Data Science & Artificial Intelligence

High Performance Computing

PERFORMANCE EVALUATION OF OPENMPI COLLECTIVE OPERATIONS

Pranay Narsipuram
SM3800006

JULY 2024

OBJECTIVE

1. To evaluate and compare the performance of various algorithms in the OpenMPI library for collective operations, specifically focusing on broadcast and gather operations.

2. To develop and validate performance models that predict the latency of these implementations.

EXPERIMENTAL SETUP

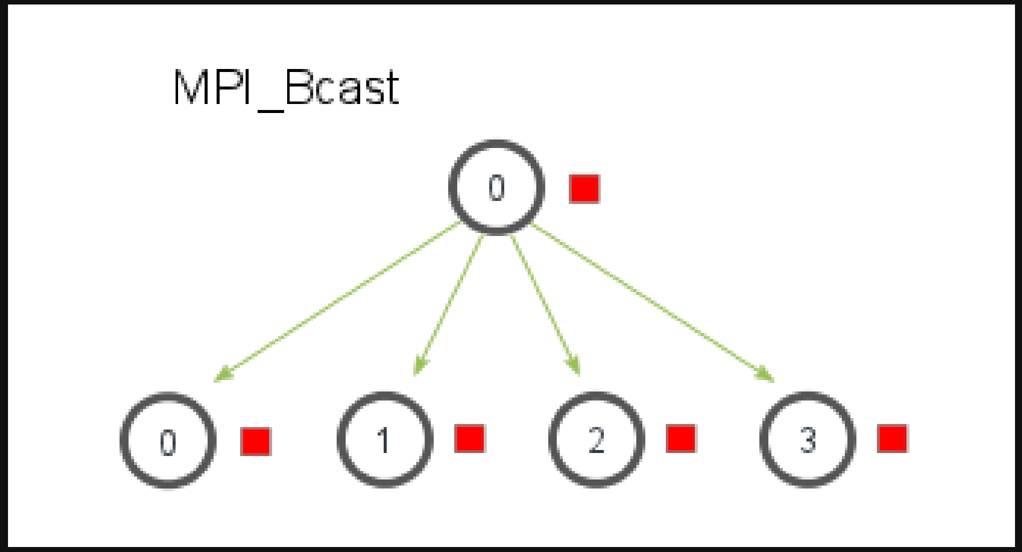
- ORFEO cluster: 2 EPYC nodes, 128 cores per node.
- OpenMPI 4.1.5
- OSU Micro-Benchmarks (osu_bcast, osu_gather)
- Scripts to automate data collection.
- –map-by core policy.
- Collect measures varying the number of processes from 2 to 256 and the size of the messages from 1 to 524288 bytes.

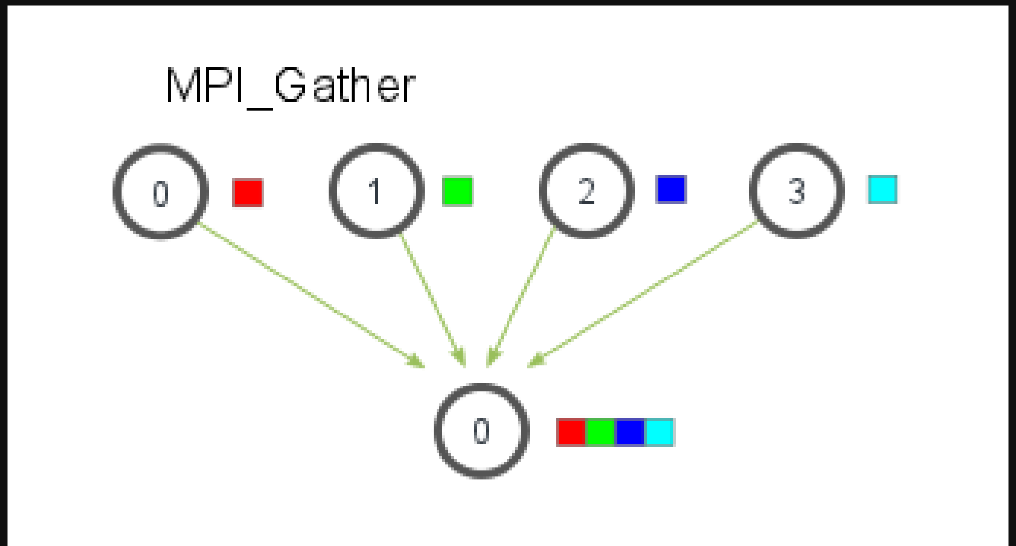
ORFEO CLUSTER

DATA COLLECTION

scan for
benchmark_job.sh

- Usage of benchmark_job.sh script.
- Usage: ./benchmark_job.sh <operation> <benchmark_type>
- Operations: Broadcast and Gather.
- Algorithms: ignore, chain, pipeline, binary_tree, basic_linear, binomial, linear_sync.
- Fixed and variable message sizes.





OSU MICRO BENCHMARKS

- A suite of benchmarks developed by Ohio State University.
- Key Benchmarks:
  osu_latency: Measures point-to-point latency.
  osu_bw: Measures point-to-point bandwidth.
  osu_bcast: Measures performance of MPI_Bcast.
  osu_gather: Measures performance of MPI_Gather.
- Benchmarking Methodology:
  Varying number of processes and message sizes.
- Automated benchmarking scripts.

Broadcast Operation

Evaluated the performance of different broadcast algorithms using OpenMPI on the ORFEO cluster.

- Ignore Algorithm:
  Baseline for comparison, essentially a no-operation broadcast.
  Results: Minimal latency, serves as a control to compare other algorithms.

- Chain Algorithm:
  Each process forwards the message to the next sequentially.
  Results: Higher latency due to sequential forwarding. Suitable for small-scale systems.
  Analysis: Latency increases linearly with the number of processes.

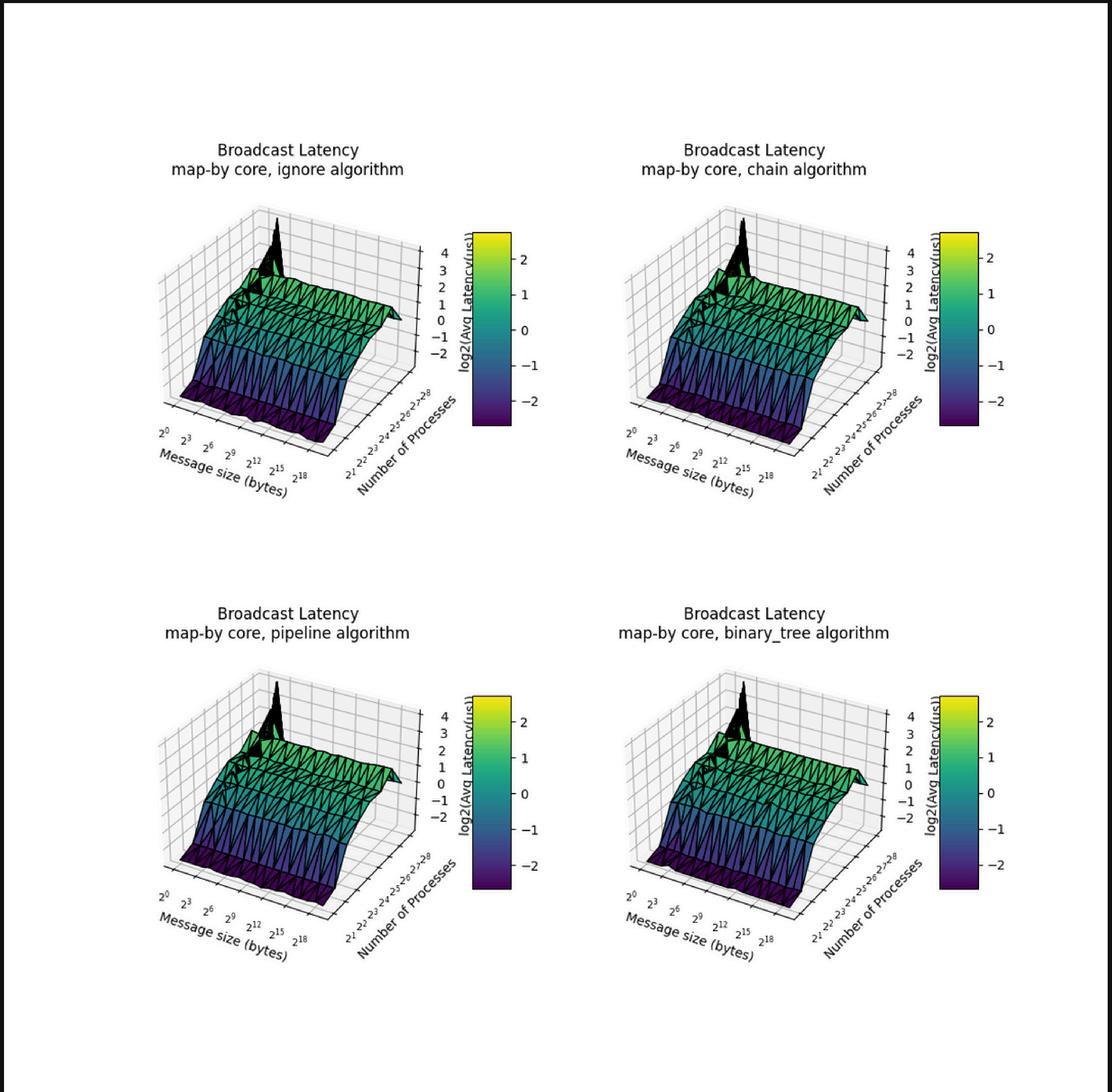
- Pipeline Algorithm:
  Processes arranged in a pipeline, reducing latency by overlapping communication with computation.
  Results: Better performance than the chain algorithm due to overlapping.
  Analysis: Latency improves but increases with larger message sizes.

- Binary Tree Algorithm:
  Root sends messages to two children, forming a binary tree.
  Results: Best performance among tested algorithms. Efficient for larger process counts due to logarithmic scaling.
  Analysis: Latency increases logarithmically with the number of processes, making it highly scalable.

- Comparison:
- Fixed Message Size: Analyzed latency with a fixed message size across different algorithms.
  Results: Binary tree algorithm consistently outperforms others.

- Variable Message Size: Analyzed latency changes with varying message sizes.
  Results: Pipeline and binary tree algorithms show better scalability with increasing message sizes.

Visualizing the performance of broadcast algorithms using 3D heatmaps on the ORFEO cluster.



Gather Operation

Evaluated the performance of different gather algorithms using OpenMPI on the ORFEO cluster.

- Ignore Algorithm:
  Baseline for comparison, essentially a no-operation gather.
  Results: Minimal latency, serves as a control to compare other algorithms.

- Basic Linear Algorithm:
  Each process sends its data directly to the root.
  Results: Simple implementation. High latency due to all processes sending data to the root.
  Analysis: Latency increases linearly with the number of processes.

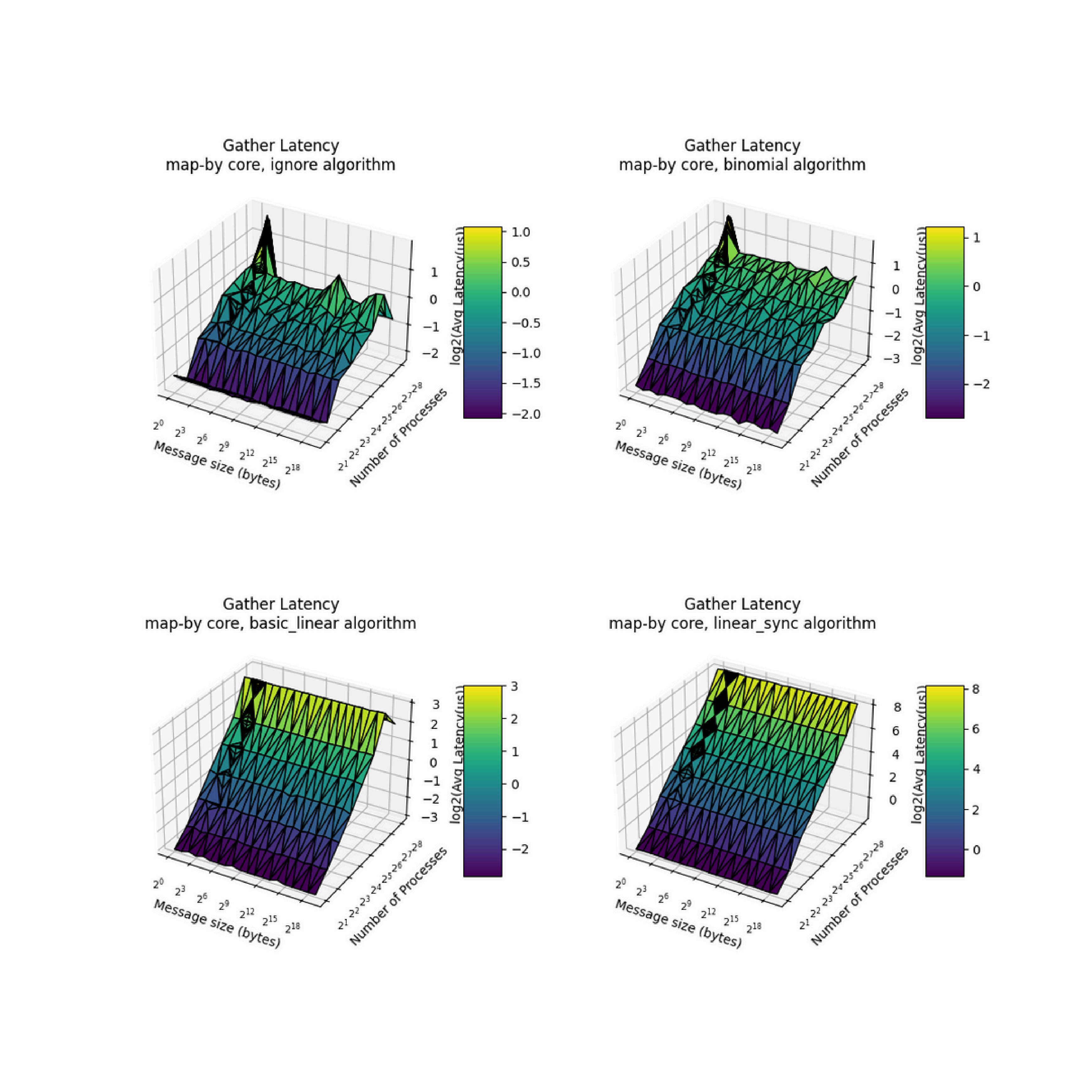
- Binomial Algorithm:
  Data is aggregated in a hierarchical binomial tree structure.
  Results: Better performance than the basic linear algorithm due to hierarchical aggregation.
  Analysis: Latency scales logarithmically with the number of processes.

- Linear Sync Algorithm:
  Similar to basic linear but includes synchronization steps.
  Results: Ensures data consistency and reduces network congestion.
  Analysis: Slightly higher overhead due to synchronization, but better performance in terms of consistency.

- Comparison:
  Fixed Message Size: Analyzed latency with a fixed message size across different algorithms.
  Results: Binomial algorithm consistently outperforms others in fixed message size scenarios.

  Variable Message Size: Analyzed latency changes with varying message sizes.
  Results: Binomial and linear sync algorithms show better scalability with increasing message sizes.

Visualizing the performance of gather algorithms using 3D heatmaps on the ORFEO cluster.



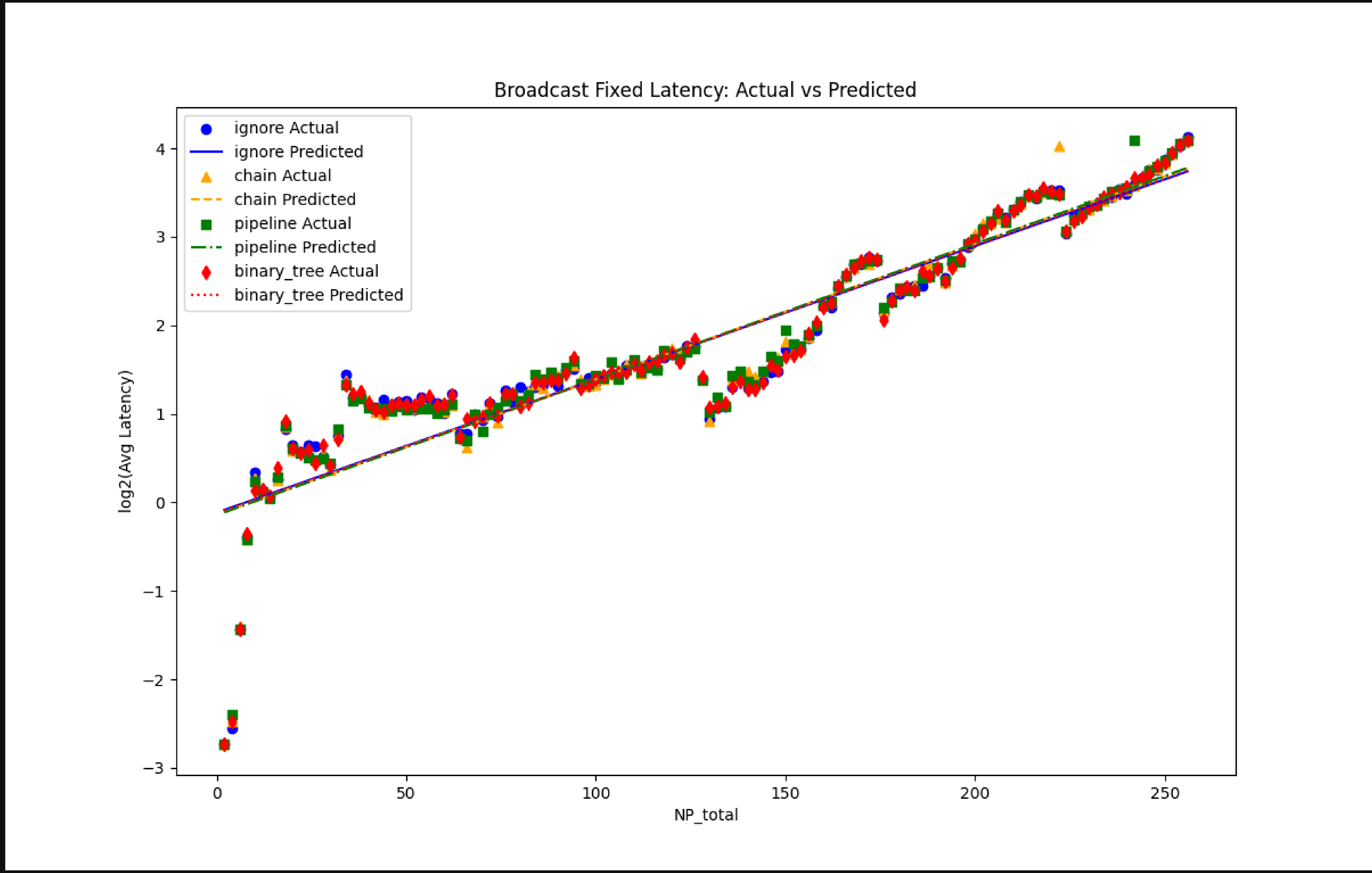
Performance Model - Broadcast fixed

- Model shows strong predictive power, closely matching actual latencies.
- Effective across different core counts, ensuring reliability.

Model Equation:

$$\log_2(\text{avg-lat}) = \beta_1 \cdot \text{proc-num} + \beta_2 \cdot \log_2(\text{mess-size}) + \beta_3 \cdot (\log_2(\text{mess-size}))^2$$

- Polynomial term captures non-linear trends in latency.



Performance Model - Gather Fixed

- Model shows strong predictive power, closely matching actual latencies.
- Effective across different core counts, ensuring reliability.

Model Equation:

$$\log_2(\text{avg-lat}) = \delta_1 \cdot \text{proc-num} + \delta_2 \cdot \log_2(\text{mess-size}) + \delta_3 \cdot (\log_2(\text{mess-size}))^2$$

- Polynomial term captures non-linear trends in latency.