MPI

- MPI (Message Passing Interface) is a **language agnostic cross-platform standard for message passing computation**. This interface allows and facilitates a **Distributed Memory multi-computer system**.
- This standard actuates on a network that allows communications between pairs of CPU and Memory.
- Because it is language agnostic, can be programmed in various languages (C, Fortran, C++)
- The open source implementation is called **OpenMPI**, which has a lot of collaborators, like Intel, nVIDIA, ADM, IBM, Cisco, Facebook, and more.

Syntax basics

- Primitive communication functions
- Barrier and Broadcast
- Non-blocking Communication

Primitive communication functions

These functions covers the basic functionalities of MPI. By stablishing a MPI communication protocol (*MPI_Comm* variable), one process can communicate with another one

- **MPI_Send**
  - Makes possible a process send a message to another one.
  - Uses **rank** variable as an argument. It defines where to send.
- **MPI_Recv**
  - Makes possible a process receive a message from another one.
  - Uses **rank** variable as an argument. It defines where to receive from.

Is important to note that a communicate link between processes will only be possible if **both sides have theses functions** *MPI_Send* and *MPI_Recv* respectively.

- **MPI_Sendrecv**
  - A simultaneous transmitting (*MPI_Send*) and receiving (*MPI_Recv*) operation.

Barrier and Broadcast

- **MPI_Barrier**

    - When a process calls it, it will stop and wait every other process in the same communication protocol (e.g. *MPI_Comm*) calls *MPI_Barrier()*. Then, when all processes reach this point, all the processes continues running at the same time.
    - Uses **communication protocol** variable as an argument.
- **MPI_Bcast**
    - Uses **rank** variable as an argument.
    - When a process calls it, if its rank its the same as the argument passed into the *MPI_Bcast()* function, then it will **send** data. Otherwise, it will **receive** data from the broadcaster.

Non-blocking Communication

Maybe the Non-blocking Communication is the most important feature of MPI. This type of communication allows processes to **run concurrently sending and receiving data** from other processes. In another words, sending/receiving data **without interrupting the computation**.

For sending and receiving without interrution, there are two functions:

- **MPI_ISend**
    - Whether this function is called, returns to work before the data is copied out of the process buffer. It will **overwrite the buffer** whenever a new function *MPI_ISend()* is called.
- **MPI_IRecv**
    - Whether this function is called, returns to work before the data is received and copied into the process buffer. It will **want to use the data from the buffer** the instant the function *MPI_IRecv()* is called.

Is important to note that these functions will operate based on the buffer, and will not check if the data has already been sent (in *MPI_ISend*) or has already been updated (in *MPI_IRecv*) from/in this buffer.
For control purpose, each one of then has a **request** variable as an argument. This request will be used to trigger two other functions that can be used to control the process:

- **MPI_Test**
    - This function will **check** if the request variable has been completed.
- **MPI_Wait**
    - This function will **wait** for the request variable to be completed. It will stop the process and will wait for a *MPI_Test()* to be successful.
    - Even though it effectively depends of the validity of *MPI_Test()* function, the latter can be called before *MPI_Wait()* (without interrupting the process), and if it checks that the request has been completed, *MPI_Wait()* will be skipped immediately when called.

**