Machine Learning Driven Auto-tuning for Non-uniform All-to-All Collectives

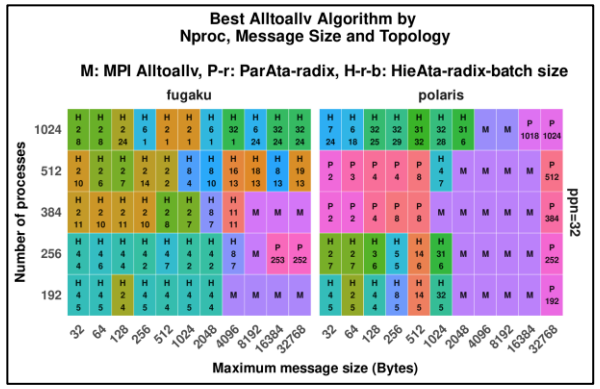
Kunting Qi, Ke Fan, Jens Domke, Seydou Ba, Venkatram Vishwanath, Michael Papka, ***Sidharth Kumar***

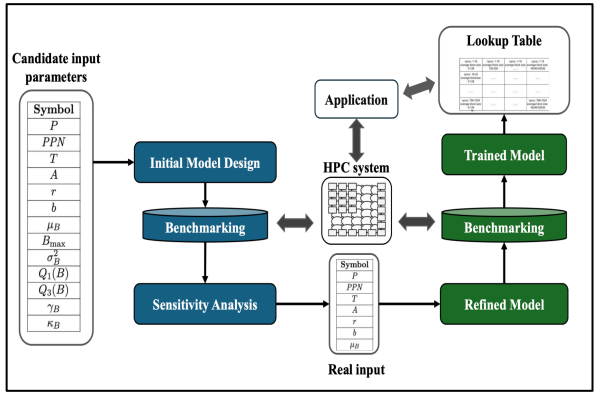
Outline

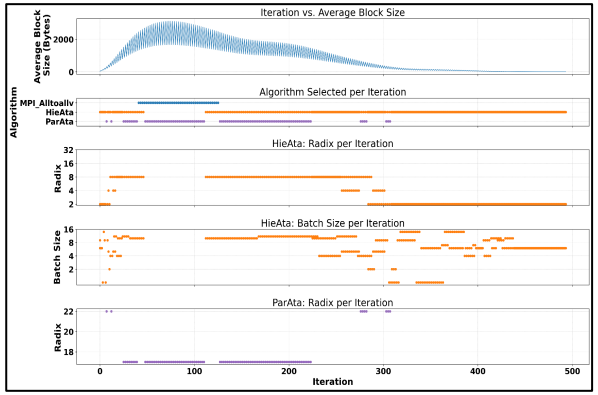
Motivation and challenges

Proposed approach

Results



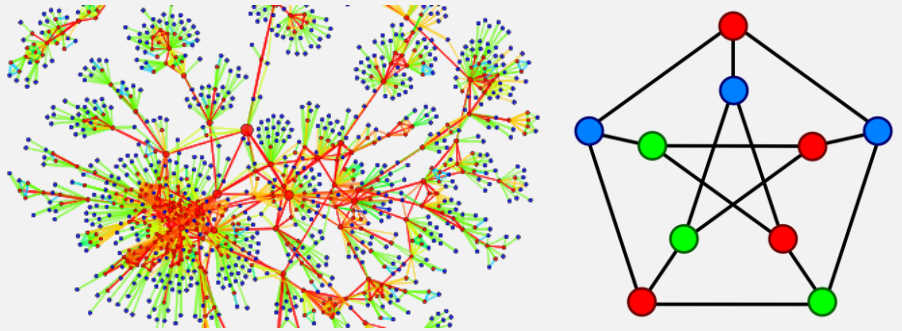




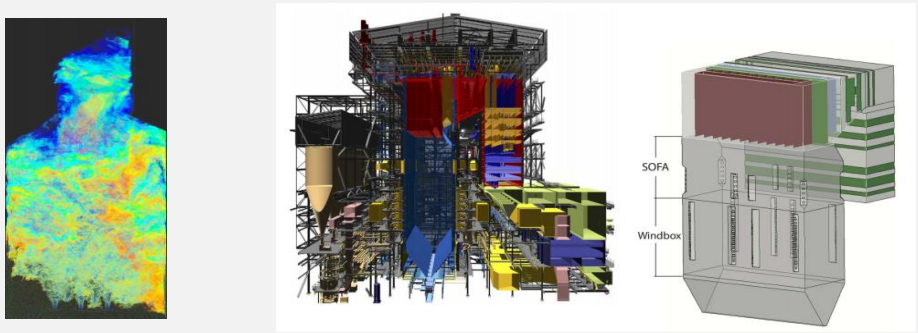
Emergence of Exascale computing and large-scale applications

Exascale Supercomputers

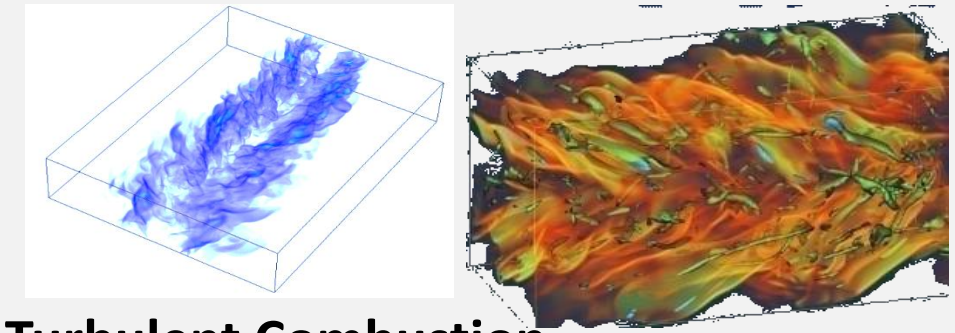
Simulations / Applications



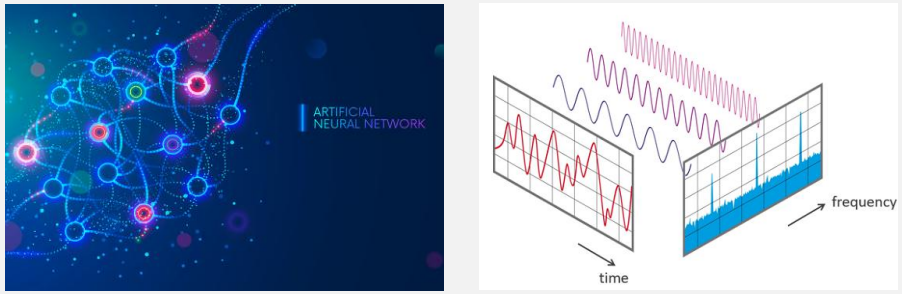
Large scale graph mining



Modeling of a 1000 MWe coal boiler

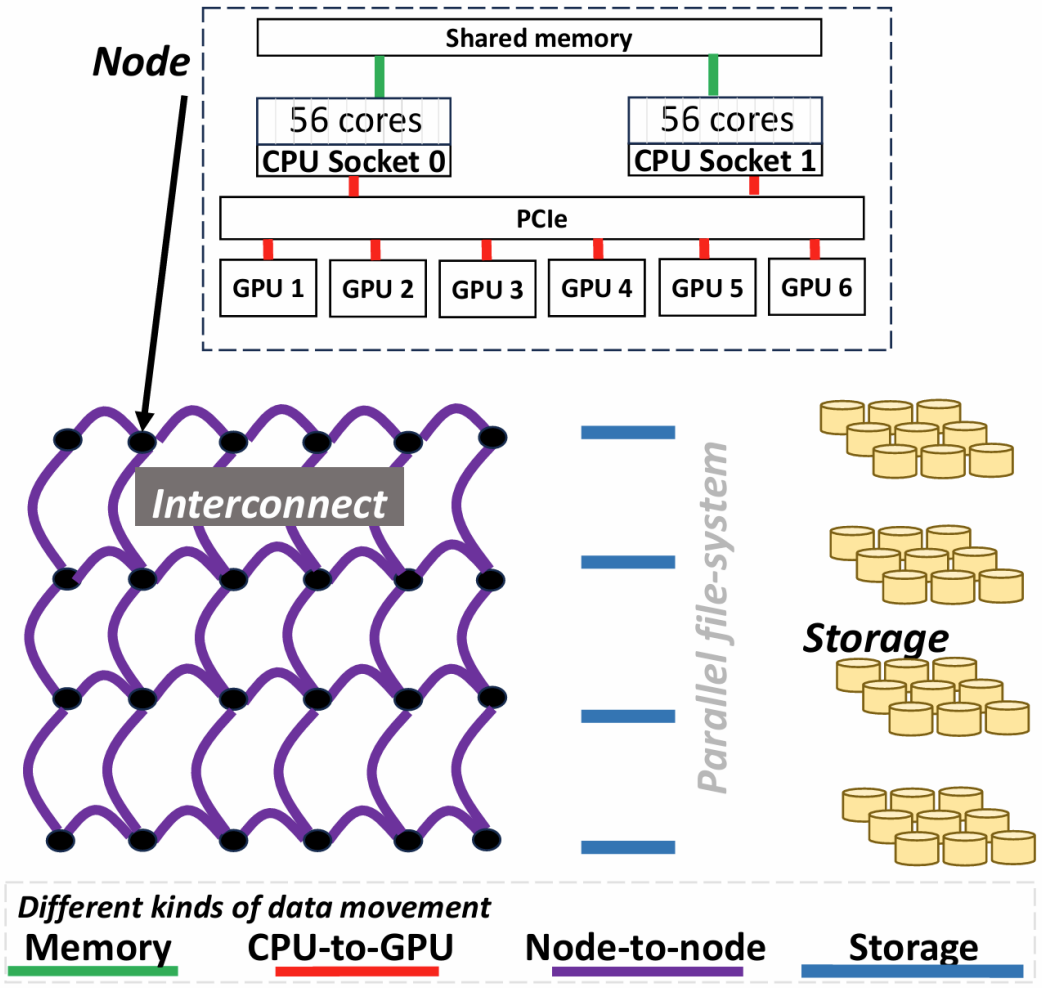


Turbulent Combustion

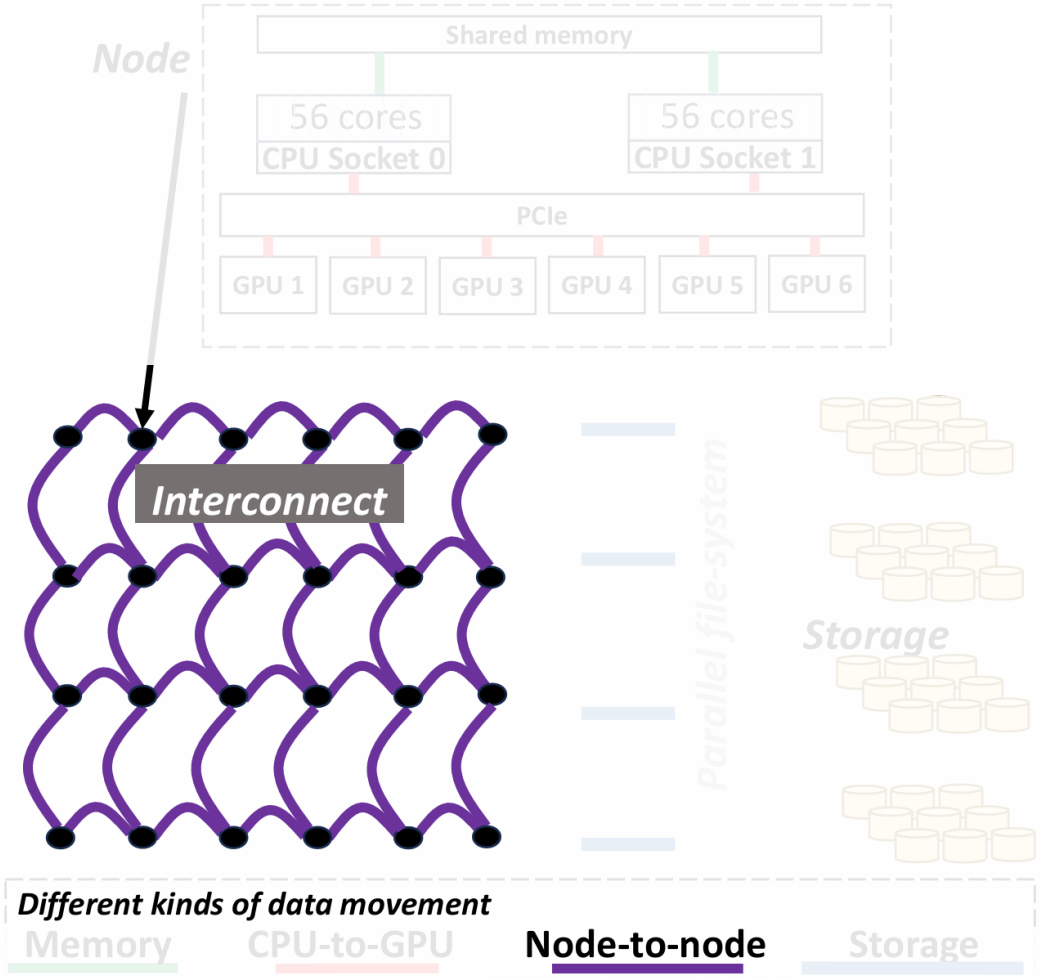


Parallel Data-driven AI/ML

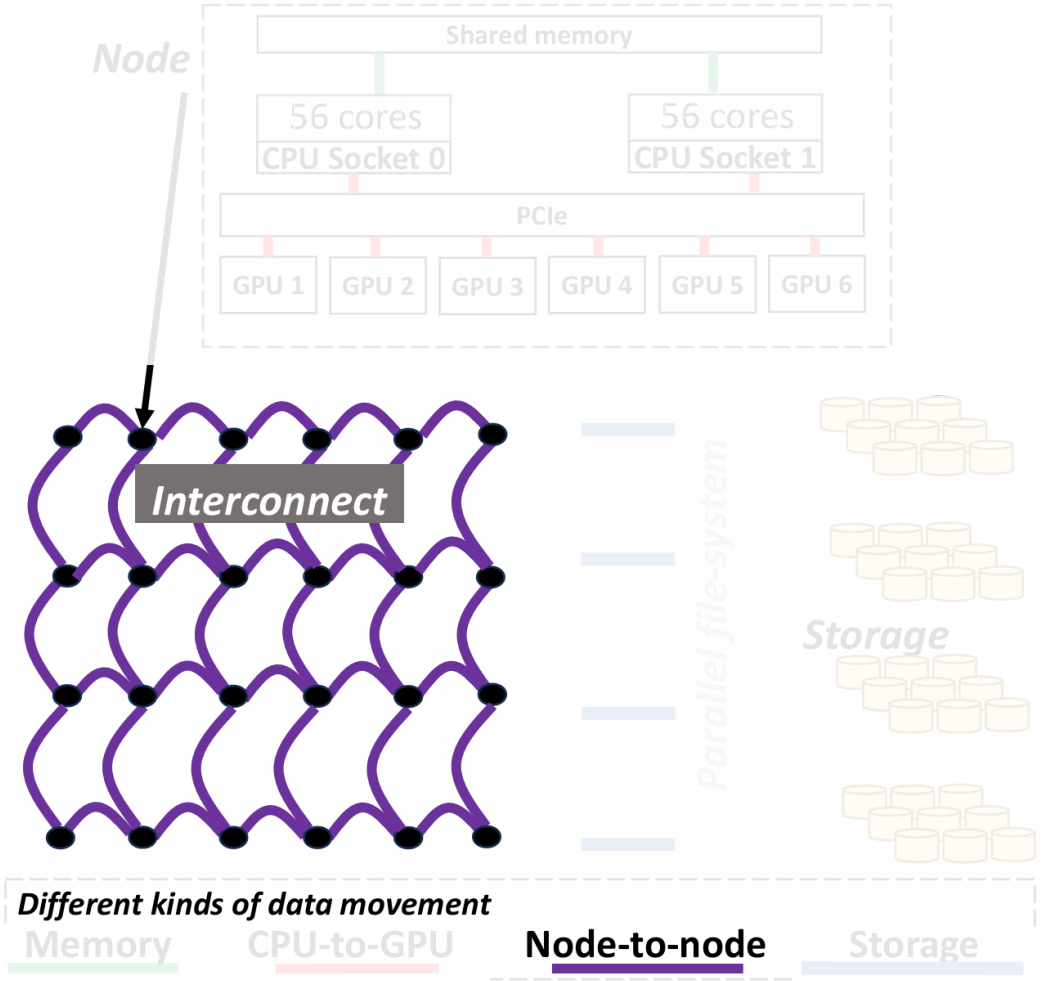
Data movements in exascale

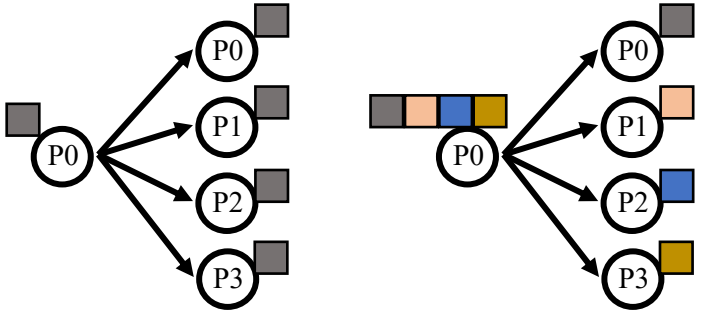


Data movements in exascale



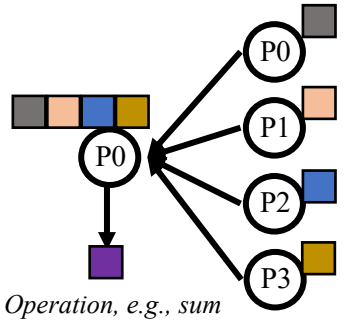
Data movements in exascale - Collectives



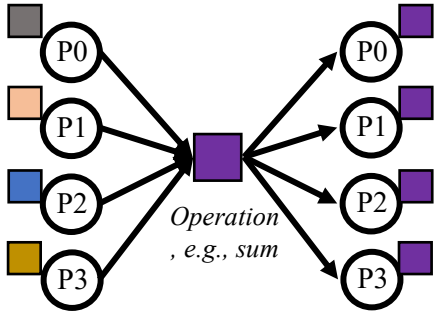


(1) Broadcast

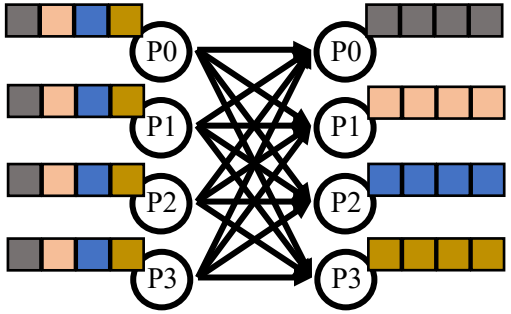
(2) Scatter



(3) Reduction

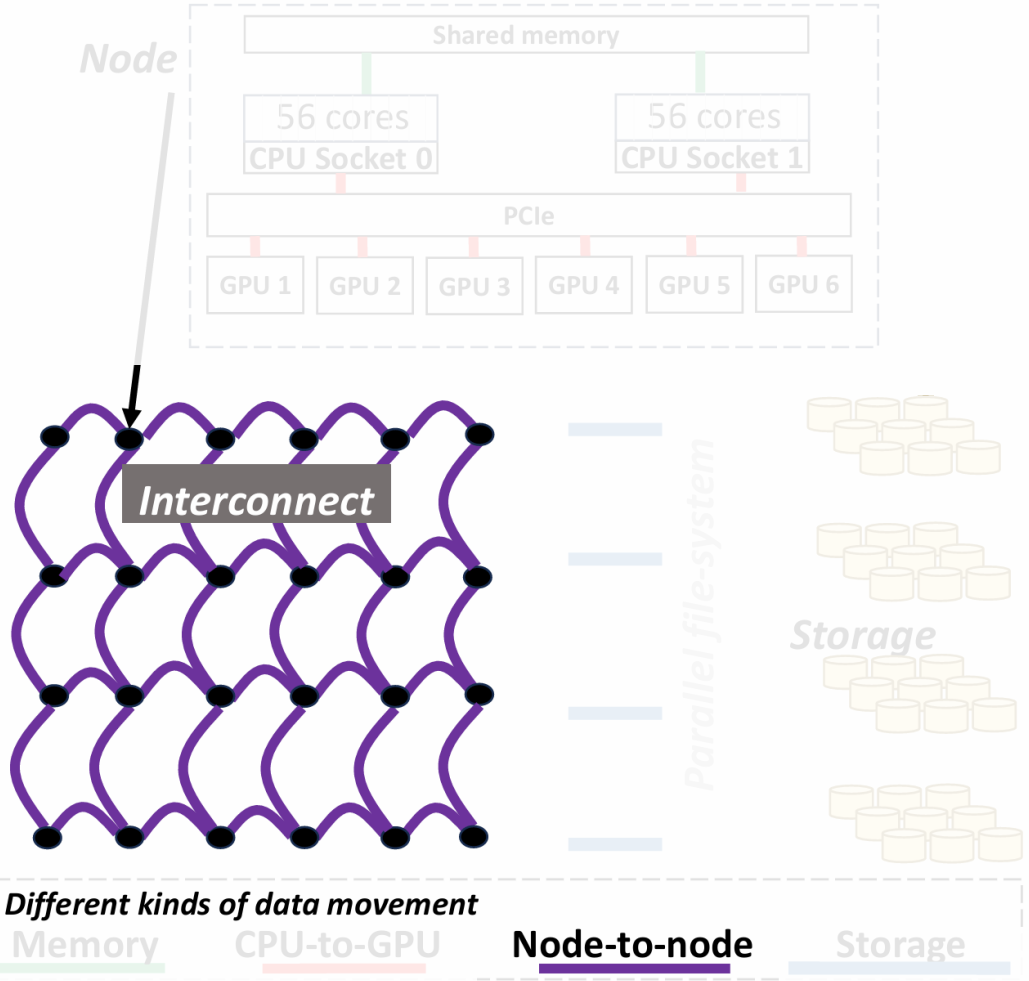


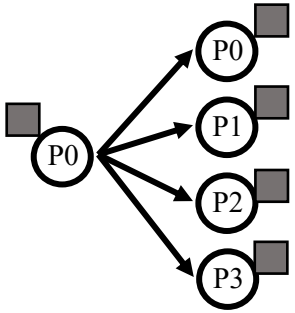
(4) All Reduction



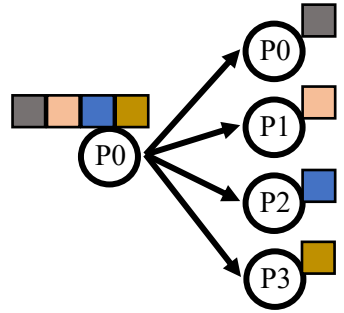
(5) All-to-all

Data movements in exascale - Collectives

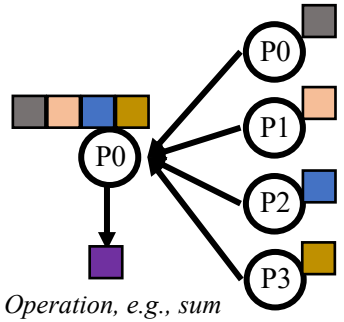




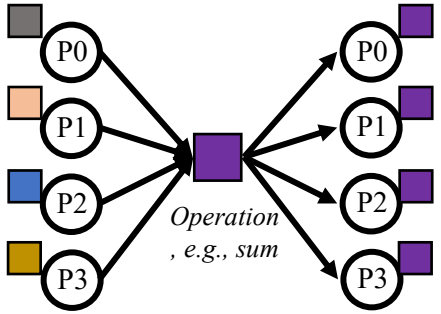
(1) Broadcast



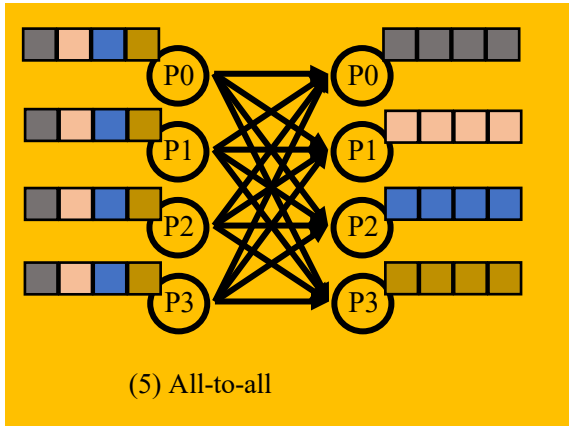
(2) Scatter



(3) Reduction

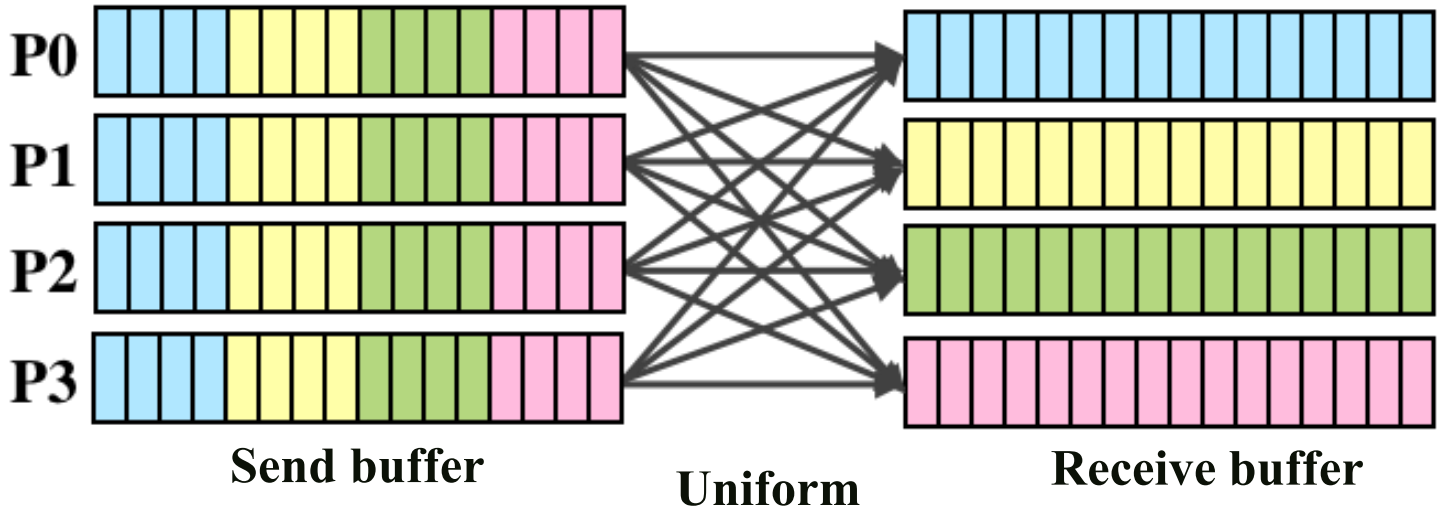


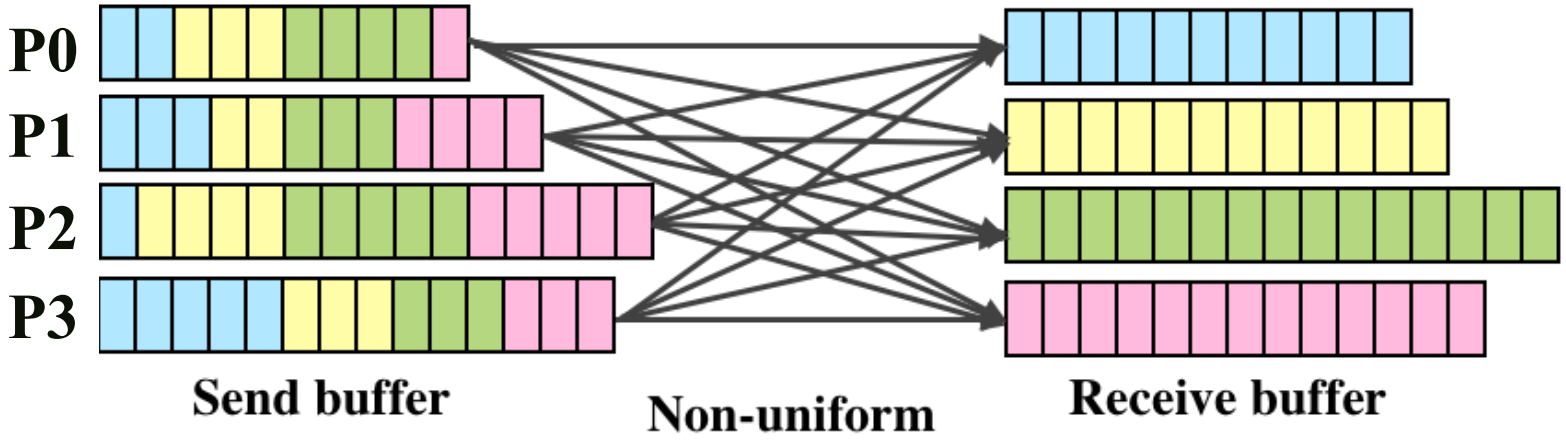
(4) All Reduction



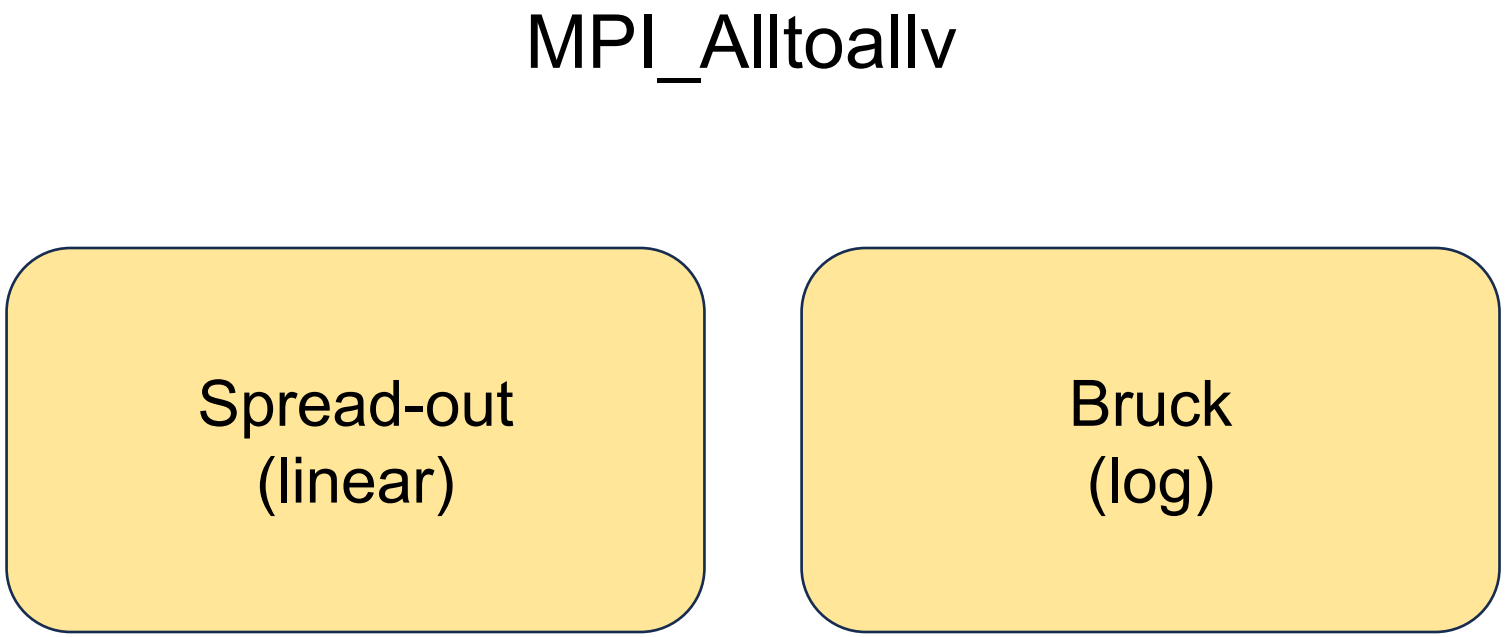
(5) All-to-all

All-to-all – Every process Sends and receives data from every other process

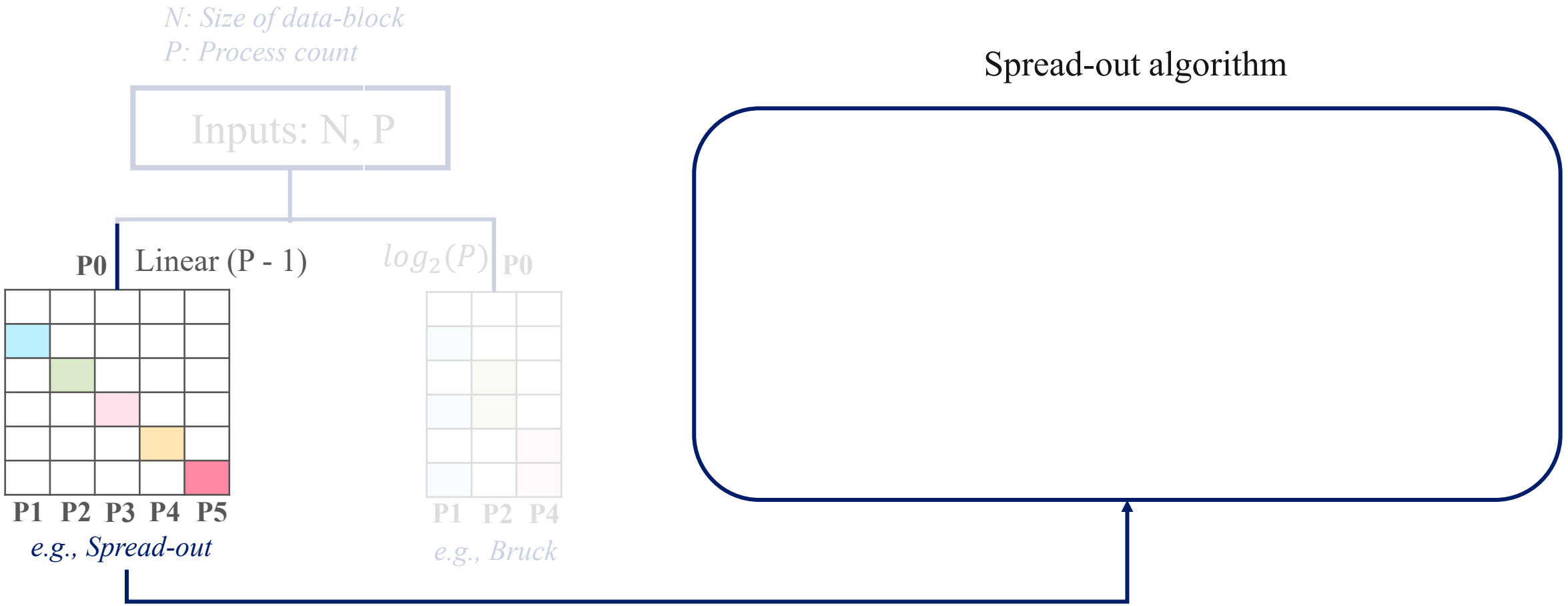




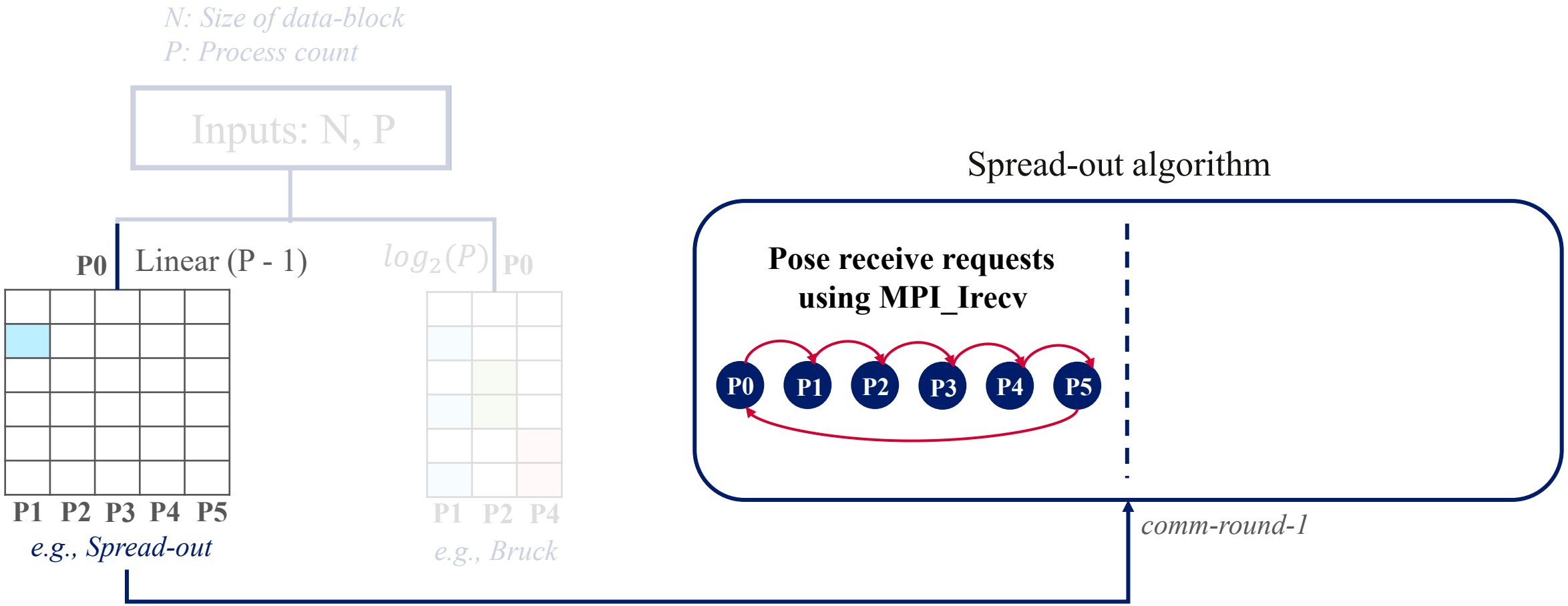
How is non-uniform all-to-all implemented



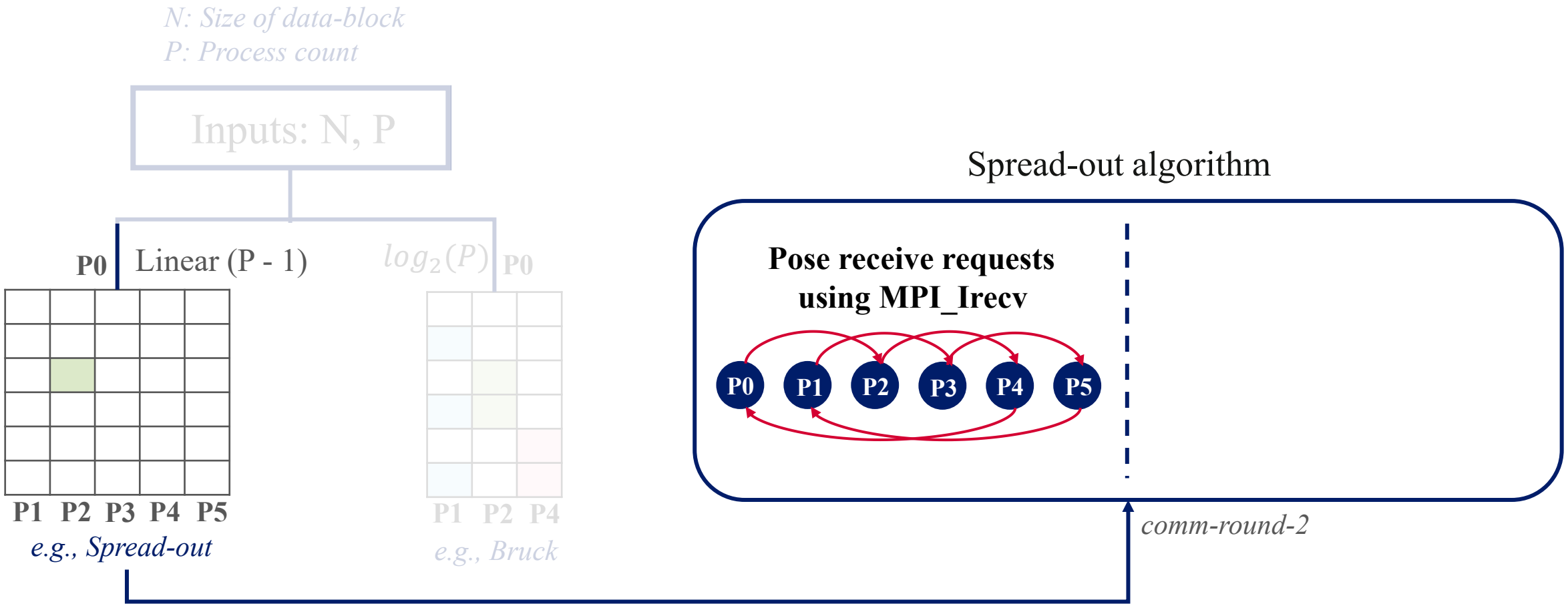
Standard MPI All-to-allv Implementations: Spread-out



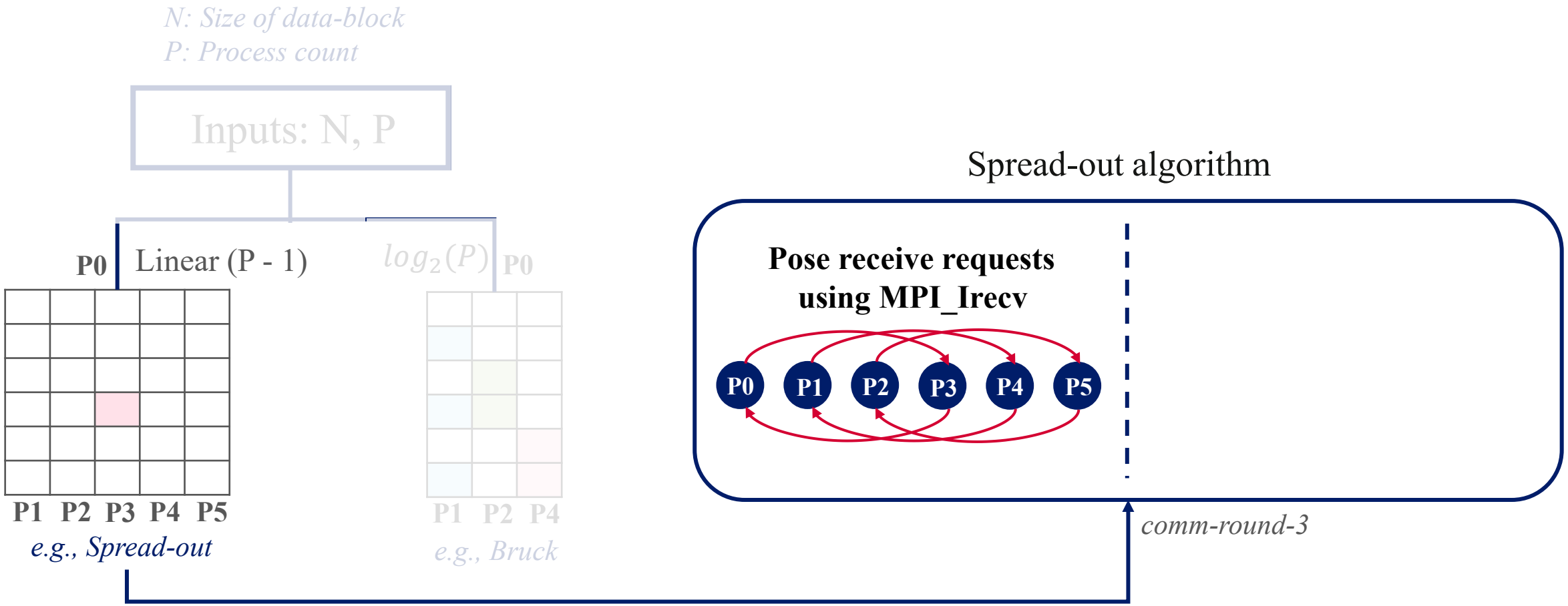
Standard MPI All-to-allv Implementations: Spread-out



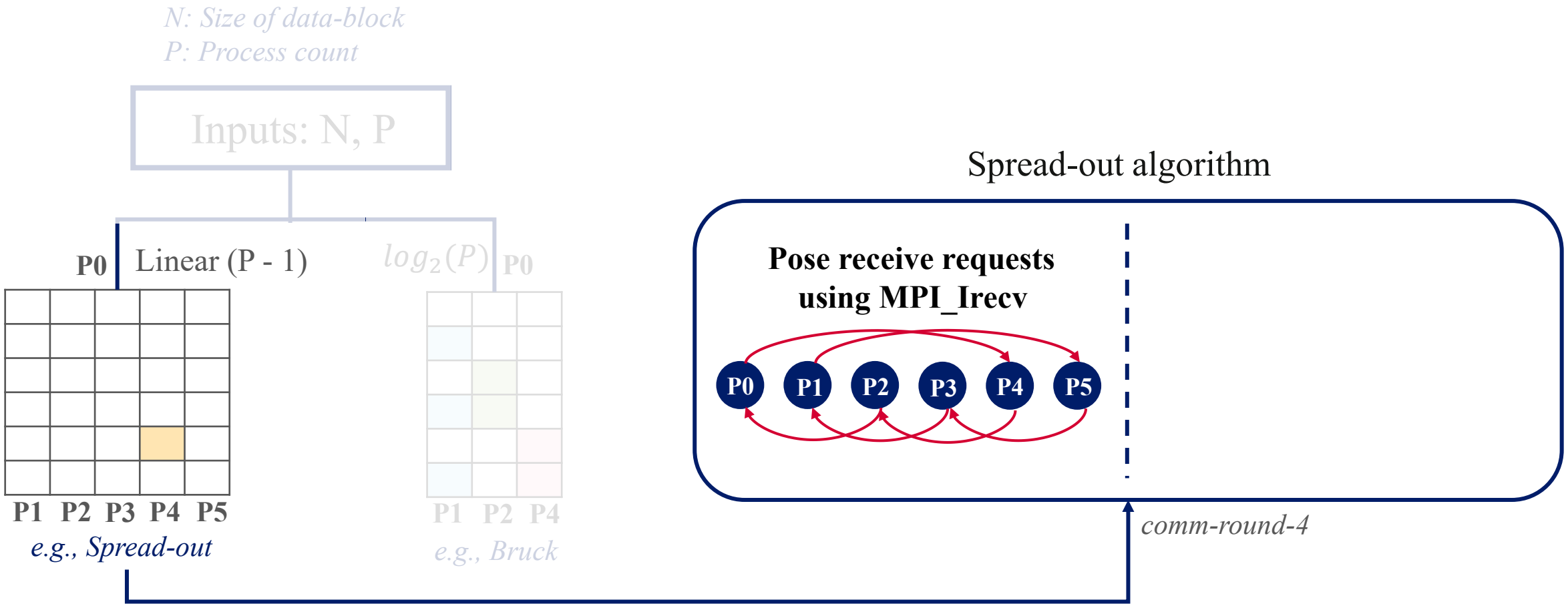
Standard MPI All-to-allv Implementations: Spread-out



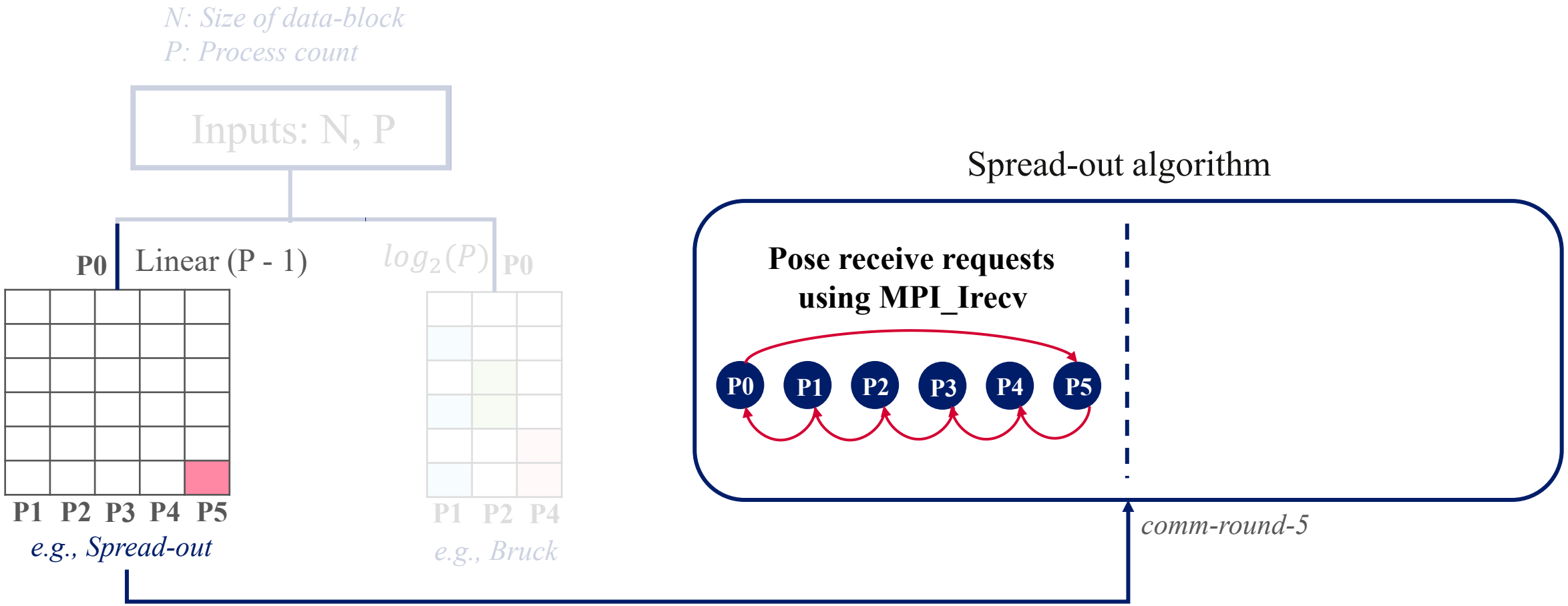
Standard MPI All-to-allv Implementations: Spread-out



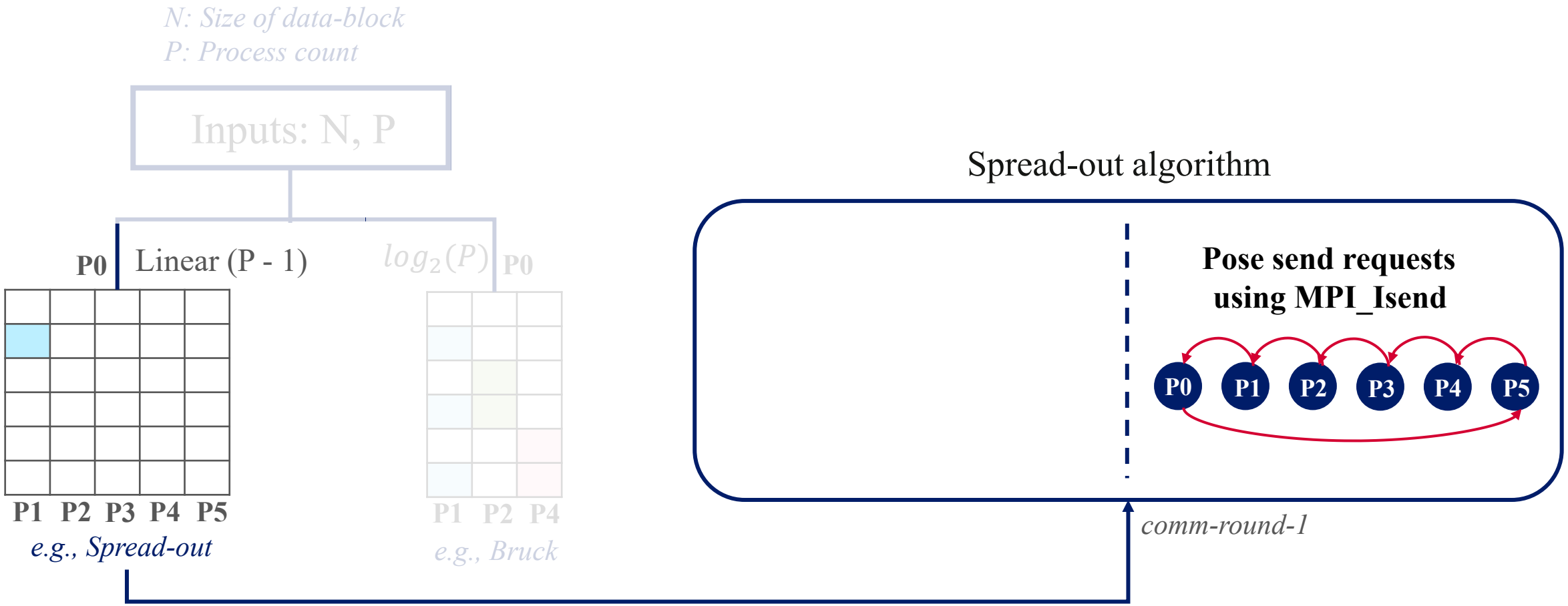
Standard MPI All-to-allv Implementations: Spread-out



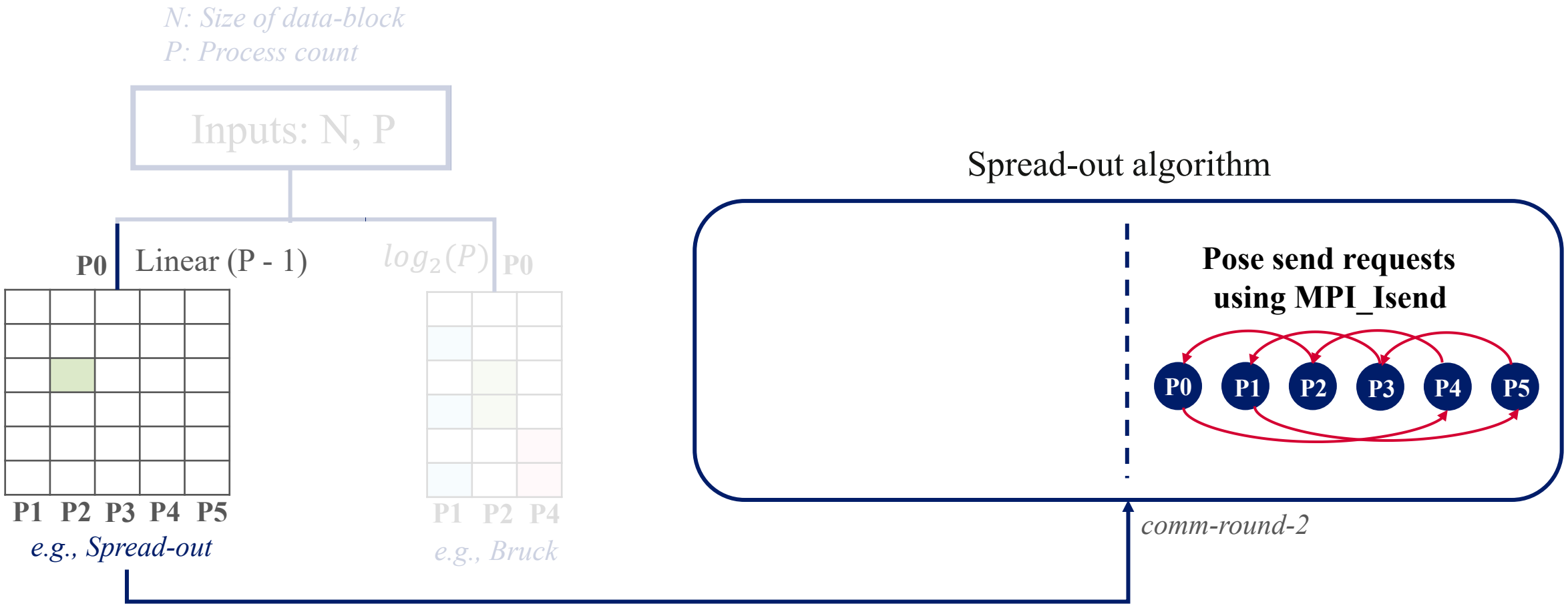
Standard MPI All-to-allv Implementations: Spread-out



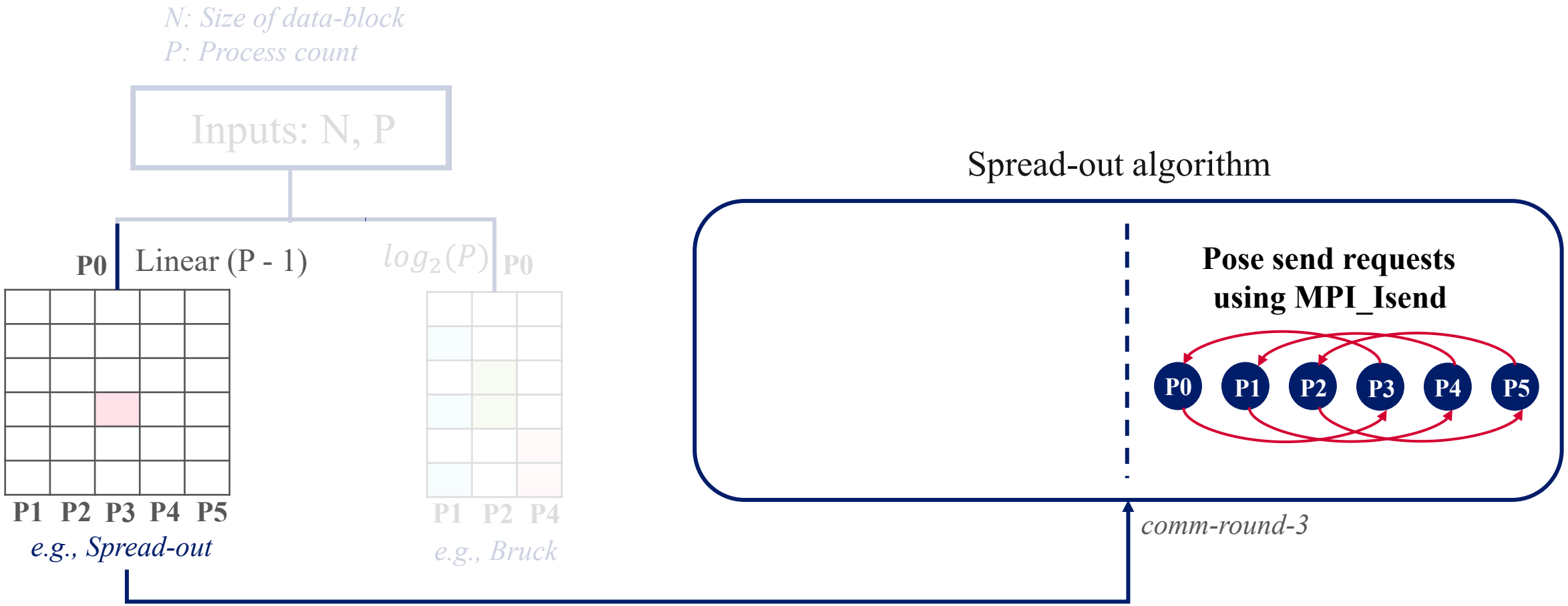
Standard MPI All-to-allv Implementations: Spread-out



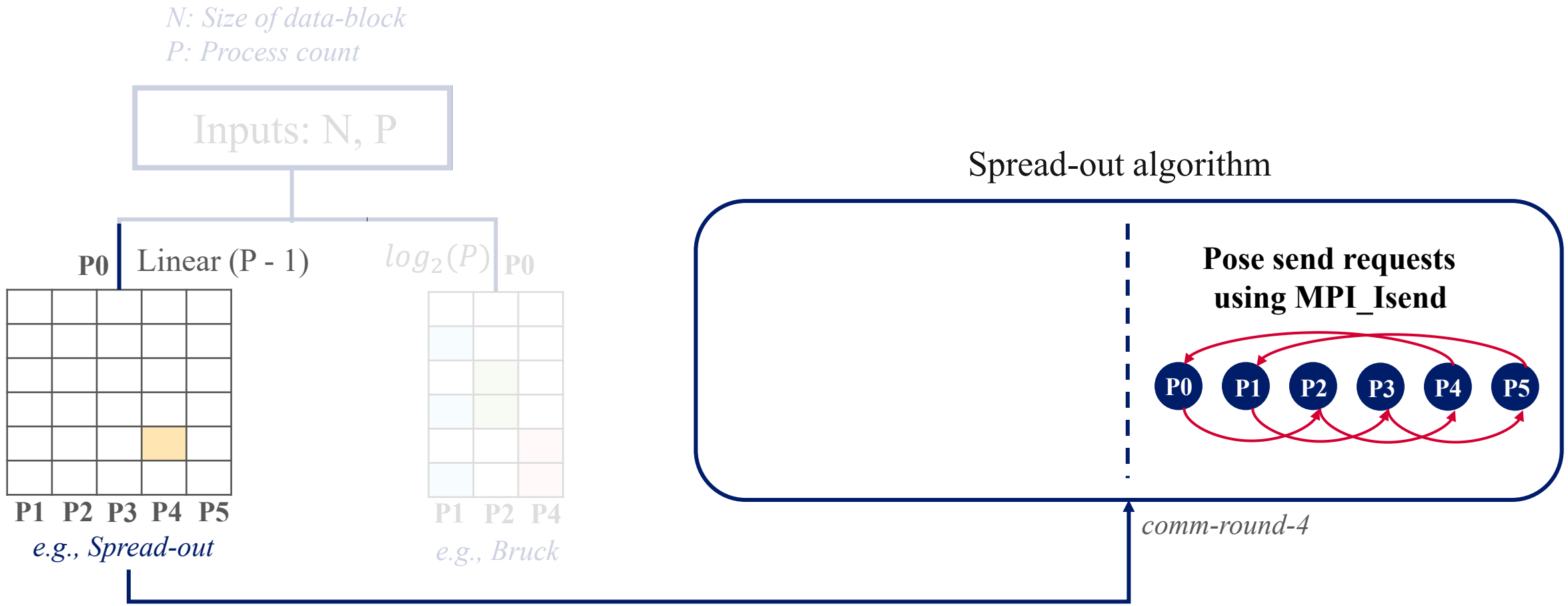
Standard MPI All-to-allv Implementations: Spread-out



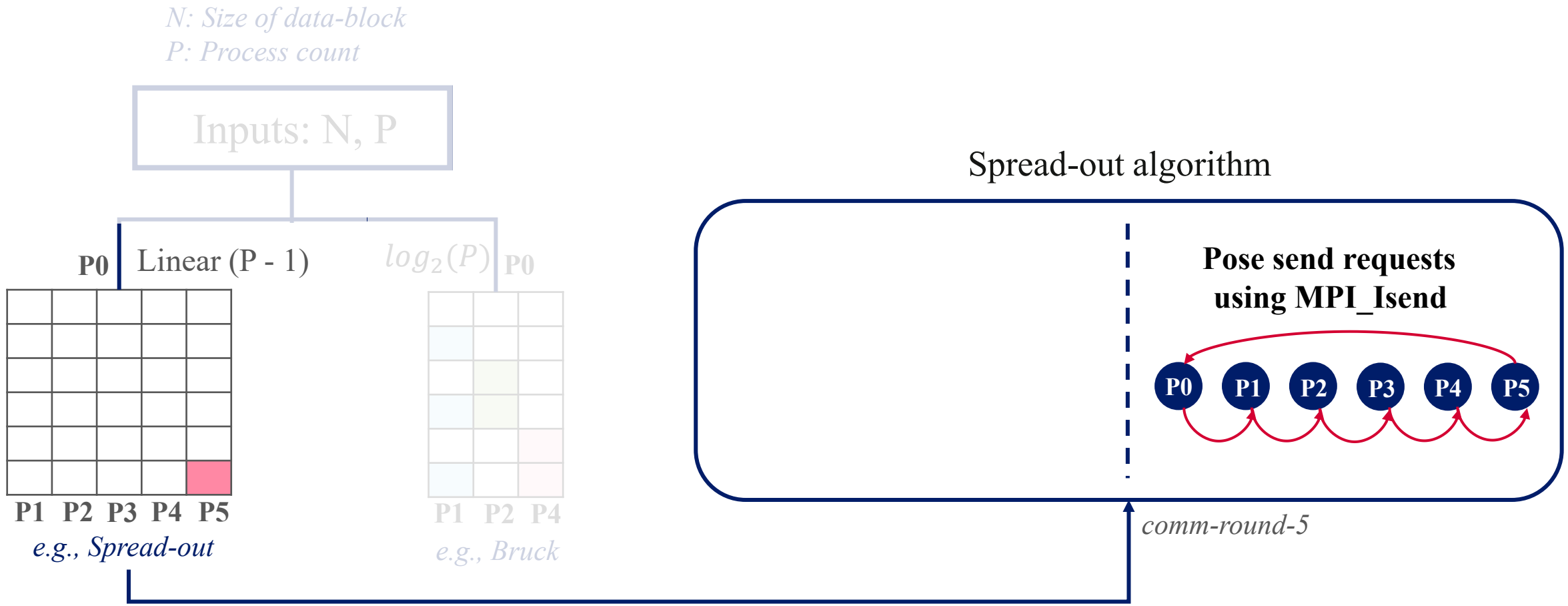
Standard MPI All-to-allv Implementations: Spread-out



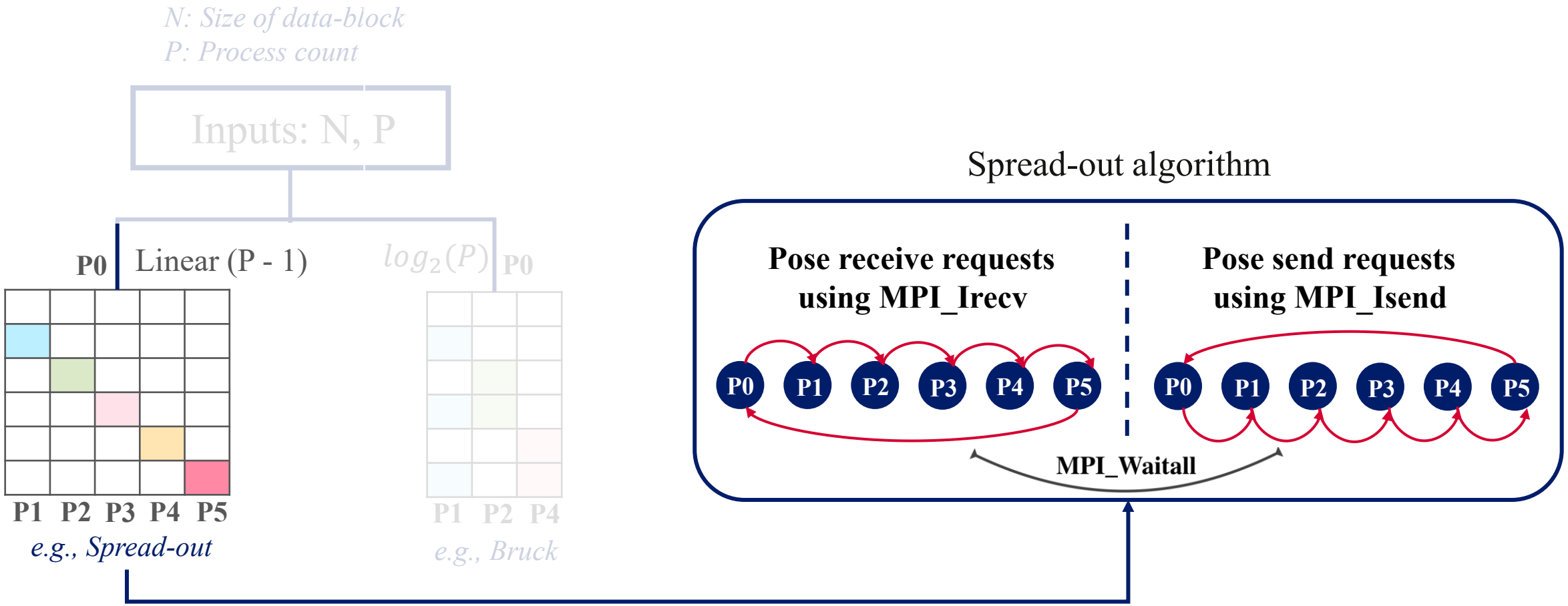
Standard MPI All-to-allv Implementations: Spread-out



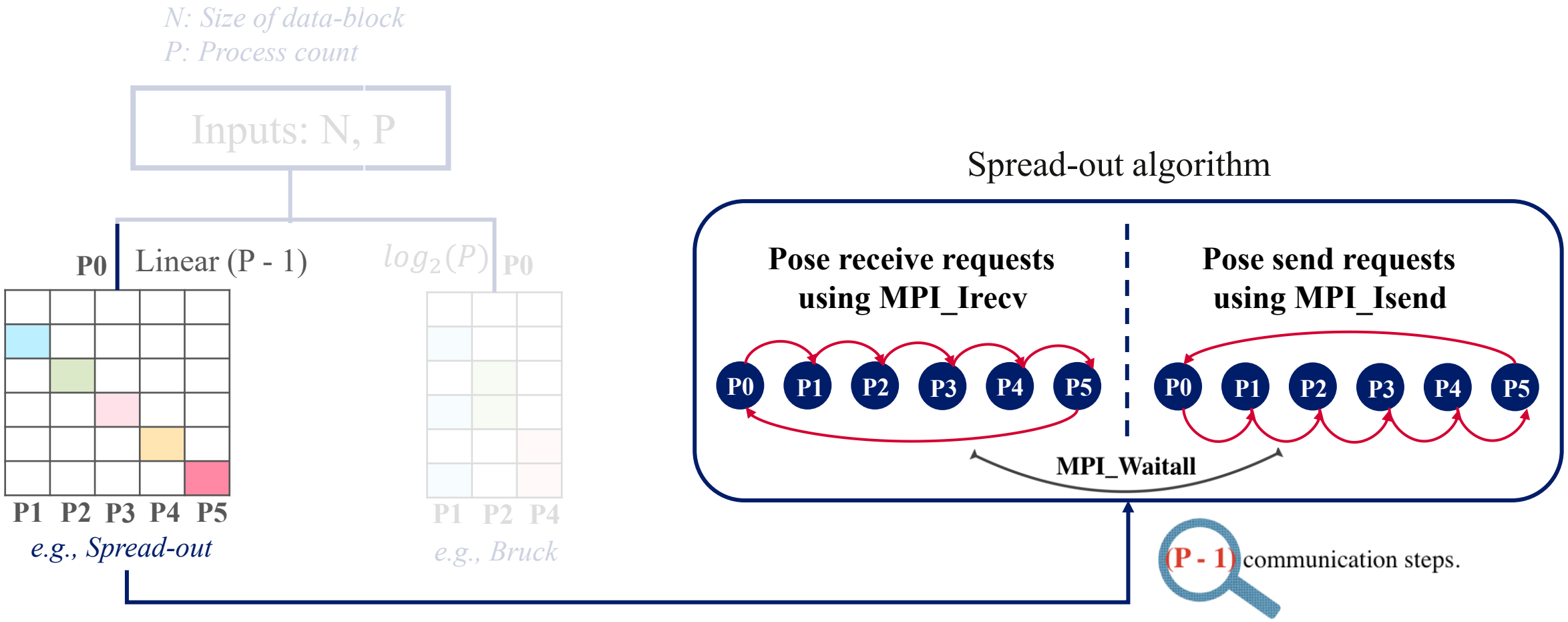
Standard MPI All-to-allv Implementations: Spread-out



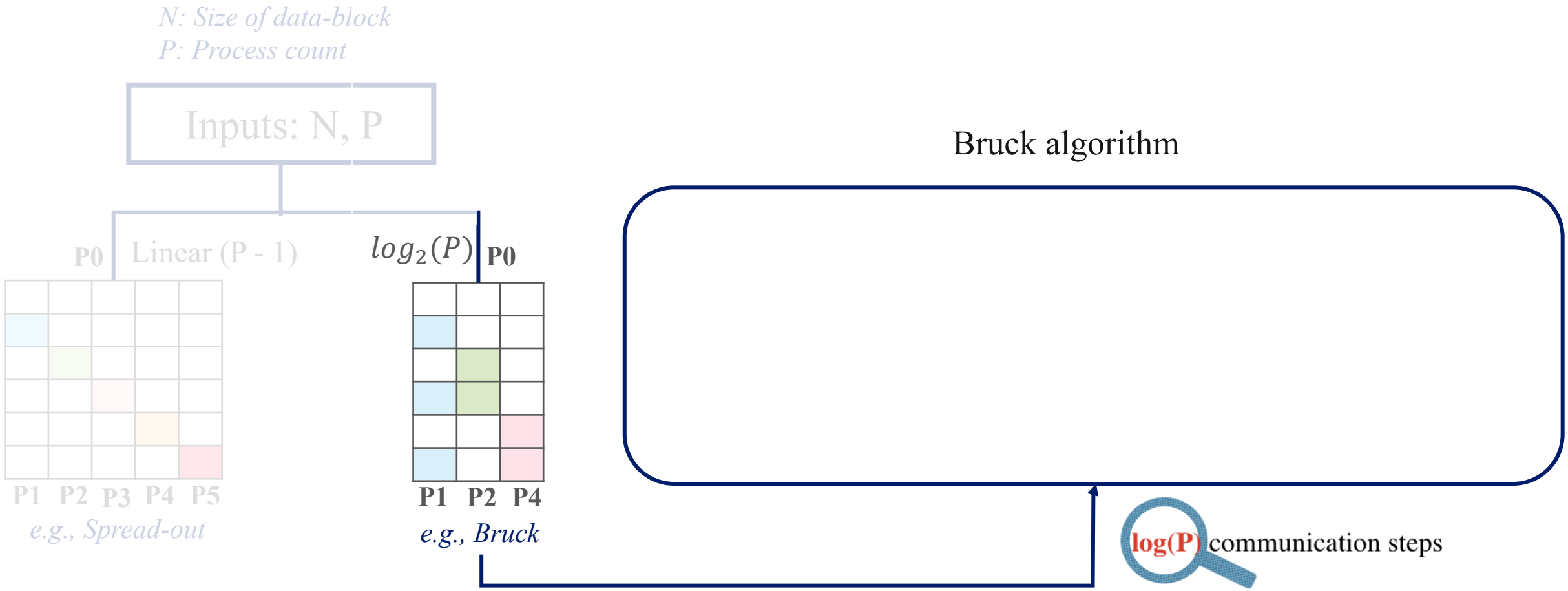
Standard MPI All-to-allv Implementations: Spread-out



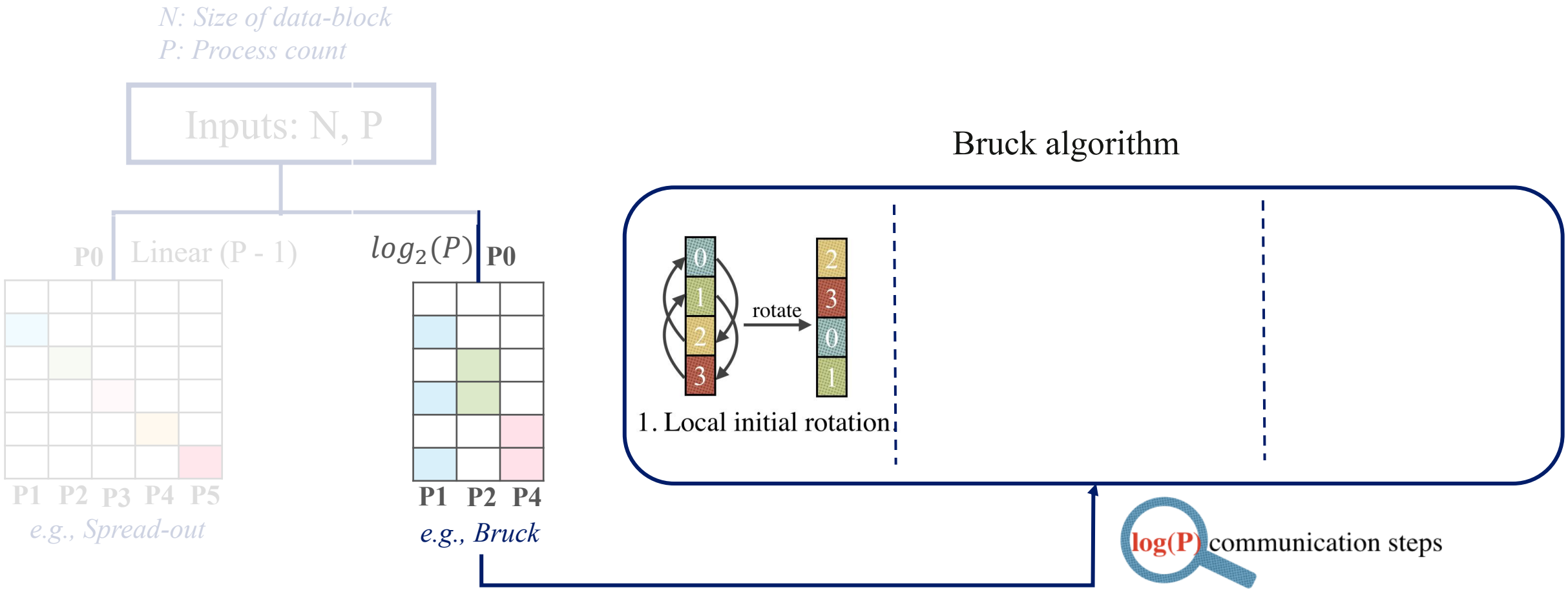
Standard MPI All-to-allv Implementations: Spread-out



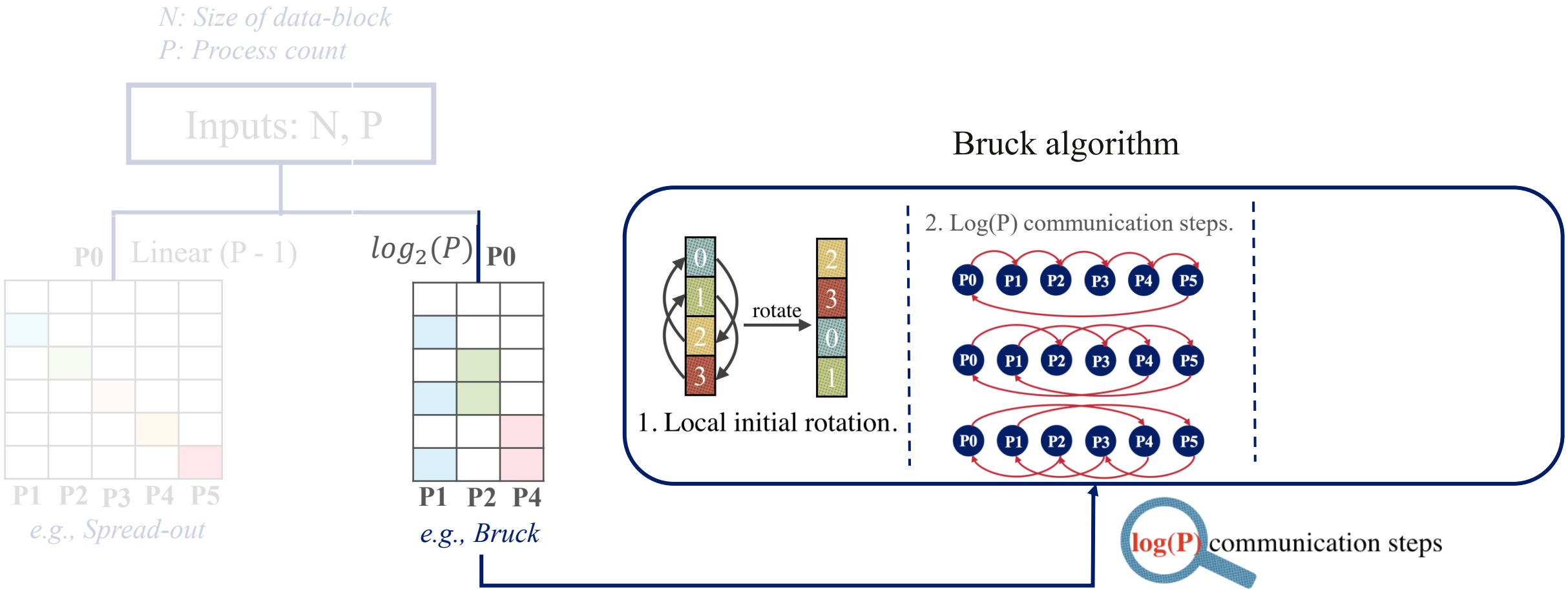
Standard MPI All-to-allv Implementations: Bruck



Standard MPI All-to-allv Implementations: Bruck



Standard MPI All-to-allv Implementations: Bruck



Standard MPI All-to-allv Implementations: Comm-0

N: Size of data-block
P: Process count

Inputs: N, P

P0 | Linear (P - 1)

P1 P2 P3 P4 P5

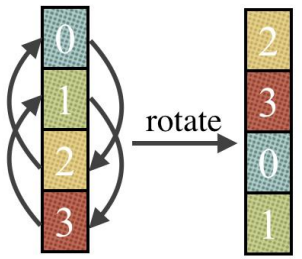
e.g., Spread-out

$log_2(P)$ | P0

P1 P2 P4

e.g., Bruck

Bruck algorithm



1. Local initial rotation

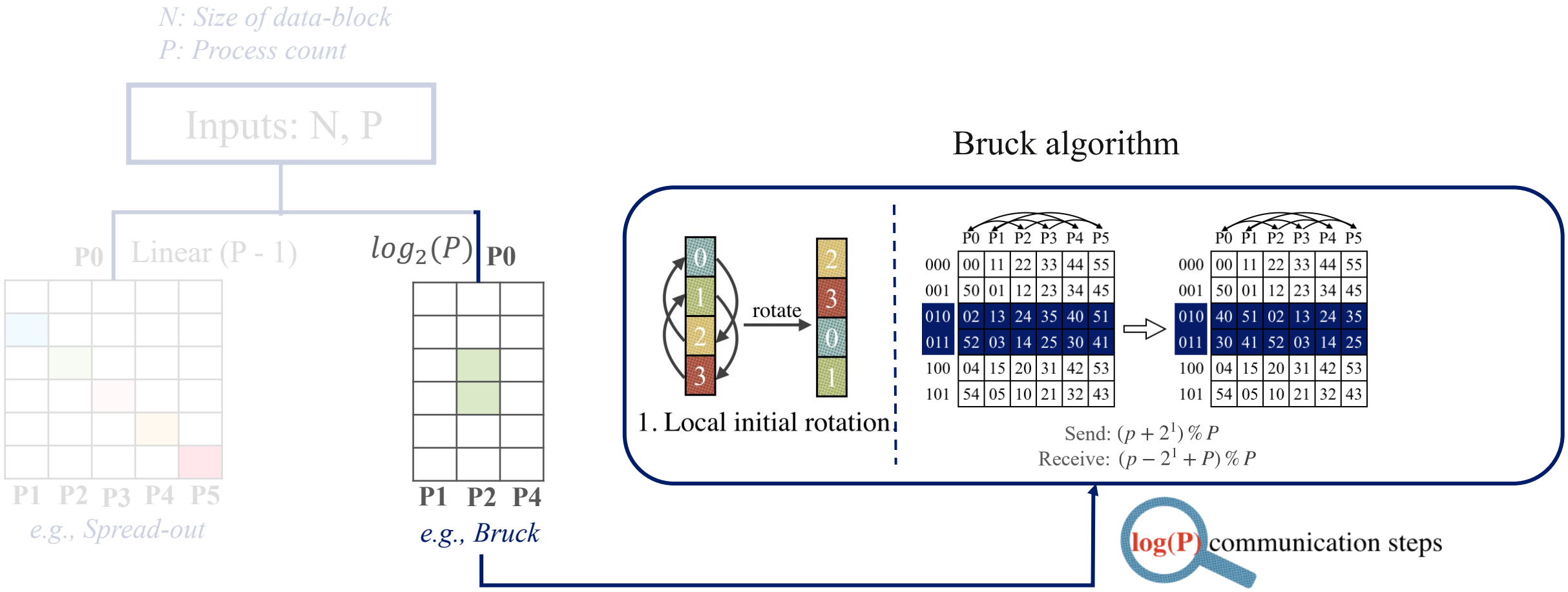
	P0	P1	P2	P3	P4	P5
000	00	11	22	33	44	55
001	01	12	23	34	45	50
010	02	13	24	35	40	51
011	03	14	25	30	41	52
100	04	15	20	31	42	53
101	05	10	21	32	43	54

	P0	P1	P2	P3	P4	P5
000	00	11	22	33	44	55
001	50	01	12	23	34	45
010	02	13	24	35	40	51
011	52	03	14	25	30	41
100	04	15	20	31	42	53
101	54	05	10	21	32	43

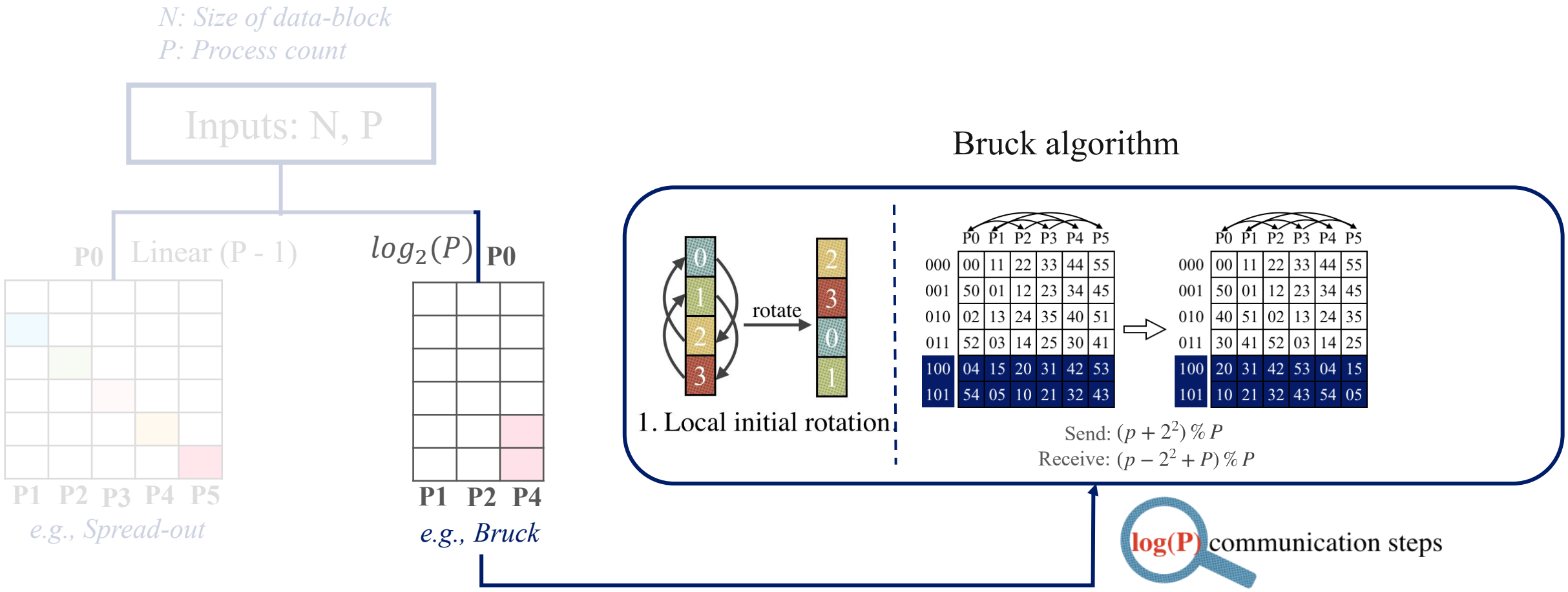
Send: $(p + 2^0)\ \%\ P$
Receive: $(p - 2^0 + P)\ \%\ P$

log(P) communication steps

Standard MPI All-to-allv Implementations: Comm-1



Standard MPI All-to-allv Implementations: Comm-2



Standard MPI All-to-allv Implementations: Bruck

N: Size of data-block
P: Process count

Inputs: N, P

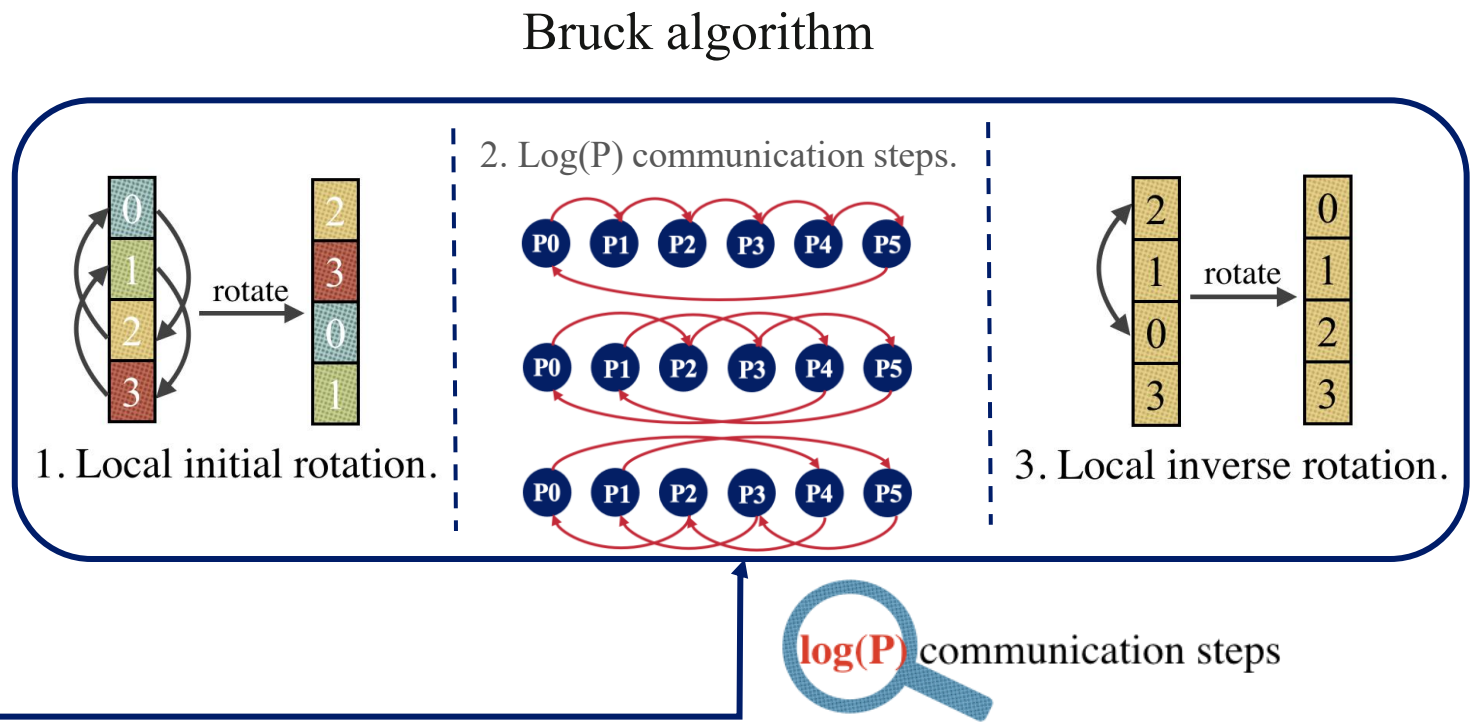
P0 | Linear (P - 1) | $log_2(P)$ P0

P1 P2 P3 P4 P5

e.g., Spread-out

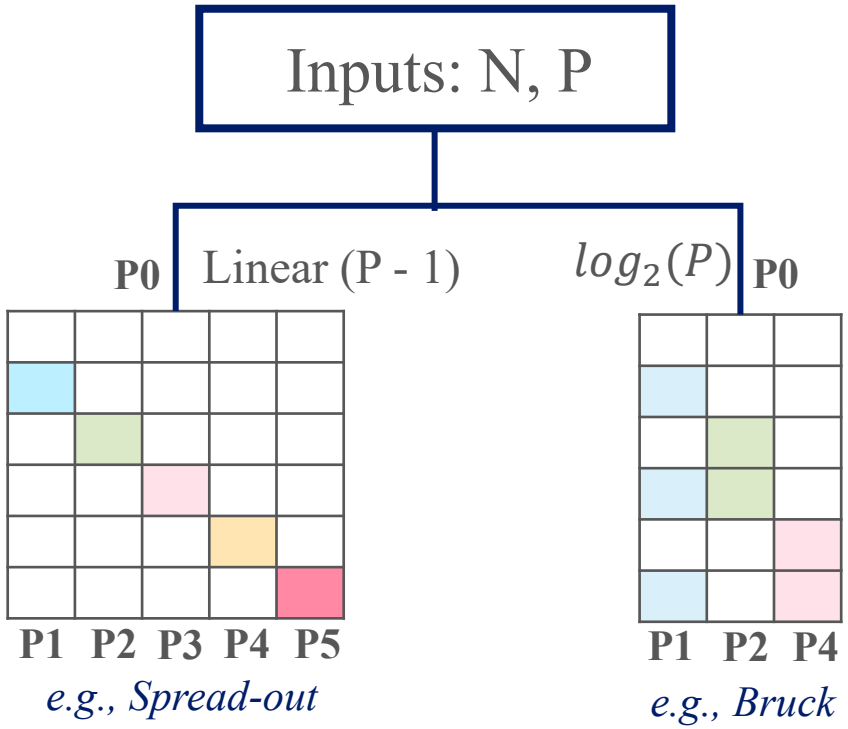
P1 P2 P4

e.g., Bruck



Comparison of These Two Algorithms

N: Size of data-block
P: Process count



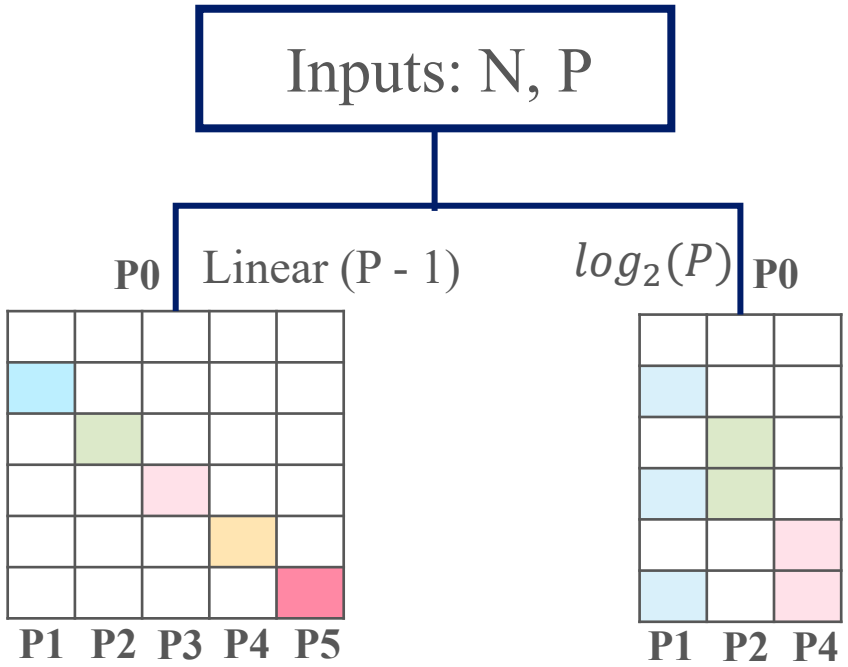
e.g., Spread-out

e.g., Bruck

Exchanging 5 data-blocks via 5 rounds

Comparison of These Two Algorithms

N: Size of data-block
P: Process count

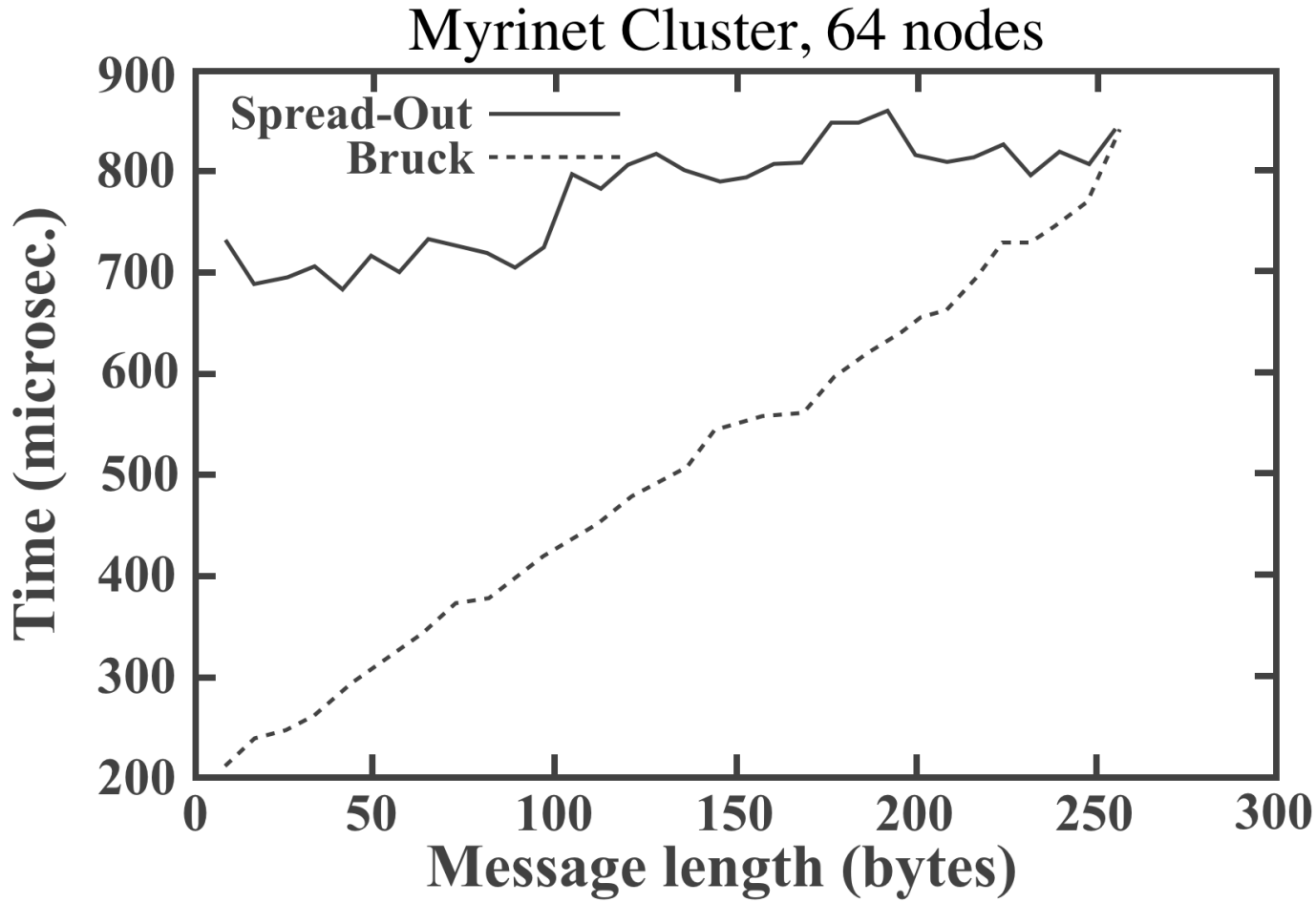


e.g., Spread-out

Exchanging 5 data-blocks via 5 rounds

e.g., Bruck

Exchanging 7 data-blocks via 3 rounds



New algorithms for non-uniform all-to-allv

ParAta (parameterized All-to-Allv)

Generalized bruck algorithm with configurable radix

HieAta (Hierarchical All-to-allv)

Intra-node: ParAta
Inter-node: batched spread-out with tunable batch size

New algorithms for non-uniform all-to-allv

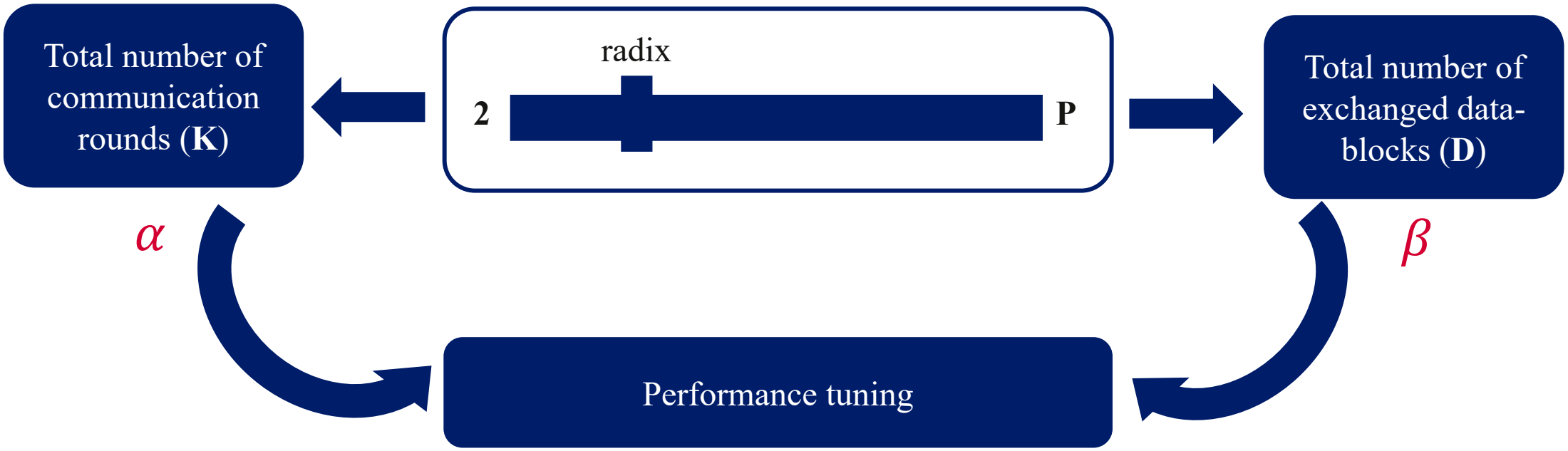
ParAta (parameterized All-to-Allv)

Generalized bruck algorithm with configurable radix

HieAta (Hierarchical All-to-allv)

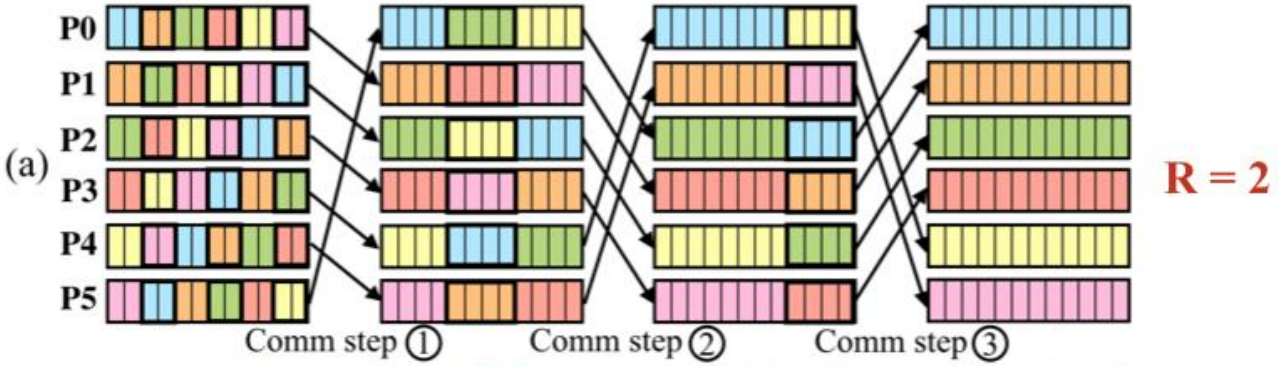
Intra-node: ParAta
Inter-node: batched spread-out with tunable batch size

Tuning Radices Enables Performance Tuning

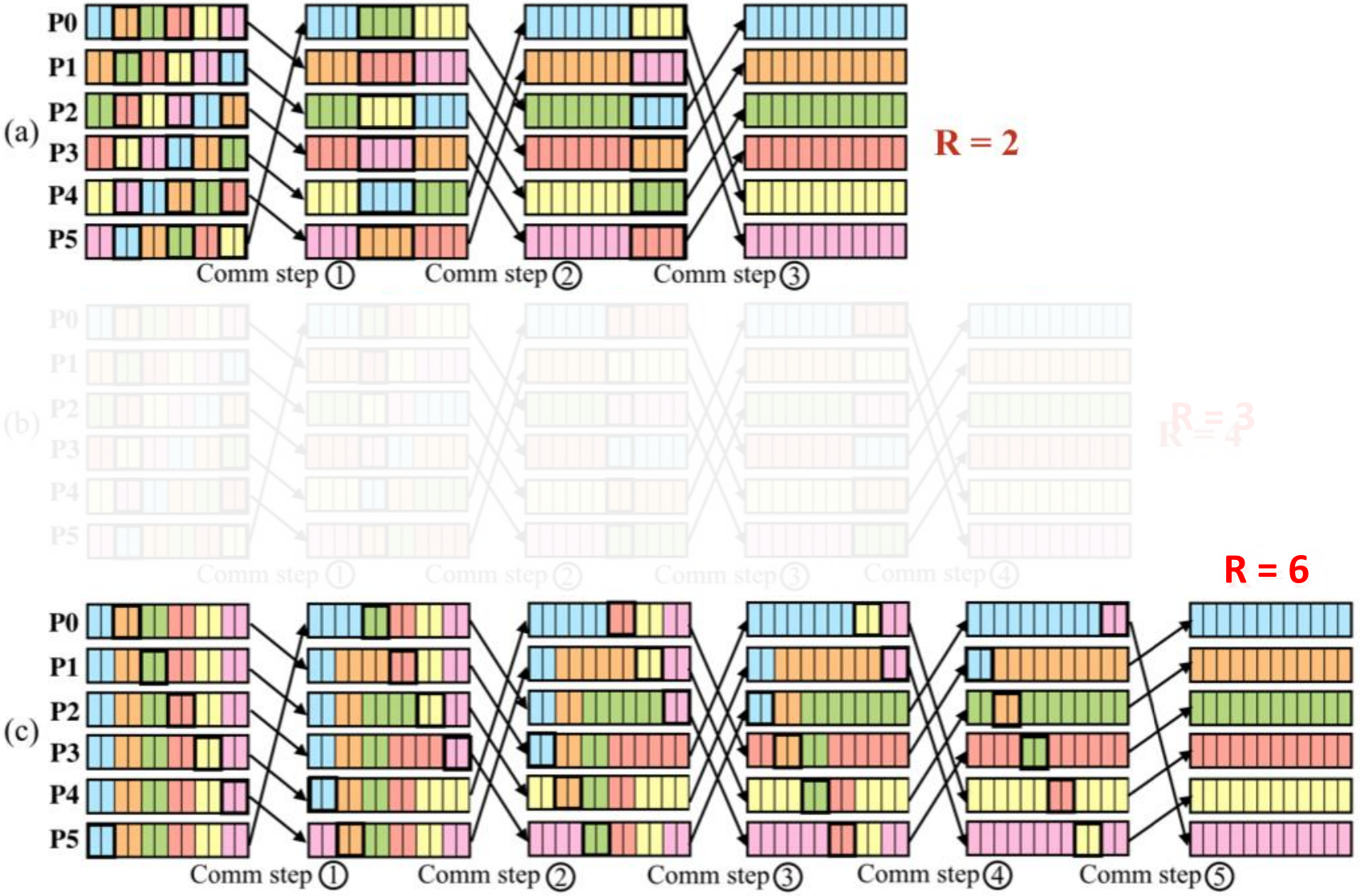


ParAta (parameterized All-to-Allv)

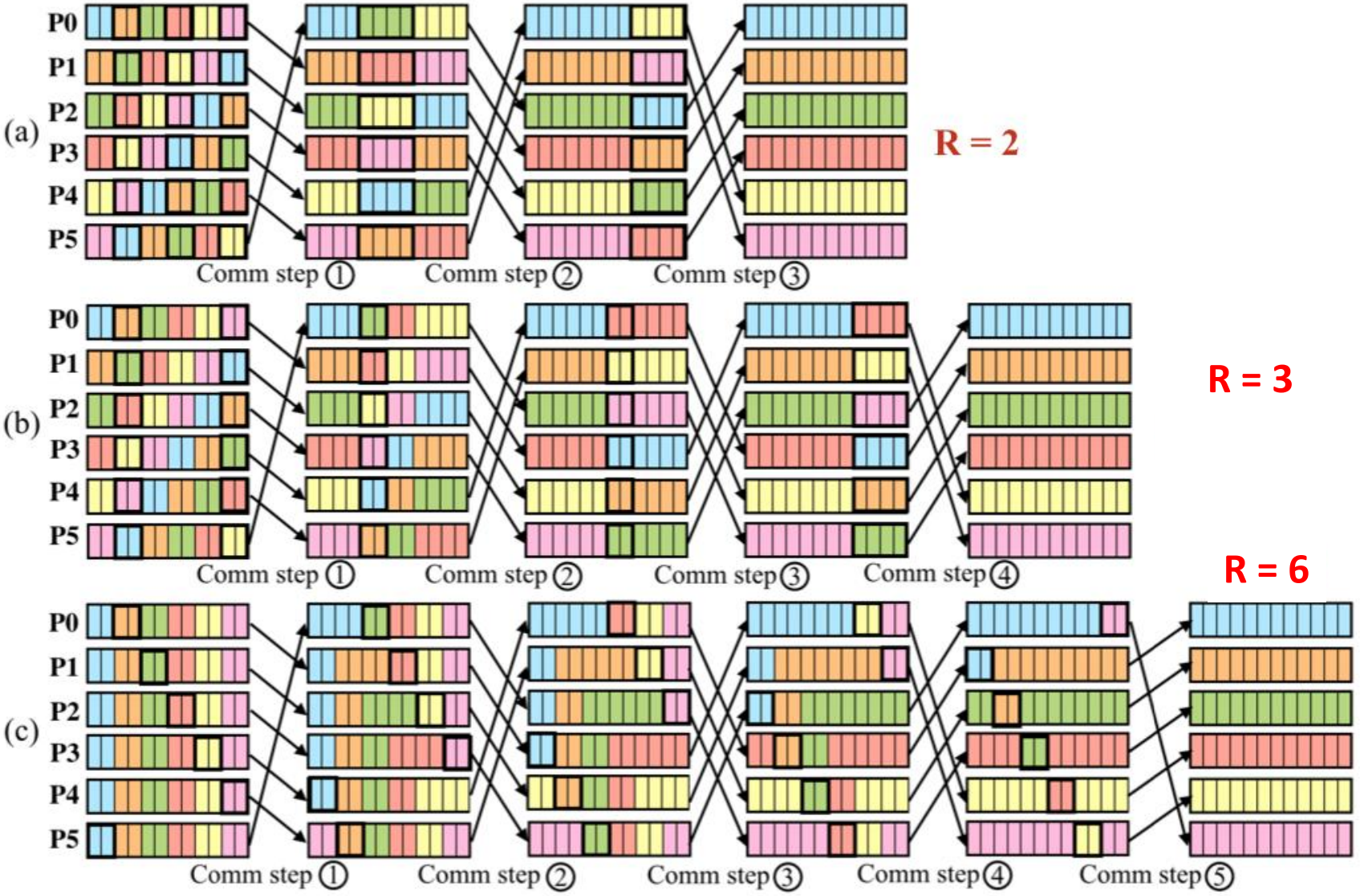
ParAta, that lets one pick the radix r



ParAta, that lets one pick the radix r



ParAta, that lets one pick the radix r



New algorithms for non-uniform all-to-allv

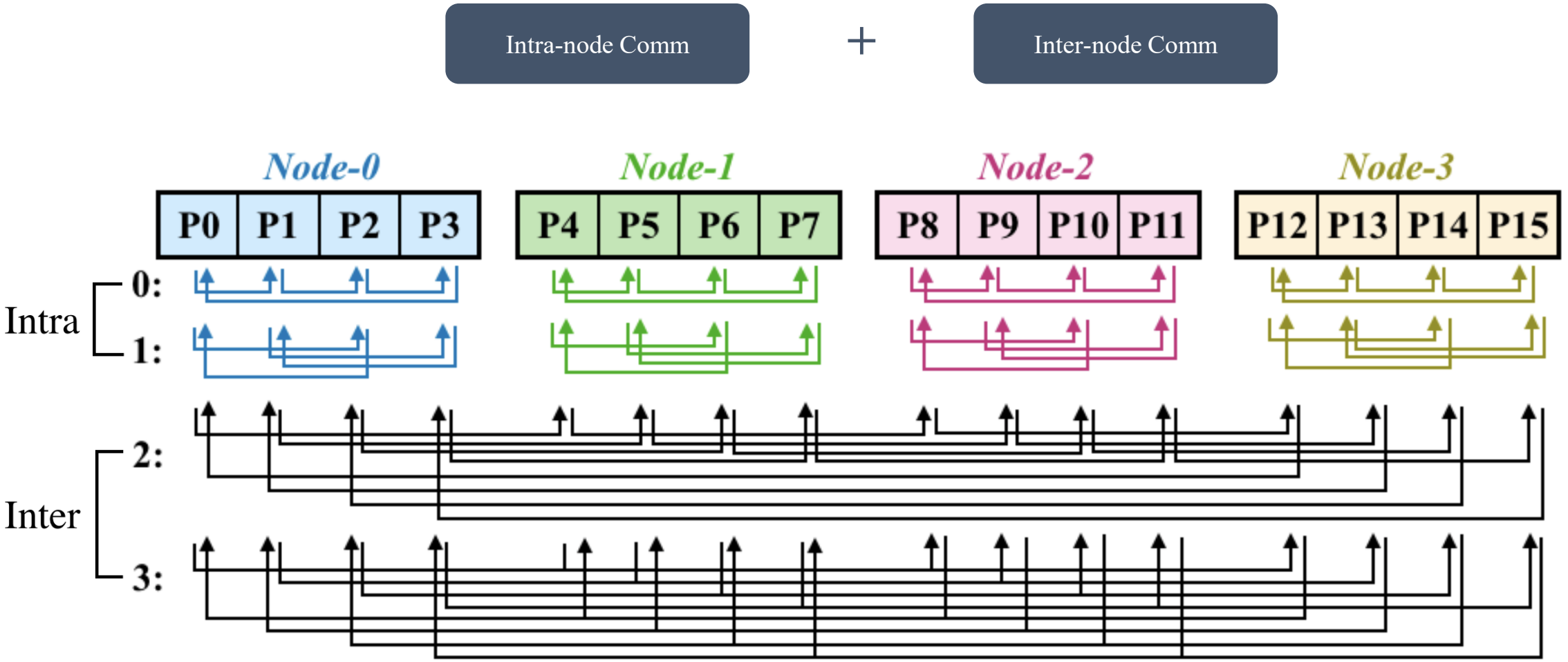
ParAta (parameterized All-to-Allv)

Generalized bruck algorithm with configurable radix

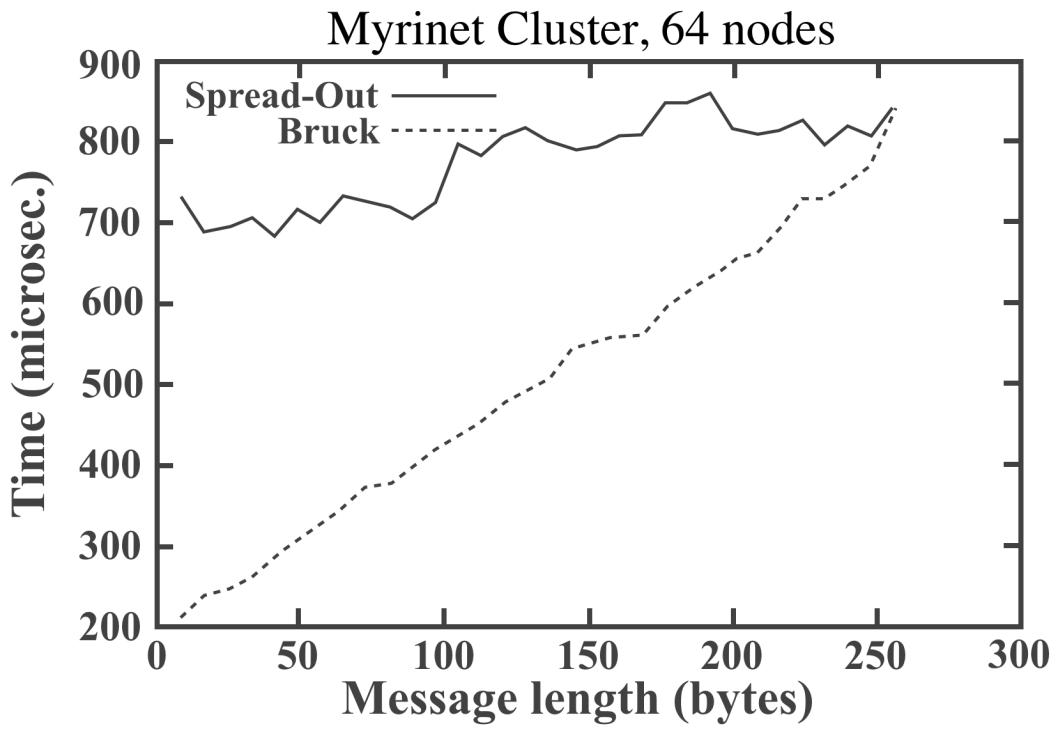
HieAta (Hierarchical All-to-allv)

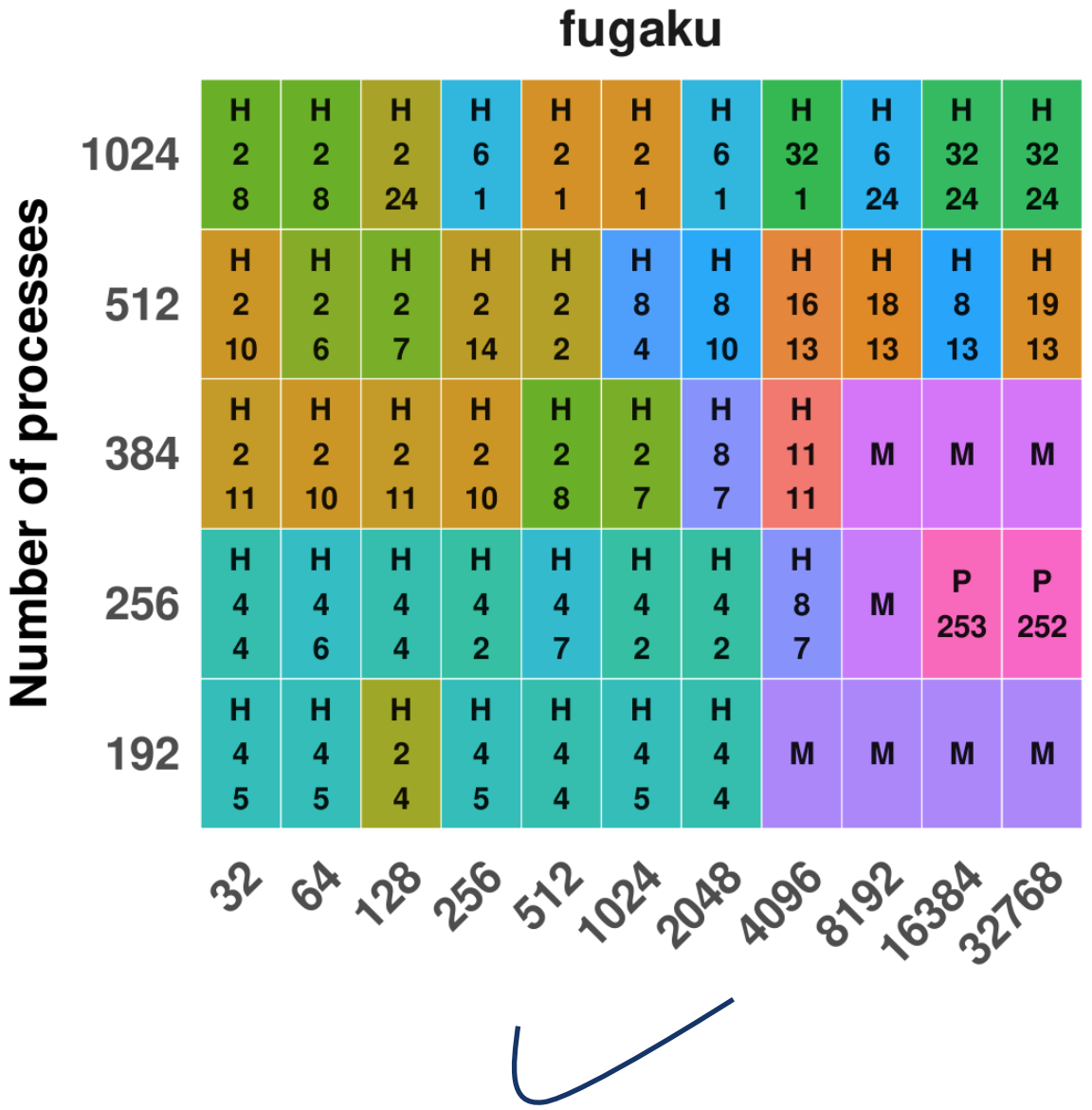
Intra-node: ParAta
Inter-node: batched spread-out with tunable batch size

HieAta (Hierarchical All-to-allv) – lets one exploit system hierarchy

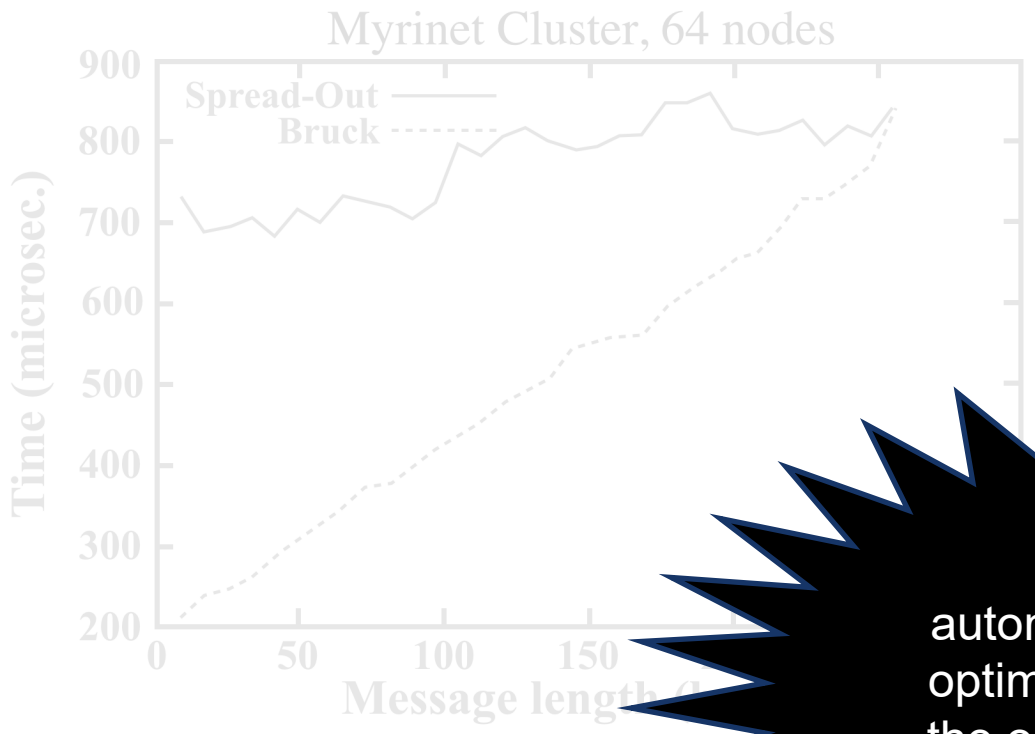


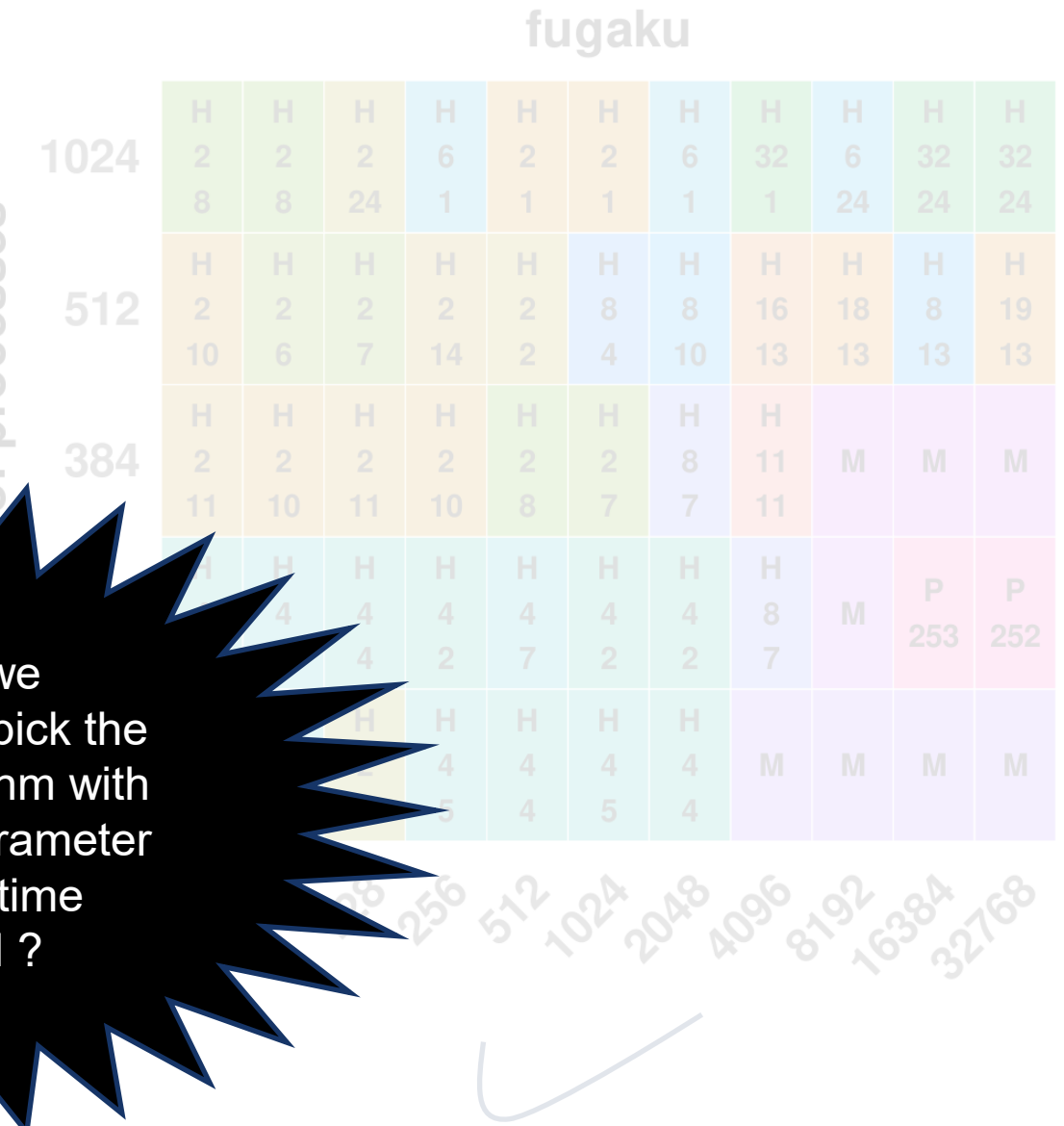
Need a better way to select the optimal algorithm and parameters





Need a better way to select the optimal algorithm and parameters





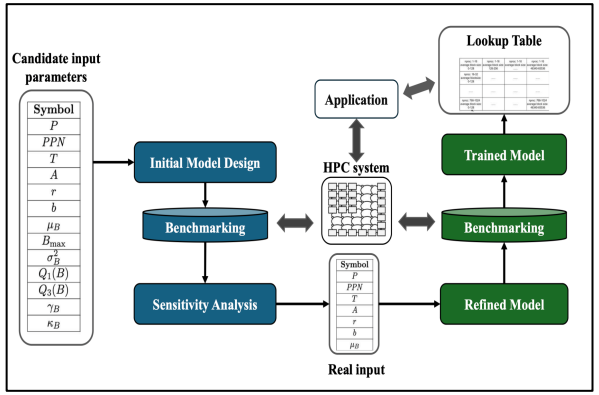
How do we automatically pick the optimal algorithm with the optimal parameter without runtime overhead ?

Outline

Motivation and challenges

Proposed approach

Results

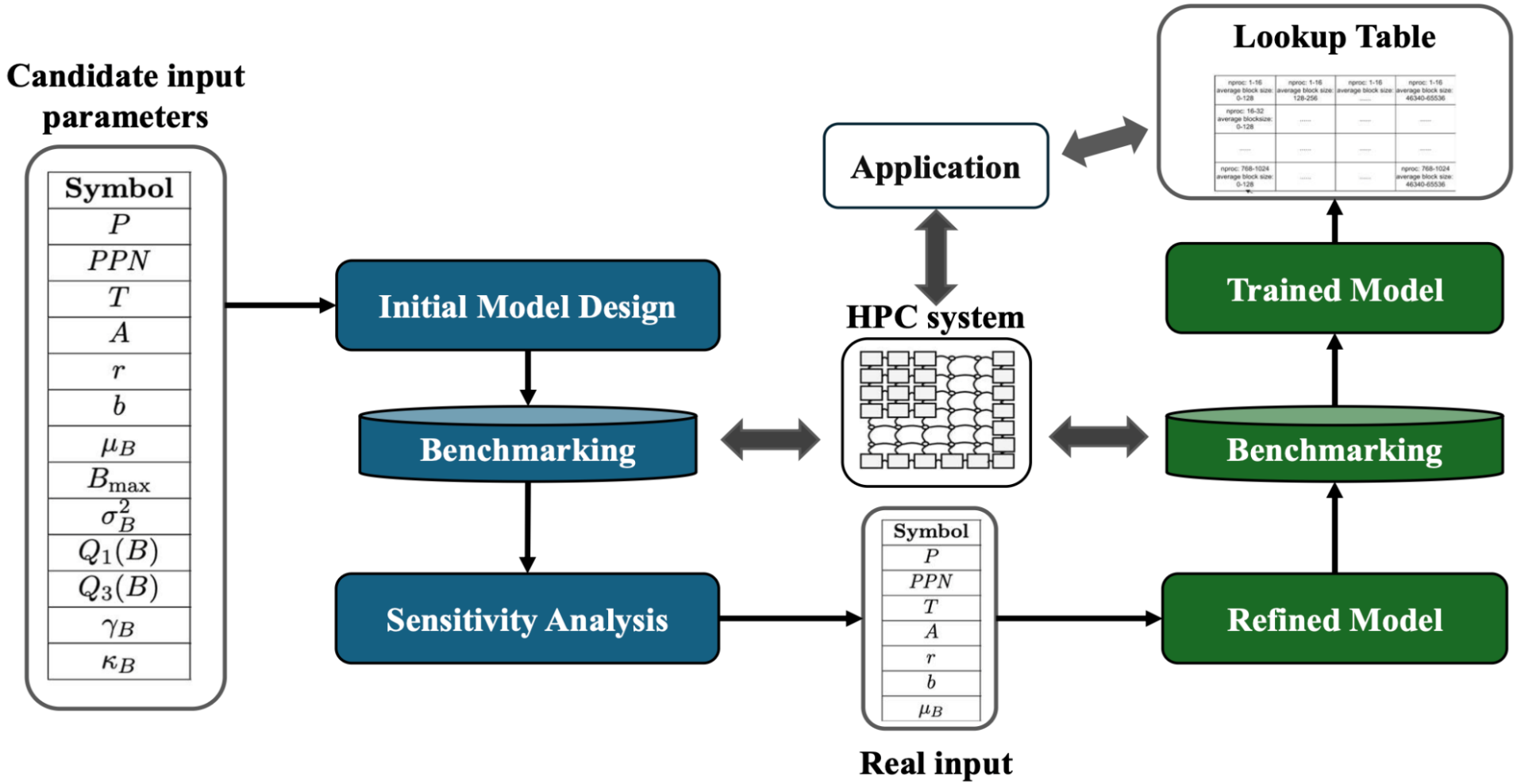


Goals

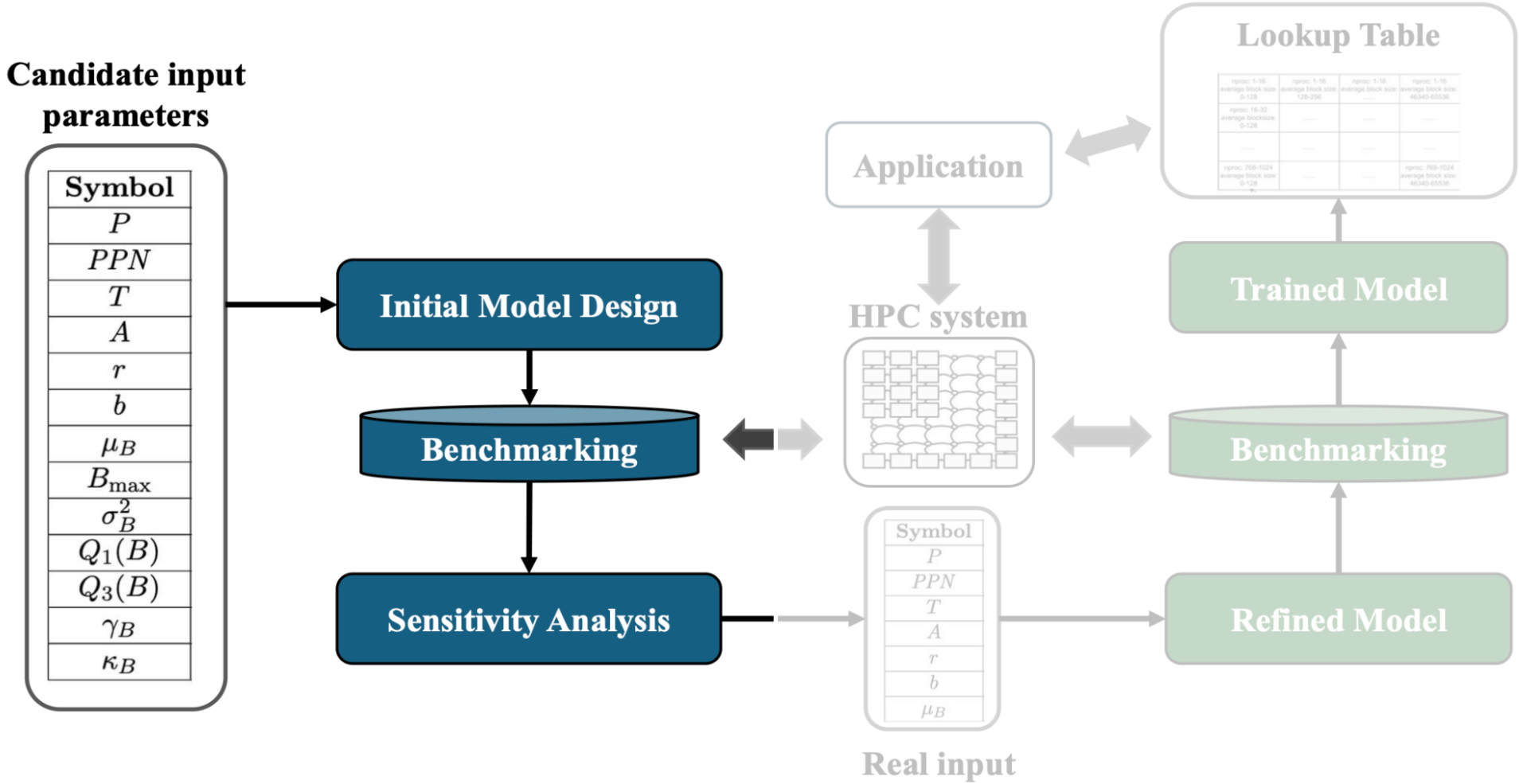
Develop a data-driven, ML-based autotuning framework for non-uniform AlltoAll

- Automatic algorithm selection at runtime
- Adaptive parameter tuning (radix, batch size) to match workload
- Handle irregular and dynamic communication patterns without manual tuning
- Minimal runtime overhead

Methodology and Workflow



Methodology and Workflow



Initial Model

The factors initially considered as the input for the performance prediction model.

Symbol	Description	Type
P	Total number of processes	Execution
PPN	Number of processes per node	Execution
τ	Network topology	Platform-specific
A	Algorithm selection	Model-specific
r	Radix of algorithm	Algorithmic-specific
b	batch_size: max number of comm requests	Algorithmic-specific
μ_B	Average of data-block sizes	Statistical
$B_{\max}$	Maximum of data-block sizes	Statistical
σ_B^2	Variance of data-block sizes	Statistical
$Q_1(B)$	Sample quantile of data-block sizes (25%)	Statistical
$Q_3(B)$	Sample quantile of data-block sizes (75%)	Statistical
γ_B	Skewness of data-block sizes	Statistical
κ_B	Kurtosis of data-block sizes	Statistical

Initial Model

Here are the system parameters

Symbol	Description	Type
P	Total number of processes	Execution
PPN	Number of processes per node	Execution
τ	Network topology	Platform-specific
A	Algorithm selection	Model-specific
r	Radix of algorithm	Algorithmic-specific
b	batch_size: max number of comm requests	Algorithmic-specific
μ_B	Average of data-block sizes	Statistical
$B_{\max}$	Maximum of data-block sizes	Statistical
σ_B^2	Variance of data-block sizes	Statistical
$Q_1(B)$	Sample quantile of data-block sizes (25%)	Statistical
$Q_3(B)$	Sample quantile of data-block sizes (75%)	Statistical
γ_B	Skewness of data-block sizes	Statistical
κ_B	Kurtosis of data-block sizes	Statistical

Initial Model

Here are the algorithmic-specific parameters such as what algorithm model to choose and the parameter values for the chosen algorithm

Symbol	Description	Type
P	Total number of processes	Execution
PPN	Number of processes per node	Execution
τ	Network topology	Platform-specific
A	Algorithm selection	Model-specific
r	Radix of algorithm	Algorithmic-specific
b	batch_size: max number of comm requests	Algorithmic-specific
μ_B	Average of data-block sizes	Statistical
$B_{\max}$	Maximum of data-block sizes	Statistical
σ_B^2	Variance of data-block sizes	Statistical
$Q_1(B)$	Sample quantile of data-block sizes (25%)	Statistical
$Q_3(B)$	Sample quantile of data-block sizes (75%)	Statistical
γ_B	Skewness of data-block sizes	Statistical
κ_B	Kurtosis of data-block sizes	Statistical

Initial Model

Here are the statistacal parameters regarding the varing message size during non-uniform Alltoall

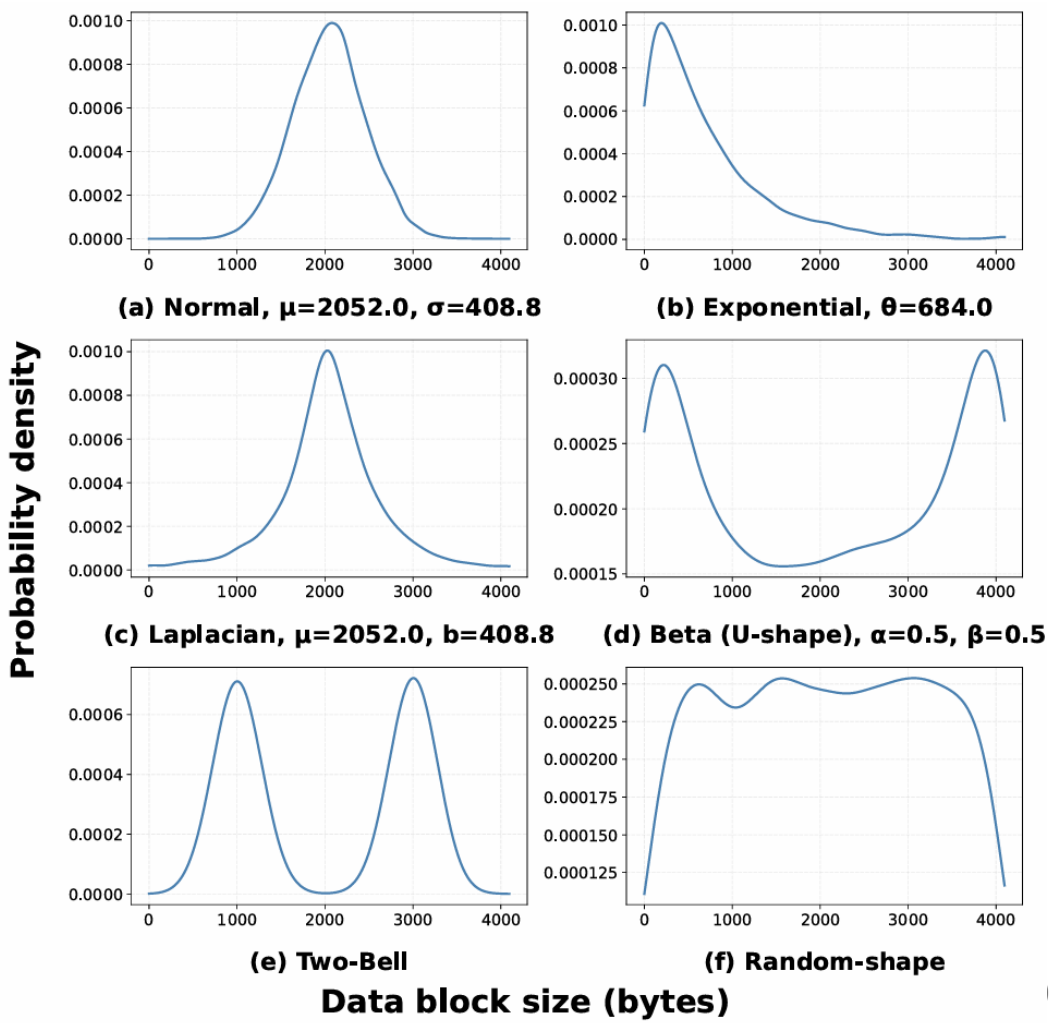
However, there are too much statistical parameters and use all of them will cause significant overhead on-the-fly.

Symbol	Description	Type
P	Total number of processes	Execution
PPN	Number of processes per node	Execution
τ	Network topology	Platform-specific
A	Algorithm selection	Model-specific
r	Radix of algorithm	Algorithmic-specific
b	batch_size: max number of comm requests	Algorithmic-specific
μ_B	Average of data-block sizes	Statistical
$B_{\max}$	Maximum of data-block sizes	Statistical
σ_B^2	Variance of data-block sizes	Statistical
$Q_1(B)$	Sample quantile of data-block sizes (25%)	Statistical
$Q_3(B)$	Sample quantile of data-block sizes (75%)	Statistical
γ_B	Skewness of data-block sizes	Statistical
κ_B	Kurtosis of data-block sizes	Statistical

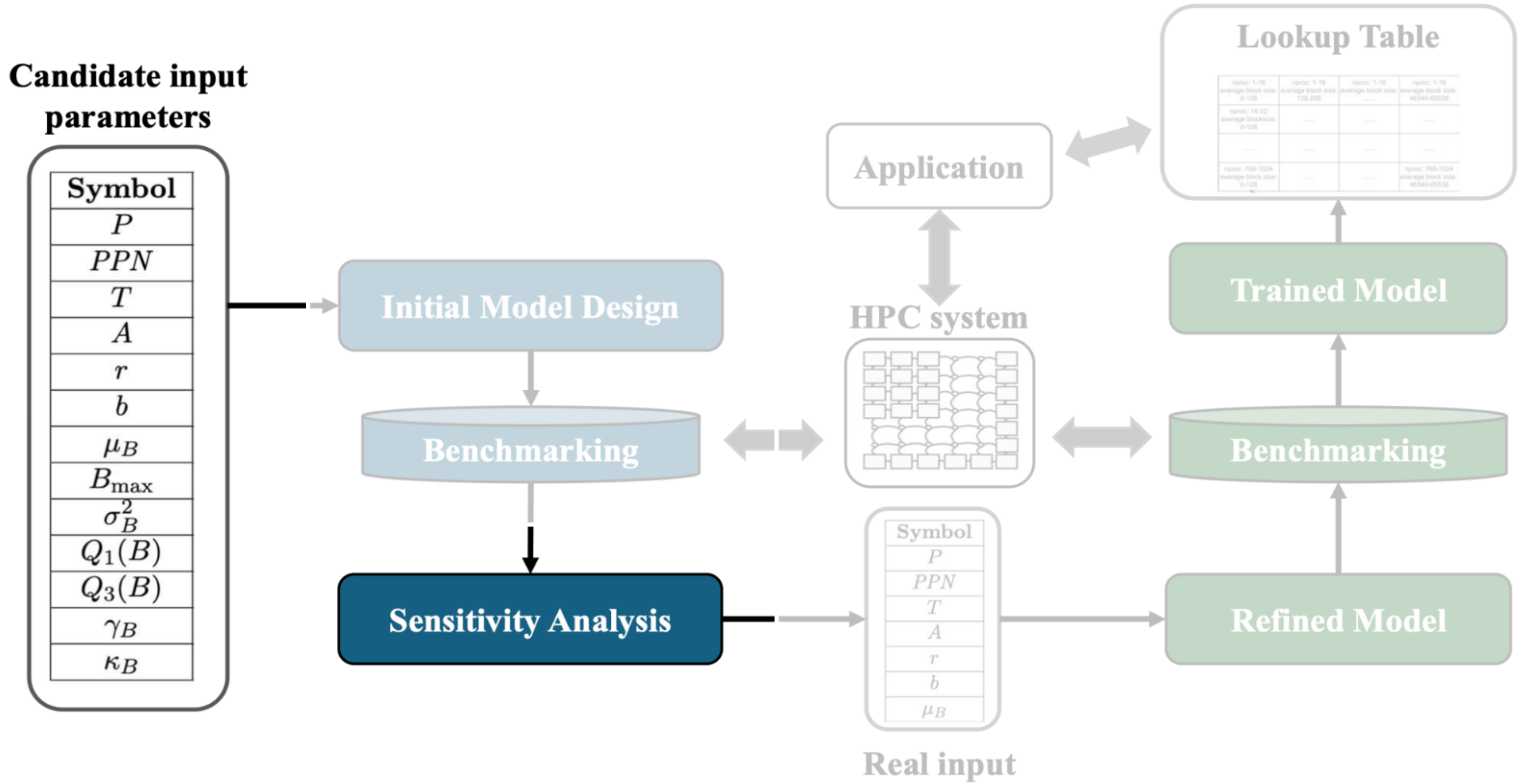
Benchmarking for Sensitivity Analysis

We perform benchmarking in which the varying message sizes generated from different distribution

- Platform: Polaris
- Nproc: 32, 128, 512, 1024, 2048, 4096
- Maximum message size: 8, 32, 128, 512, 2048, 8192
- MPI Alltoallv algorithms:
  - MPI Alltoallv
  - HieAta
    - Radix value: 2,$\sqrt{ppn}$, ppn
    - Batch size value: num_node
  - ParAta
    - Radix value: 2, $\sqrt{p}$, p



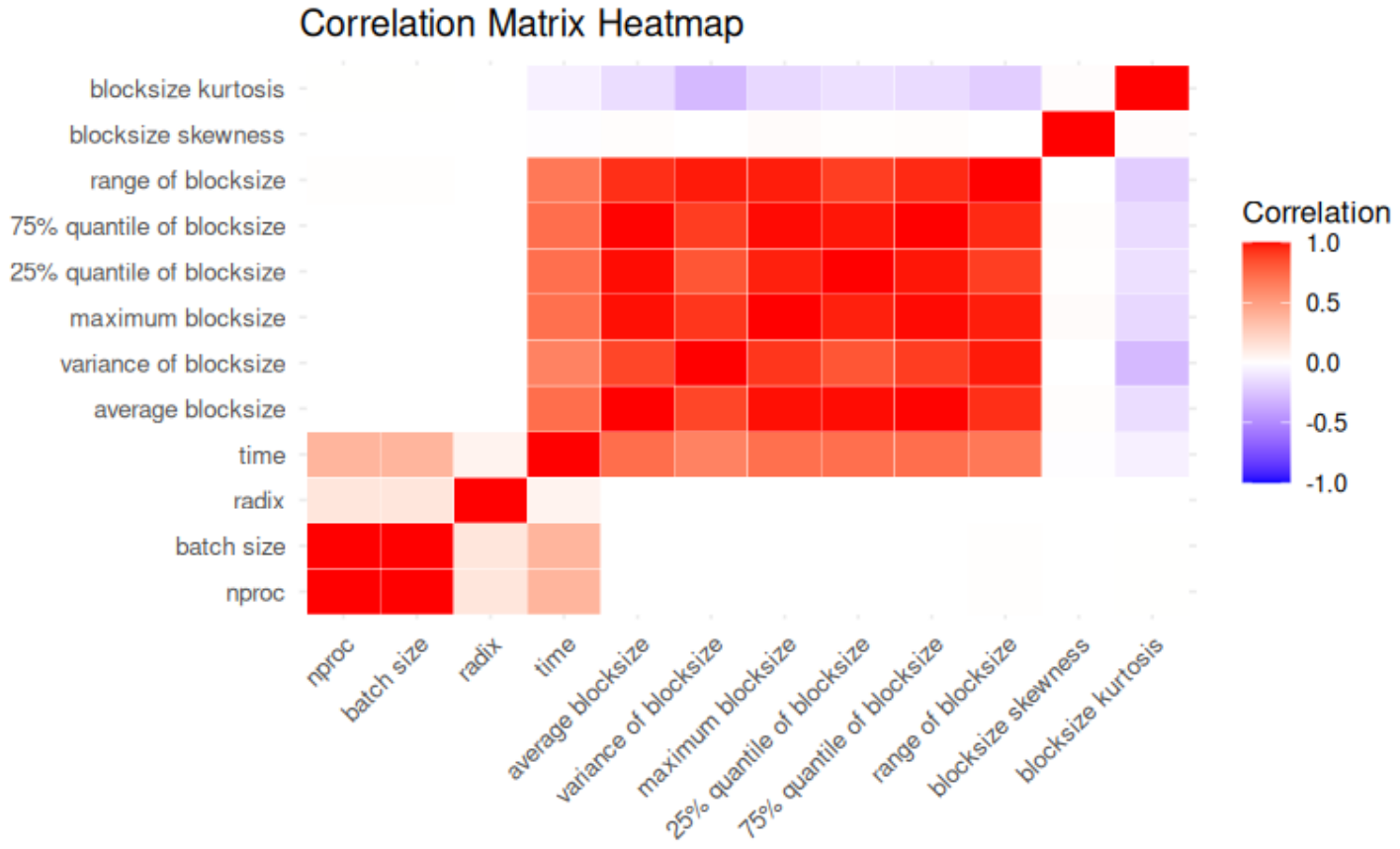
Methodology and Workflow



Sensitivity Analysis Result

In practice, more variables means more accurate model. But two much variable will introduce excessive overhead on-the-fly.

So we perform exploratory benchmarking and conduct sensitivity analysis remove unnecessary variables.



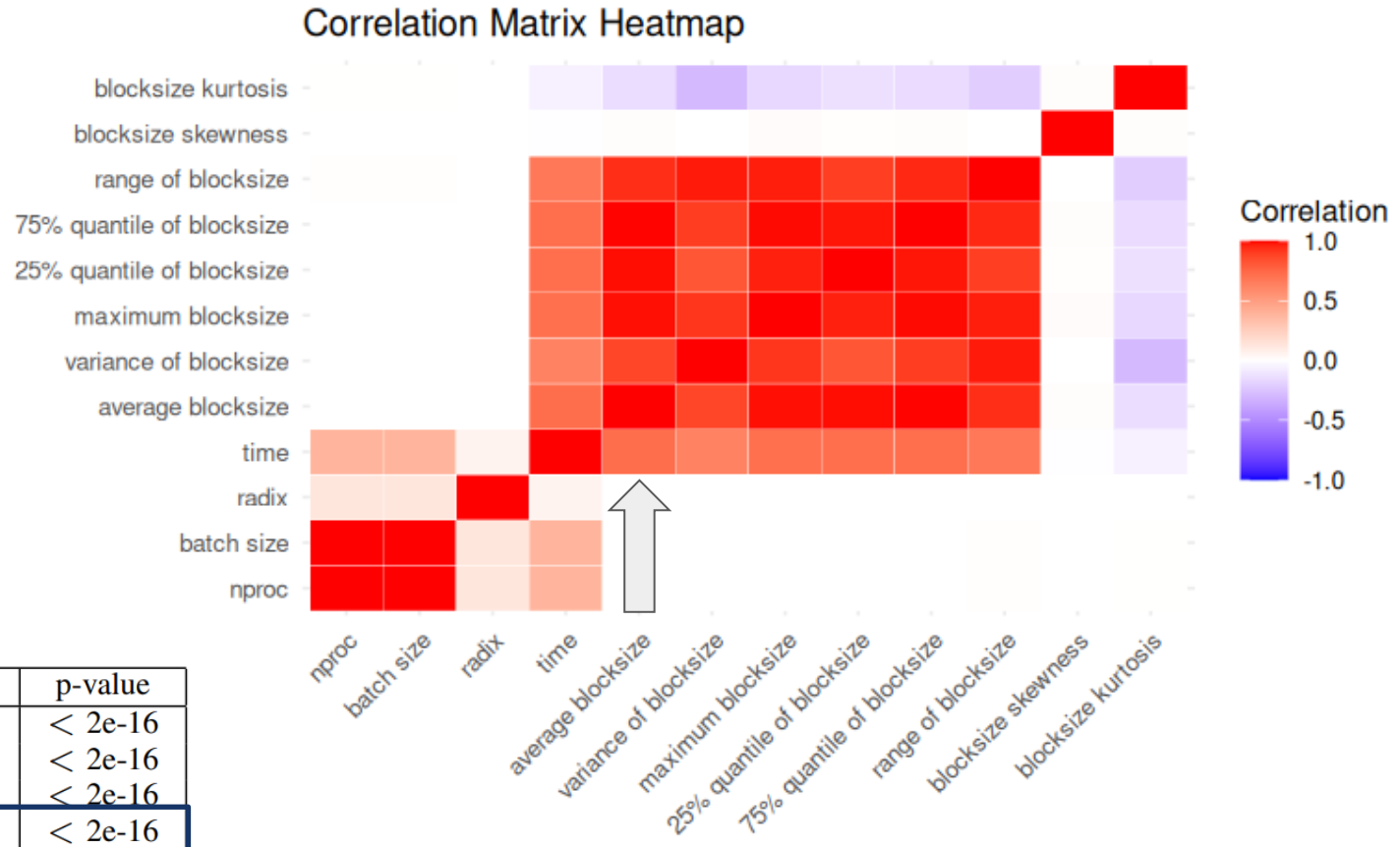
Following variables are equally important, either of them is enough.

- Average block size
- Maximum block size
- 25% and 75% quantile of blocksize

Sensitivity Analysis Result

In practice, more variables means more accurate model. But two much variable will introduce excessive overhead on-the-fly.

So we perform exploratory benchmarking and conduct sensitivity analysis remove unnecessary variables.

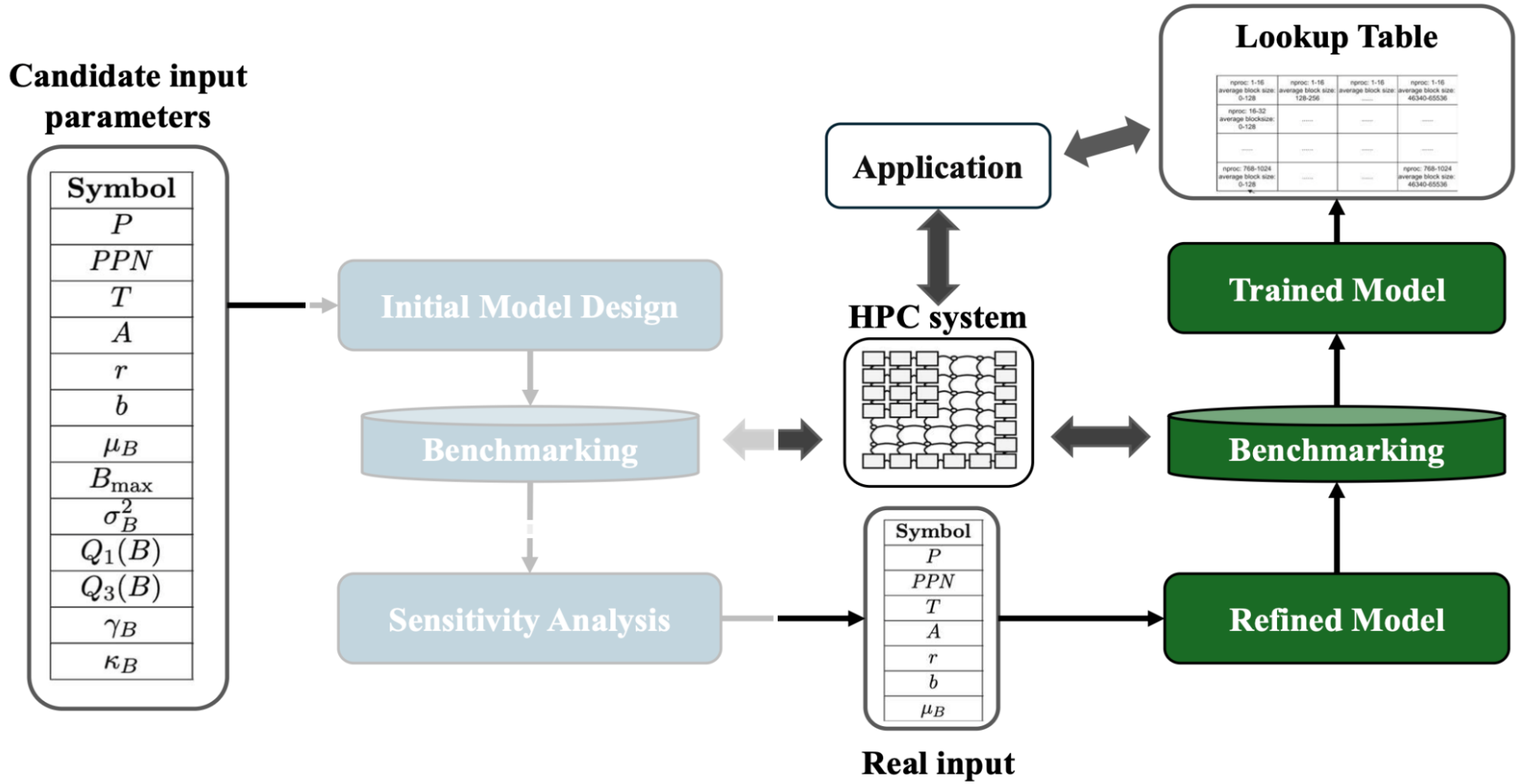


Parameters	Df	Sum of Squares	Mean Squares	F value	p-value
A	3	5404	2702	1.371e+04	$<$ 2e-16
P	1	51666	51666	2.622e+05	$<$ 2e-16
r	1	1284	1284	6517	$<$ 2e-16
μ_B	1	265153	265153	1.346e+06	$<$ 2e-16
σ_B^2	1	334	334	1697	$<$ 2e-16
$B_{\max}$	1	99	99	504.4	$<$ 2e-16
$Q_1(B)$	1	150	150	763.0	$<$ 2e-16
$Q_3(B)$	1	22	22	110.8	$<$ 2e-16
γ_B	1	9	9	45.60	1.46e-11
κ_B	1	10	10	52.86	3.63e-13

Following variables are equally important, either of them is enough.

- Average block size
- Maximum block size
- 25% and 75% quantile of blocksize

Methodology and Workflow



Initial Model and Reduced Model

- nproc

- algorithm type
- ppn
- topology
- radix
- batch size

- average block size
- blocksize variance
- maximum block size
- blocksize variance
- 25% and 75% quantile of blocksize
- skewness of blocksize

- nproc

- algorithm type
- ppn
- topology
- radix
- batch size

- average block size

- Output:
  - non-uniform Alltoall time

Comprehensive Benchmarking

Nproc: 32, 64, 96, 128, 192, 256, 384, 512, 1024, 2048 (10 levels in total)

Block Sizes are randomly generated from 1 to maximum block size,

- *Compare to exploratory benchmarking, only random uniform message are considered*
- *but finer levels for message size and nproc*

Max blocksize: 2, 4, 6, 8, ….., 3072, 4096

PPN: 32, 16, 8

Topology: Dragonfly on Polaris, 6D Mesh Toru on Fugaku

Algorithm to be benchmarked:

MPI All_to_Allv

ParAta

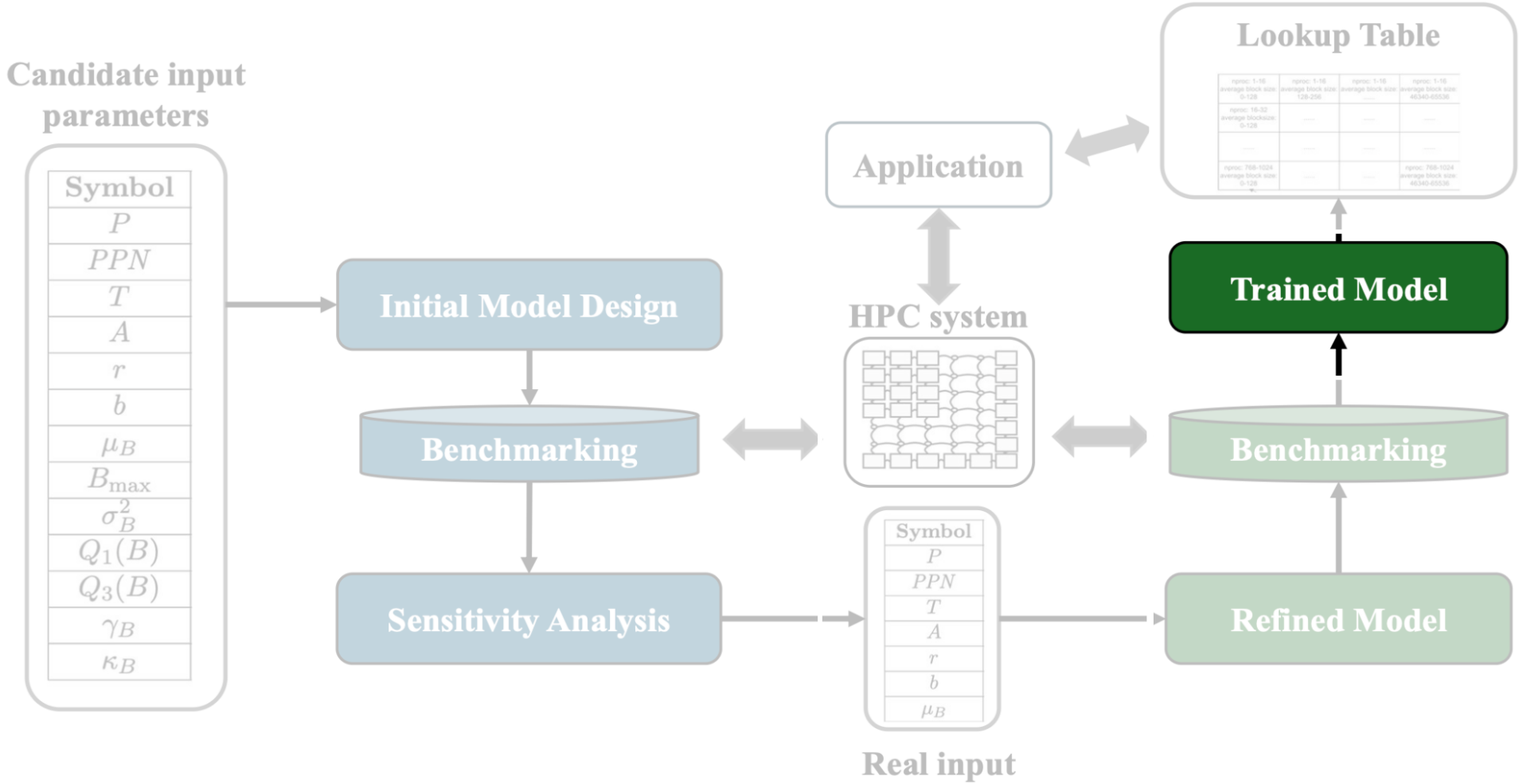
Radix = {2, 4, 8, 16, …., p} and fine-grid benchmarking around radix = 2, $\sqrt{p}$, and p

HieAta

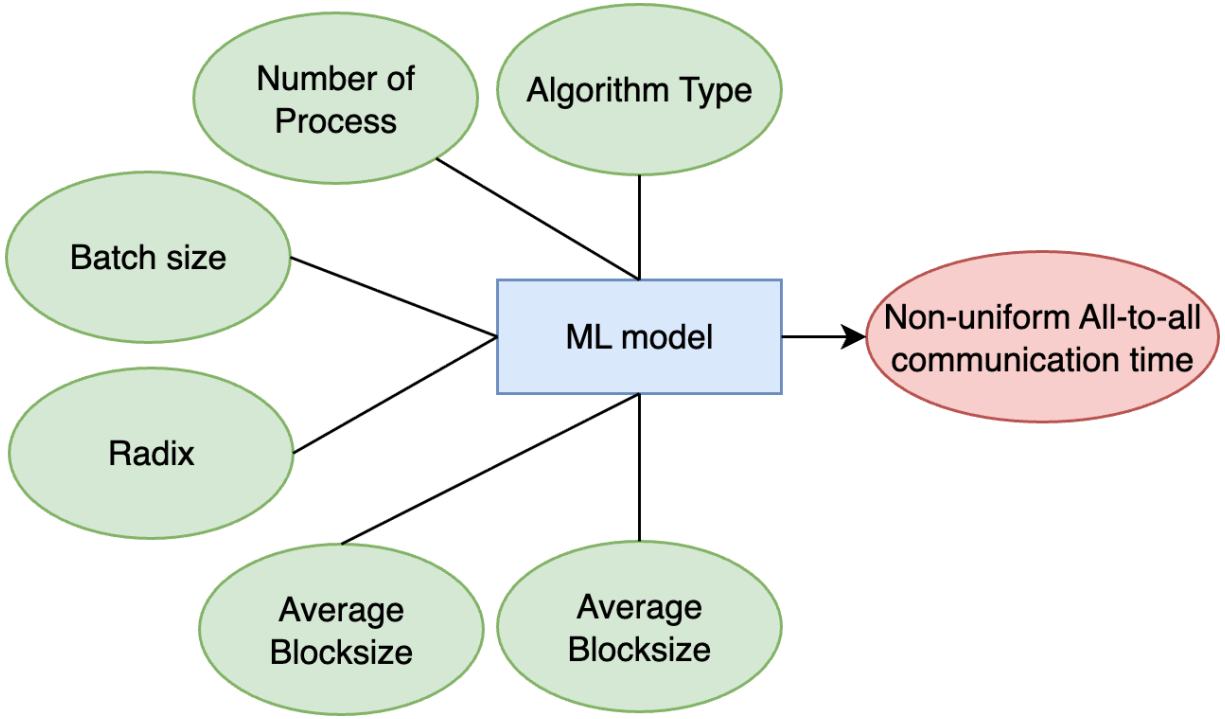
Batch size: 1, 2, 3, … num_node

Radix = {2, 4, 6, 8, …, ppn} and fine-grid benchmarking around radix = 2, $\sqrt{ppn}$, and ppn

Methodology and Workflow



Performance Prediction Model



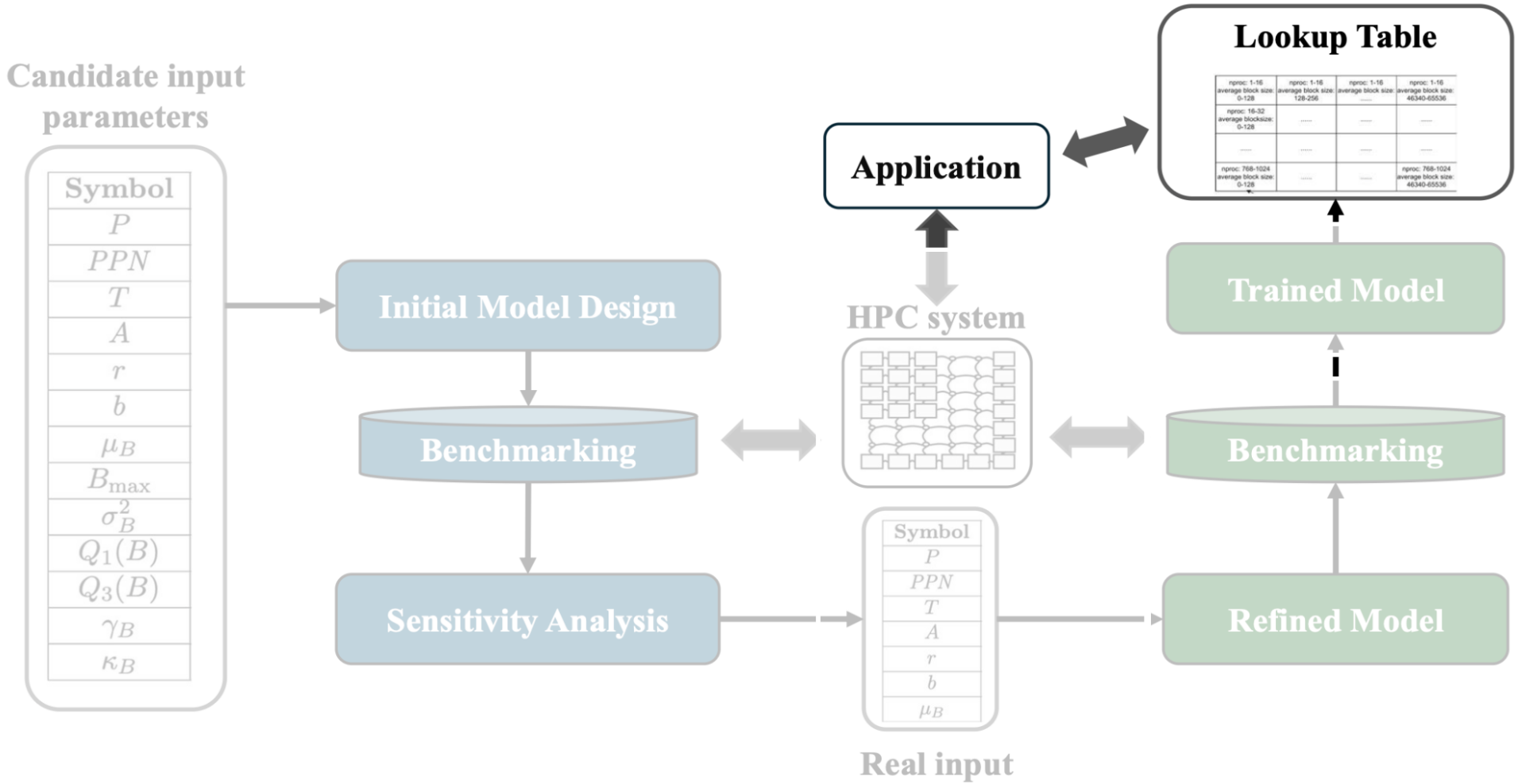
ML model attempted:

- Linear Regression
- Random Forest
- Regression Tree
- Lasso/Ridge Regression
- Neural Network (MLP)

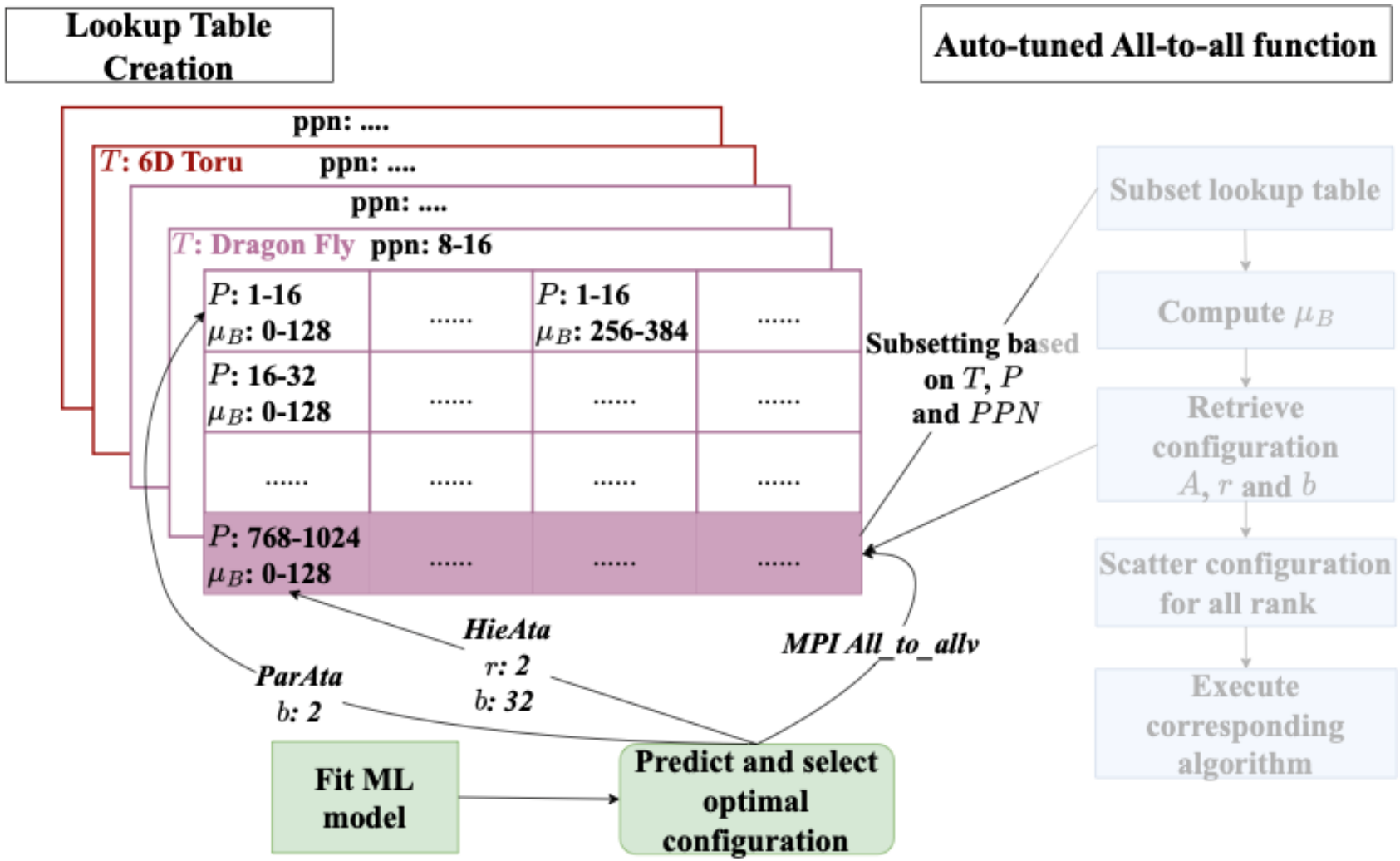
Dataset partition:

- 70% data for training
- 10% data for validation (hyperparameter tuning)
- 20% data for testing

Methodology and Workflow

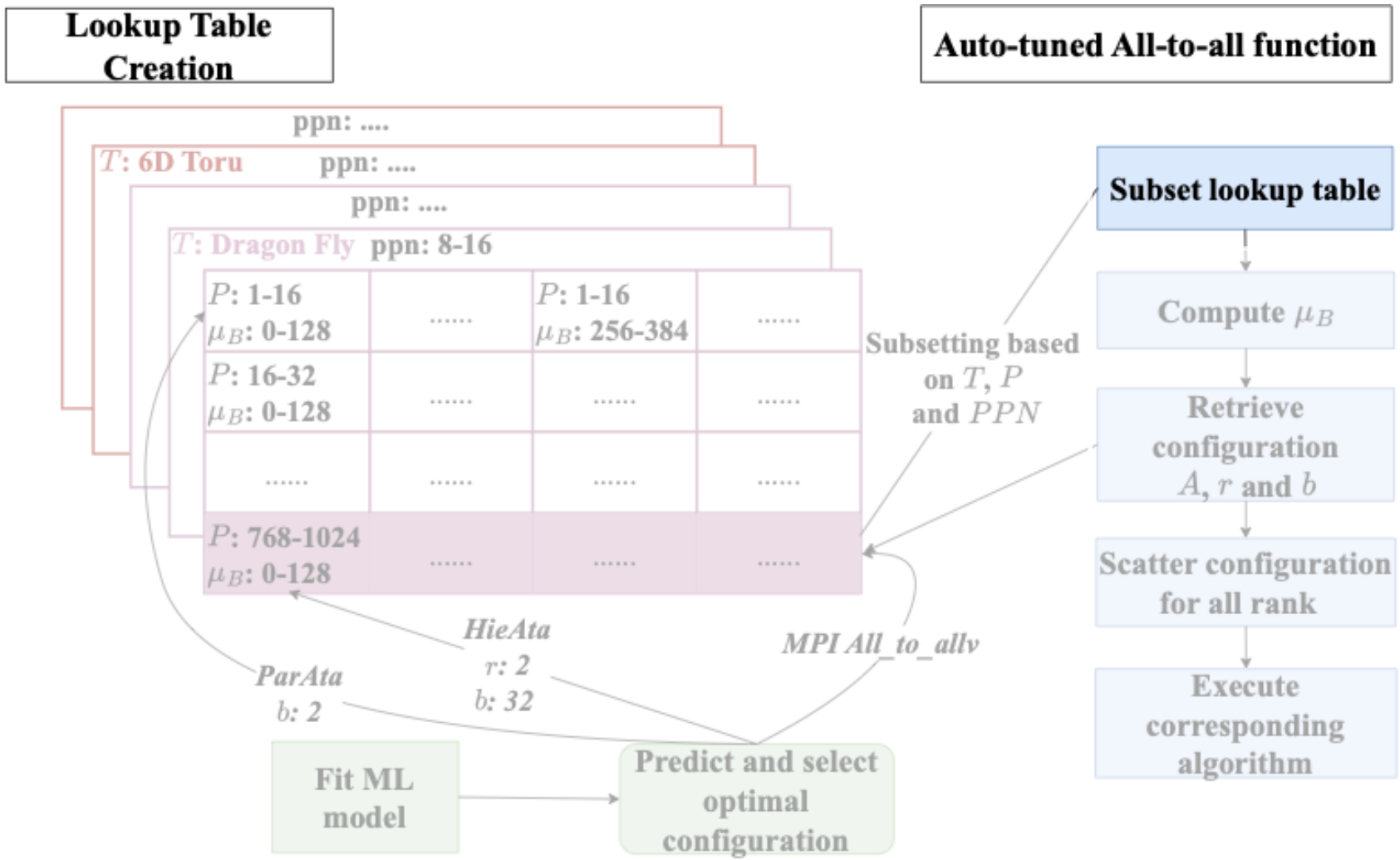


Lookup Table and Autotune Alltoall function



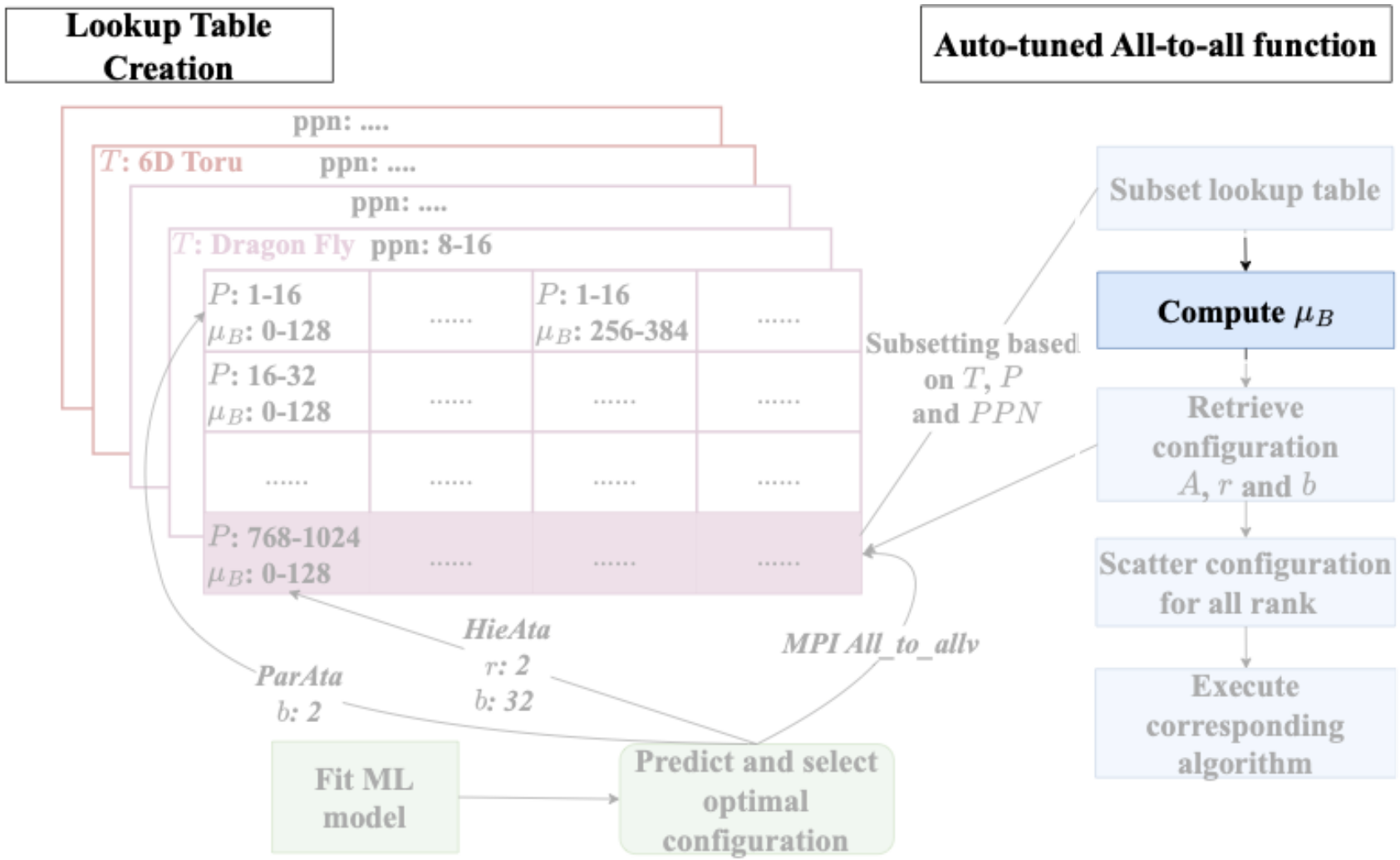
Use fitted machine learning model to search for the optimal algorithm configuration for each combination of scenario and export it into json table

Lookup Table and Autotune Alltoall function



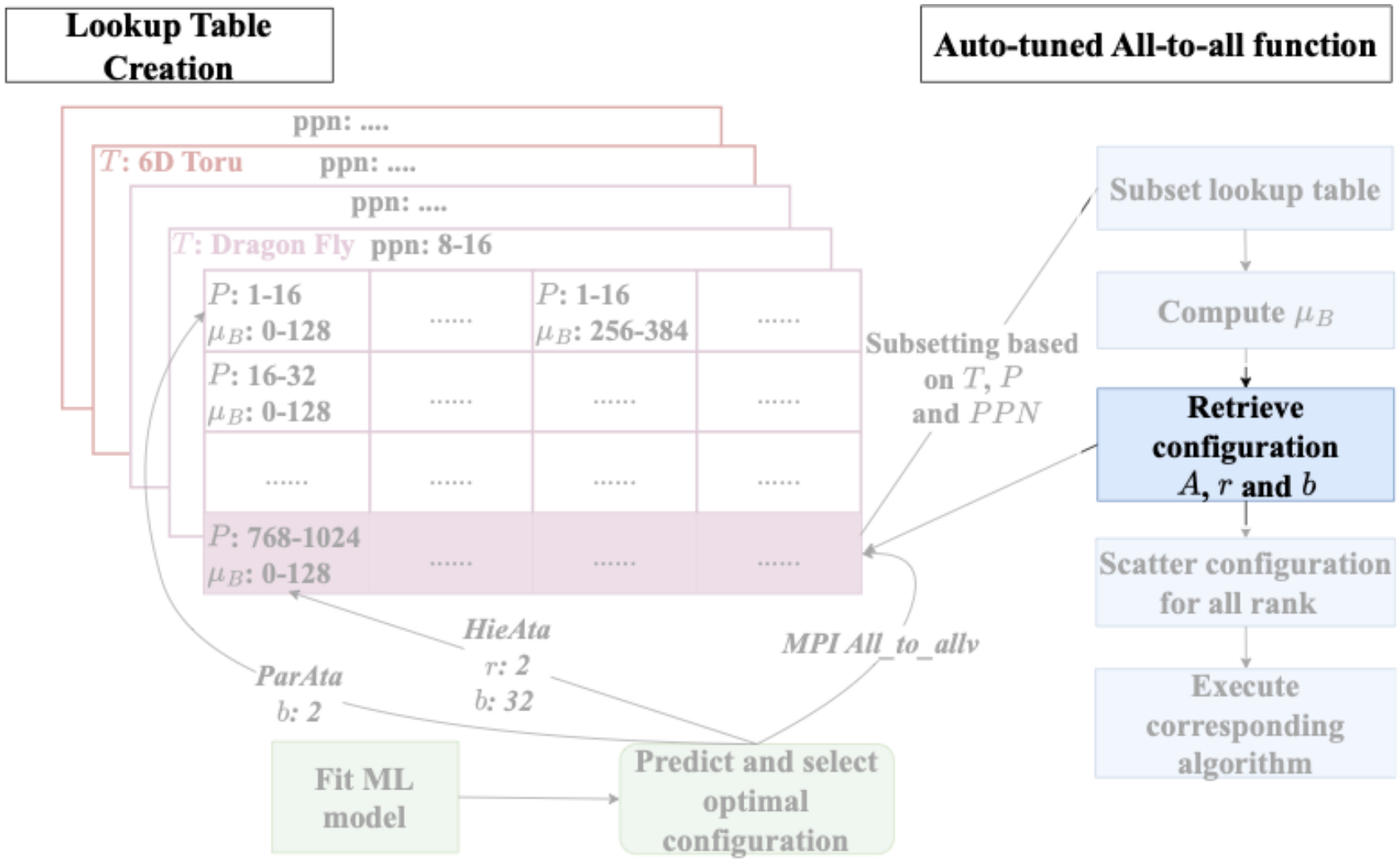
Pick the part of the table that matches the number of processes, processes per node, and network topology type

Lookup Table and Autotune Alltoall function



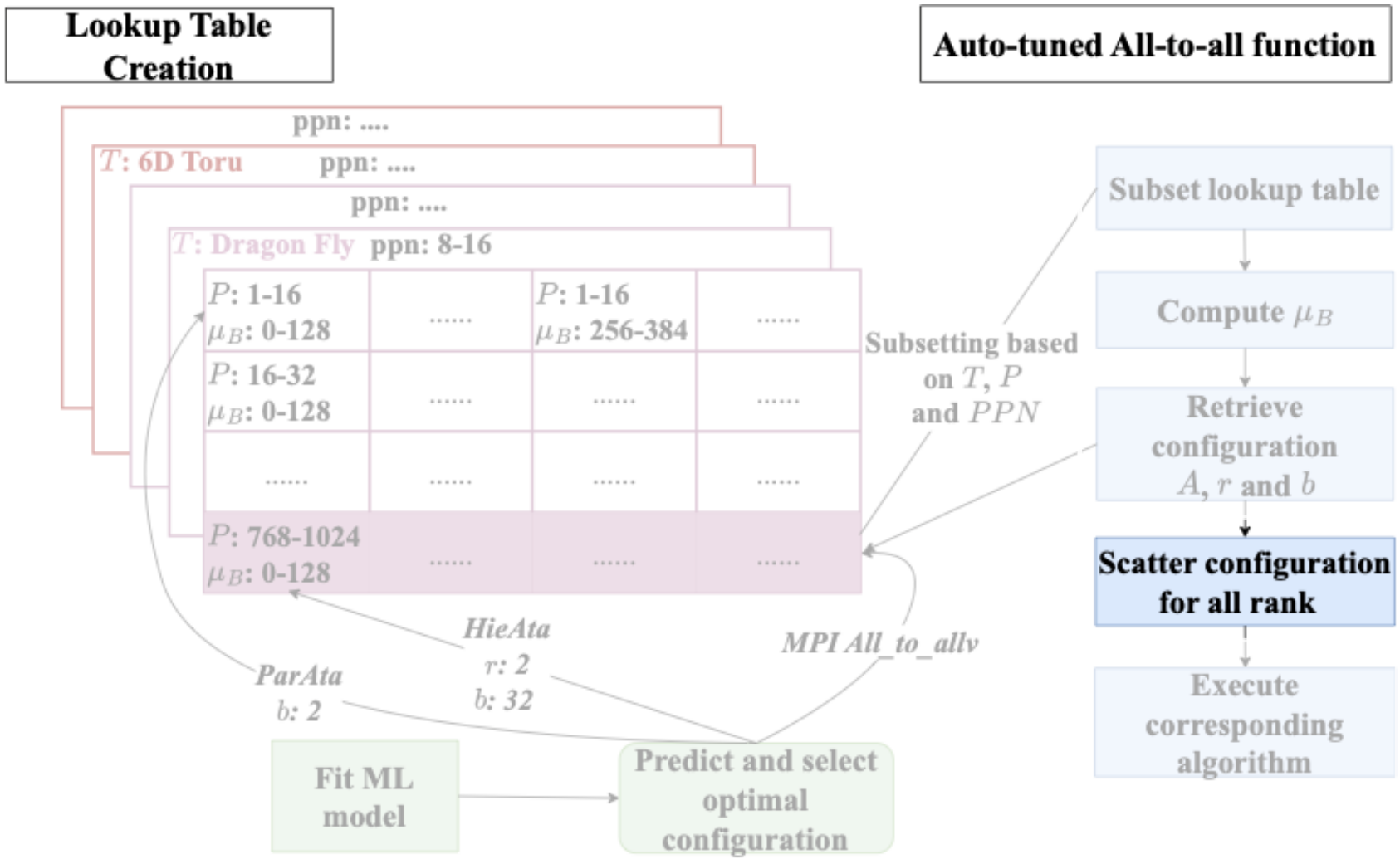
Gather block size information from all processes and calculate the average size

Lookup Table and Autotune Alltoall function



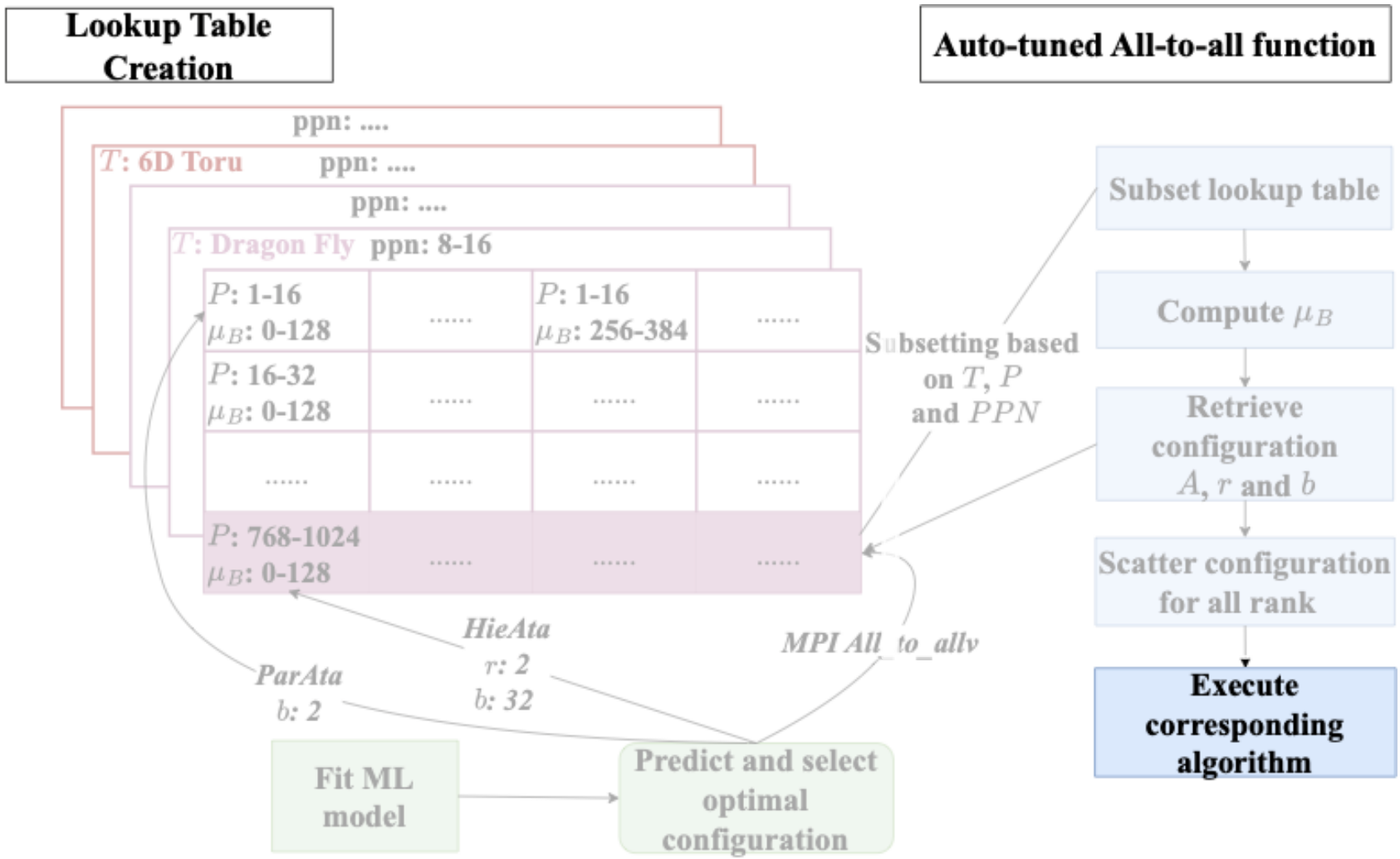
Find the best algorithm and its settings (algorithm type, radix value, and batch size) from the table

Lookup Table and Autotune Alltoall function



Share the chosen algorithm and settings with all processes

Lookup Table and Autotune Alltoall function



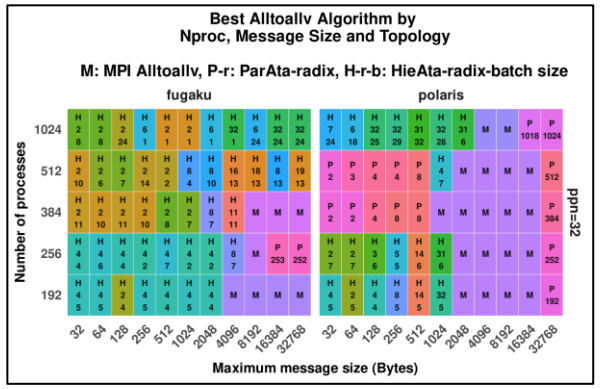
Run the selected Alltoallv algorithm using the given settings

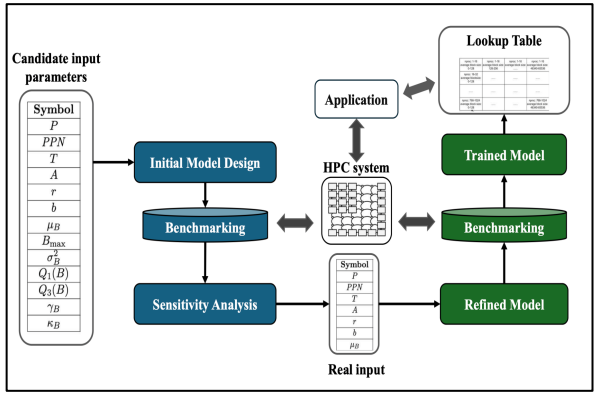
Outline

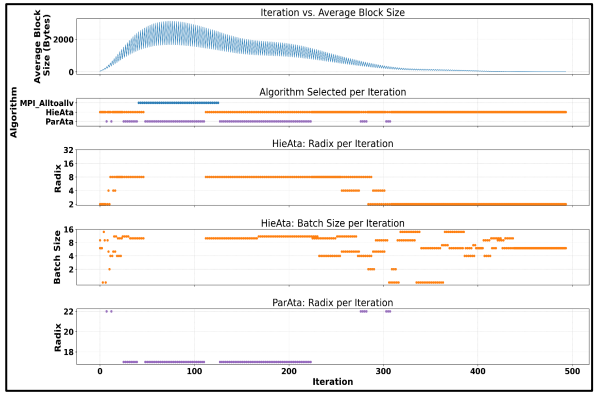
Motivation and challenges

Proposed approach

Results







Evaluation and Results

- Accuracy of performance prediction model
- Performance of autotuned MPI function
- Analysis of Autotuned MPI function behavior

Performance Prediction Model

For each trained model, we evaluate its test accuracy in test dataset.

We choose Decision Tree as final model for its balance between accuracy and speed.

Model	Absolute Percentage of Test Error
Linear Regression	48.52%
Lasso Regression	48.38%
Ridge Regression	48.48%
Decision Tree	2.22%
Random Forest	1.91%
Multiple Perceptron Layer	6.11%

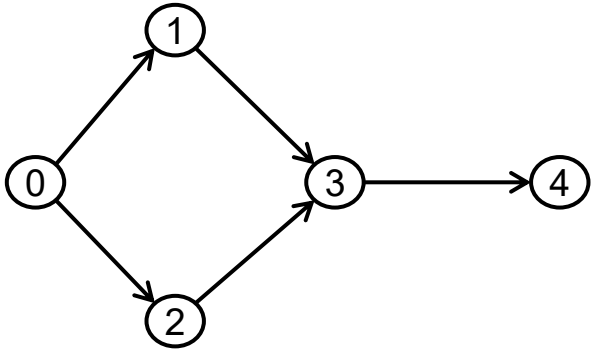
Application: Path-finding on large-scale graphs

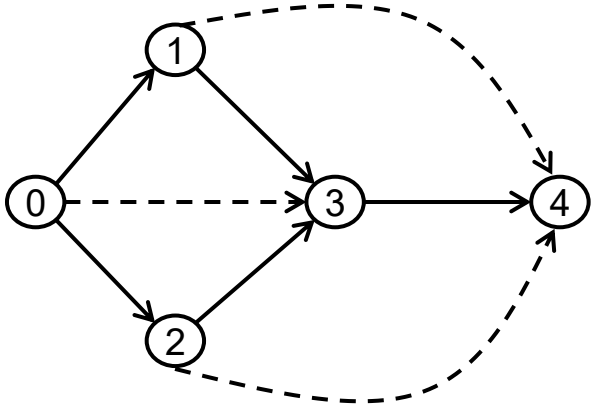
0	1
0	2
1	3
2	3
3	4

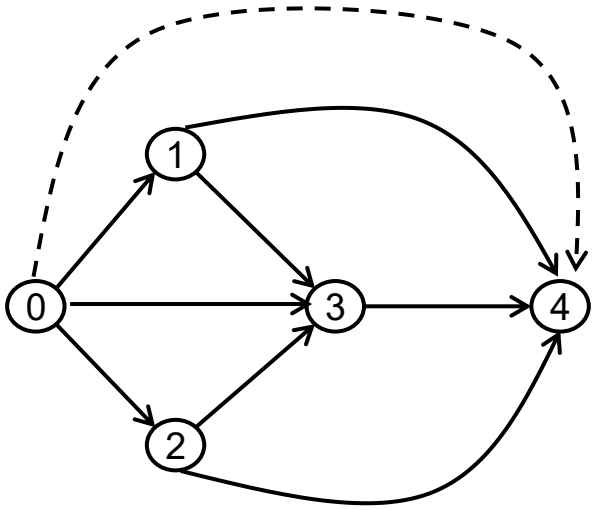
0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4

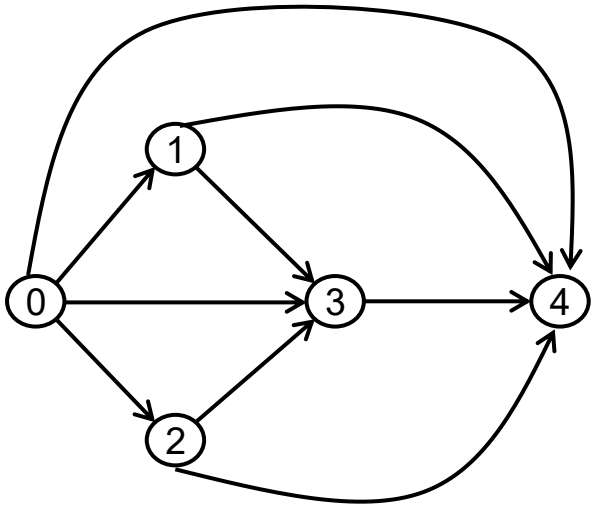
0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4
0	4

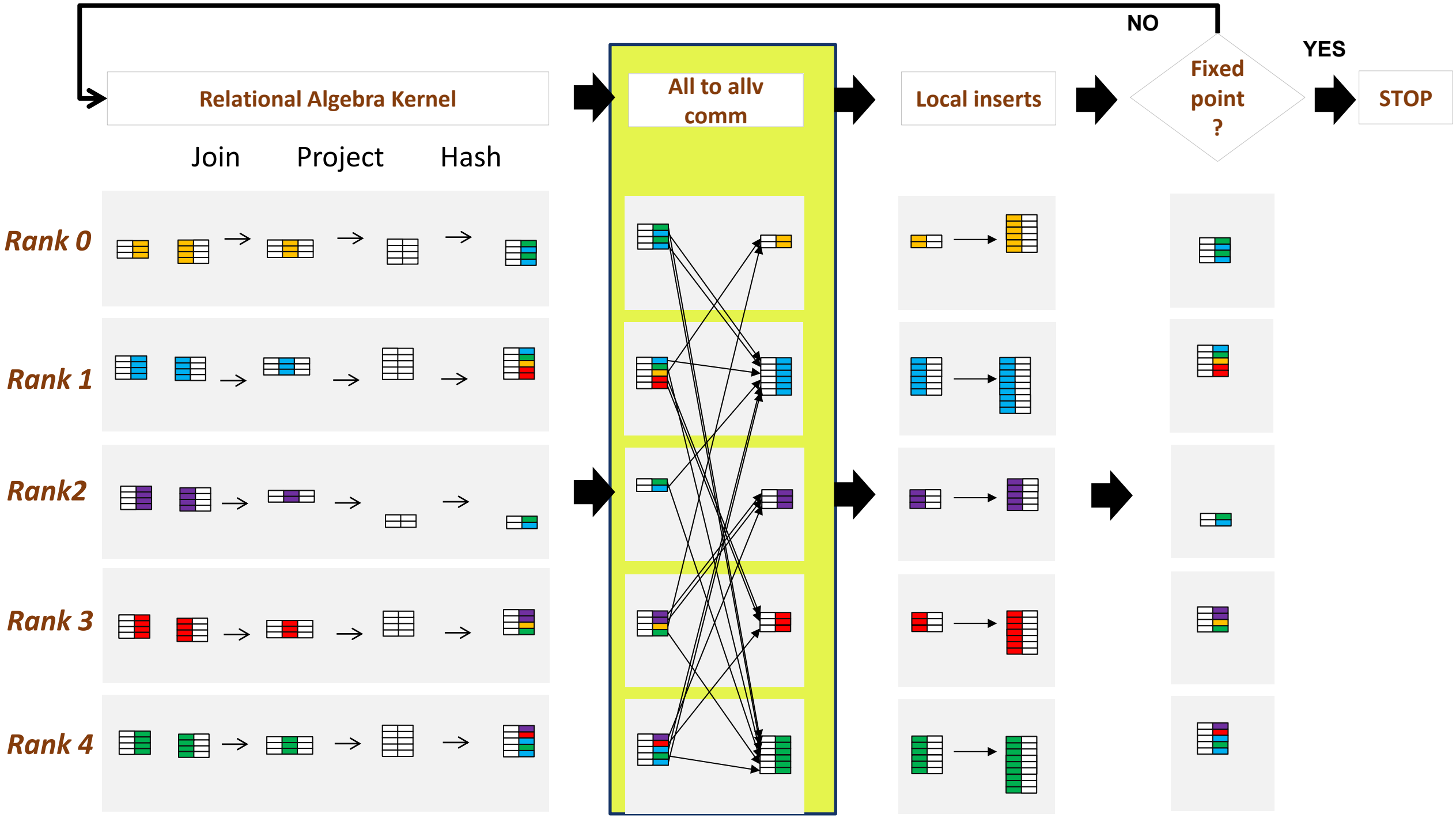
0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4
0	4







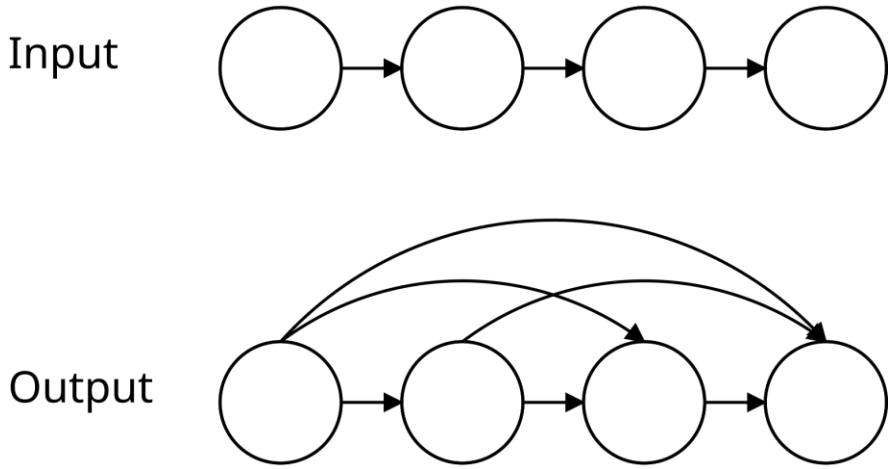



NO
YES
Relational Algebra Kernel
All to allv comm
Local inserts
Fixed point ?
STOP
Join
Project
Hash
Rank 0
Rank 1
Rank2
Rank 3
Rank 4

Speedup of Autotuned All-to-all function

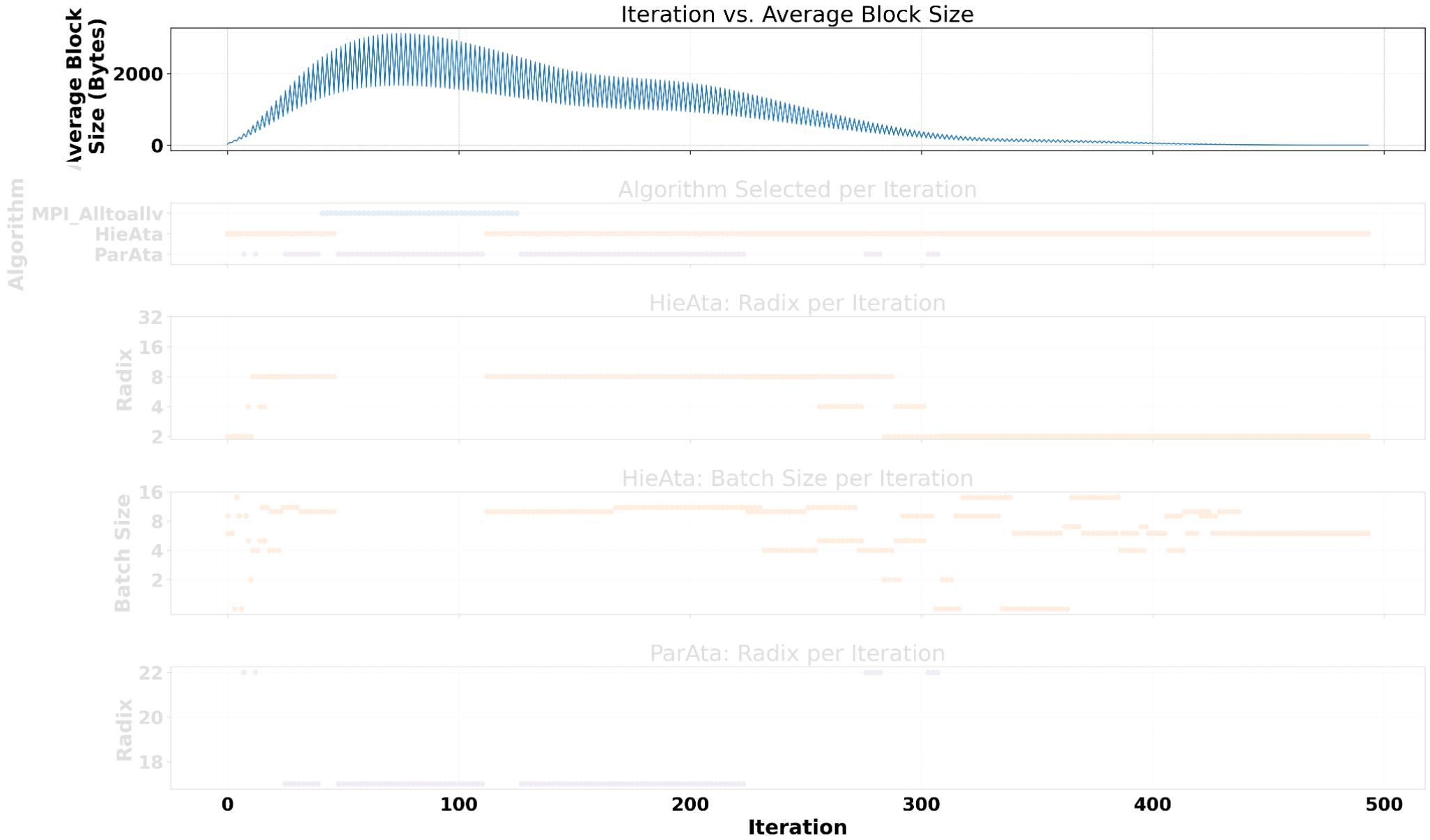
Evaluation is MPI parallel path-finding.

Each iteration, new edges will be computed, iterations will be repeated until no edge to add

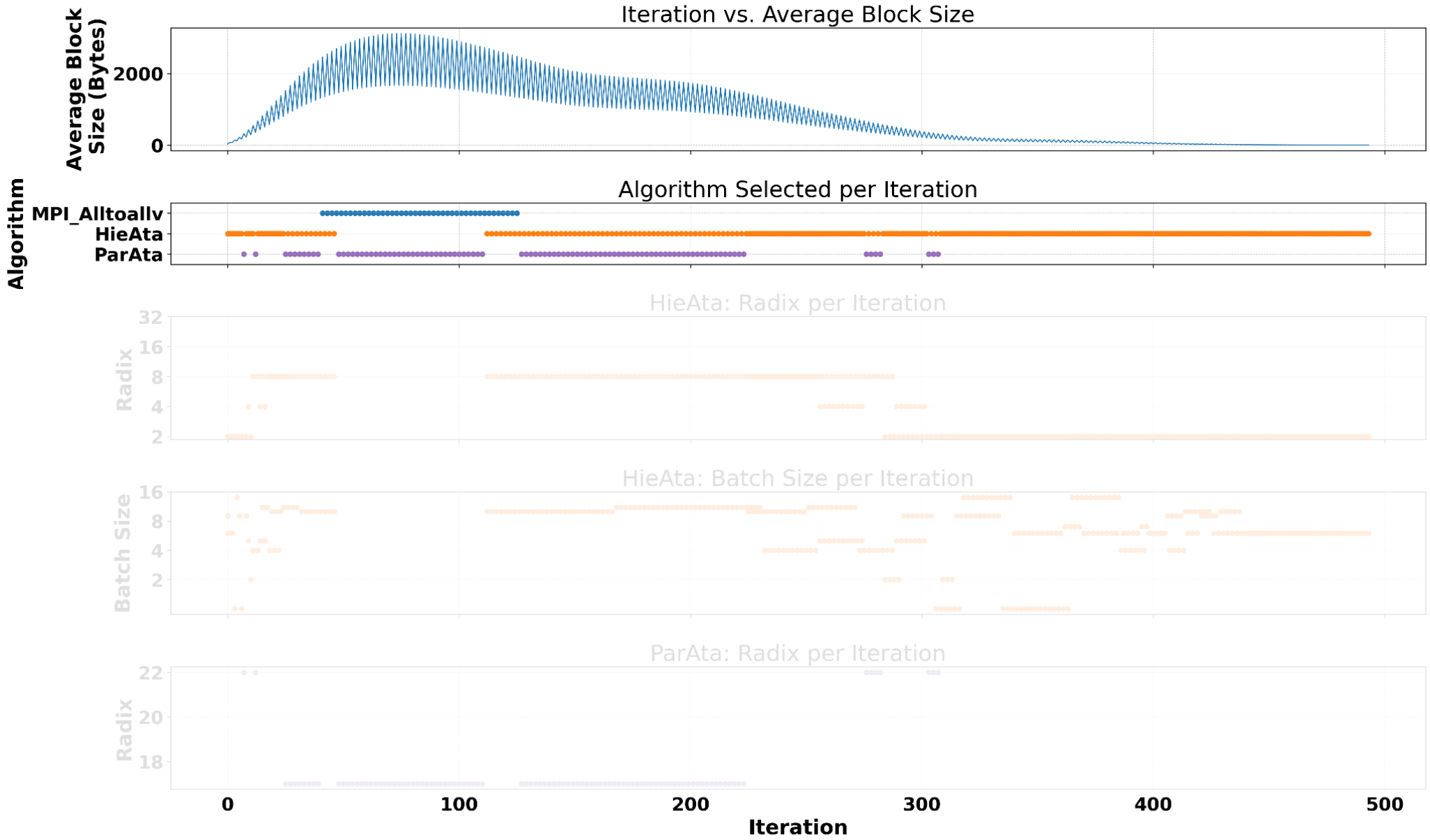


P	Implementation	Comm. Time (s)
256	Official `MPI_Alltoallv`	11.68
256	Auto-tuned (*Ours*)	3.93
256	Speedup (Off/Ours)	2.97×
512	Official `MPI_Alltoallv`	14.02
512	Auto-tuned (*Ours*)	3.60
512	Speedup (Off/Ours)	3.89×
1024	Official `MPI_Alltoallv`	11.52
1024	Auto-tuned (*Ours*)	1.91
1024	Speedup (Off/Ours)	6.03×
200	Official `MPI_Alltoallv`	8.69
200	Auto-tuned (*Ours*)	3.57
200	Speedup (Off/Ours)	2.43×
400	Official `MPI_Alltoallv`	13.14
400	Auto-tuned (*Ours*)	3.39
400	Speedup (Off/Ours)	3.88×
600	Official `MPI_Alltoallv`	11.20
600	Auto-tuned (*Ours*)	1.89
600	Speedup (Off/Ours)	5.93×

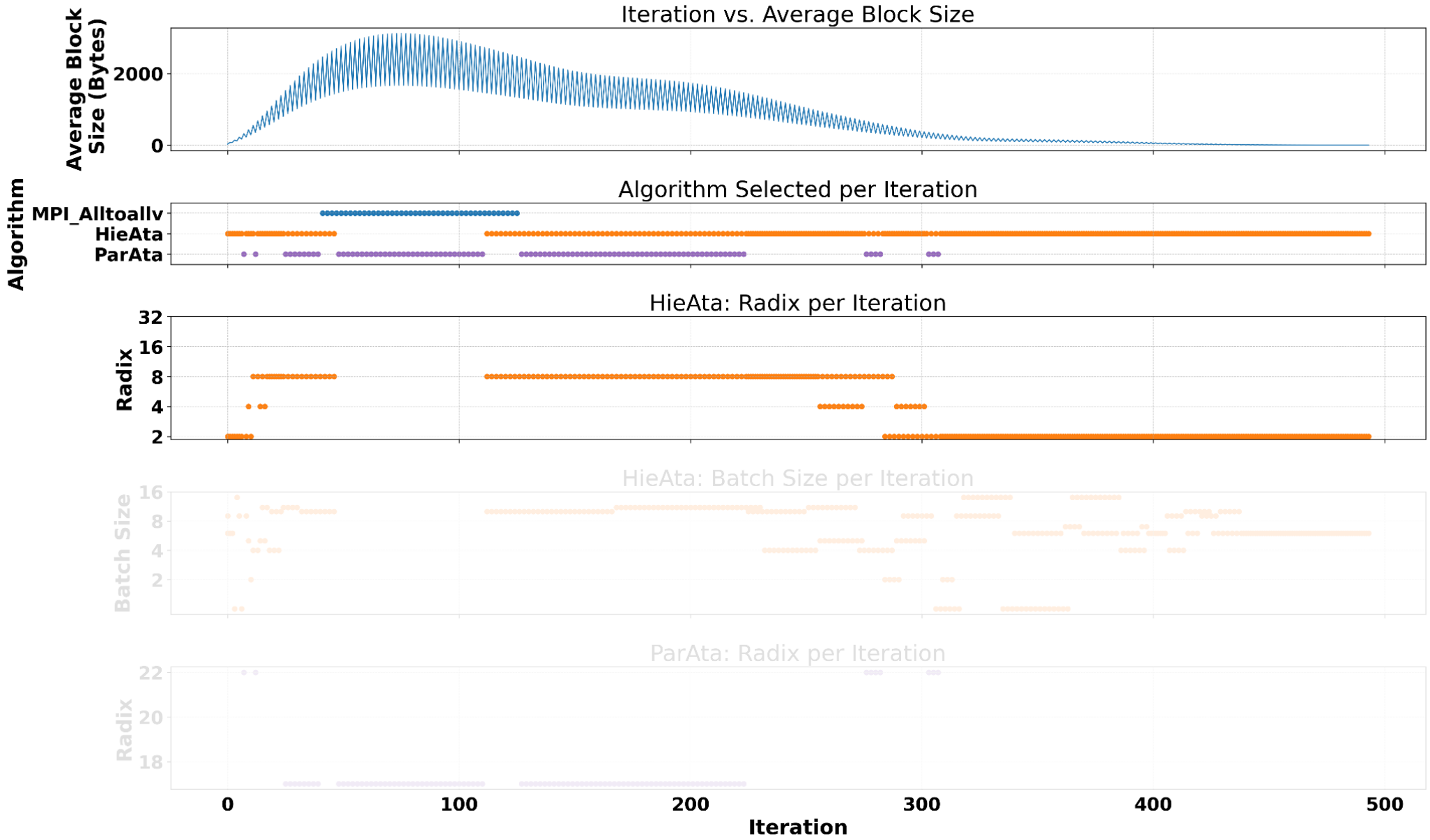
Analysis of Autotune MPI function



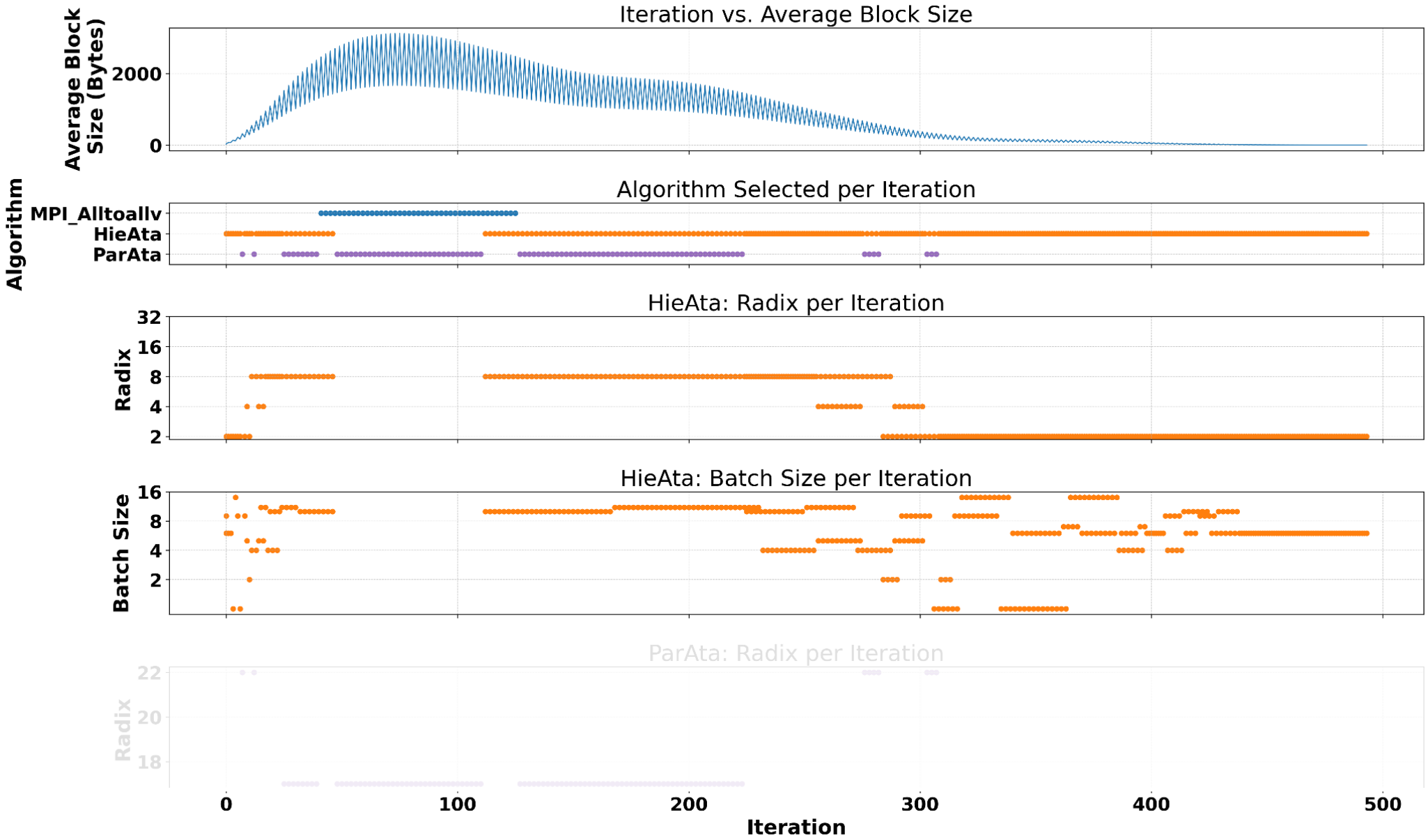
Analysis of Autotune MPI function



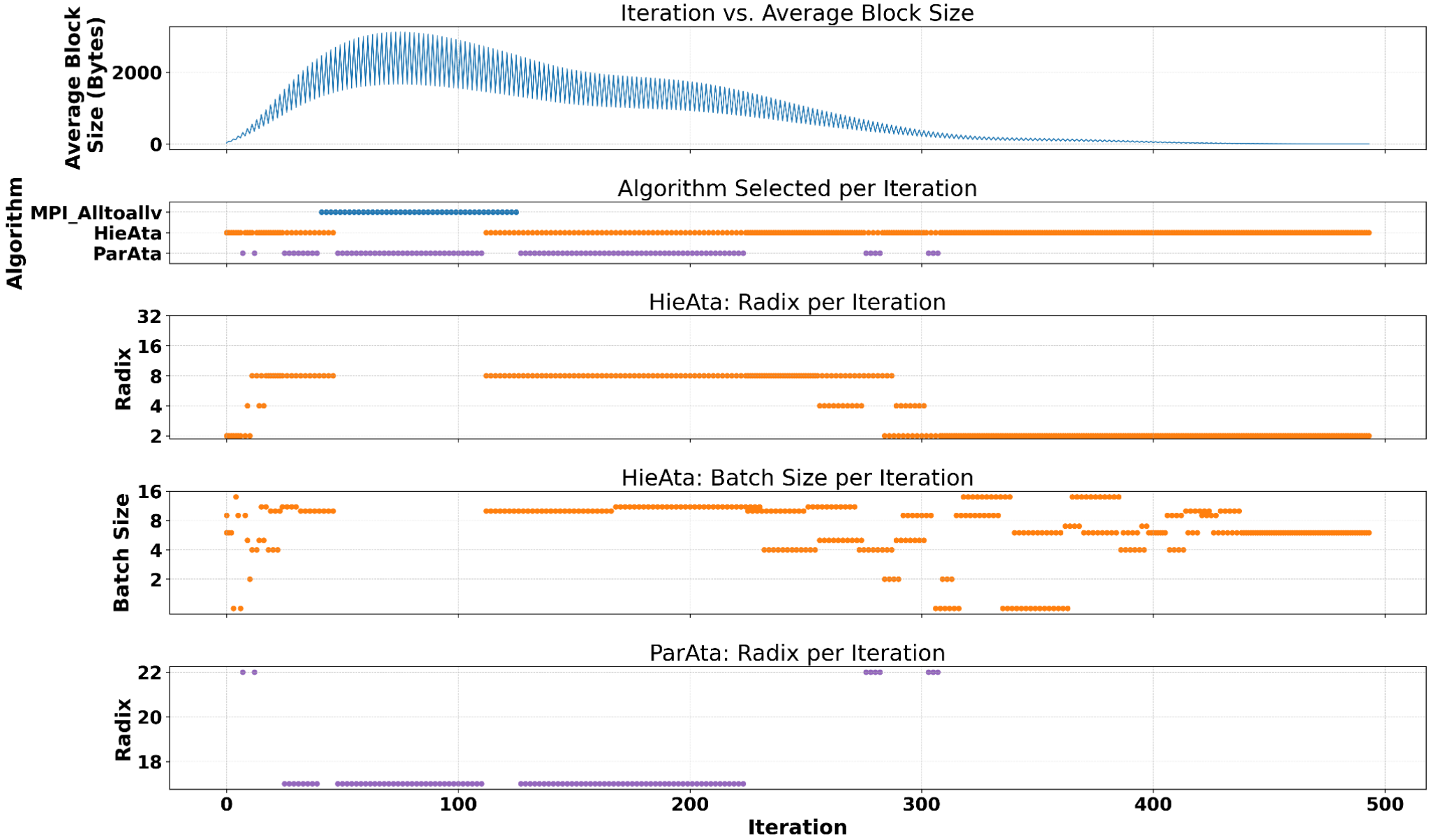
Analysis of Autotune MPI function



Analysis of Autotune MPI function



Analysis of Autotune MPI function



Conclusion

- Presented ML-driven auto tuner for non-uniform MPI_Alltoallv
- Combines sensitivity analysis + ML models + lookup table
- Achieve 2.4 - 6x faster communication
- Runtime overhead is minimal

Thank you so much!