Report Exercise 1
High Performance Computing

Alessandro Della Siega

May 22, 2024

1 Introduction

The OpenMPI library implements several algorithms to perform collective blocking operations according to many different parameters. In this exercise, two operations are take into consideration: broadcast and reduce. In order to evaluate the performance of those collective operations, the OSU benchmark tool is used. The computational resources used to complete the exercise are the ones provided by the ORFEO cluster.

1.1 Architecture

The ORFEO system architecture consists of different machines architecture, in this project the THIN partition is used. The THIN partition consist of 12 Intel nodes: two equipped with Xeon Gold 6154 and 10 equipped with Xeon Gold 6126 cpus. The main difference in the hardware is the available RAM, the first ones have . Internal connectivity in ORFEO is handled by a 25 Gbit/s Ethernet Network for general TCP/IP and by a 100 Gbit/s Infiniband link for Remote Direct Memory Access (RDMA) between the compute nodes.

1.2 Broadcast

The broadcast operation `MPI_Bcast` is a one-to-all communication operation which allows a process (typically the root process) to distribute information across many processes. This operation is implemented in OpenMP using different algorithms. The algorithms that are considered in this analysis are:

- **basic linear**, the root process sends to all the other process the message;
- **chain**, if we order the processes with respect to their rank, each process sends the message to the next process;
- **split binary tree**, suppose that each process is a node of a binary tree and the root node is the root process, each parent node sends the message to its children. As the name of the algorithm suggests, the message is split in two halves before transmission and when each half of the message reaches its destination, it is necessary to merge the two halves.

1.3 Reduce

The reduce operation `MPI_Reduce` combines the elements provided in the input buffer of each process in the group using the specified operation, and returns the combined value in the output buffer of the root process.

- **linear**, the root process sends a release message to each process;
- **double ring**, if we order the processes with respect to their rank in a circular fashion, each process send a release message to the next process. A process is released only after it receives the message for the second time

- **recursive doubling**, at step k, node with rank r exchanges message with node $rXOR2^k$

to be continued...