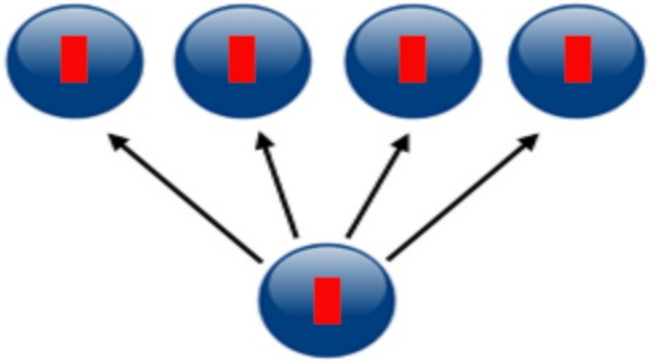
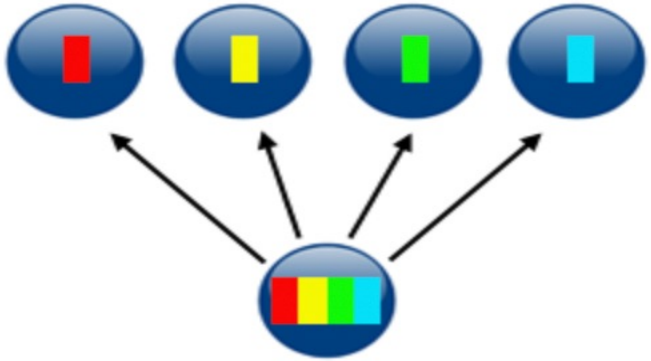
Performance Assessment of MPI Collective Operations





Nigam Vishal
University of Trieste
July 2024

Objective

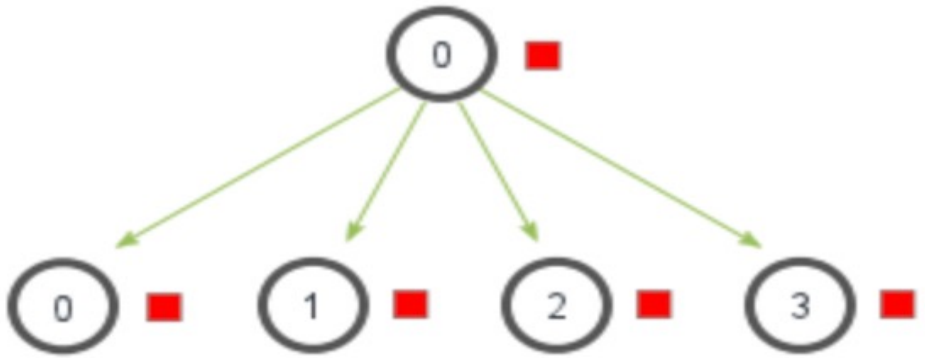
Performance Comparison of MPI algorithms for below operations:

- Broadcast->MPI_Bcast (default,chain,pipeline and binary tree)
- Scatter->MPI_Scatter (default,linear,binomial,non-blocking linear)

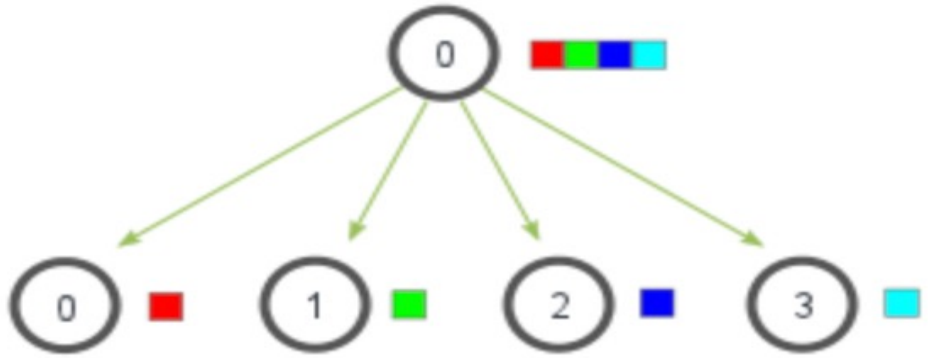
*Usage of osu benchmark tool to collect latencies of these operations
And analysis of performance model

Algorithms

MPI_Bcast



MPI_Scatter



Experimental Plan

ORFEO CLUSTER

- CPU ->2 x AMD EPYC 7H12
- EPYC partition
- 2 nodes -> 256 cores
- Choice of map by core vs others
- Bash script for benchmark data collection for fixed and varying data sizes
- Python script for plots and images
- Slurm jobs for above 2
- Assessment of performance model

OMB Mapping Policy

- Choice of the algorithm significantly influenced the optimal map-by strategy
- `mpirun --mca coll_tuned_use_dynamic_rules true --mca coll_tuned_${operation}_algorithm $alg_num --map-by core -np $np $MPI_DIR/$EXECUTABLE -m ${sizes[0]} -x $WARMUP_ITERATIONS -i $TOTAL_ITERATIONS`

Test Type	Message Size	Avg Latency (us)
Map by socket - Basic Linear Algorithm	1	14.30
Map by core - Basic Linear Algorithm	1	5.70
Map by socket - Binomial Algorithm	1	5.20
Map by core - Binomial Algorithm	1	3.80

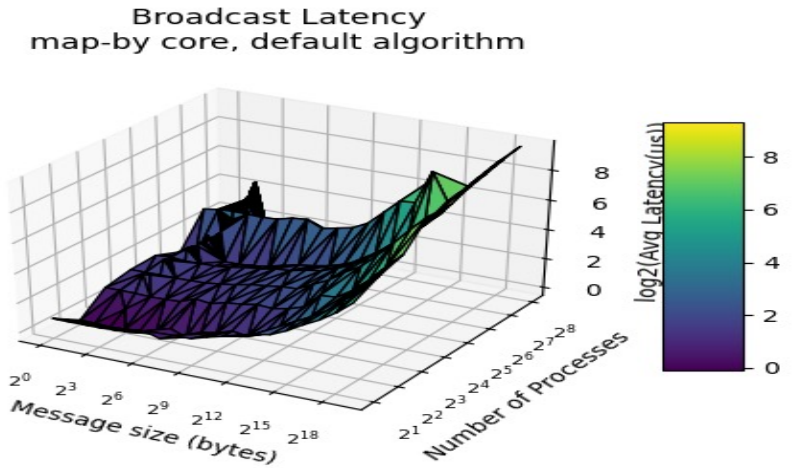
Task-1

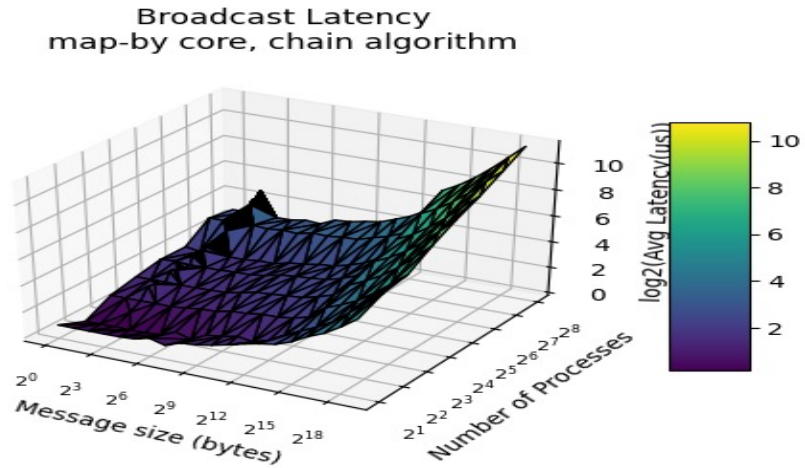
- Ran algorithms for both broadcast and scatter operations and collected benchmark latencies data for full and fixed benchmark types.

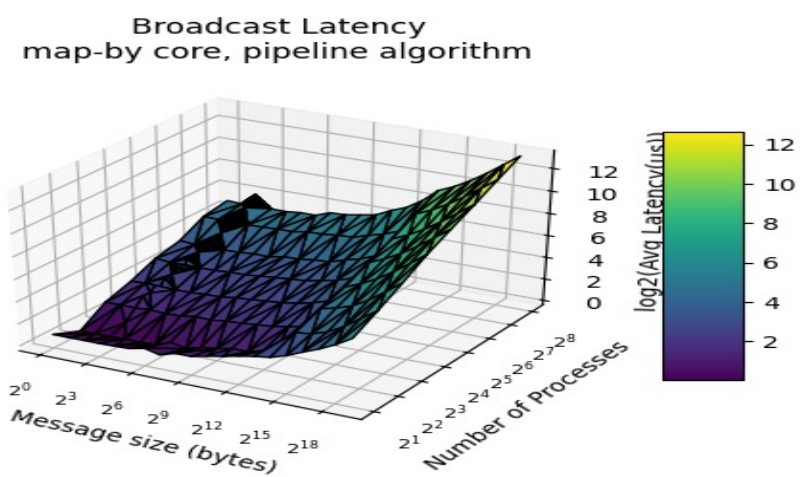
- Parameters used:

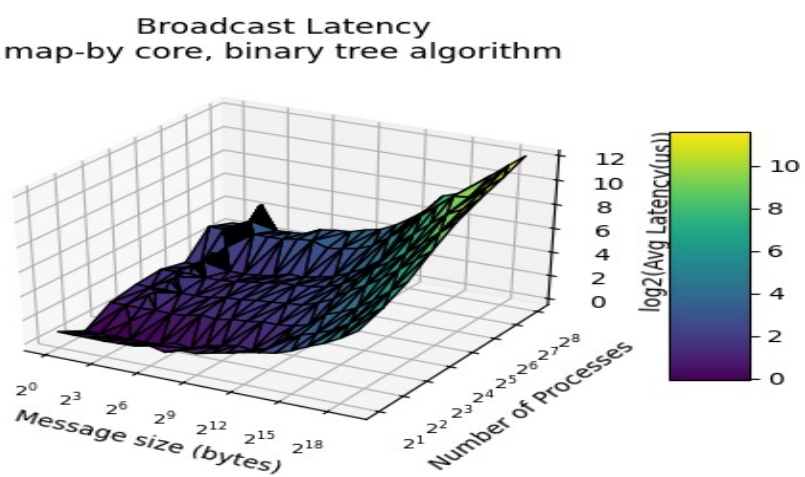
a) Message sizes: from 1 to 2^{19} bytes
b) Mapping policy: by core
c) Number of processes: 2 to 256 step 2

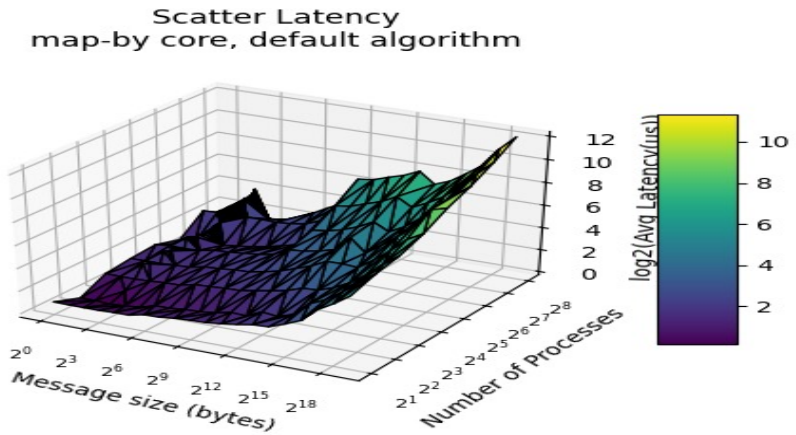
Some Visualizations ahead….

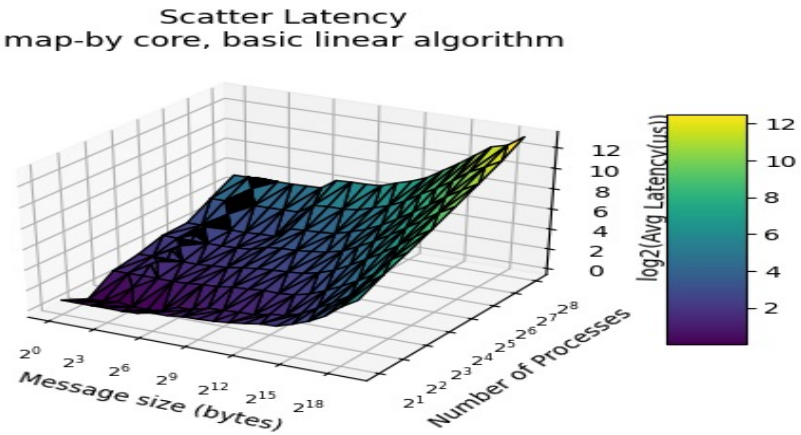

Broadcast Latency
map-by core, default algorithm
Message size (bytes)
Number of Processes
log2(Avg Latency(us))

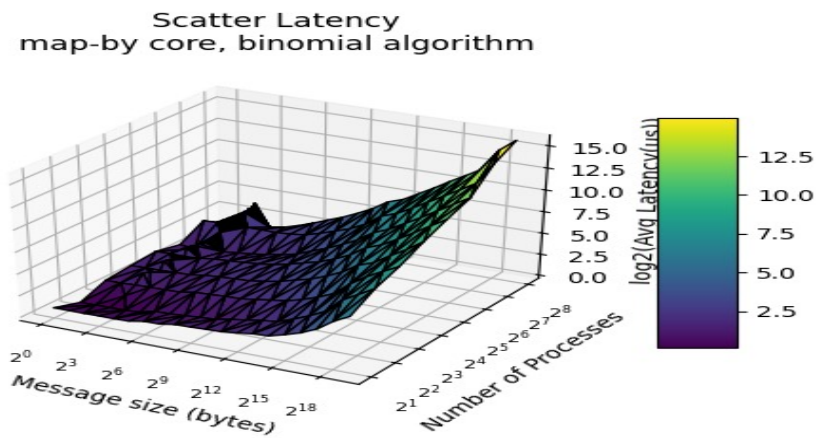

Broadcast Latency
map-by core, chain algorithm
Message size (bytes)
Number of Processes
log2(Avg Latency(us))

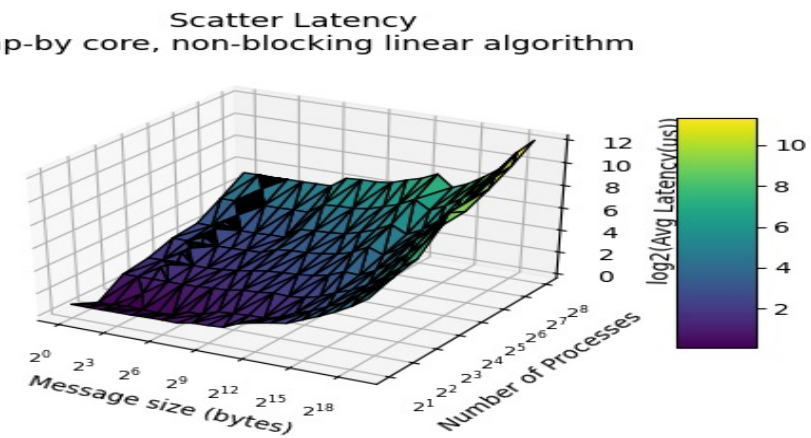

Broadcast Latency
map-by core, pipeline algorithm
Message size (bytes)
Number of Processes
log2(Avg Latency(us))


Broadcast Latency
map-by core, binary tree algorithm
Message size (bytes)
Number of Processes
log2(Avg Latency(us))


Scatter Latency
map-by core, default algorithm
Message size (bytes)
Number of Processes
log2(Avg Latency(us))


Scatter Latency
map-by core, basic linear algorithm
Message size (bytes)
Number of Processes
log2(Avg Latency(us))

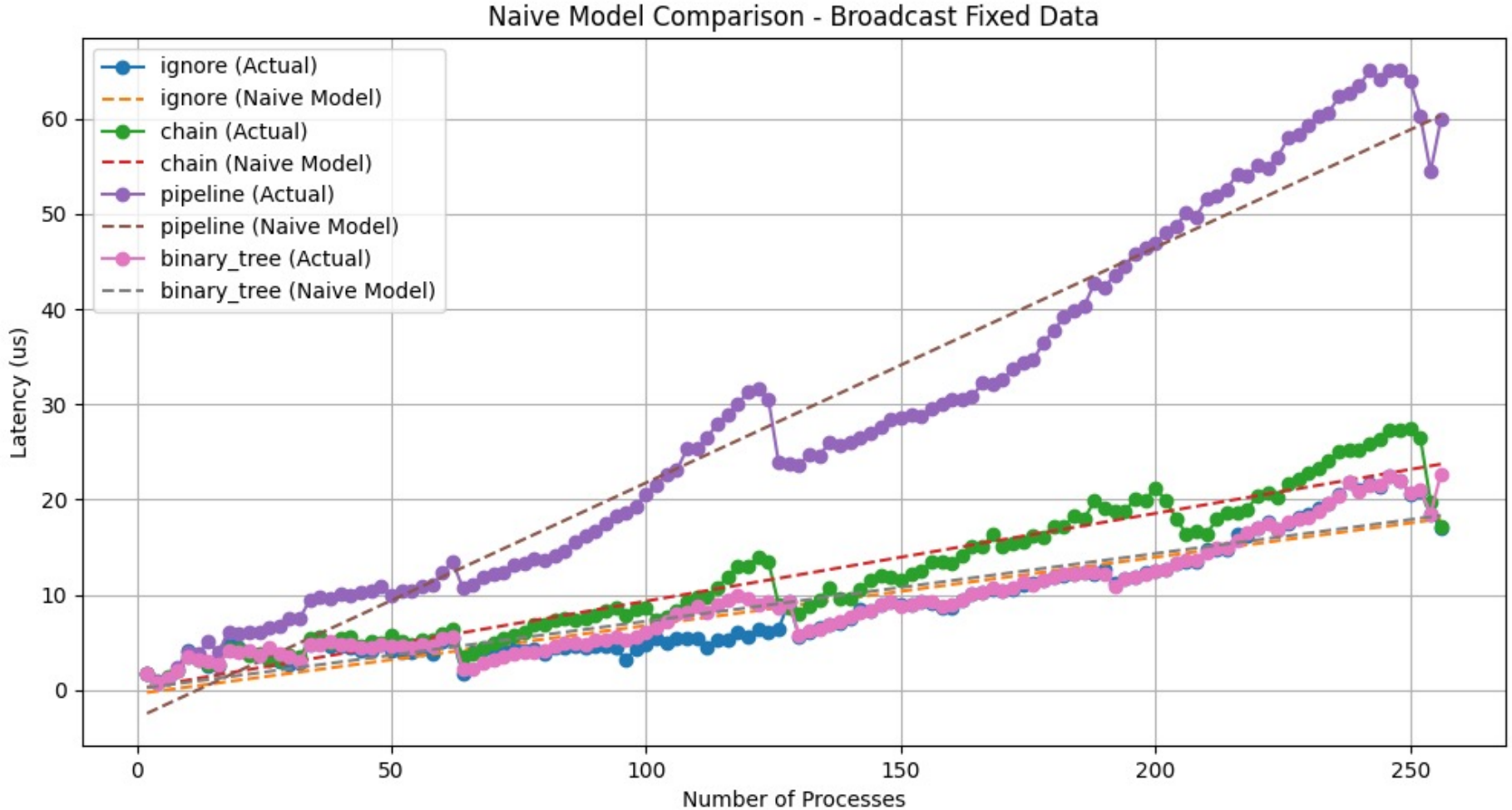

Scatter Latency
map-by core, binomial algorithm
Message size (bytes)
Number of Processes
log2(Avg Latency(us))

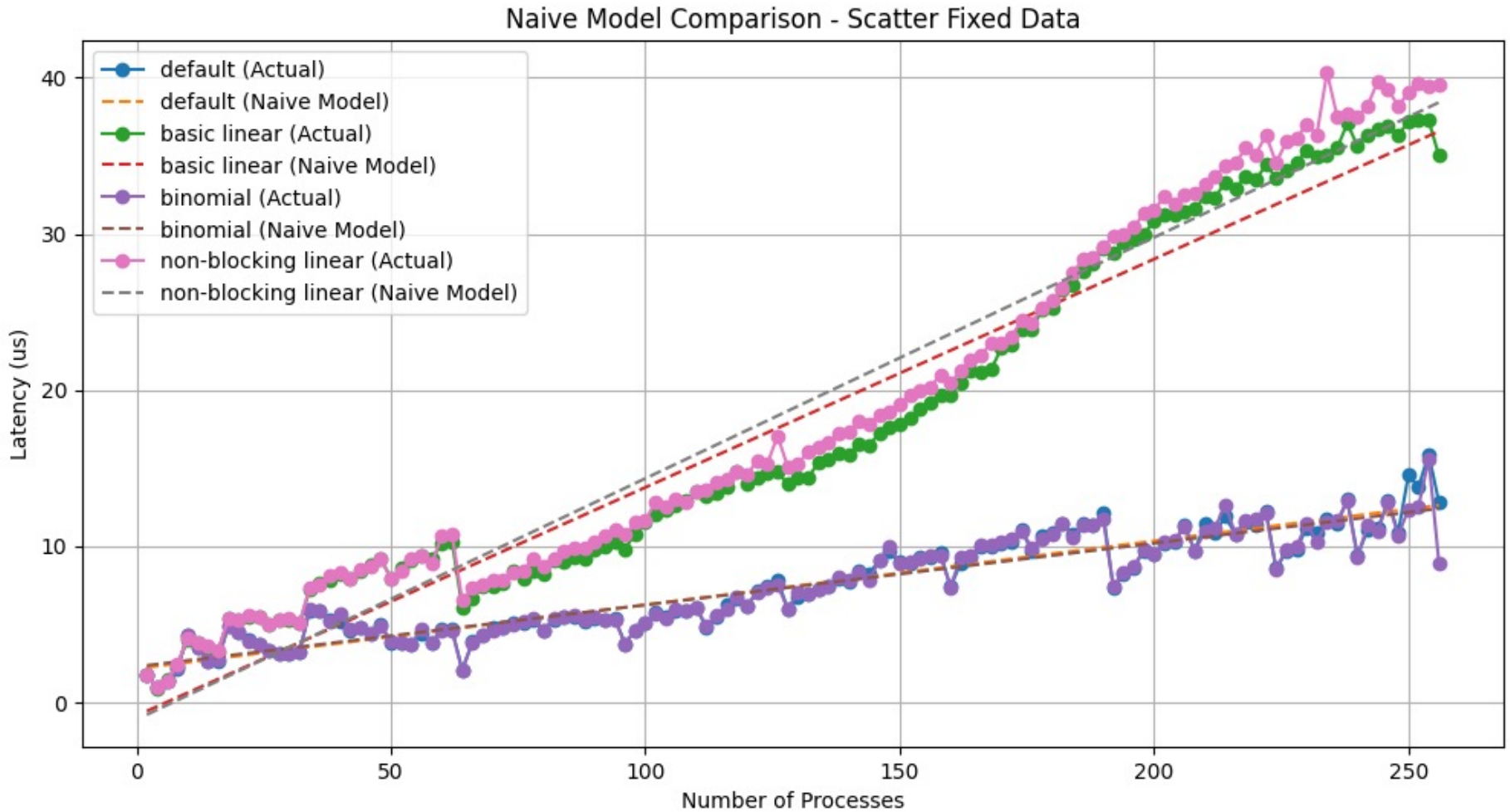

Scatter Latency
map-by core, non-blocking linear algorithm
Message size (bytes)
Number of Processes
log2(Avg Latency(us))

Performance Model

- Linear Regression Model as naïve model for latency across broadcast and scatter algorithms

$$Latency = Intercept + \beta 1 * Number\ of\ Processes + \beta 2 * Message\ Size$$


Naive Model Comparison - Broadcast Fixed Data
ignore (Actual)
ignore (Naive Model)
chain (Actual)
chain (Naive Model)
pipeline (Actual)
pipeline (Naive Model)
binary_tree (Actual)
binary_tree (Naive Model)
Latency (us)
Number of Processes
0
10
20
30
40
50
60
0
50
100
150
200
250


Naive Model Comparison - Scatter Fixed Data
default (Actual)
default (Naive Model)
basic linear (Actual)
basic linear (Naive Model)
binomial (Actual)
binomial (Naive Model)
non-blocking linear (Actual)
non-blocking linear (Naive Model)
Latency (us)
Number of Processes
0
10
20
30
40
0
50
100
150
200
250

Conclusion

- **Naive Models**: Provide a simplified estimation of latency trends but lack the precision needed for detailed performance predictions.
- **Actual Data**: Shows that real-world performance varies significantly due to additional overheads and optimizations not captured by naive models.
- **Estimation**: The naive models tend to overestimate the latency for both broadcast and scatter operations, especially as the number of processes increases.