Topic: MPI Programming

Objective

- MPI Basics - Learn about MPI functions
- Compiling a MPI program
- Running MPI program
- MPI in a Fortran program

What is MPI

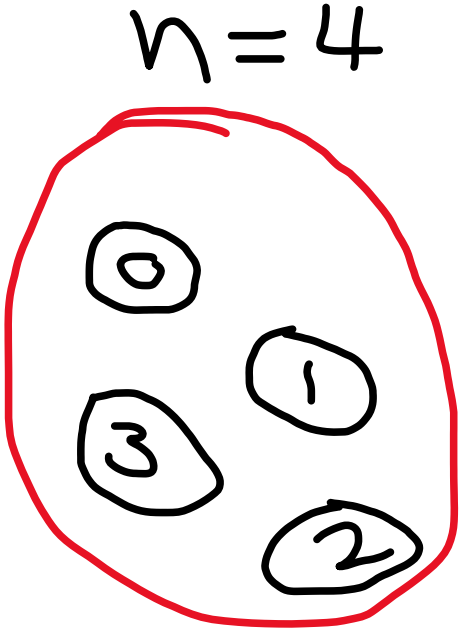
- For parallel computing
- A Message-Passing Interface (MPI) communication protocols or specifications
  - not language or compiler specific
  - not hardware specific
- High performance, efficient and portable
- Designed for providing access to the advanced parallel computer hardware for end users and tool developers
- MPI library: IntelMPI, openMPI, MVAPICH, MPICH, etc

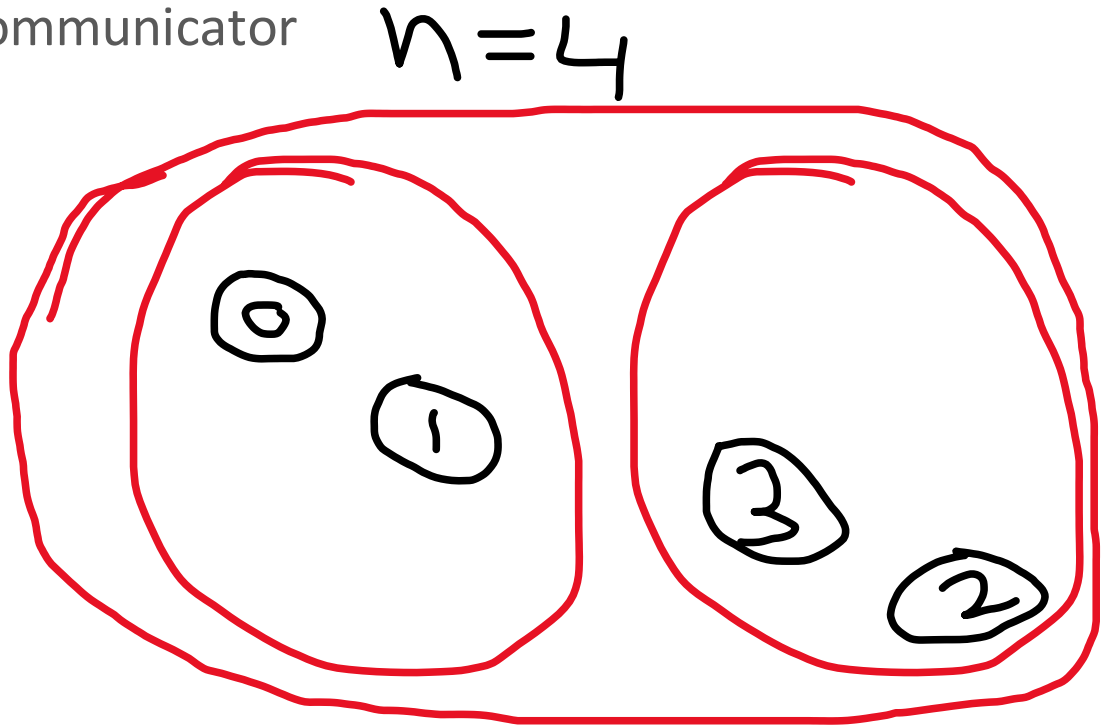
Message passing

- P1: send(message, P2) & P2: receive(message, P1)
- Each process must be assigned a unique identifier: integer numbers
- Parent (or root) and child process
- MPI communicators
- Subroutine/function name start with MPI_
- Use *call* statement in Fortran program for MPI subroutine
- MPI constants

MPI communicators

- Default communicator: MPI_COMM_WORLD
- There can be more than one communicator





- MPI_COMM_WORLD

- MPI_COMM_WORLD, MPI_NEW_COMM1, MPI_NEW_COMM2

Six MPI functions

- A total of more than 125 functions available (397 functions in the latest version of mpich library)
- Biggest advantage: One need not master all parts of MPI to use it
- Most of the MPI programs can be written with just SIX MPI functions

MPI_Init
MPI_Comm_Size
MPI_Comm_Rank
MPI_Finalize
MPI_Bcast
MPI_Reduce

Compiler wrappers – Open MPI library

These are provided by the MPI library package

Language	Compiler	Command
C	gcc	mpicc
C++	g++	mpicxx
Fortran	gfortran	mpif90

Install Open MPI library using,

sudo apt install libopenmpi-dev

How to compile and run the job

- Compile: mpif90 mpihello.90 -o mpihello.x
- Run: mpirun -np XX ./mpihello.x
  - where XX is the number of processes
- Use '-show' or '-showme' to display all libraries and flags used along with the compiler name. Eg: mpif90 -show

- gfortran -I/usr/lib/openmpi/include -pthread -I/usr/lib/openmpi/lib -Wl,-rpath -Wl,/usr/lib/openmpi/lib -Wl,--enable-new-dtags -L/usr/lib/openmpi/lib -lmpi_usempif08 -lmpi_usempi_ignore_tkr -lmpi_mpifh -lmpi

examples

- Run: mpirun -np XX ./mpihello.x
  - where XX is the number of processes

Why are we using mpirun command?

```
 $ mpif90 p1.f90 -o p1.x
 $ mpirun -np 1 ./p1.x
Sum of first   100   numbers is:    5050
 $
 $ mpirun -np 2 ./p1.x
Sum of first   100   numbers is:    5050
Sum of first   100   numbers is:    5050
 $
 $ mpirun -np 4 ./p1.x
Sum of first   100   numbers is:    5050
Sum of first   100   numbers is:    5050
Sum of first   100   numbers is:    5050
Sum of first   100   numbers is:    5050
 $
 $
```

Hello World

```
program main
   implicit none
   include 'mpif.h'
   integer :: ierr

   call MPI_INIT( ierr )

   write(*,*)  'Hello, world! '

   call MPI_FINALIZE( ierr )
end
```

Hello World

```
program he
 implicit none
 include 'mpif.h'

 integer :: id, np, ierr

 call MPI_Init(ierr)
 call MPI_Comm_Size(MPI_Comm_World, np, ierr)
 call MPI_Comm_rank(MPI_Comm_World, id, ierr)

 write(*,"(a,i1,a,i1)") "Hello world - rank/totproc = ",id,"/",np

 call MPI_Finalize(ierr)

end program he
```

Hello World -- output

```
program he
 implicit none
 include 'mpif.h'

 integer :: id, np, ierr

 call MPI_Init(ierr)
 call MPI_Comm_Size(MPI_Comm_World, np, ierr)
 call MPI_Comm_rank(MPI_Comm_World, id, ierr)

 write(*,"(a,i1,a,i1)") "Hello world - rank/totproc = ",id,"/",np

 call MPI_Finalize(ierr)

end program he
```

```
$ mpirun -np 8 ./a.out
Hello world - rank/totproc = 2/8
Hello world - rank/totproc = 7/8
Hello world - rank/totproc = 5/8
Hello world - rank/totproc = 1/8
Hello world - rank/totproc = 4/8
Hello world - rank/totproc = 6/8
Hello world - rank/totproc = 3/8
Hello world - rank/totproc = 0/8
```

Hello World – rank

```
program he
 implicit none
 include 'mpif.h'

 integer :: id, np, ierr

 call MPI_Init(ierr)
 call MPI_Comm_Size(MPI_Comm_World, np, ierr)
 call MPI_Comm_rank(MPI_Comm_World, id, ierr)

 if(mod(id,2)==0) then
   write(*,"(a,i1,a,i1)") "Hello world - rank/totproc = ",id,"/",np
 endif

 call MPI_Finalize(ierr)

end program he
```

Hello World – rank -- output

```
program he
 implicit none
 include 'mpif.h'

 integer :: id, np, ierr

 call MPI_Init(ierr)
 call MPI_Comm_Size(MPI_Comm_World, np, ierr)
 call MPI_Comm_rank(MPI_Comm_World, id, ierr)

 if(mod(id,2)==0) then
   write(*,"(a,i1,a,i1)") "Hello world - rank/totproc = ",id,"/",np
 endif

 call MPI_Finalize(ierr)

end program he
```

```
$ mpirun -np 8 ./a.out
Hello world - rank/totproc = 0/8
Hello world - rank/totproc = 2/8
Hello world - rank/totproc = 6/8
Hello world - rank/totproc = 4/8
$
```

Hello World – last rank

```
program he
 implicit none
 include 'mpif.h'

 integer :: id, np, ierr

 call MPI_Init(ierr)
 call MPI_Comm_Size(MPI_Comm_World, np, ierr)
 call MPI_Comm_rank(MPI_Comm_World, id, ierr)

 if(id==np-1) then
   write(*,"(a,i1,a,i1)") "Hello world - rank/totproc = ",id,"/",np
 endif

 call MPI_Finalize(ierr)

end program he
```

Hello World – output

```
program he
 implicit none
 include 'mpif.h'

 integer :: id, np, ierr

 call MPI_Init(ierr)
 call MPI_Comm_Size(MPI_Comm_World, np, ierr)
 call MPI_Comm_rank(MPI_Comm_World, id, ierr)

 if(id==np-1) then
   write(*,"(a,i1,a,i1)") "Hello world - rank/totproc = ",id,"/",np
 endif

 call MPI_Finalize(ierr)

end program he
```

```
$ mpirun -np 4 ./a.out
Hello world - rank/totproc = 3/4
$
$ mpirun -np 8 ./a.out
Hello world - rank/totproc = 7/8
$
```

What will be the output?

```
program test_mpi
 implicit none

 integer :: i, N, sum

 N = 100

 sum=0
 do i = 1, N
   sum = sum + i
 enddo

 write(*,"(a,i5,2x,a,i7)") "Sum of first ",N," numbers is: ", sum
end program test_mpi
```

Topic: MPI Programming

What will be the output?

```
program test_mpi
 implicit none

 integer :: i, N, sum

 N = 100

 sum=0
 do i = 1, N
   sum = sum + i
 enddo

 write(*,"(a,i5,2x,a,a,i7)") "Sum o
end program test_mpi
```

```
 $ mpif90 p1.f90 -o p1.x
 $ mpirun -np 1 ./p1.x
Sum of first   100   numbers is:    5050
 $
 $ mpirun -np 2 ./p1.x
Sum of first   100   numbers is:    5050
Sum of first   100   numbers is:    5050
 $
 $ mpirun -np 4 ./p1.x
Sum of first   100   numbers is:    5050
Sum of first   100   numbers is:    5050
Sum of first   100   numbers is:    5050
Sum of first   100   numbers is:    5050
 $
 $
```

Template

```
program sample_mpi
 Implicit none
 Include ‘mpif.h’
[other includes]
 integer :: ierr, nproc, rank
 [other declarations]
 call mpi_init(ierr)
 call mpi_comm_size(MPI_COMM_WORLD, nproc, ierr)
 call mpi_comm_rank(MPI_COMM_WORLD, rank, ierr)
:
Main part of the code
:
 call mpi_finalize(ierr)
end program
```

Hands-on

1. Write a MPI Fortran program to print "Successful" if the process id is divisible by 3.
2. Write a MPI Fortran program to calculate the average of the 'N' (take N=3000) random numbers for each process and print the result to the standard output (use nproc=4; use rand() internal function to generate random numbers)