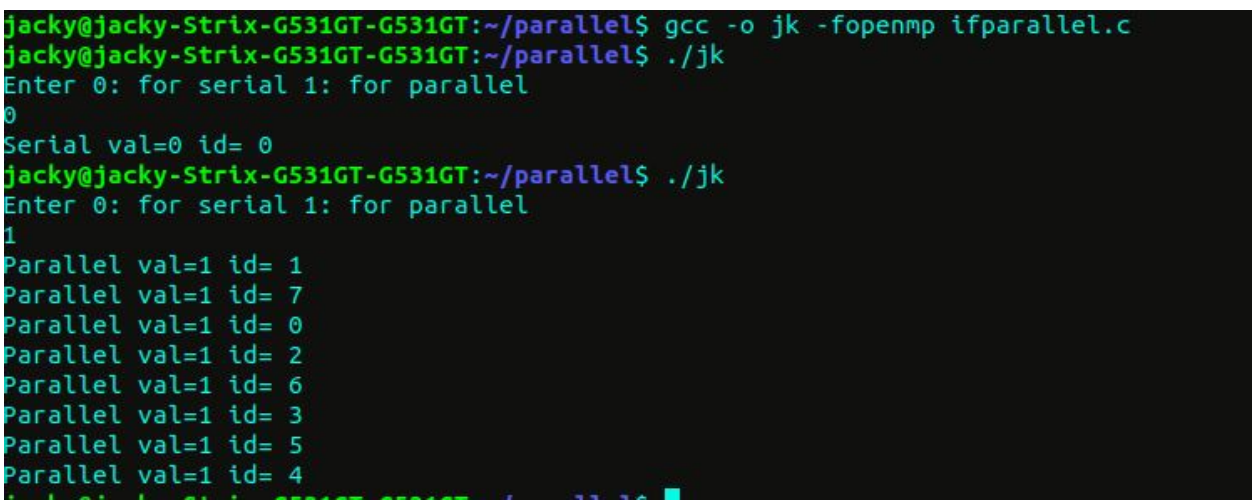
Use these various methods to set number of threads and mention the method of setting the same.

2. Check the output of following program:

```
/*ifparallel.c*/

#include<omp.h>

int main(){

int val;

//omp_set_num_threads(8);

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());

}

}
```

OUTPUT :



```
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o jk -fopenmp ifparallel.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./jk
Enter 0: for serial 1: for parallel
0
Serial val=0 id= 0
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./jk
Enter 0: for serial 1: for parallel
1
Parallel val=1 id= 1
Parallel val=1 id= 7
Parallel val=1 id= 0
Parallel val=1 id= 2
Parallel val=1 id= 6
Parallel val=1 id= 3
Parallel val=1 id= 5
Parallel val=1 id= 4
```

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);

}

}
```

OUTPUT:

Here number of threads are 4 and 7 respectively.

```
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o jk -fopenmp num_threads.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./jk
Hello world from thread=0
Hello world from thread=1
Hello world from thread=3
Hello world from thread=2
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o jk -fopenmp num_threads.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./jk
Hello world from thread=0
Hello world from thread=1
Hello world from thread=6
Hello world from thread=3
Hello world from thread=4
Hello world from thread=5
Hello world from thread=2
```

If tid is not declared with omp_get_thread_num() , then output is a default/junk value

```
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o jk -fopenmp num_threads.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./jk
Hello world from thread=32764
Hello world from thread=32764
Hello world from thread=32764
Hello world from thread=32764
```

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]);

}

}
```

Checking the output by varying Chunk Size and Number of Threads:

a. Taking chunk size=5,No of threads=4

```
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o jk -fopenmp addarray.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./jk
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= 2 i=10,c[10]=50
Thread id= 2 i=11,c[11]=55
Thread id= 2 i=12,c[12]=60
Thread id= 2 i=13,c[13]=65
Thread id= 2 i=14,c[14]=70
Thread id= 3 i=15,c[15]=75
Thread id= 0 i=3,c[3]=15
Thread id= 0 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= 1 i=8,c[8]=40
Thread id= 1 i=9,c[9]=45
Thread id= 3 i=16,c[16]=80
Thread id= 3 i=17,c[17]=85
Thread id= 3 i=18,c[18]=90
Thread id= 3 i=19,c[19]=95
```

Here the allotment i for each thread are:

for thread 0: 0,1,2,3,4

for thread 1: 5,6,7,8,9

for thread 2: 10,11,12,13,14

for thread 3: 15,16,17,18,19.

b.Taking chunk size=10,No of threads=4

```
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o jk -fopenmp addarray.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./jk
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= 0 i=8,c[8]=40
Thread id= 0 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= 1 i=16,c[16]=80
Thread id= 1 i=17,c[17]=85
Thread id= 1 i=18,c[18]=90
Thread id= 1 i=19,c[19]=95
```

From observation, if chunk size is more then some of the threads are not used as the program finishes earlier .

c.Taking chunk size=4,No of threads=8

```
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ gcc -o jk -fopenmp addarray.c
jacky@jacky-Strix-G531GT-G531GT:~/parallel$ ./jk
Thread id= 4 i=16,c[16]=80
Thread id= 4 i=17,c[17]=85
Thread id= 4 i=18,c[18]=90
Thread id= 4 i=19,c[19]=95
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
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= 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
```

NAME: BHAJAN KUMAR BARMAN
ROLL NO: 181IT211