Parallel Computing Systems and Applications
MPI

Salvatore Filippone

salvatore.filippone@uniroma2.it

The MPI standard

MPI: Message Passing Interface

- Anybody can participate https://www.mpi-forum.org/
- Started in the early 90s;
- Currently at version 3.1;
- Version 4.0 currently under development;
- Implemented by multiple vendors/groups;

Combined the experience of multiple research project active in the 80s and 90s.
Open source implementations (in many Linux distributions):

- MPICH https://www.mpich.org/
- OpenMPI https://www.open-mpi.org/
- MVAPICH https://mvapich.cse.ohio-state.edu/

Plus multiple vendors (e.g. Intel, Cray etc.)

The MPI standard

From the MPI 3.1 standard document

MPI (Message-Passing Interface) is a message-passing library interface specification.

All parts of this definition are significant.

- MPI addresses primarily the message-passing parallel programming model, in which data is moved from the address space of one process to that of another process through cooperative operations on each process.
- Extensions to the "classical" message-passing model are provided in collective operations, remote-memory access operations, dynamic process creation, and parallel I/O. MPI is a specification, not an implementation; there are multiple implementations of MPI.
- This specification is for a library interface; MPI is not a language, and all MPI operations are expressed as functions, subroutines, or methods, according to the appropriate language bindings which, for C and Fortran, are part of the MPI standard.

The MPI standard

Whenever you are in doubt

Go read the standard

You can get it from the MPI Forum website.

But, be prepared for the need to interpret what you read

Hello World

```
#include <stdio.h>
#include <string.h>
#include <mpi.h>

main(int argc, char* argv[])
  {
    int my_rank;
    int p;
    int source;
    int dest;
    int tag=0;
    char message[100];
    MPI_Status status;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COM_WORLD, &my_rank);
    MPI_Comm_size(MPI_COM_WORLD, &p);
```

Hello World

```
  if (my_rank != 0) {
    sprintf(message,"Greetings from process %d!",my_rank);
    dest = 0;
    MPI_Send(message,strlen(message)+1,MPI_CHAR,
                  dest,tag,MPI_COMM_WORLD);
  } else {
    for(source=1; source <p; source++){
      MPI_Recv(message,100,MPI_CHAR,
                  source,tag,MPI_COMM_WORLD,&status);
      printf("%s\n",message);
    }
  }

  MPI_Finalize();
}
```

Notes on Hello World

Compilation process

Every MPI implementation provides scripts that take care of includes files and library links:

- Commonly used script names: `mpicc`, `mpifort`, `mpicxx`;
- May change among implementations (ex: `mpiCC`, `ftn`);
- To be used in place of `cc` and friends;

Running your program

The process of starting a program determines how many processes are available:

- Commonly used: `mpirun -np N`
- Alternatives: `mpiexec`, `srun`;
- You should **never** hardwire the number of processes in your code