CS528 OpenMP and MPI

A Sahu

Dept of CSE, IIT Guwahati

Outline

- Implicit/Auto Thread Pooling: OpenMP
- Distributed Memory Programming Model
  - Message Passing Interface (MPI)
  - MPI Programming

Trend of HPC

- **HPC system**
  - Multi Nodes/Computer/Blades
  - **Programming Model MPI**

Motivation for Parallel Programming

- Solving large problem **(HPC)**
  - Scientific simulation ,computation, CFD, data analytics, ..

Programming Model

- Shared memory Programming Model
  - Pthread, Cilk, OpenMP, Vectorized
- **Distributed Meory Programming Model**
  - **MPI**
  - **Large Scale**

Writing Parallel Program using MPI

Writing Parallel Program

- Given a problem
- Design Solution/algorithm
- Design Solution with Parallel Algorithmic Technique

Generic Parallel Algorithm Design: Foster's Methodology

- **Partitioning :** Process of dividing the computation and data into pieces
  - A good partitioning spilt both into many pieces
  - **Domain decomposition :** divide data into pieces and associate computation with data
- **Communication :** Indentify communication pattern between partition and intra partition
- **Agglomeration:** Process of grouping partition/task into larger task to reduce communications
- **Mapping :** Assigning task to processor

Message Passing Interface (MPI)

The Message-Passing Model

- A *process* is program with ex. with PC and address space.
- Processes may have multiple *threads* (PCs and associated stacks) sharing a single addr. space.

The Message-Passing Model

- A *process* is program with ex. with PC and address space.
- Processes may have multiple *threads* (PCs and associated stacks) sharing a single addr. space.
- MPI is for communication among processes, which have separate address spaces.
- Interprocess communication consists of
  - Synchronization
  - Movement of data from one process's address space to another's.

How to install MPI in Linux machine

```
$sudo apt-get install mpich2 mpich2-doc
```

Or

```
$sudo dnf install openmpi
```

How to install MPI in Linux machine

```
$sudo apt-get install mpich2 mpich2-doc
```

Or

```
 $sudo dnf install openmpi
```

```
$mpicc hello_mpi.c –o hello_mpi
```

How to install MPI in Linux machine

```
$sudo apt-get install mpich2 mpich2-doc
```

Or

```
$sudo dnf install openmpi
```

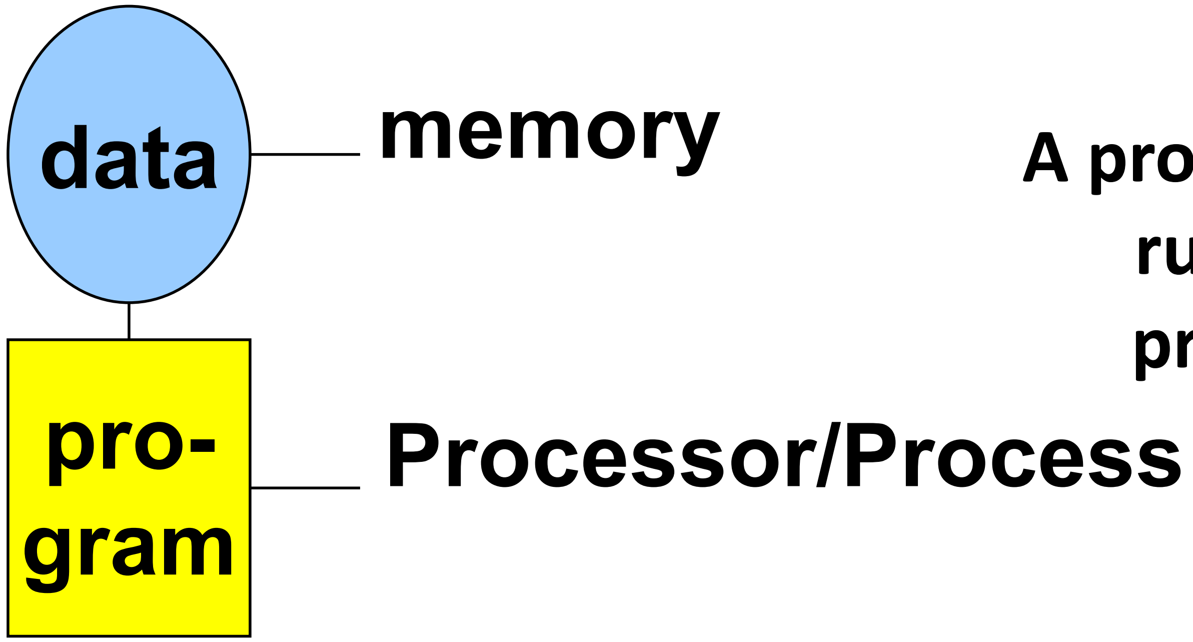
```
$mpicc hello_mpi.c –o hello_mpi
$mpirun –np 4 ./hello_mpi
```

mpiCC, mpicxx, mpif77, mpif90

4 copy of hello_mpi process will run

The Message-Passing Programming Paradigm

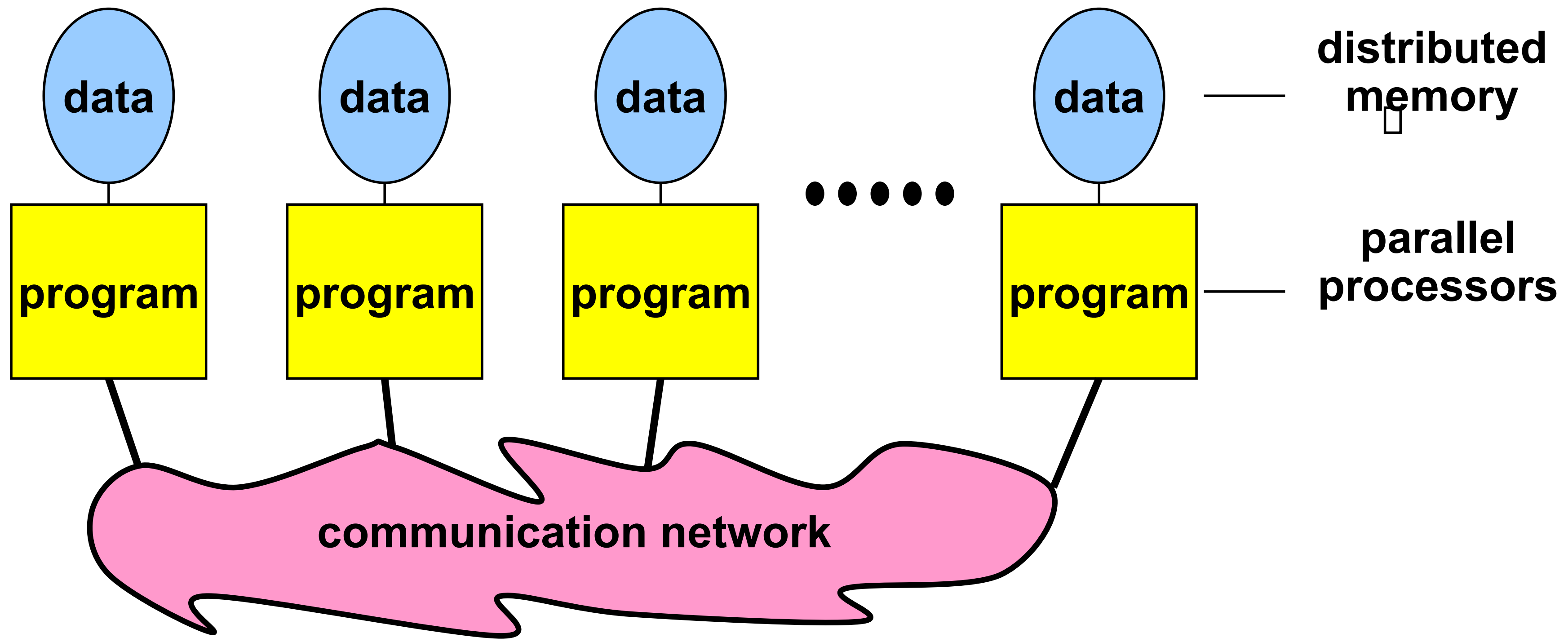
- Sequential Programming Paradigm



A processor may run many processes

The Message-Passing Programming Paradigm

- **Message-Passing Programming Paradigm**



The Message-Passing Programming Paradigm

- A **process** is a program performing a task on a **processor**
- Each processor/process in a message passing program runs a instance/copy of a ***program***
- Written in a conventional sequential language, e.g., C or Fortran,

The Message-Passing Programming Paradigm

- Typically a single program operating of multiple dataset
- The variables of each sub-program have
  - **The same name**
  - **But different locations (distributed memory) and different data!**
  - **i.e., all variables are local to a process**
- Communicate via special send & receive routines (***message passing***)

Every process of MPI are different

- Hi : single person : you do
  - Touch you nose by left hand
  - Hi : Touch you head by right hand
- Hi: all persons of this hall do:
  - Touch your nose

Every process of MPI are different

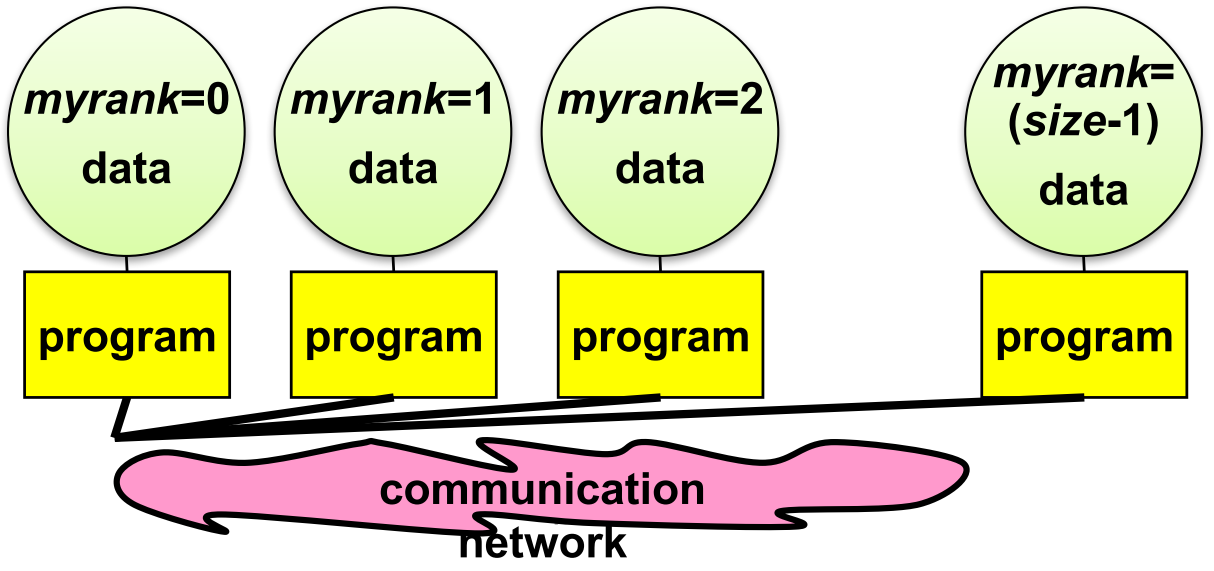
- How to do work collaboratively : MPI program
- Assume 10 persons : want to do sum of n numbers
- First person : manager responsible for I/O
  - **Get input from KBD**
  - **Send one data to each person**
  - Get Sum from 2nd person
  - **Display the SUM**
- All persons : every person have rank/ID-number
  - Receive a data from master
  - Receive a SUM from rank+1 person if i<10
  - If rank=10 SUM = Number else SUM=SUM+Number
  - Send the number of rank-1 person.

Do work collaboratively : MPI program

```
main(){  int D, SUM, rank, data[N]; //private data
 if (rank==MASTER){
      Get_inputs_from_KBD()
      Send_one_data_to_each_person();//SCATER();
      Get_ Sum_(SUM,From_2ND_person); S=NUM+SUM;
      Display_the_SUM();
 }  else {
    Receive_a_data_from_master(D, MASTER);
    if i<10
      Receive  a  SUM from rank+1 person
    If (rank==N) SUM = D;   else SUM=SUM+D
     Send_the_number(SUM, RANK-1);
    }
}
```

Data and Work Distribution

- To communicate together mpi-processes need identifiers: **rank = identifying number**
- all distribution decisions are based on the ***rank***
  - i.e., which process works on which data



What is SPMD

- **S**ingle **P**rogram, **M**ultiple **D**ata
- Same (sub-)program runs on each processor
- MPI allows also MPMD, i.e., **Multiple** Program, ...
  - but some vendors may be restricted to SPMD
  - MPMD can be emulated with SPMD

Emulation of MPMD

```
main(int argc, char **argv){
     if (myrank < XX){
        ocean( /* arguments */ );
     }else{
       weather( /* arguments */ );
     }
  }
```