Topic: Parallel Programming with OpenMP

Objective

- OpenMP introduction
- How to run OpenMP program
- First program
- Examples

Topic: Parallel Programming with OpenMP

Basics

- OpenMP is a shared-memory parallelism model
- Compilation: use flag -fopenmp for gfortran
- All OMP directives begin with !$omp
- Unlike MPI, multiple parallel regions within a single program can exist
- In a parallel region, variables are shared by default
- Number of threads can be controlled with environment variable
  - export OMP_NUM_THREADS=N

First program

```
program hello
 use omp_lib
 implicit none


  write(*,*) "Hello, World!"


end program hello
```

First program

```
program hello
 use omp_lib
 implicit none

  !$omp parallel

  write(*,*) "Hello, World!"

  !$omp end parallel

end program hello
```

First program

```
program hello
 use omp_lib
 implicit none

  !$omp parallel

  write(*,*) "Hello, World!"

  !$omp end parallel

end program hello
```

export OMP_NUM_THREADS=N

What is the output?

```
program hello
 use omp_lib
 implicit none

  !$omp parallel


  !$omp end parallel

  write(*,*) "Hello, World!"

end program hello
```

What is the output?

```
program hello
 use omp_lib
 implicit none
 integer :: msg

  !$omp parallel
   msg=10

  !$omp end parallel

  write(*,*) "Hello, World!", msg

end program hello
```

What is the output?

```
program hello
 use omp_lib
 implicit none
 integer :: msg

  !$omp parallel private(msg)
   msg=10

  !$omp end parallel

  write(*,*) "Hello, World!", msg

end program hello
```

omp parallel directive

- By default, all variables are shared among threads
- clauses
  - private(v1, v2, v3, ..), shared(v4,v5), default(shared|private|none)
  - num_threads(np), reduction(operator:var1, var2,...)

print thread id number

```
program hello
 use omp_lib
 implicit none
 integer :: id

 !$omp parallel

 write(*,*) "hello from process: ",
omp_get_thread_num(),"/",omp_get_num_threads()

!$omp end parallel
end program hello
```

What is the output?

```
program hello_world
    use omp_lib
    implicit none

    integer :: num_threads, thread_id

    num_threads = omp_get_max_threads()

    !$omp parallel private(thread_id)

      thread_id = omp_get_thread_num()
      write(*,*) 'ad', thread_id, 'out of', num_threads

    !$omp end parallel
    write(*,*) 'Hello, from thread', thread_id, '/', num_threads

    ! end parallel section

end program hello_world
```

Define a variable in master thread

```
program hello
 use omp_lib
 implicit none
 integer :: msg

  !$omp parallel private(msg)

  !$omp master
    msg=20
  !$omp end master
   write(*,*) "Hello, World!", msg
  !$omp end parallel

end program hello
```

Q: Broadcast the variable ‘msg’

```
program hello
 use omp_lib
 implicit none
 integer :: msg

  !$omp parallel private(msg)

  !$omp master
    msg=20
  !$omp end master
   write(*,*) "Hello, World!", msg
  !$omp end parallel

end program hello
```

Barrier - synchronization

```
program hello
use omp_lib
integer :: id

!$omp parallel private(id)

    id = omp_get_thread_num()

    do i=0,omp_get_max_threads()
        if (i == id) then
          write(*,*) "hello from process: ", id
        end if
    !$omp barrier
    end do
!$omp end parallel

end program hello
```

Topic: Parallel Programming with OpenMP - sum of N numbers

```
program test
  use omp_lib
  implicit none
  integer, parameter :: N = 10000
  integer :: total, i, partial_sum, nt, id

  total = 0

  ! Parallel region starts
  !$omp parallel private(i, partial_sum)
  partial_sum = 0

  !$omp do
  do i = 1, N
    partial_sum = partial_sum + i
  end do
  !$omp end do
```

```
  !$omp critical
  total = total + partial_sum
  !$omp end critical

  ! End of parallel region
  !$omp end parallel

  write(*,*) "Sum: ", total

end program test
```

deadlock

Examples

```
!$OMP CRITICAL
  !$OMP BARRIER
!$OMP END CRITICAL

!$OMP SINGLE
  !$OMP BARRIER
!$OMP END SINGLE

!$OMP MASTER
  !$OMP BARRIER
!$OMP END MASTER
```

Hands-on

- Write a parallel program using OpenMP for computing the sum of an array of numbers. Calculate the time taken with 1, 2, and 4 threads. Fortran program is given (sum_array.f90)

Topic: Parallel Programming with OpenMP

Hands on

- https://curc.readthedocs.io/en/latest/programming/OpenMP-Fortran.html