Introduction to MPI

ICHEC
Irish Centre for High-End Computing

An Roinn Post, Fiontar agus Nuálaíochta
Department of Jobs, Enterprise and Innovation
sfi science foundation ireland
AN ROINN OIDEACHAIS AGUS SCILEANNA
DEPARTMENT OF EDUCATION AND SKILLS
HEA Higher Education Authority
an tÚdarás um Ard-Oideachas
www.ichec.ie

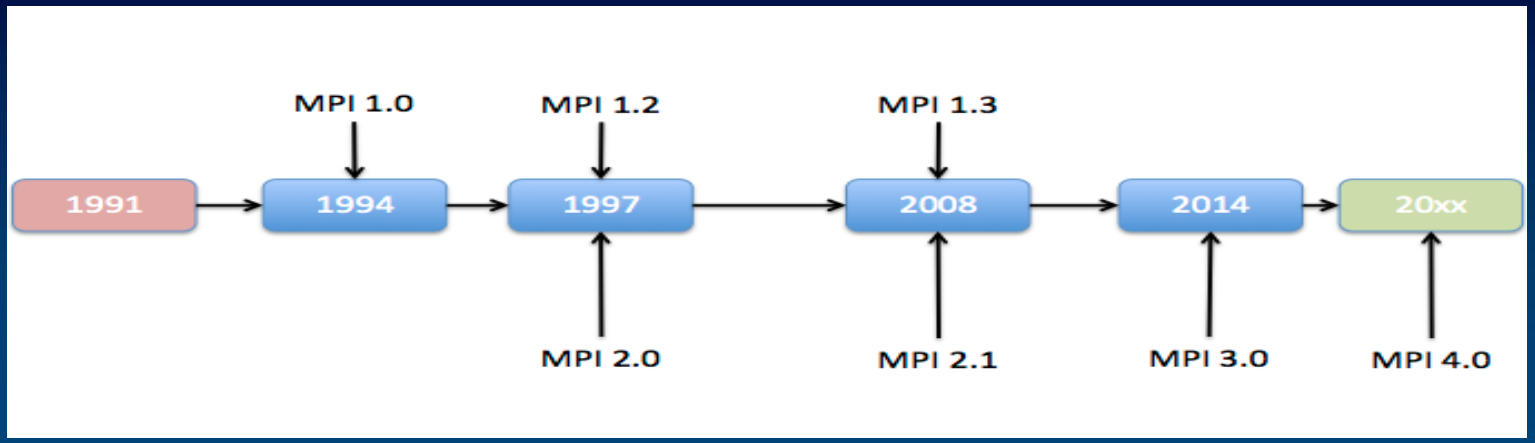
ICHEC
Irish Centre for High-End Computing

MPI (Message Passing Interface)?

- Standardized message passing library specification (IEEE)
  - for parallel computers, clusters and heterogeneous networks
  - not a specific product, compiler specification etc.
  - many implementations, MPICH, LAM, OpenMPI …
- Portable, with Fortran and C/C++ interfaces.
- Many functions
- Real parallel programming
- Notoriously difficult to debug

ICHEC
Irish Centre for High-End Computing

Information about MPI

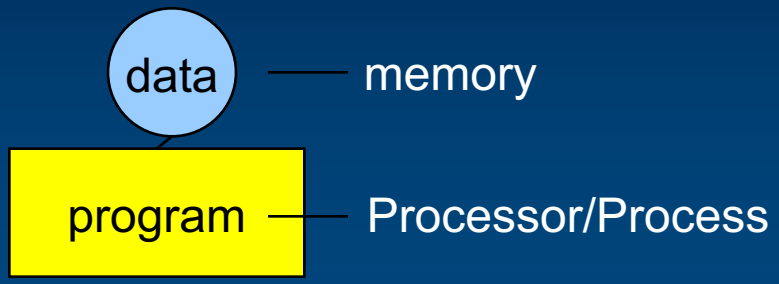


- **http://www.mpi-forum.org/docs/**
- **MPI: The Complete Reference**, Marc Snir and William Gropp et al, The MIT Press, 1998 (2-volume set)
- **Using MPI: Portable Parallel Programming With the Message-Passing Interface** and **Using MPI-2: Advanced Features of the Message-Passing Interface.** William Gropp, Ewing Lusk and Rajeev Thakur, MIT Press, 1999 – also available in a single volume *ISBN 026257134X.*
- **Parallel Programming with MPI**, Peter S. Pacheco, Morgen Kaufmann Publishers, 1997 - *very good introduction.*
- **https://computing.llnl.gov/tutorials/mpi/**

ICHEC
Irish Centre for High-End Computing

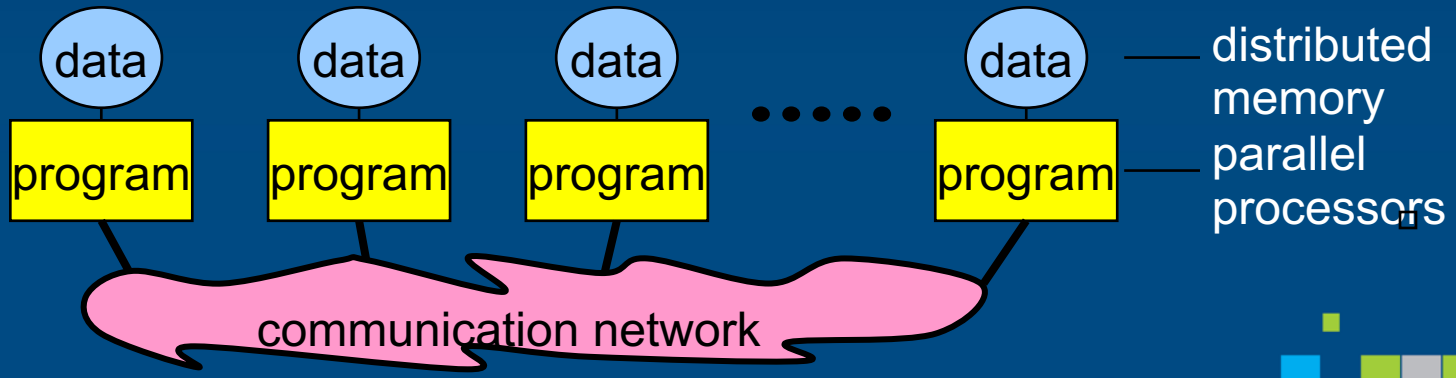
The Message-Passing Programming Paradigm

- **Sequential Programming Paradigm**



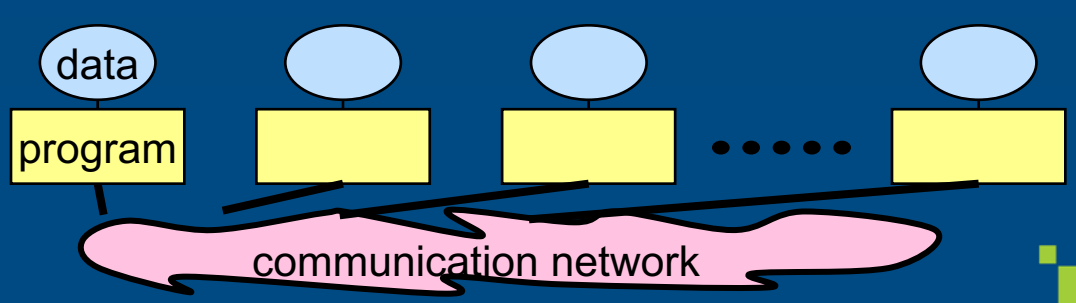
A processor may
run many processes

- **Message Passing Programming Paradigm**



ICHEC
Irish Centre for High-End Computing

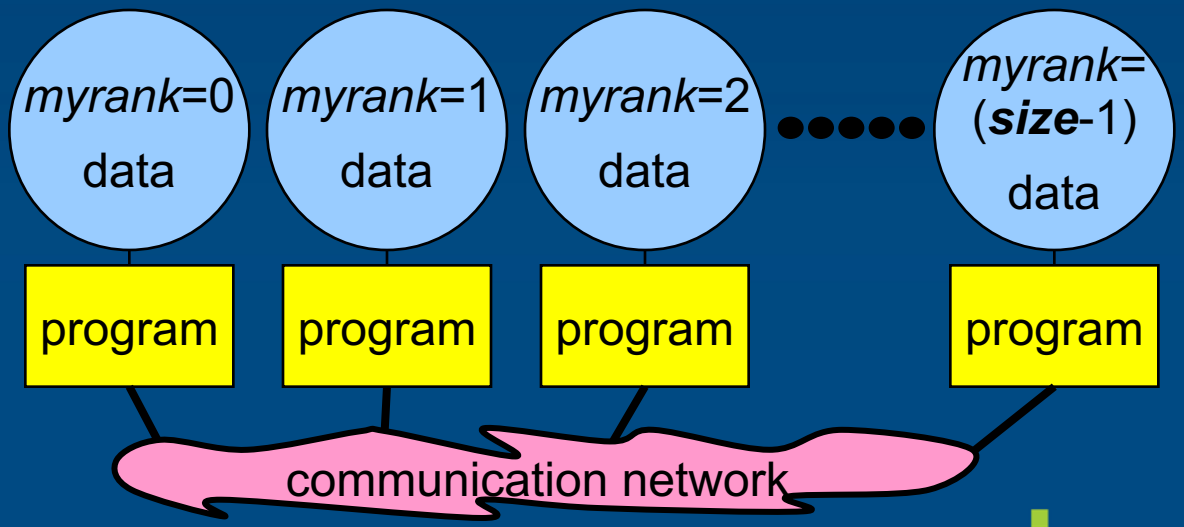
- A **process** is a program performing a task on a **processor/core**
- Each processor/process in a message passing program runs a instance/copy of a ***program:***
  - written in a conventional sequential language: C/C++, Fortran, python
  - typically a single program operating on multiple dataset
  - 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***)



ICHEC
Irish Centre for High-End Computing

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



ICHEC
Irish Centre for High-End Computing

Example : sum of elements of a vector

- **Sequential code**

```
sum = 0
for (int i = 0; i < 1000 ;++i)
    sum = sum + array[i] ;
```

- **parallel code**

```
sum = 0
chunkSize = 1000 / numProc
for (int i= rank*chunkSize; i< (rank + 1)*chunkSize; ++i)
                sum = sum + array[i] ;

    if( rank != root)
      send (partialsum) to root
    else
      receive(partialsum) from workers
```

ICHEC
Irish Centre for High-End Computing

On each processor

```
sum = 0
for (int i= 0; i < 500; ++i)
    sum = sum + array[i] ;
```

P_0

```
sum = 0
for (int i = 500; i < 1000; ++i)
    sum = sum + array[i] ;
```

P_1

- The same program
- The same variables, but different values

ICHEC
Irish Centre for High-End Computing

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

ICHEC
Irish Centre for High-End Computing

Emulation of MPMD

- C/C++:

```
main(int argc, char **argv) {
if (myrank < .... /* process should run the ocean model */){
      ocean( /* arguments */ );
   } else {
     weather( /* arguments */ );
   }
 }
```

- Fortran

```
program forecast
if (myrank < ... ) then   !! process should run the ocean model
        call  ocean ( some arguments )
ELSE
        call  weather ( some arguments )
endif
end program forecast
```

ICHEC
Irish Centre for High-End Computing

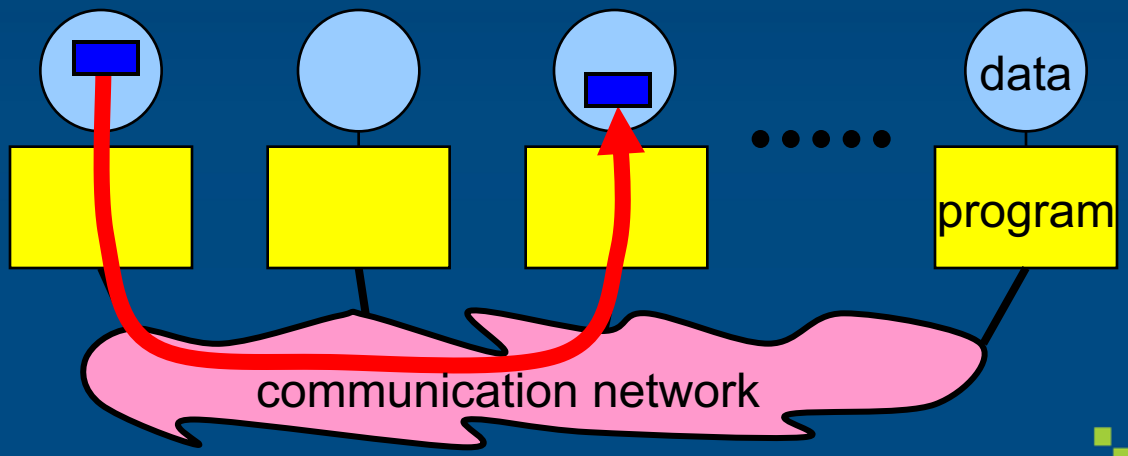
Message Passing System

- A sub-program needs to be connected to a message passing system
- A message passing system is similar to:
  - phone line
  - mail box
  - fax machine
  - etc.
- MPI:
  - program must be linked with an MPI library
  - program must be started with the MPI startup tool

ICHEC
Irish Centre for High-End Computing

Message passing

- Messages are packets of data moving between sub-programs
- Necessary information for the message passing system:
  - sending process – receiving process i.e., the ranks
  - source location – destination location
  - source data type – destination data type
  - source data size – destination buffer size



ICHEC
Irish Centre for High-End Computing

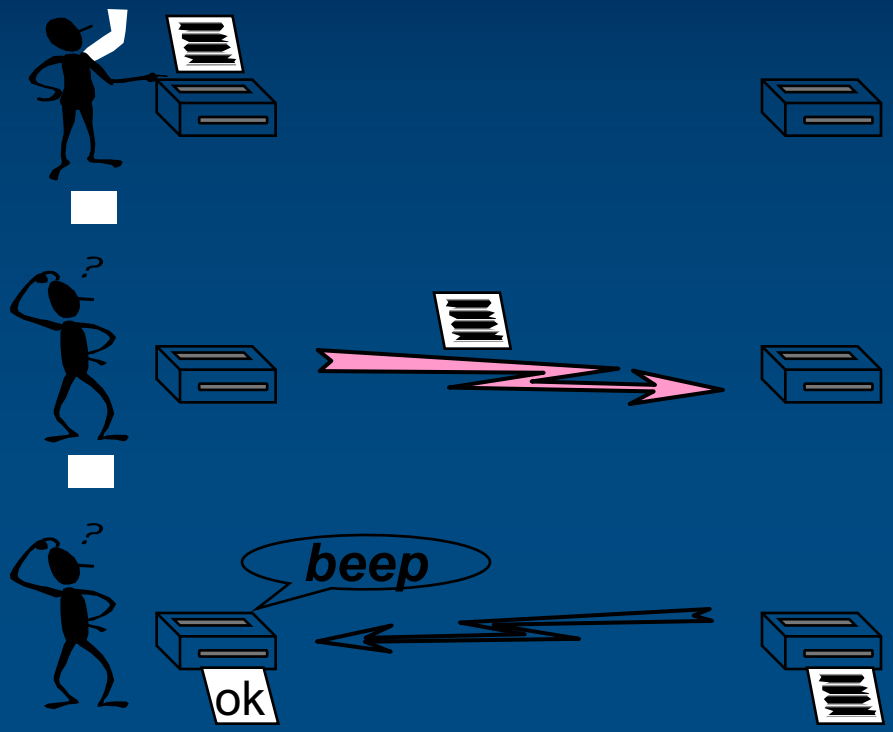
Point-to-Point Communication

- Simplest form of message passing.
- One process sends a message to another.
- Different types of point-to-point communication:
  - synchronous send
  - buffered = asynchronous send

ICHEC
Irish Centre for High-End Computing

Synchronous Sends

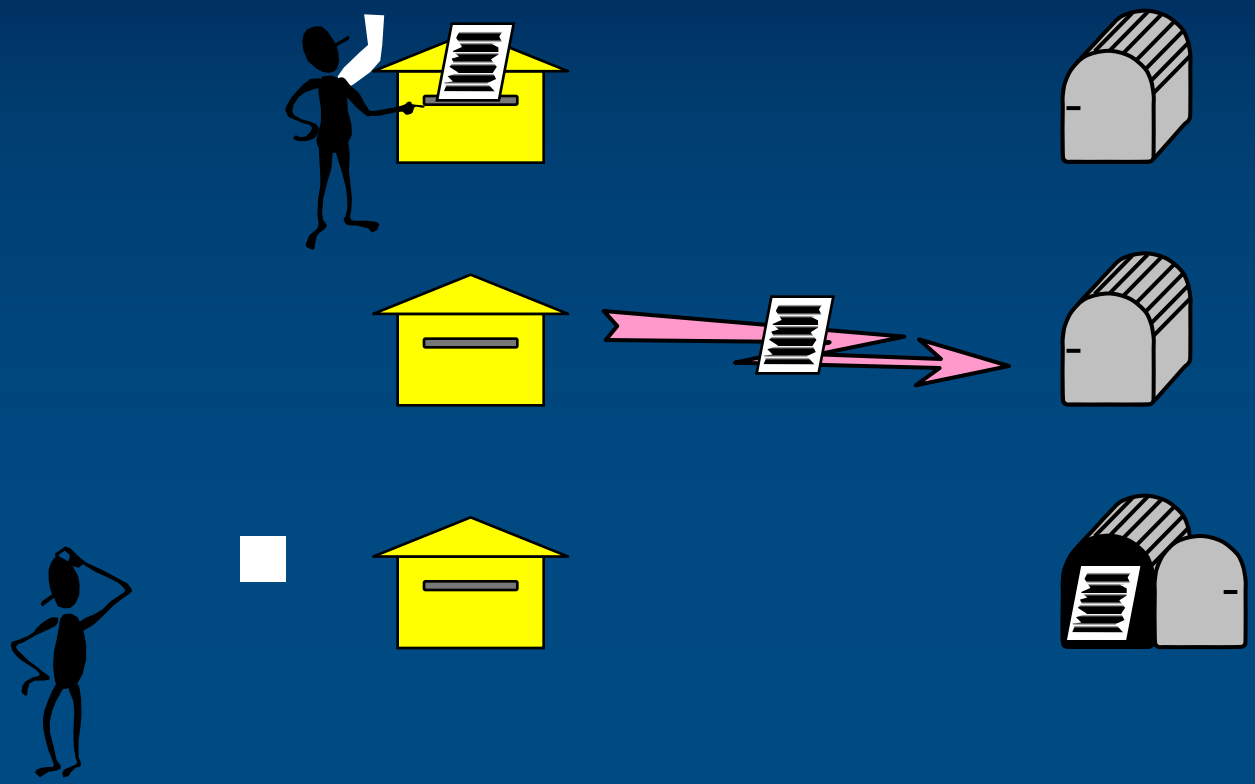
- The sender sends data and waits until it gets an information that the message is received.



ICHEC
Irish Centre for High-End Computing

Buffered = Asynchronous Sends

- Only know when the message has left.



ICHEC
Irish Centre for High-End Computing

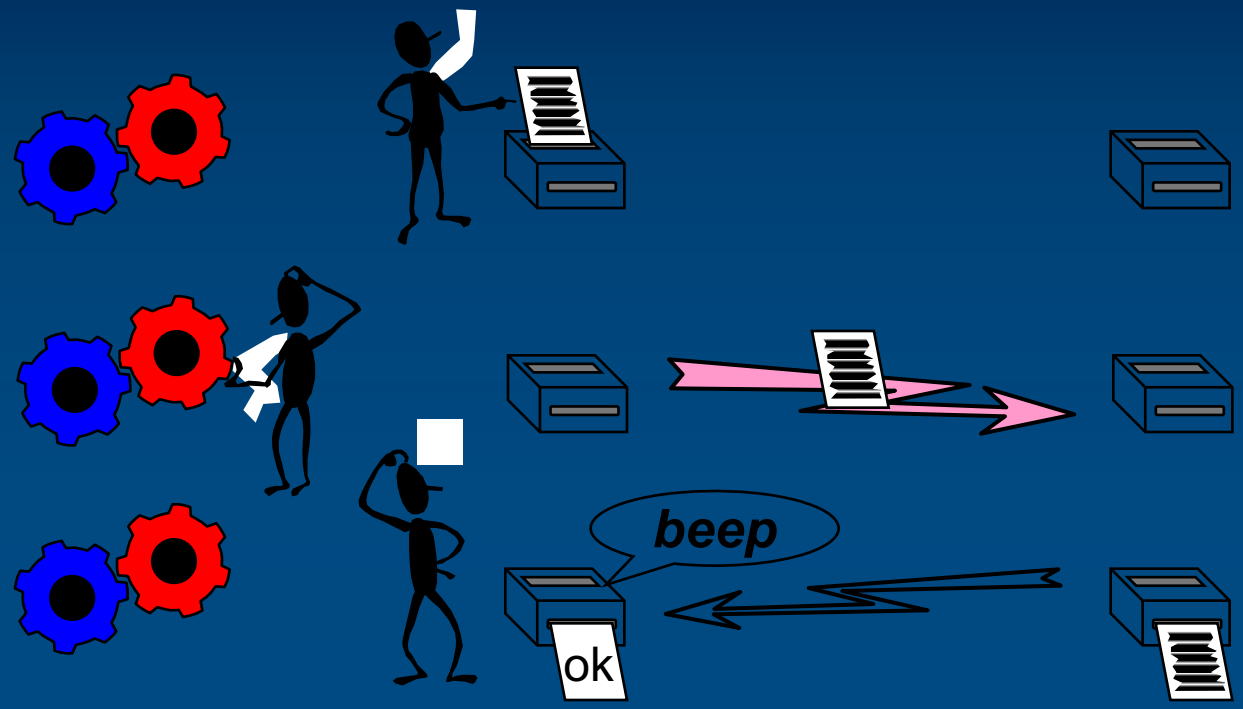
Blocking Operations

- Some sends/receives may **block** until another process acts:
  - synchronous send operation **blocks until** receive is issued;
  - receive operation **blocks until** message is sent.
- Blocking subroutine returns only when the operation has completed.

ICHEC
Irish Centre for High-End Computing

Non-Blocking Operations

- Non-blocking operations return immediately and allow the sub-program to perform other work.



ICHEC
Irish Centre for High-End Computing

Collective Communications

- Collective communication routines are higher level routines.
- Several processes are involved at a time.
- May allow **optimized internal** implementations, e.g., tree based algorithms

ICHEC
Irish Centre for High-End Computing

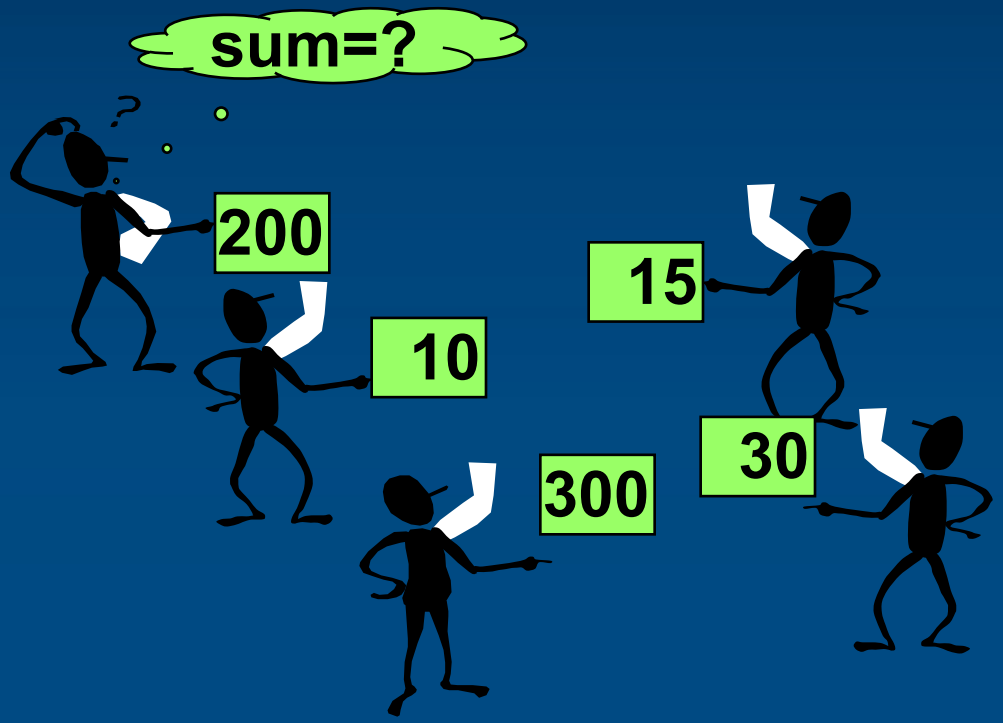
Broadcast

- A one-to-many communication.

ICHEC
Irish Centre for High-End Computing

Reduction Operations

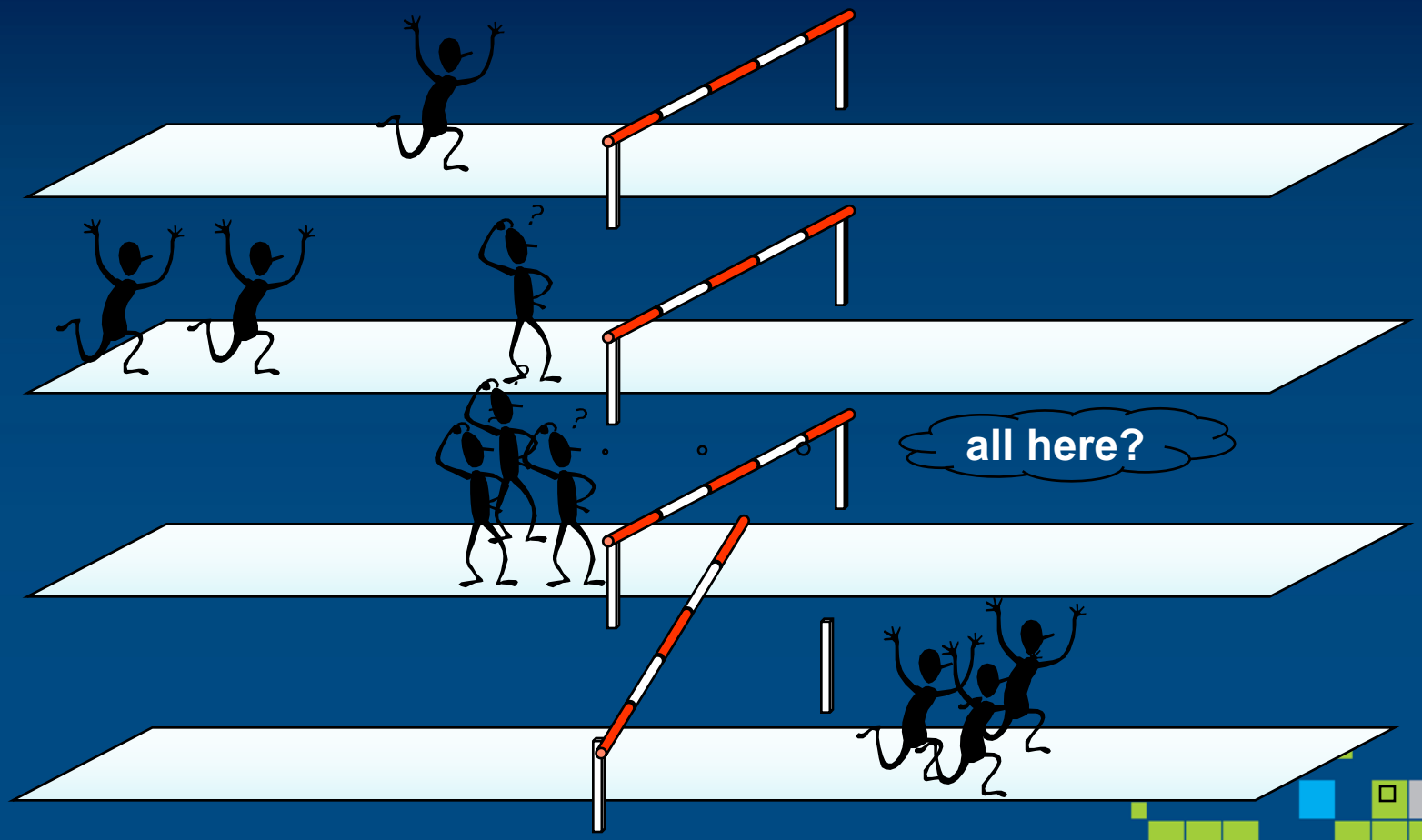
- Combine data from several processes to produce a single result.



ICHEC
Irish Centre for High-End Computing

Barriers

- Synchronize processes.



ICHEC
Irish Centre for High-End Computing

Goals and Scope of MPI

- MPI's prime goals
  - To provide a message-passing interface.
  - To provide source-code portability.
  - To allow efficient implementations.
- It also offers:
  - A great deal of functionality.
  - Support for heterogeneous parallel architectures.
- With MPI-2/MPI-3:
  - Important additional functionality.
  - Backward compatibility with MPI-1.