

# Topic: MPI Programming

---

## Objective

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

# Topic: MPI Programming

---

## 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

# Topic: MPI Programming

---

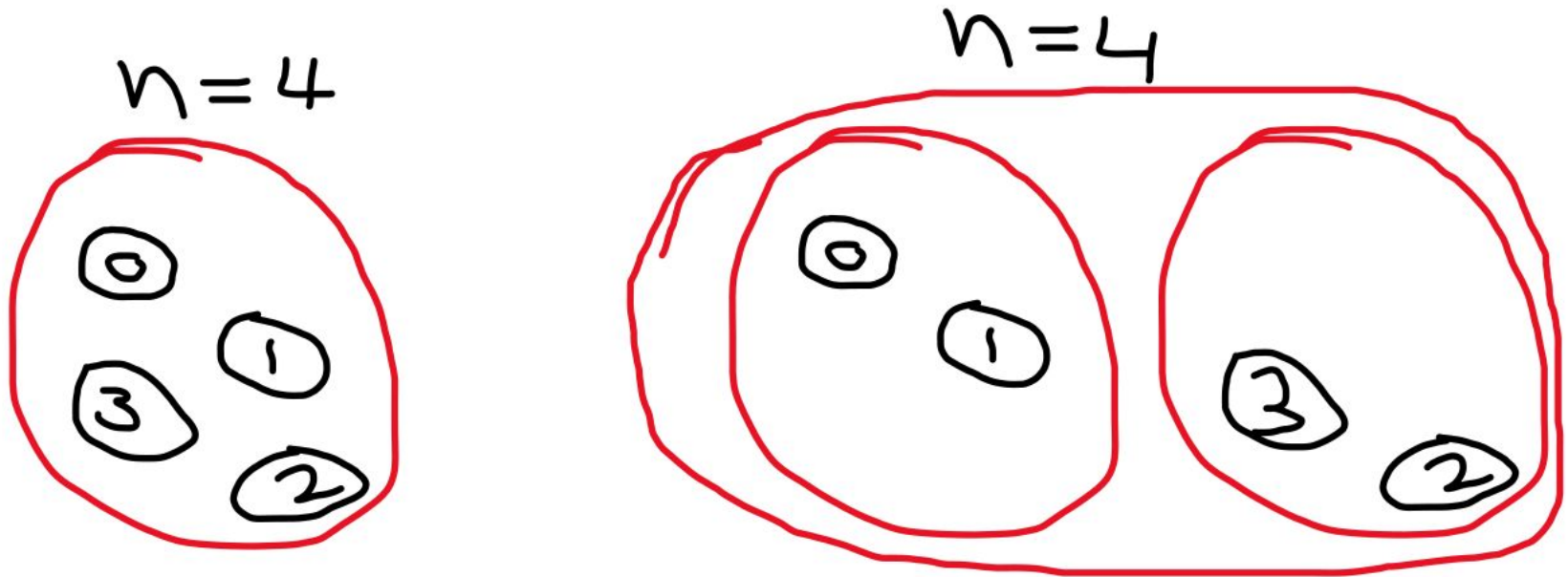
## Message passing

- 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

# Topic: MPI Programming

## 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

# Topic: MPI Programming

---

## 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

# Topic: MPI Programming

## 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*

# Topic: MPI Programming

## 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`

# Topic: MPI Programming

## 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
$
$ █
```



# Topic: MPI Programming

## Hello World

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

  call MPI_INIT( ierr )

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

  call MPI_FINALIZE( ierr )
end
```

# Topic: MPI Programming

## 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
```

# Topic: MPI Programming

## 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
```

# Topic: MPI Programming

## 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
```

# Topic: MPI Programming

## 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
$
```

# Topic: MPI Programming

## 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
```



# Topic: MPI Programming

## 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
$
```

# 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,i7)") "Sum of first ",N," numbers is: ", sum
end program test_mpi
```



# Topic: MPI Programming

## 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
```

# Topic: MPI Programming

```
program sum
```

Is this program OK?

```
  implicit none
```

```
  integer :: rank, nproc, ierr
```

```
  integer :: local_sum, global_sum, i, N
```

```
  N = 1000; call MPI_INIT(ierr) ; call MPI_COMM_SIZE(MPI_COMM_WORLD, nproc,  
  ierr)
```

```
  call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
```

```
  local_sum = 0
```

```
  do i = rank + 1, N, nproc
```

```
    local_sum = local_sum + i
```

```
  end do
```

```
  if (rank == 0) then
```

```
    write(*,*) "Sum:", local_sum
```

```
  end if
```

```
  call MPI_FINALIZE(ierr)
```

# Topic: MPI Programming

---

## Is this program OK?

program example

```
include 'mpif.h'
```

```
implicit none
```

```
integer :: rank, nproc, ierr
```

```
call MPI_INIT(ierr)
```

```
call MPI_COMM_SIZE(MPI_COMM_WORLD, nproc, ierr)
```

```
call MPI_COMM_RANK(MPI_COMM_WORLD, nproc, ierr)
```

```
write(*,*) "Hello from process", rank, "out of", nproc, "processes."
```

```
call MPI_FINALIZE()
```

```
end program example
```

# Topic: MPI Programming

Is this program OK?

```
program mpi_mistake_example
  implicit none
  integer :: rank, nproc, ierr

  call MPI_INIT(ierr); call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
  call MPI_COMM_SIZE(MPI_COMM_WORLD, nproc, ierr)

  if (rank == 0) then
    write(*,*) "Hello from process", rank, "out of", nproc, "processes."
  end if

  if (rank == 0) then
    call MPI_FINALIZE(ierr)
  end if
  call MPI_FINALIZE(ierr)
end program mpi_mistake_example
```

# Topic: MPI Programming

---

## Hands-on

1. Write a MPI Fortran program to print "Successful" if the process id is odd number.
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=2)

# Topic: MPI Programming

---

## Reading material

- Check the website given below for a well-written hands-on tutorial on MPI <https://mpitutorial.com/tutorials/>
- Peter Pacheco, "Parallel Programming with MPI"