High Performance Computing

Exercise 1

Annalisa Paladino

Introduction

Task:
To estimate the *latency* of various openMPI algorithms for the two collective operations **broadcast** and **scatter** by changing message size, number of MPI processes and allocation type. To build a *performance model* for the latency

Architecture:

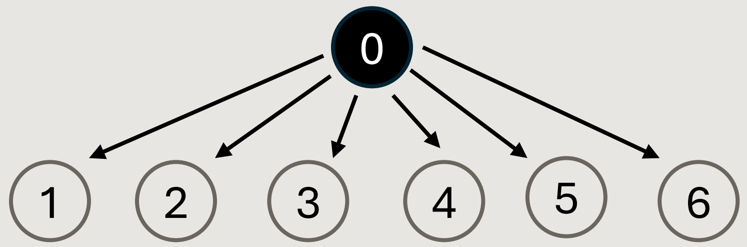
- 2 **THIN** nodes of the **ORFEO** cluster
- 2 sockets per node
- 12 cores per socket

Methodology:

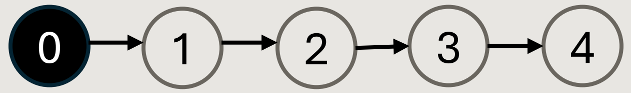
- **OSU benchmark** to estimate the latency
- Bash scripts to automate data gathering phase
- **SLURM** workload manager
- R for data analysis and models

Broadcast algorithms

1. **Basic Linear** (bcast 1):



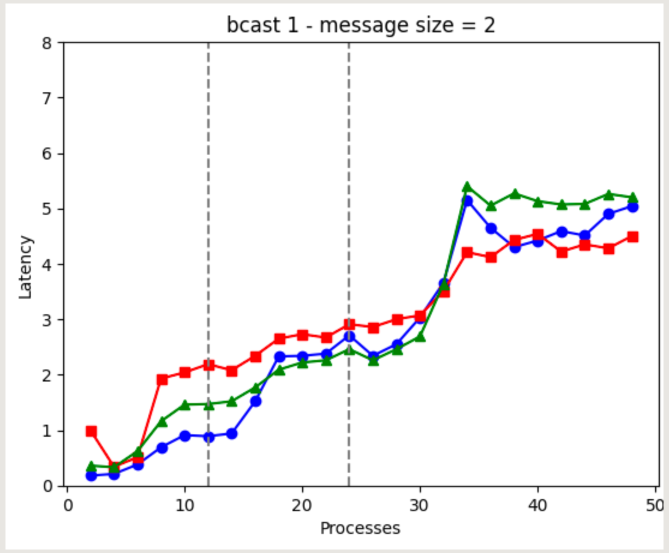
2. **Chain** (bcast 2):

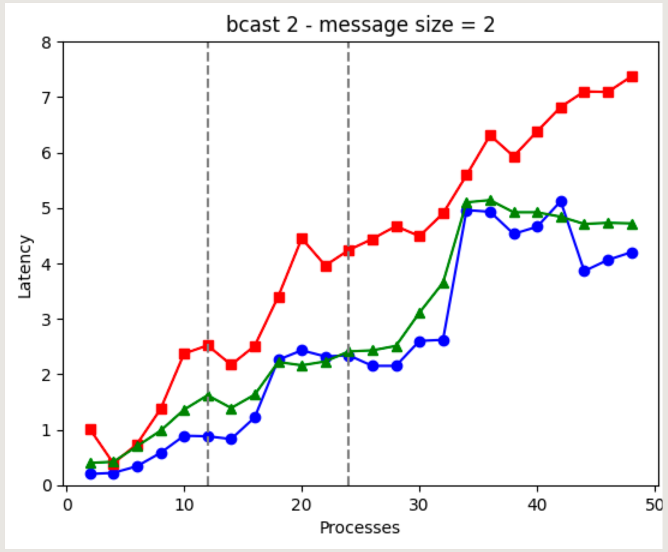


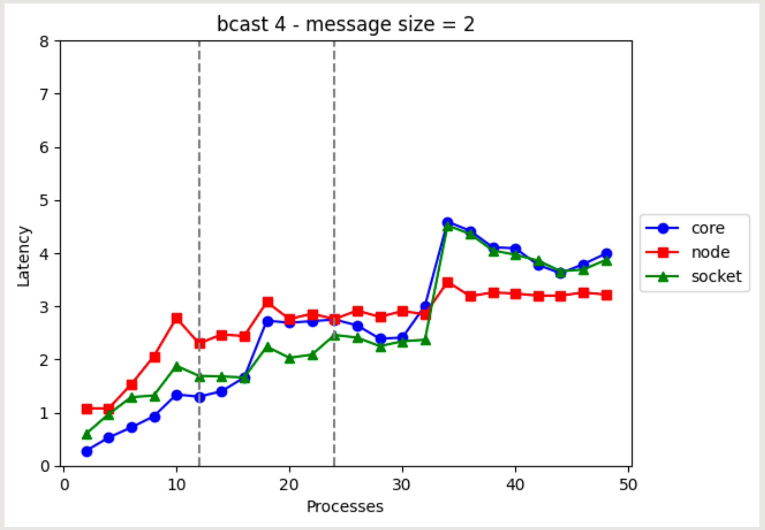
3. **Split Binary Tree** (bcast 4): The root process sends the message to two processes, and each internal process thereafter has two children. Before initiating communication, the message is divided in half. The left branch of the tree transmits the left half of the message, and similarly, the right branch transmits the right half. In the final step of the operation, the right and left nodes exchange their respective halves of the message to complete the broadcast

Bcast: allocation type comparison

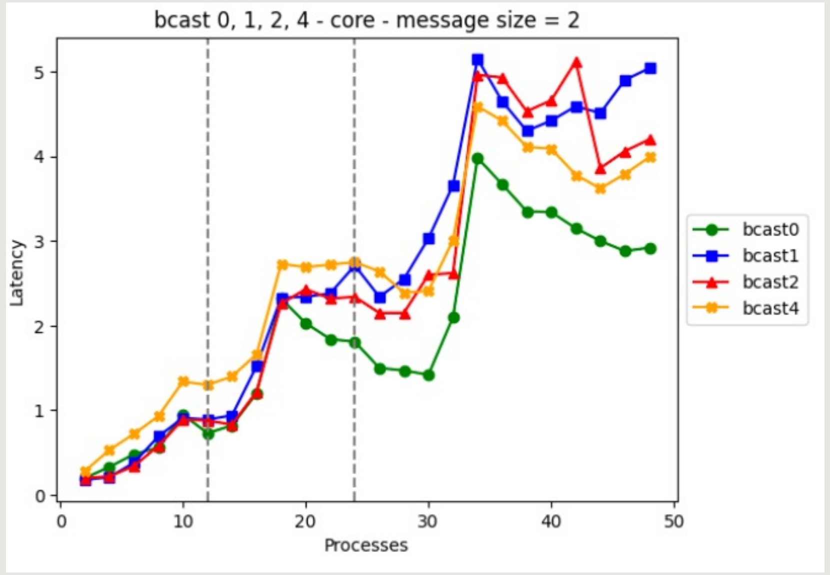
with message size fixed to 2 MPI_CHAR





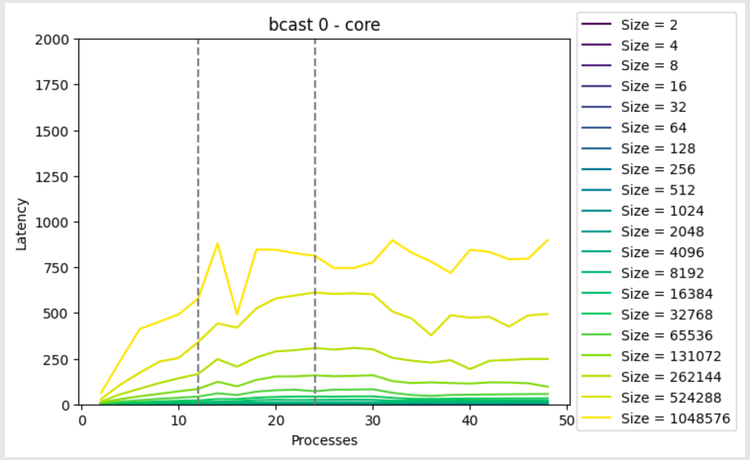


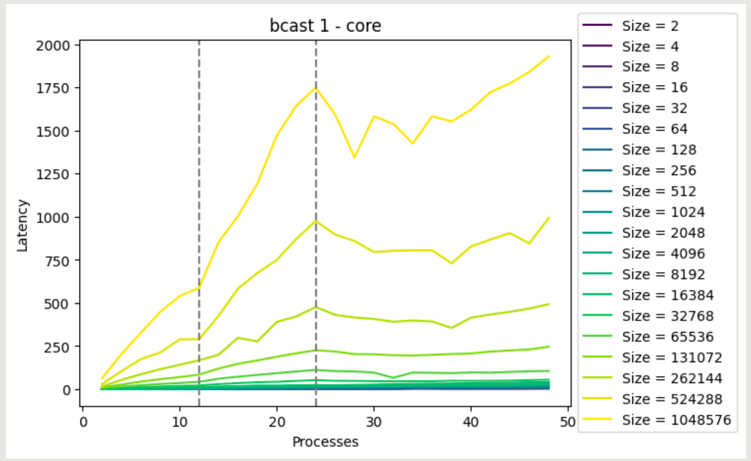
- Socket and core allocations show latency (μs) "jumps" when changing node, while node allocation progresses seamlessly
- Similar considerations apply to socket changes
- Jumps occur at 16 and 32, hinting at additional factors influencing communication
- Lines convergence: Linear vs Chain algorithm
- Split Binary Tree algorithm (expected) superior performance



Bcast: algorithms comparison

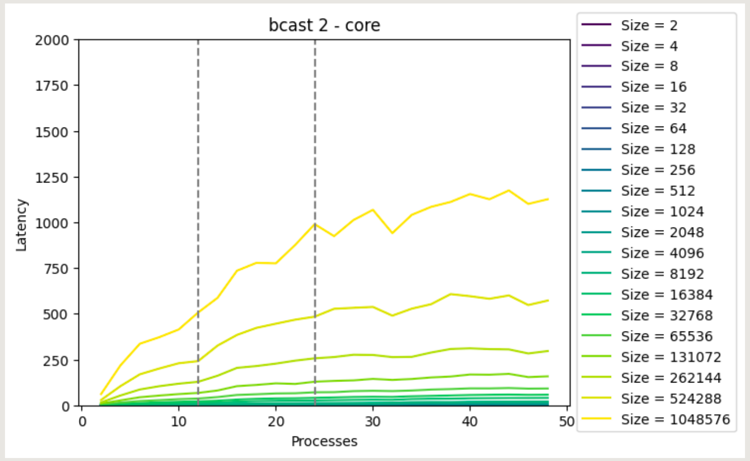
non-fixed message size

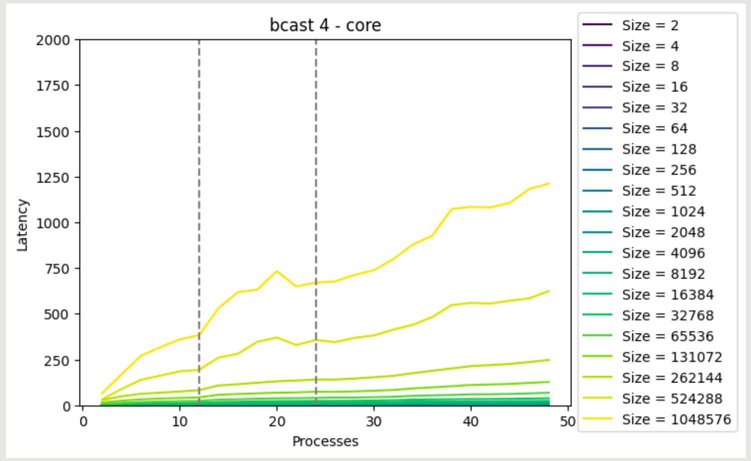




Split Binary Tree algorithm and the default implementation have superior performance:

- Tree structure efficiently distributes the communication among its branches
- Default configurations are robust and efficient in different environments, leveraging platform-specific optimizations





Increasing the message size, **Chain** outperforms **Basic Linear**:

- Chain superior performance, primarily attributable to its utilization of contiguous cores
- Chain capacity to partition messages into chunks during transmission, a feature absent in the linear

Bcast: performance models

$\rightarrow$ **Initial approach**: estimate pt2pt latency and bandwidth to construct a model resembling the *Hockney's model*

The model produced for **Basic Linear** (bcast1) is:

$$\text{latency} = -0.233587 + 0.117043 \times \text{processes}$$

with an R^2 adjusted of 0.9381.

The model produced for **Chain** (bcast2) is:

$$\text{latency} = -0.214384 + 0.109159 \times \text{processes}$$

with an R^2 adjusted of 0.8548.

The model produced for **Split Binary Tree** (bcast4) is:

$$\text{latency} = -0.523 + 5.518 \times \text{processes} - 2.529 \times \text{processes}^2$$

with an R^2 adjusted of 0.8793.

Scatter algorithms

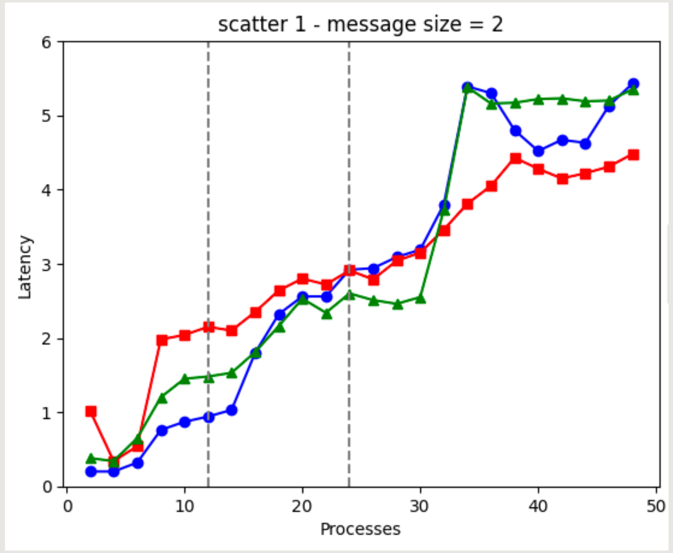
1. **Basic linear** (scatter 1): Each process receives a contiguous block of data from the root process. The root process sends data to each process in sequence.

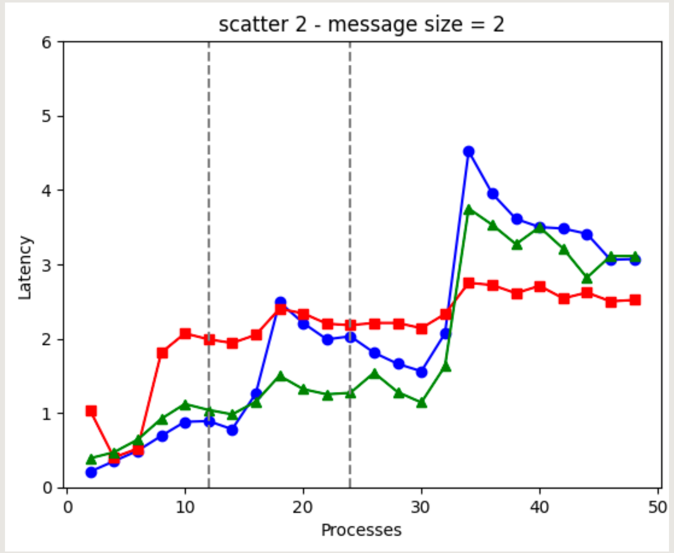
2. **Binomial** (scatter 2): It uses a binary tree structure to distribute data. Each process receives data from a process that is logarithmically distant from the root process.

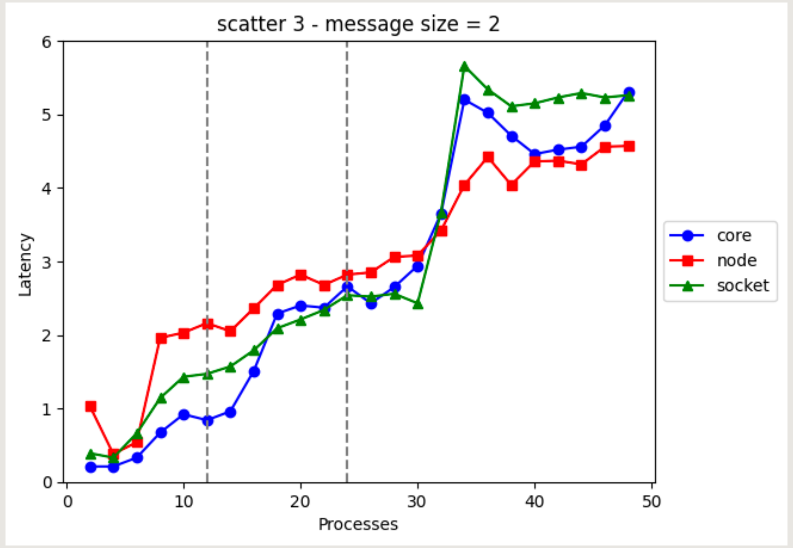
3. **Linear non-blocking** (scatter 3): The scatter operations are executed asynchronously. This means that the MPI implementation does not pause for the completion of one scatter operation before initiating the next.

Scatter: allocation type comparison

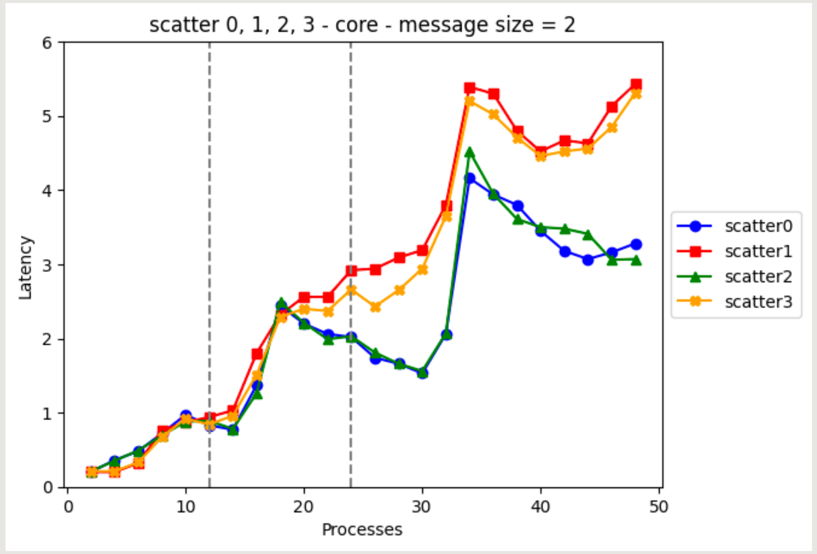
with message size fixed to 2 MPI_CHAR





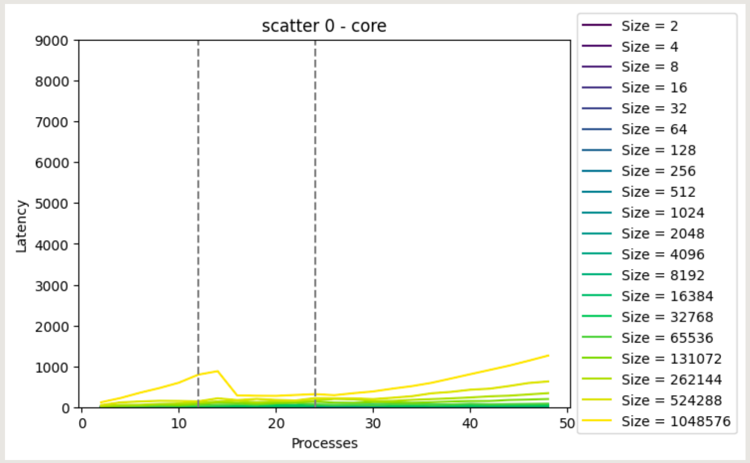


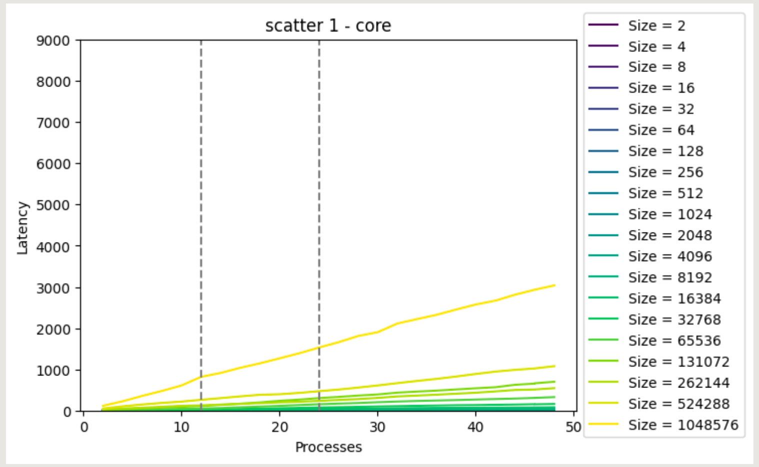
- Jumps occur at 16 and 32, hinting at additional factors influencing communication
- In scatter 2 the latency pattern of node and socket allocations appear closely tied → node-crossing impact socket communication directly
- Scatter 2 shows lower latency values
- In scatter 3 both node and socket show a steady increase until 30 processes and then a subsequent drop

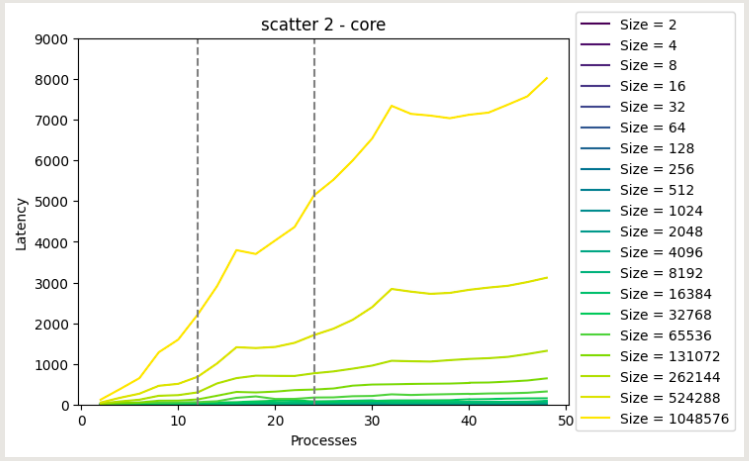


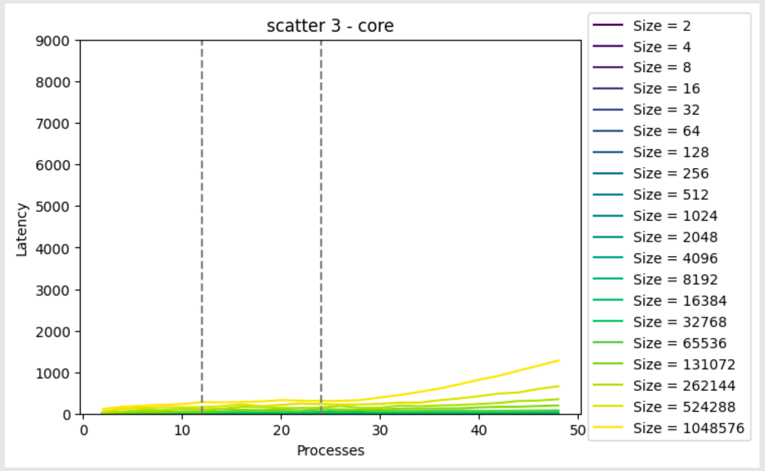
Scatter: algorithms comparison

non-fixed message size









Linear non-blocking algorithm outperform the others:
MPI implementation does not pause for the completion of one scatter operation before initiating the next → enable the system to concurrently manage communication and computation tasks

Increasing the message size the **Binomial** algorithm assume the highest values of latency among all the algorithm, while with smaller message sizes it used to be really close to the baseline model

Scatter: performance models

The model produced for **Basic linear** (Scatter 1) is:
$$\text{latency} = -0.215616 + 0.124241 \times \text{processes}$$
with an R^2 adjusted of 0.9355.

The model produced for **Binomial** (Scatter 2) is:
$$\text{latency} = 0.141486 + 0.077624 \times \text{processes}$$
with an R^2 adjusted of 0.7515.

The model produced for **Linear nb** (Scatter 3) is:
$$\text{latency} = -0.278261 + 0.120580 \times \text{processes}$$
with an R^2 adjusted of 0.9317.