

# Analysis of algorithms implemented in OpenMPI collective operations

Alessandro Della Siega

University of Trieste

May 2024

# Introduction

OpenMPI library implements blocking collective operations.

We will consider:

- ▶ MPI\_Bcast
- ▶ MPI\_Reduce

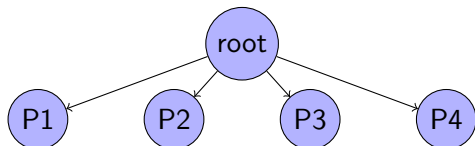
We will use the OSU Micro-Benchmark tool to evaluate the **latency** of these operations.

In particular, we are interested in analysis of these operation for different **algorithms**.

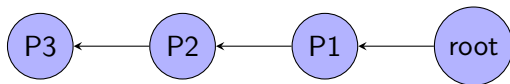
# Algorithms

For both operations, we will analyze the following algorithms:

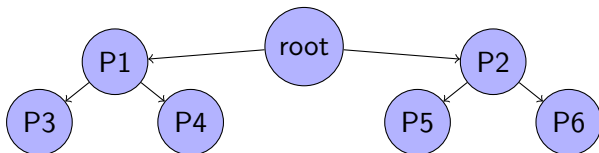
- ▶ Linear



- ▶ Chain



- ▶ (Split) Binary Tree



# Computational resources

ORFEO cluster

THIN partition

- ▶ 10× Intel Xeon Gold 6126
- ▶ 2 × 12 CPU cores each node
- ▶ 64 GiB of DDR4 RAM each core

For our analysis, we will use only 2 **THIN nodes**.

## Other parameters

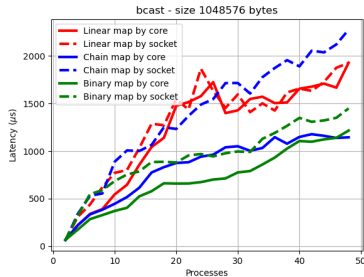
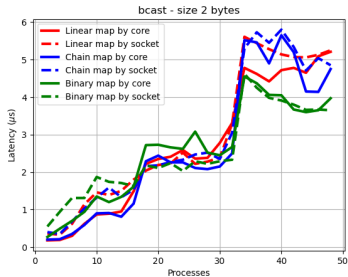
Beside **algorithm**, we will vary the other parameters as follows:

- ▶ **number of processes**: from 2 to 48 by step 2
- ▶ **mapping** of processes: by socket and by core
- ▶ **message length**: from  $2^1, 2^2, \dots, 2^{19}, 2^{20}$  MPI\_Char

For each configuration, we will perform  $10^4$  **repetitions** and compute the average latency of the operation.

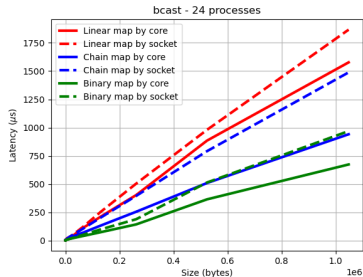
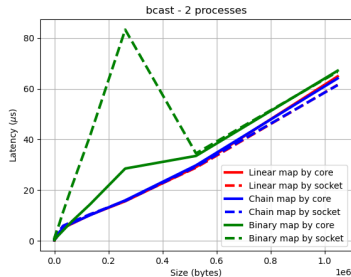
**Warm-up** is performed.

# Broadcast results



**Figure:** Broadcast average latency, for fixed message size of 2 bytes and 1 MB.

# Broadcast results



**Figure:** Broadcast average latency, for fixed number of processes: 2 and 24 .

# Reduce results

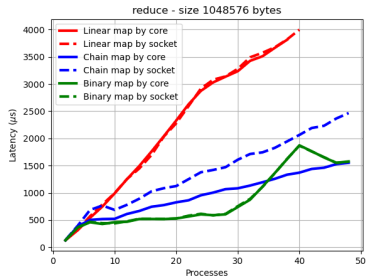
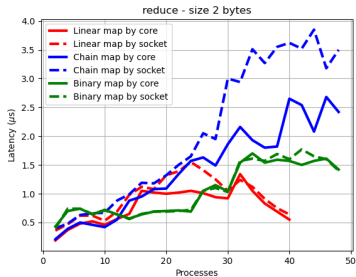


Figure: Reduce average latency, for fixed message Size: 2 and 24 .



# Reduce results

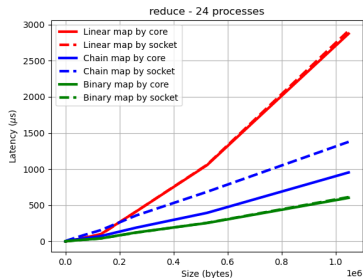
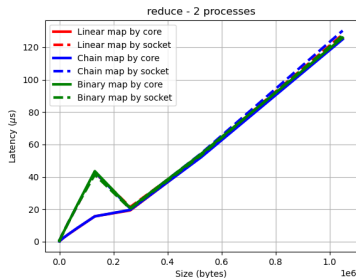


Figure: Reduce average latency, for fixed number of processes: 2 and 24 .

# Linear regression - Bcast linear algorithm mapped by core

$$Latency = \beta_0 Processes + \beta_1 Size \quad (1)$$

	Coef.	Std. Err.	t	P >  t	[0.025	0.975]
Processes	1.126	0.23	4.79	0.00	0.665	1.588
Size	0.001	0.00002	45.40	0.00	0.001	0.001

$$R^2 = 0.84 \text{ and } AIC = 6096$$

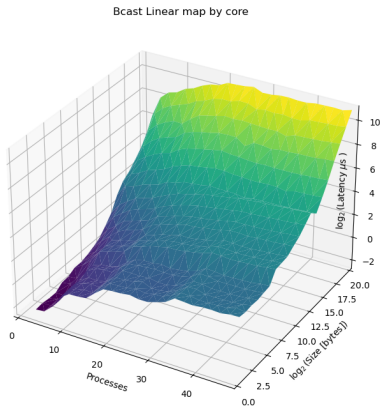
---

$$\log_2 Latency = \beta_0 Processes + \beta_1 \log_2 Size \quad (2)$$

	Coef.	Std. Err.	t	P >  t	[0.025	0.975]
Processess	0.027	0.004	6.69	0.00	0.019	0.035
log2_Size	0.325	0.0098	33.34	0.00	0.306	0.345

$$R^2 = 0.88 \text{ and } AIC = 1842.$$

# Conclusions



## ► Broadcast

- If message size is large, split binary tree is the best choice.
- Chain works better with core mapping.
- $\log\text{Latency}$  is linear with  $\log\text{Size}$  and number of Processes.

## ► Reduce

- If message size is very small, linear algorithm is the best choice.
- Mapping has no effect on the latency (except for Chain).