



Parameterized Algorithms for Non-uniform All-to-all

Presenter: Jens Domke

Authors: Ke Fan, Jens Domke, Seydou Ba, and Sidharth Kumar



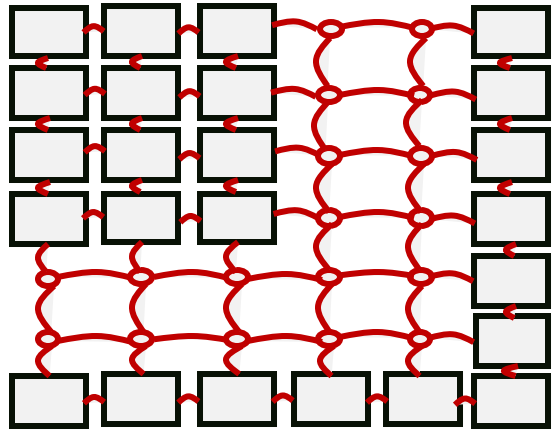
Agenda

- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- Challenges and Solutions
 - Tunability of Performance
 - Logarithmic Non-uniform All-to-all
- Parameterized Logarithmic Algorithm
- Parameterized Linear Algorithm
- Evaluation
- Application
- Conclusion

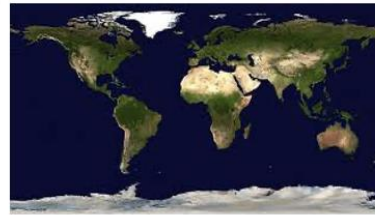
Agenda

- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- Challenges and Solutions
 - Tunability of Performance
 - Logarithmic Non-uniform All-to-all
- Parameterized Logarithmic Algorithm
- Parameterized Linear Algorithm
- Evaluation
- Application
- Conclusion

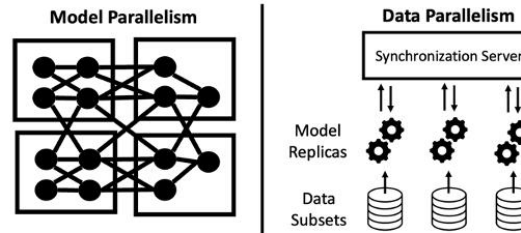
Exascale Computing Necessaries Effective Parallelism



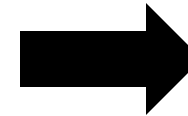
Exascale Supercomputers



Earth and Climate Modeling



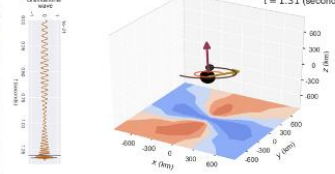
Parallel Machine Learning



**Inter-process Data
Movement**



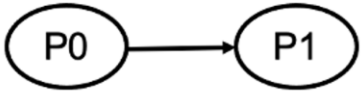
Colliding and Wobbling Black Holes



Black Holes Simulation

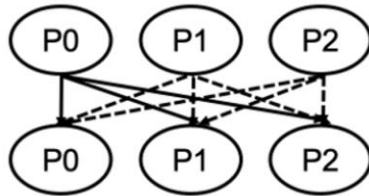
Three Fundamental Interfaces of Data Movement

Point-to-point

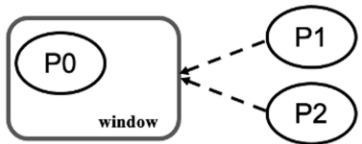


MPI_Send
MPI_Recv

Collective



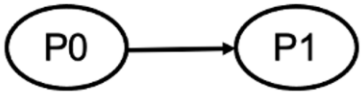
One-sided



MPI_Put
MPI_Get

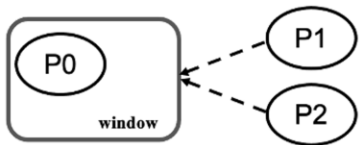
Collective Involves Communication Among All Processes

Point-to-point



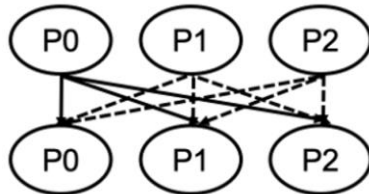
MPI_Send
MPI_Recv

One-sided

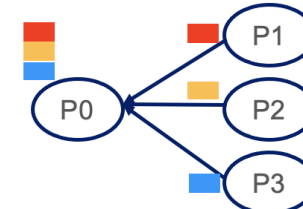


MPI_Put
MPI_Get

Collective

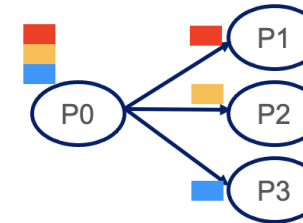


All-to-One



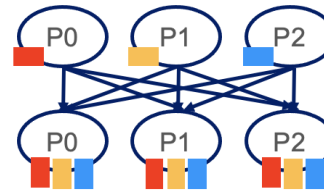
e.g., MPI_Reduce

One-to-All



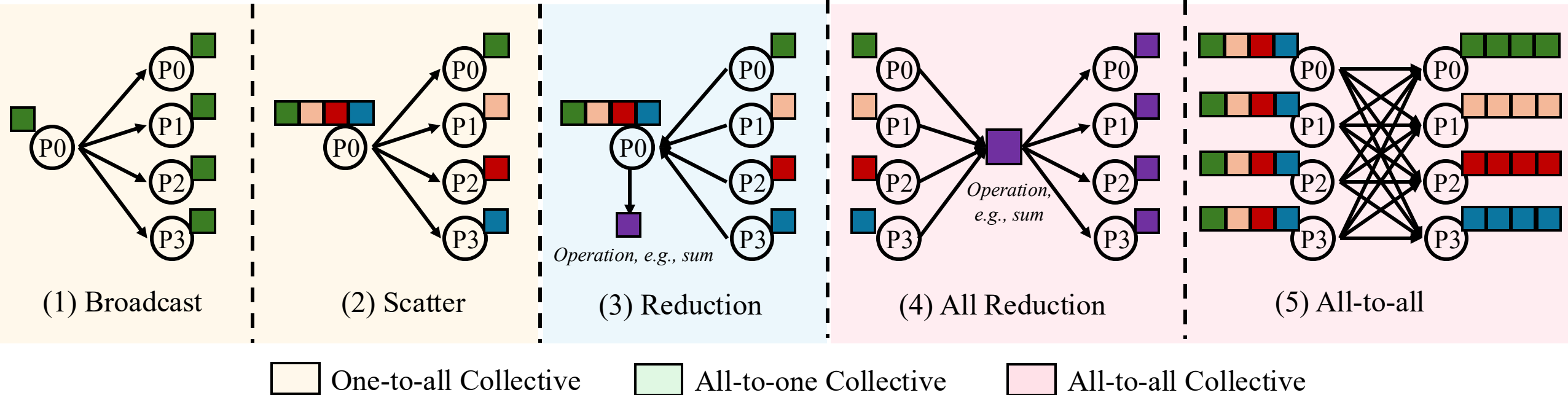
e.g., MPI_Scatter

All-to-All

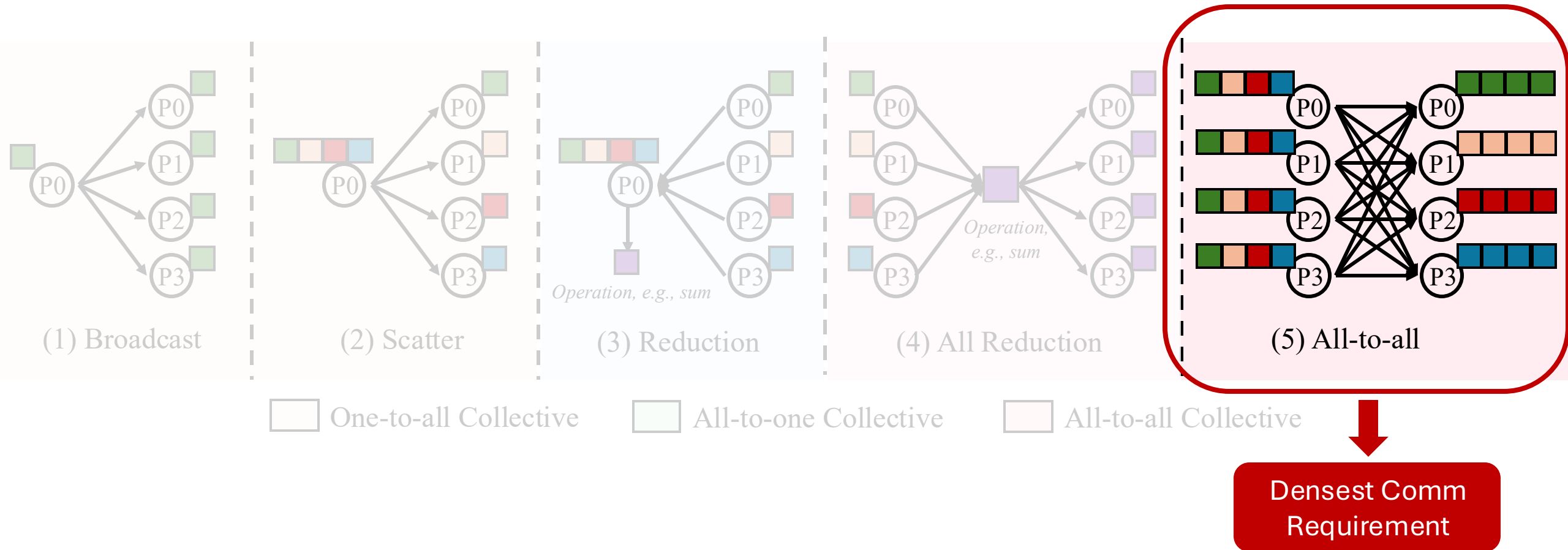


e.g., MPI_Alltoall

Some Widely Used Collective Primitives



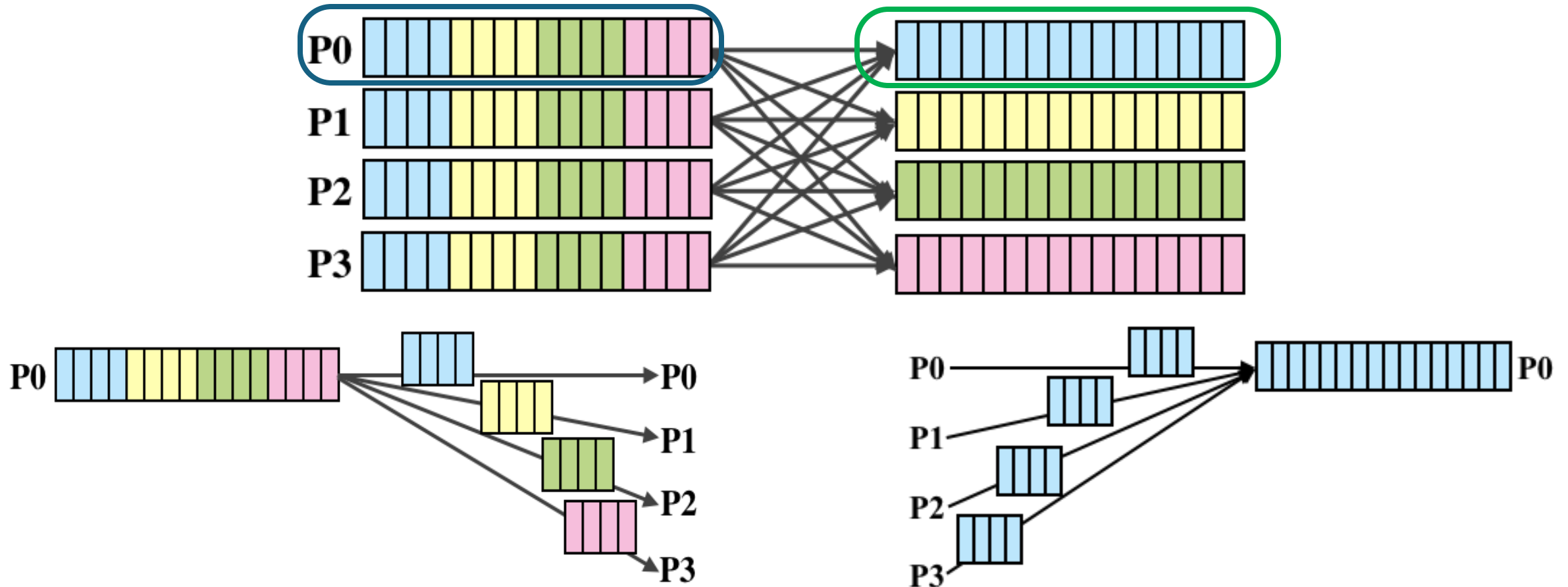
All-to-all Data Exchange is the Most Challenging to Scale and Optimize



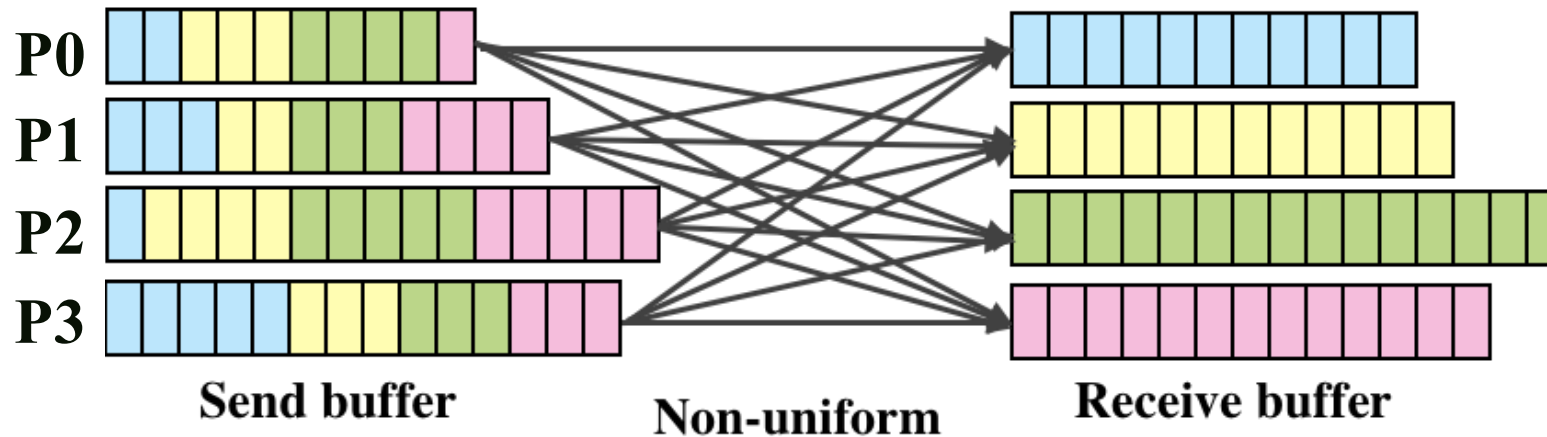
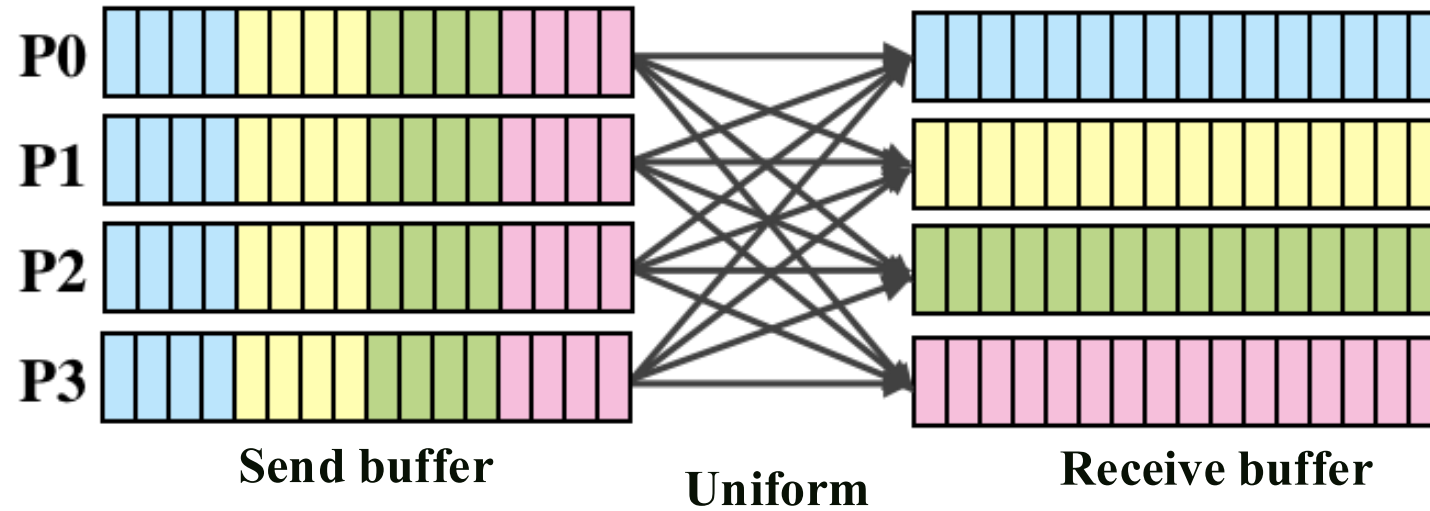
Agenda

- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- Challenges and Solutions
 - Tunability of Performance
 - Logarithmic Non-uniform All-to-all
- Parameterized Logarithmic Algorithm
- Parameterized Linear Algorithm
- Evaluation
- Application
- Conclusion

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



Uniform and Non-uniform All-to-all



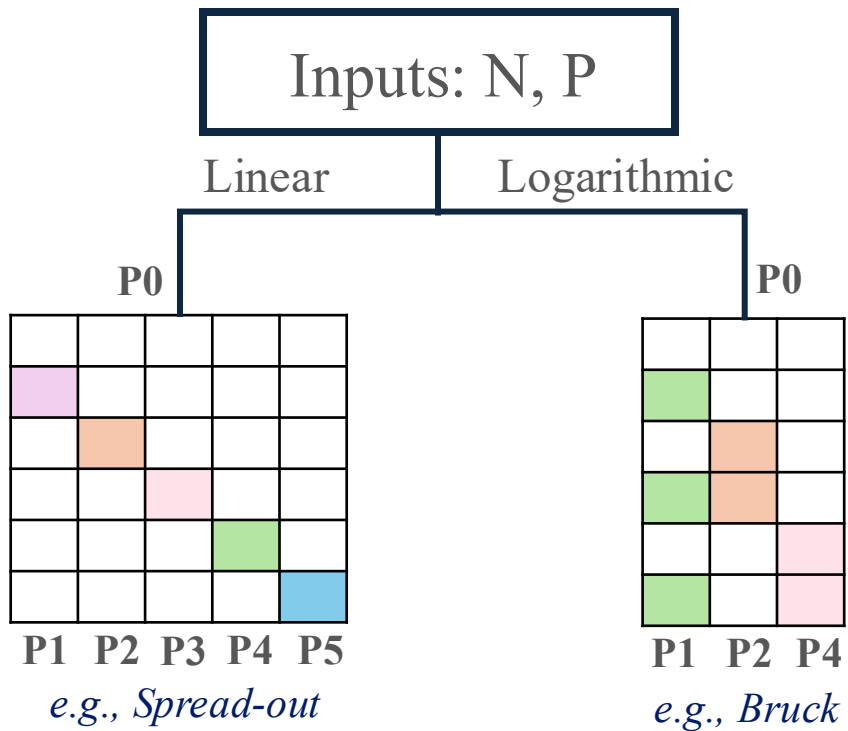
Agenda

- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- Challenges and Solutions
 - Tunability of Performance
 - Logarithmic Non-uniform All-to-all
- Parameterized Logarithmic Algorithm
- Parameterized Linear Algorithm
- Evaluation
- Application
- Conclusion

Standard MPI All-to-all Implementations

N: Size of data-block

P: Process count



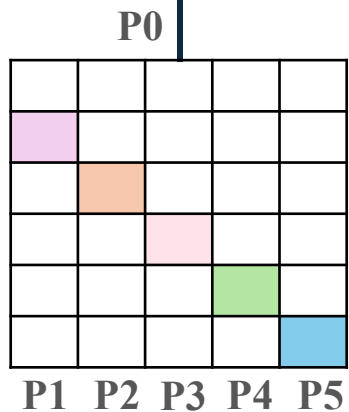
Standard MPI All-to-all Implementations: Spread-out

N: Size of data-block

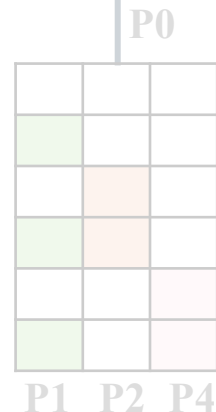
P: Process count

Inputs: N, P

Linear



e.g., Spread-out

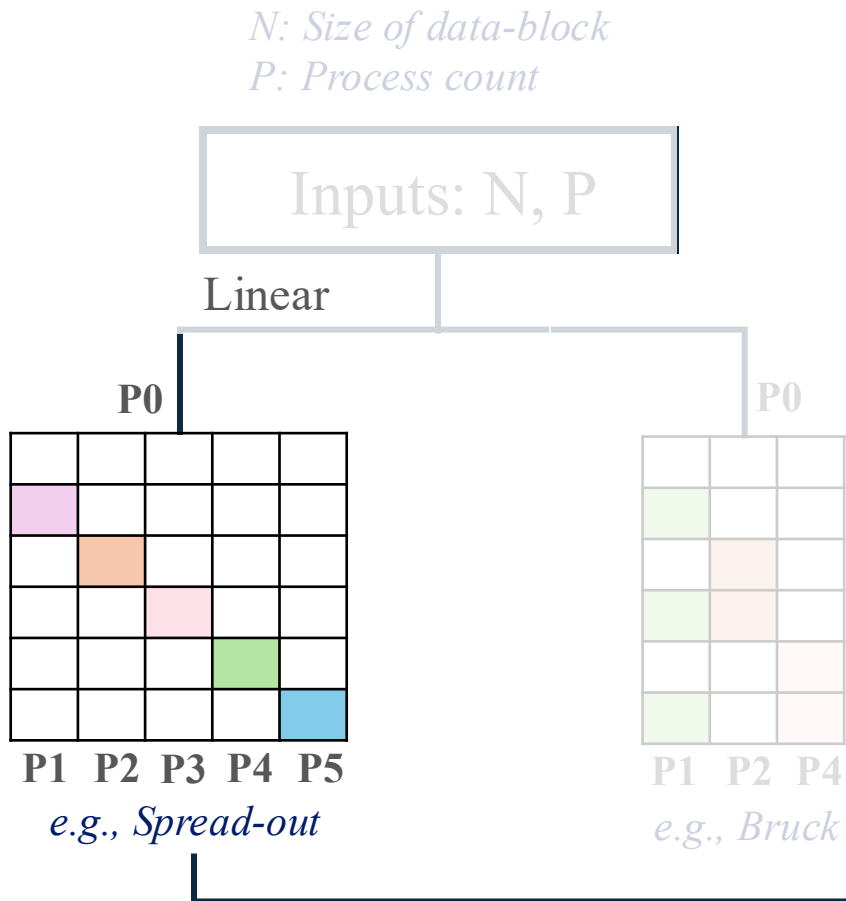


e.g., Bruck

Spread-out algorithm

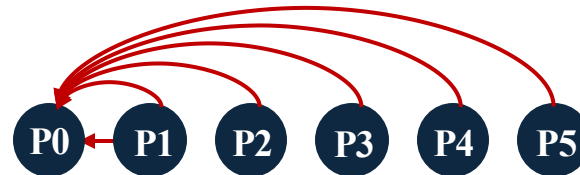


Standard MPI All-to-all Implementations: Spread-out



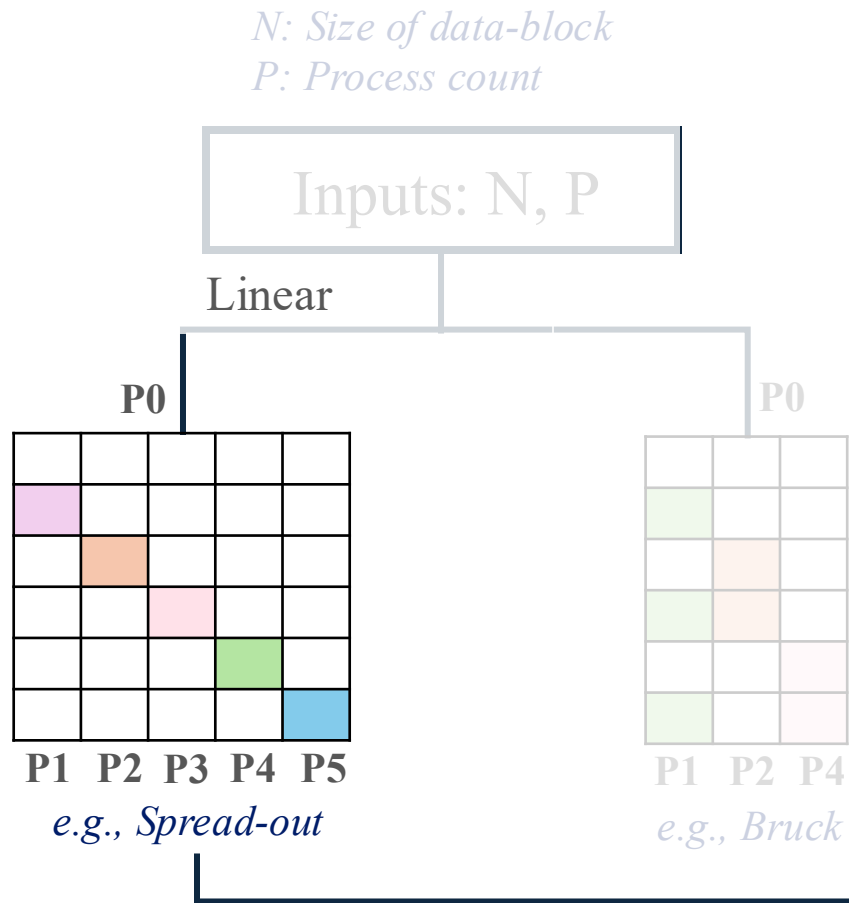
Spread-out algorithm

Pose receive requests
using `MPI_Irecv`

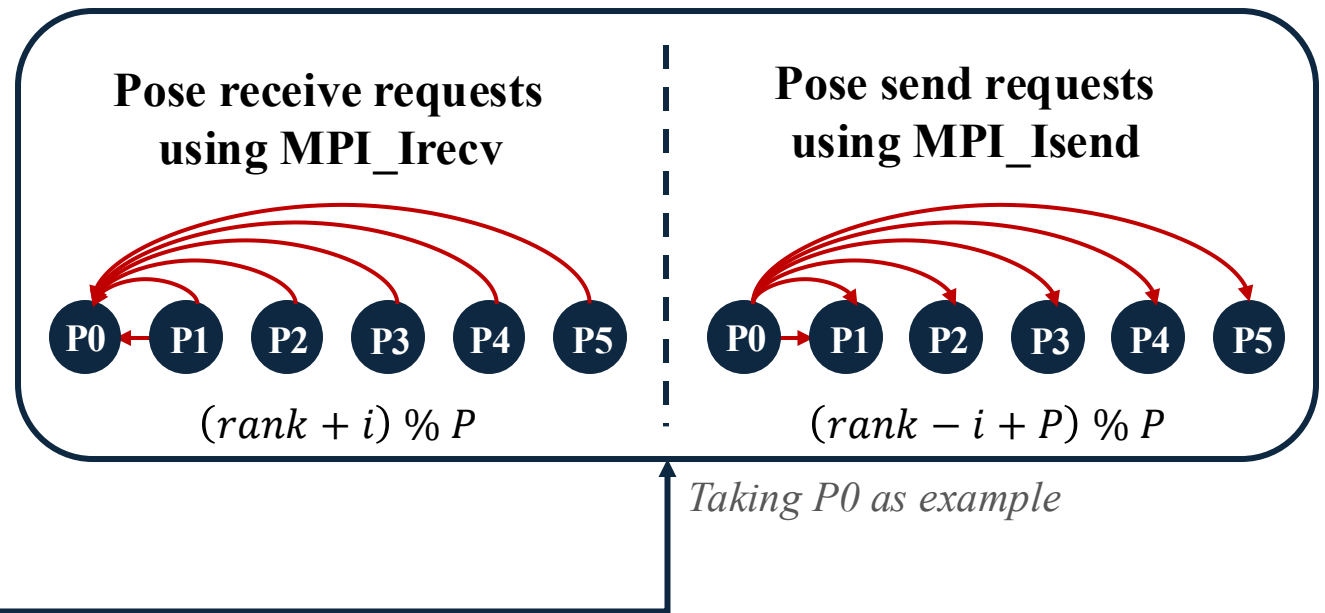


Taking P0 as example

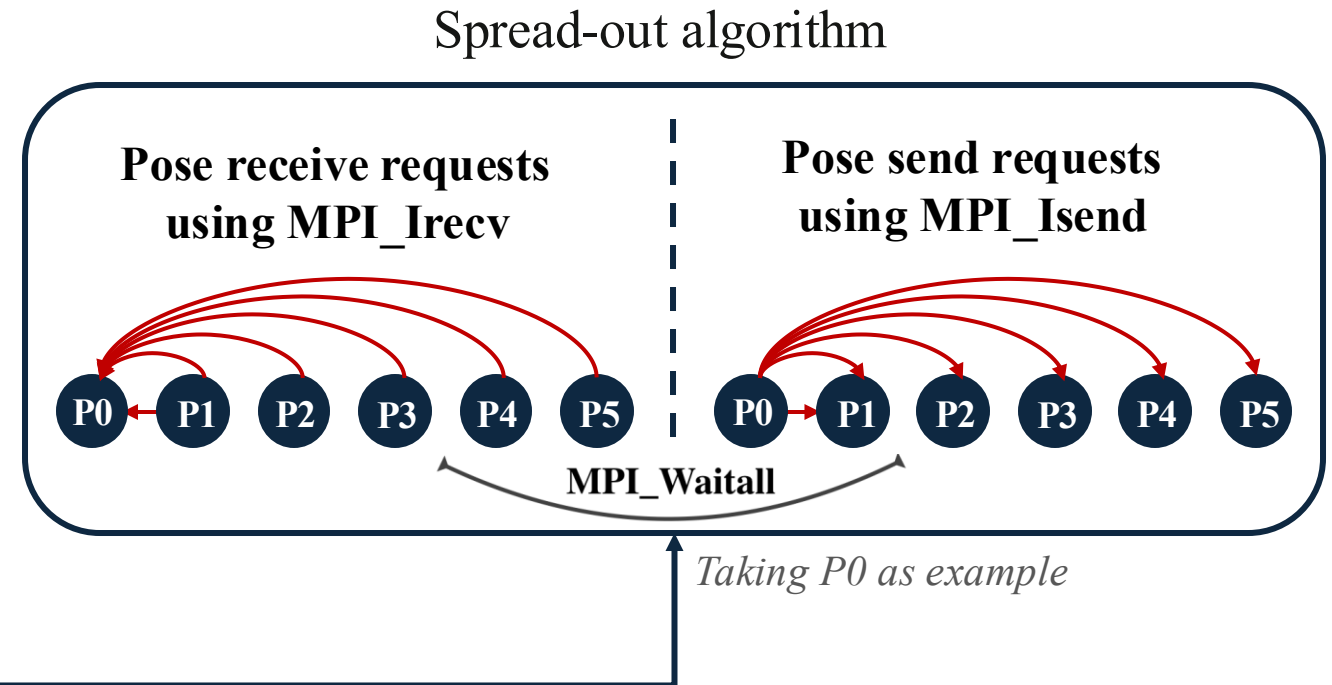
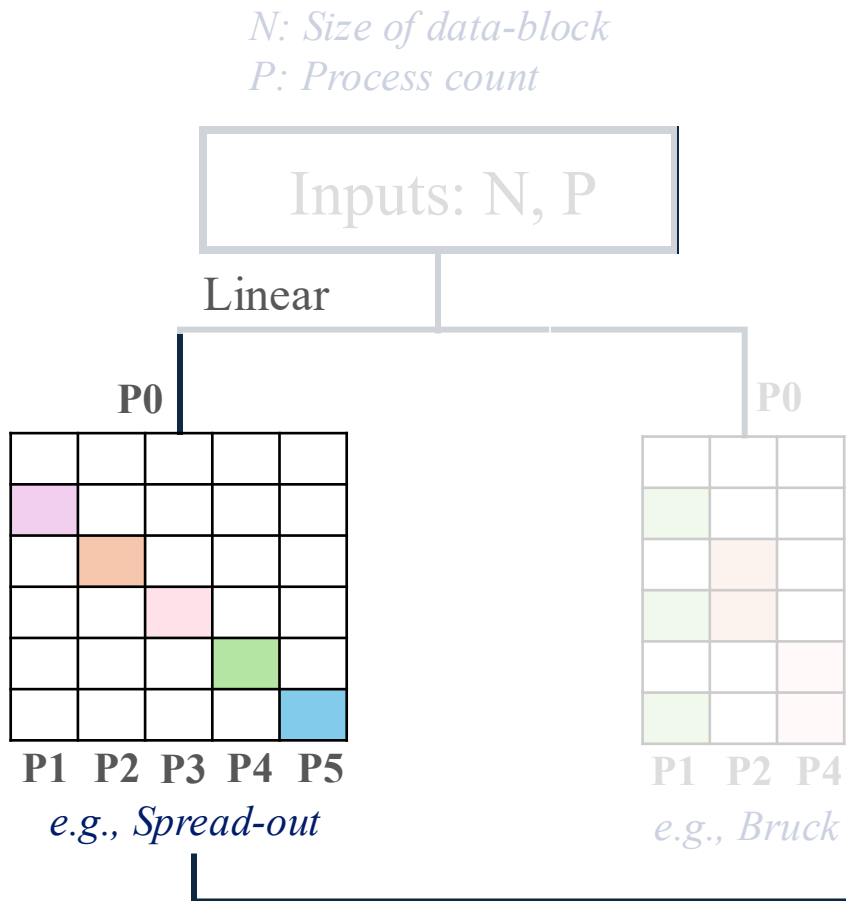
Standard MPI All-to-all Implementations: Spread-out



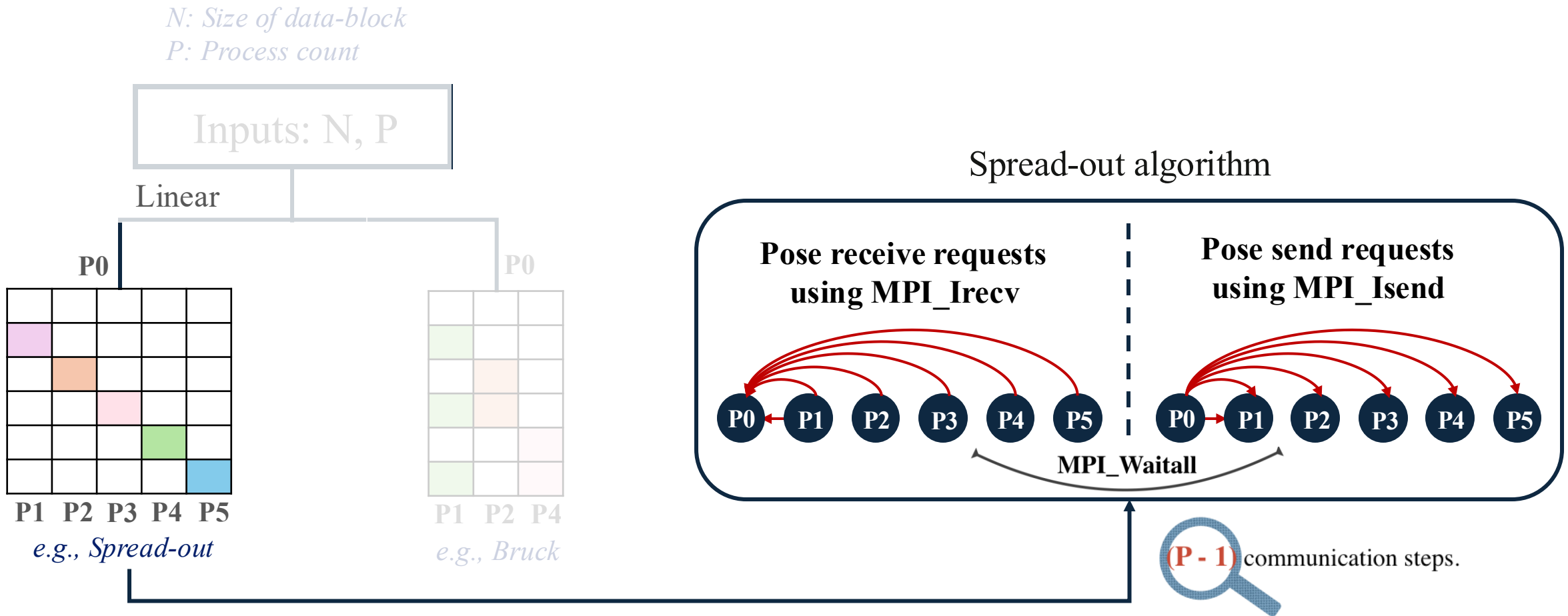
Spread-out algorithm



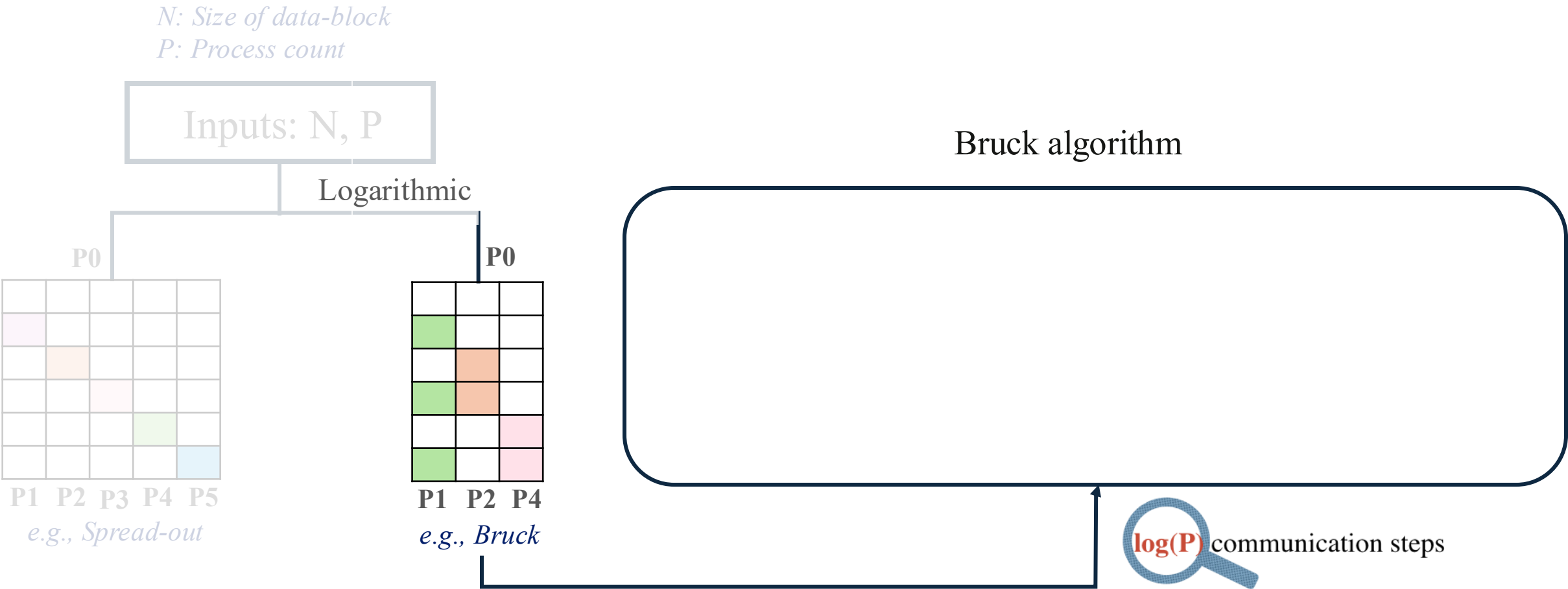
Standard MPI All-to-all Implementations: Spread-out



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



Standard MPI All-to-all Implementations: Bruck



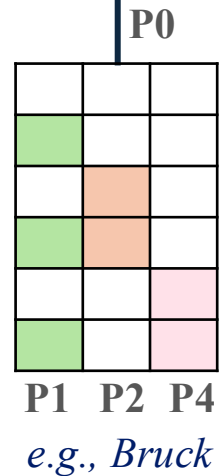
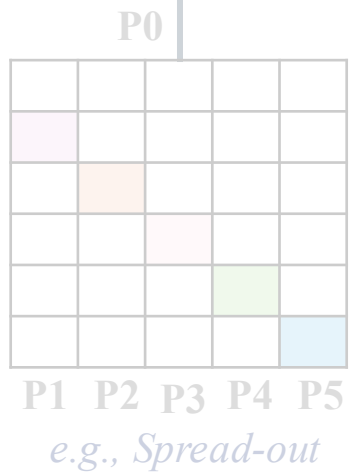
Standard MPI All-to-all Implementations: Bruck

N: Size of data-block

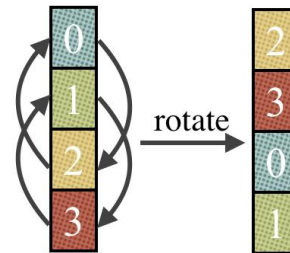
P: Process count

Inputs: N, P

Logarithmic



Bruck algorithm



1. Local initial rotation.



log(P) communication steps

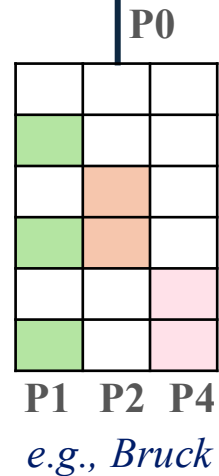
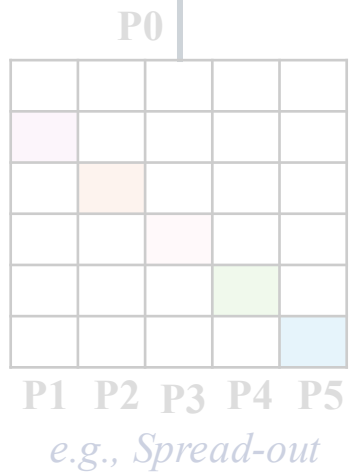
Standard MPI All-to-all Implementations: Bruck

N: Size of data-block

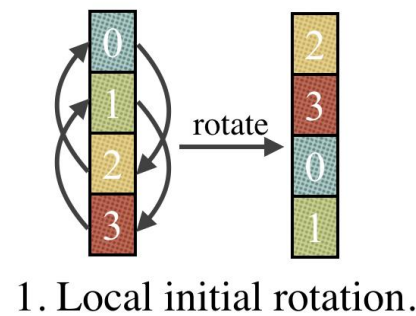
P: Process count

Inputs: N, P

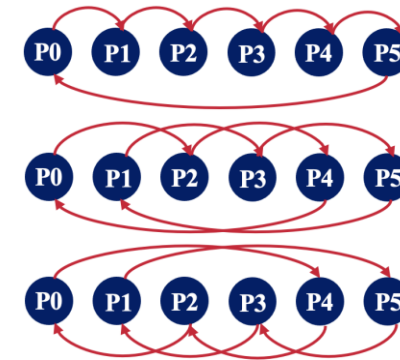
Logarithmic



Bruck algorithm



2. $\log(P)$ communication steps.



$\log(P)$ communication steps

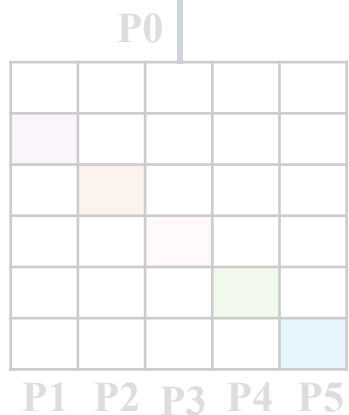
Standard MPI All-to-all Implementations: Comm-0

N : Size of data-block

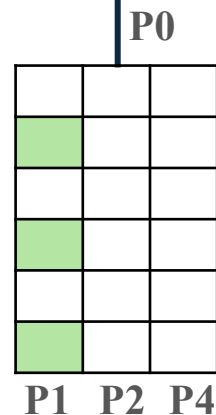
P : Process count

Inputs: N, P

Logarithmic

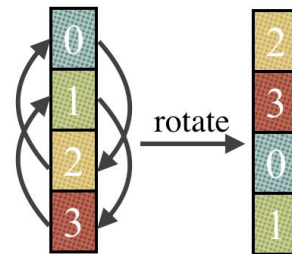


e.g., Spread-out



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

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



$\log(P)$ communication steps

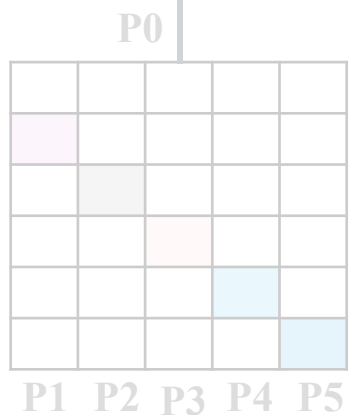
Standard MPI All-to-all Implementations: Comm-1

N: Size of data-block

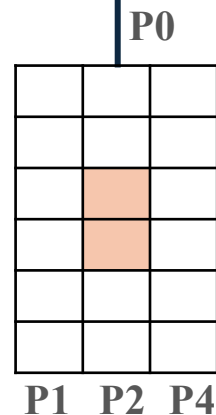
P: Process count

Inputs: N, P

Logarithmic

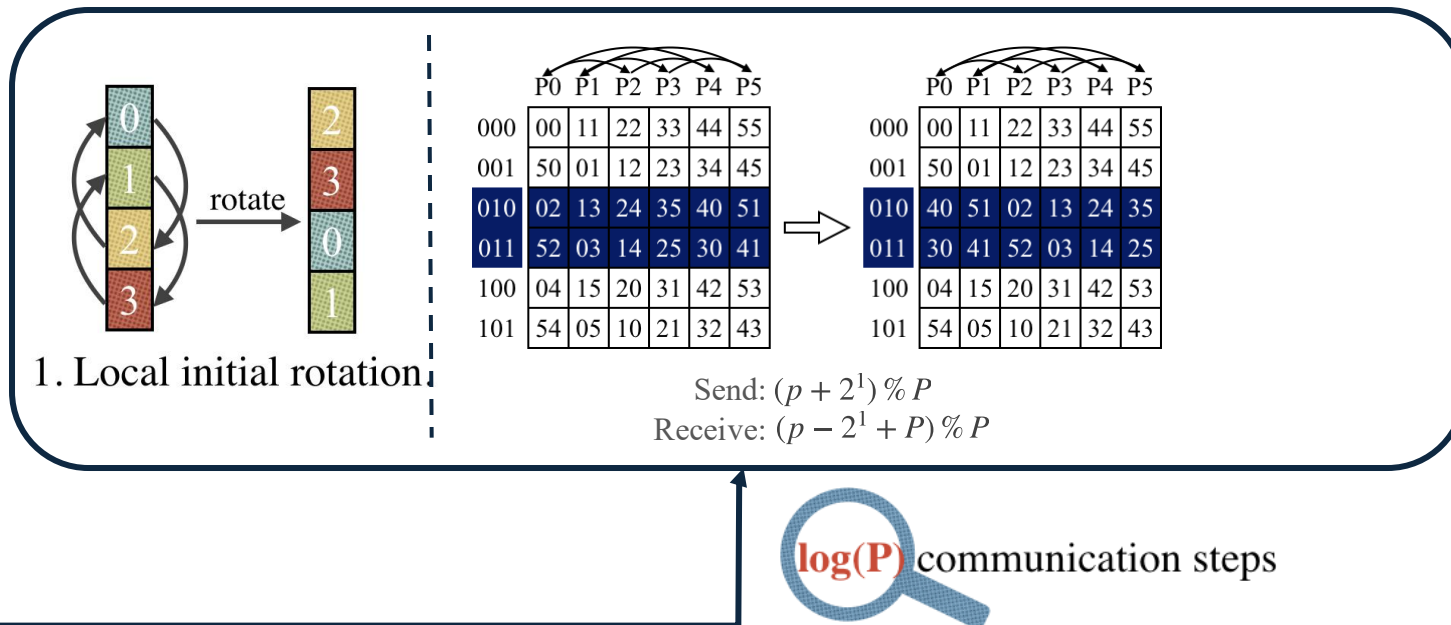


e.g., Spread-out



e.g., Bruck

Bruck algorithm



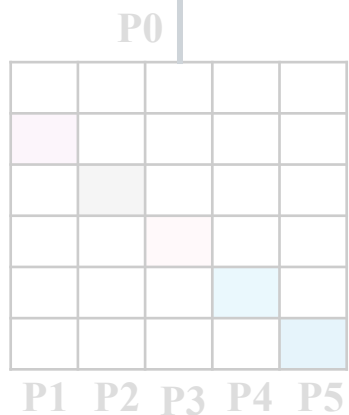
Standard MPI All-to-all Implementations: Comm-2

N: Size of data-block

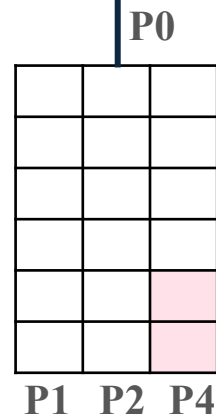
P: Process count

Inputs: N, P

Logarithmic

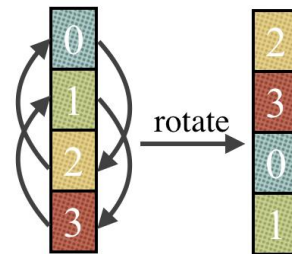


e.g., Spread-out



e.g., Bruck

Bruck algorithm



1. Local initial rotation

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



$\log(P)$ communication steps

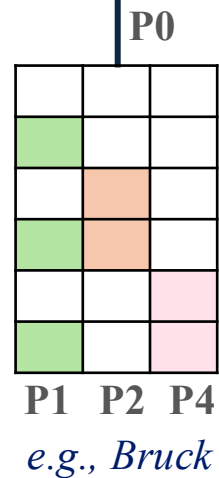
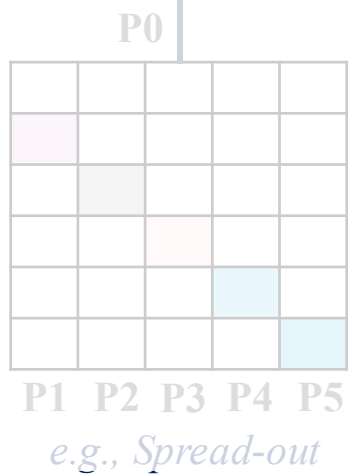
Standard MPI All-to-all Implementations: Bruck

N: Size of data-block

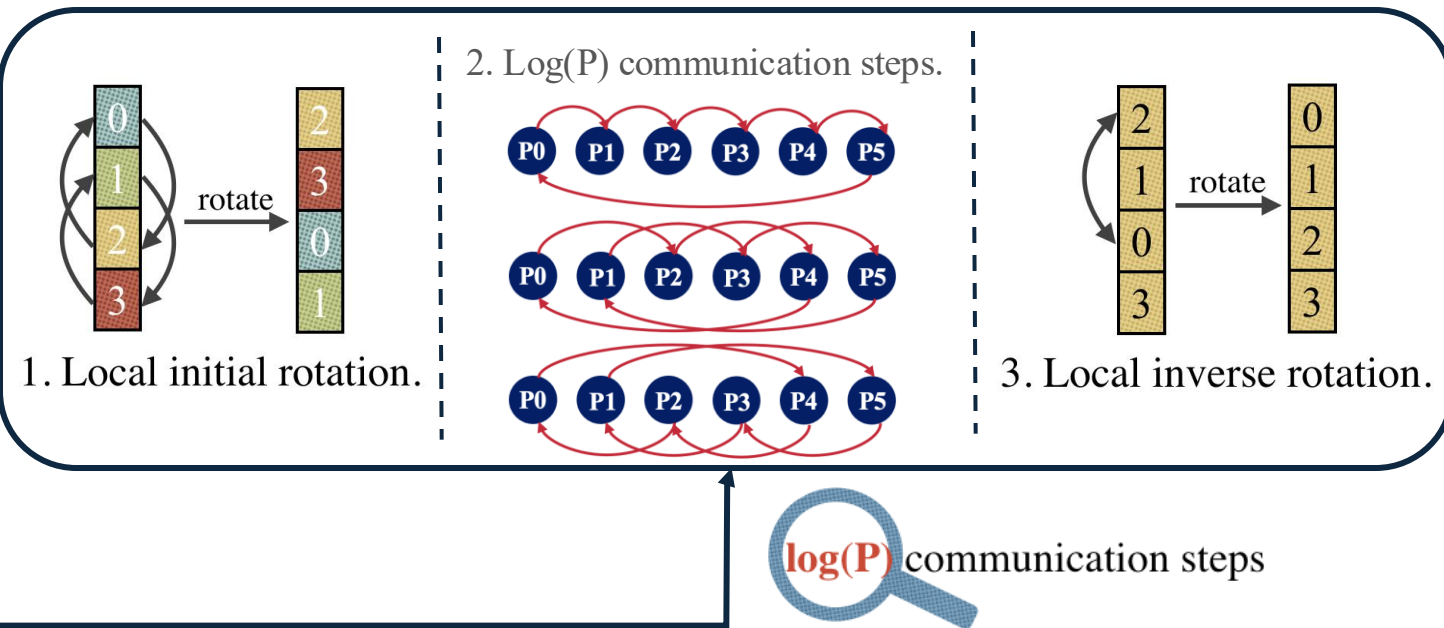
P: Process count

Inputs: N, P

Logarithmic



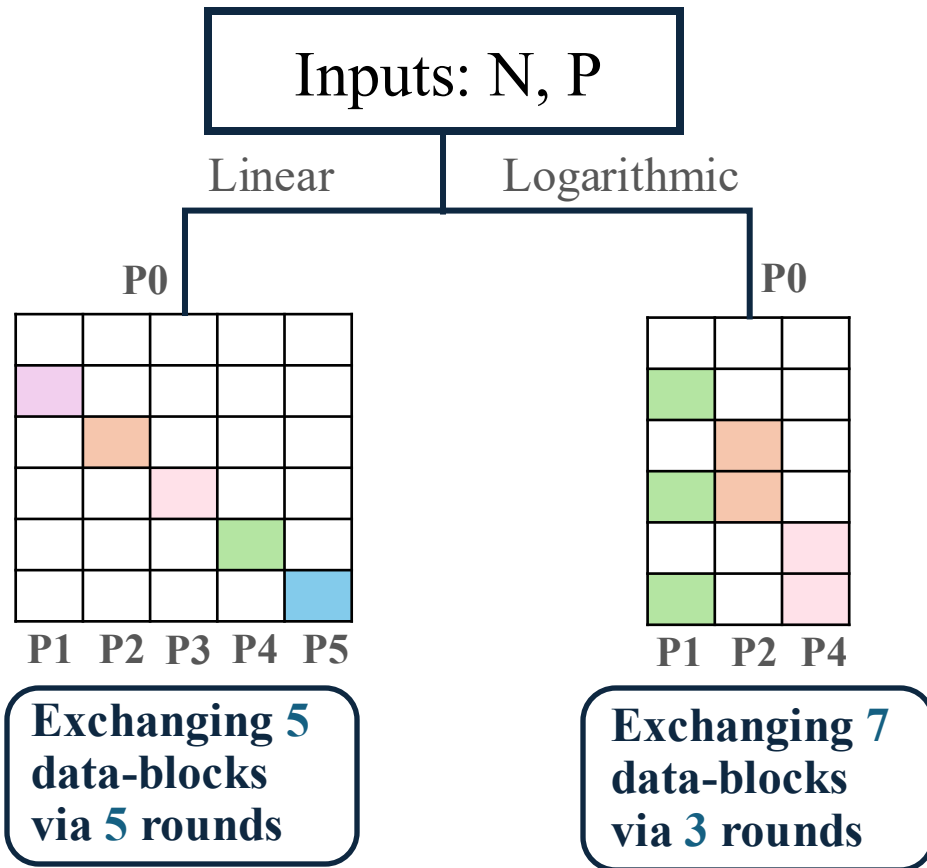
Bruck algorithm



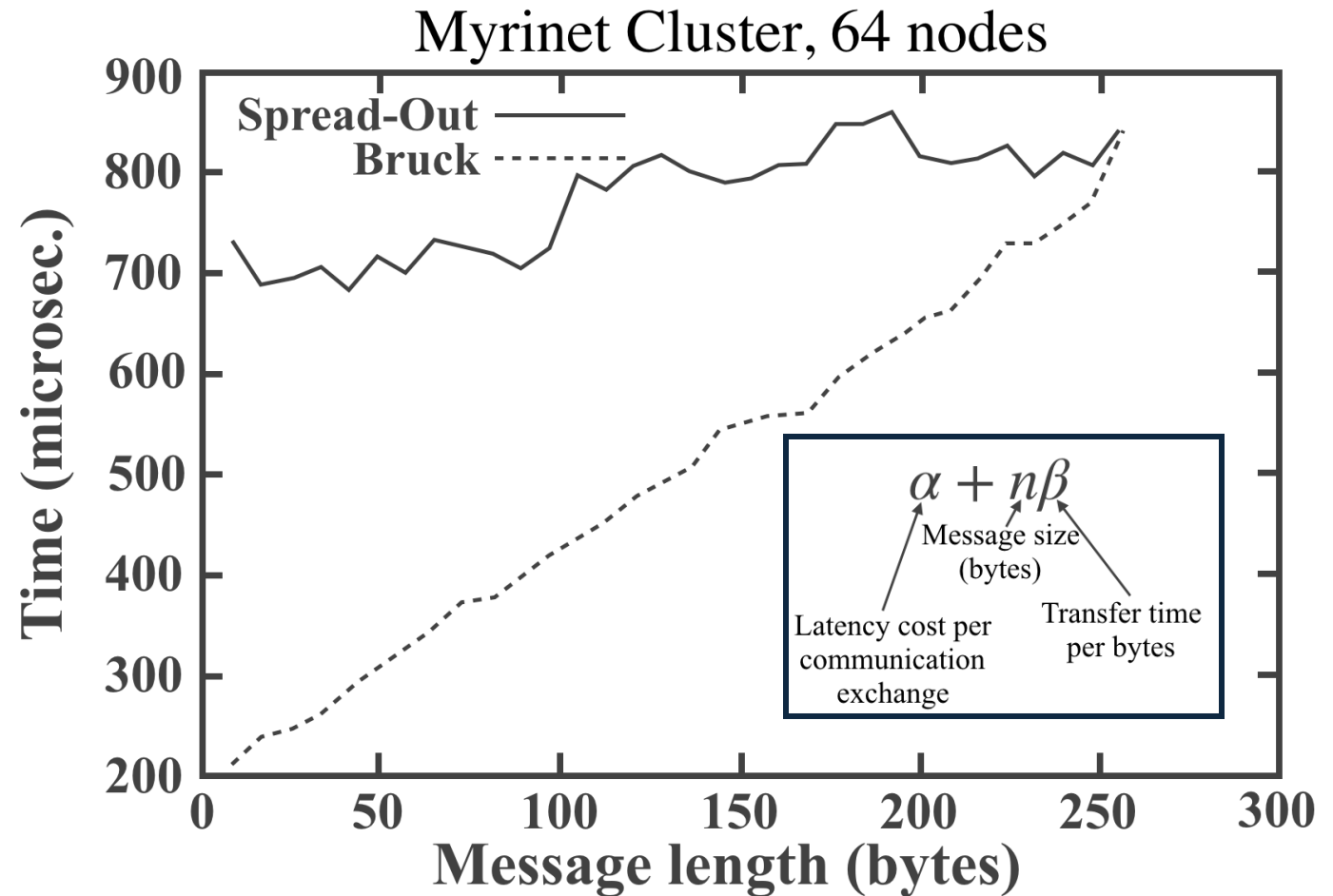
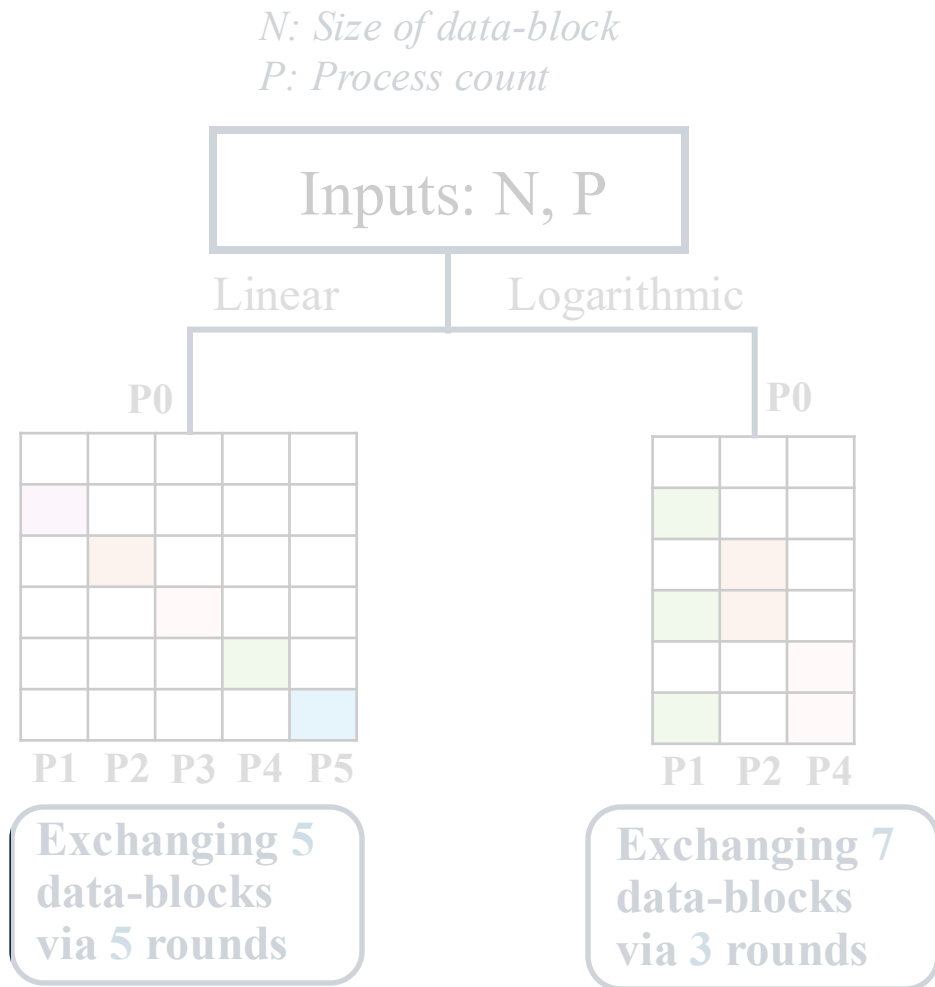
Comparison of These Two Algorithms

N: Size of data-block

P: Process count



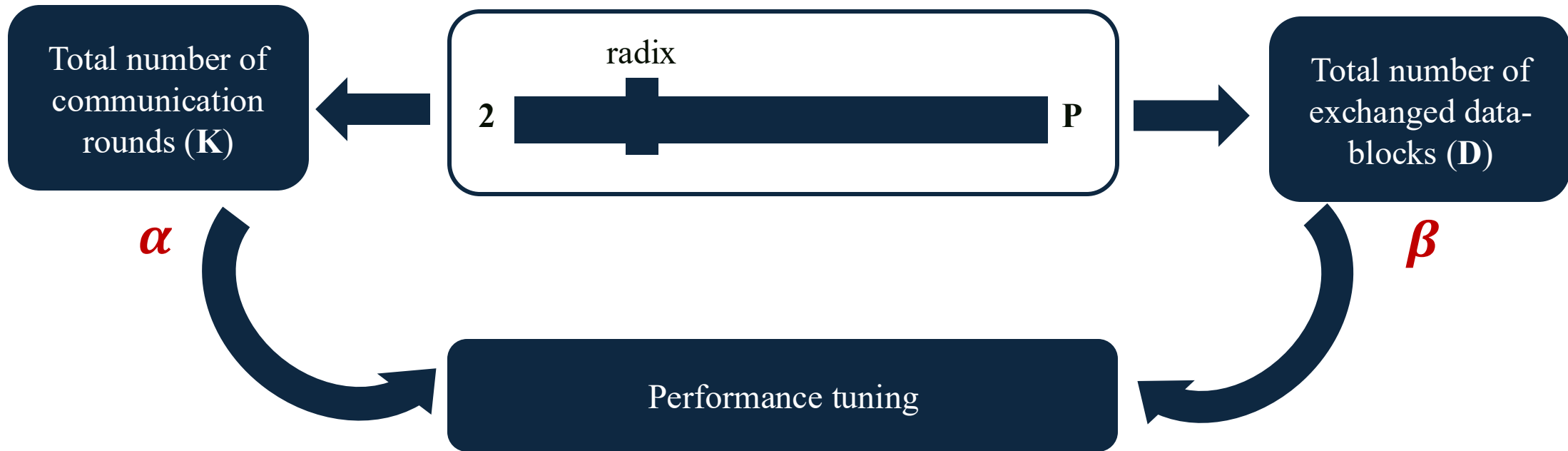
Comparison of These Two Algorithms



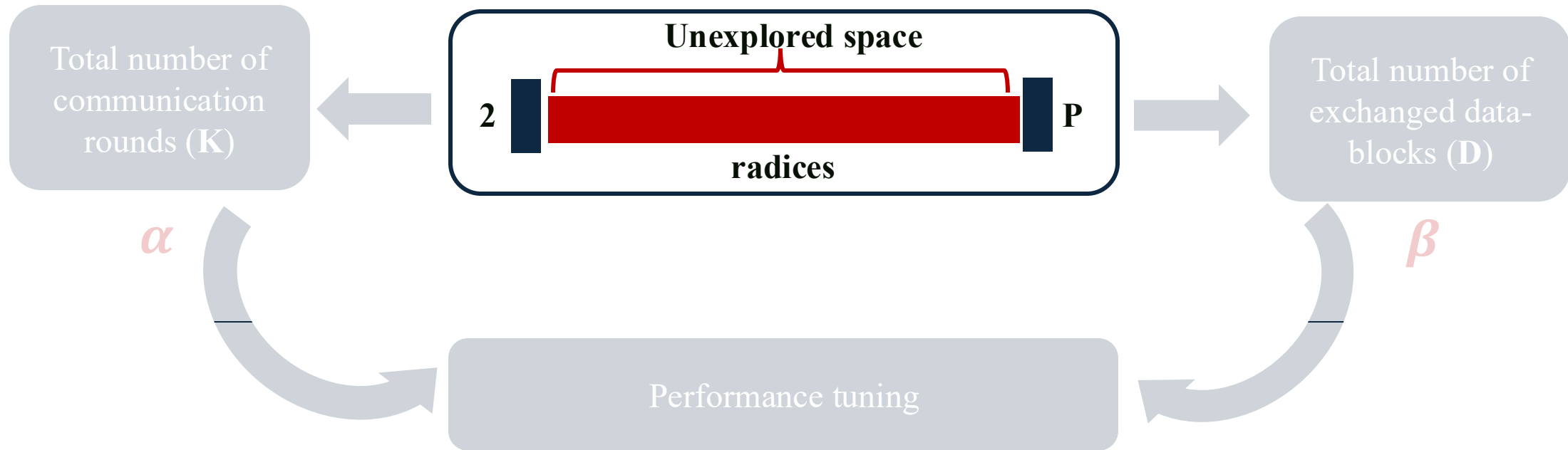
Agenda

- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- **Challenges and Solutions**
 - Tunability of Performance
 - Logarithmic Non-uniform All-to-all
- Parameterized Logarithmic Algorithm
- Parameterized Linear Algorithm
- Evaluation
- Application
- Conclusion

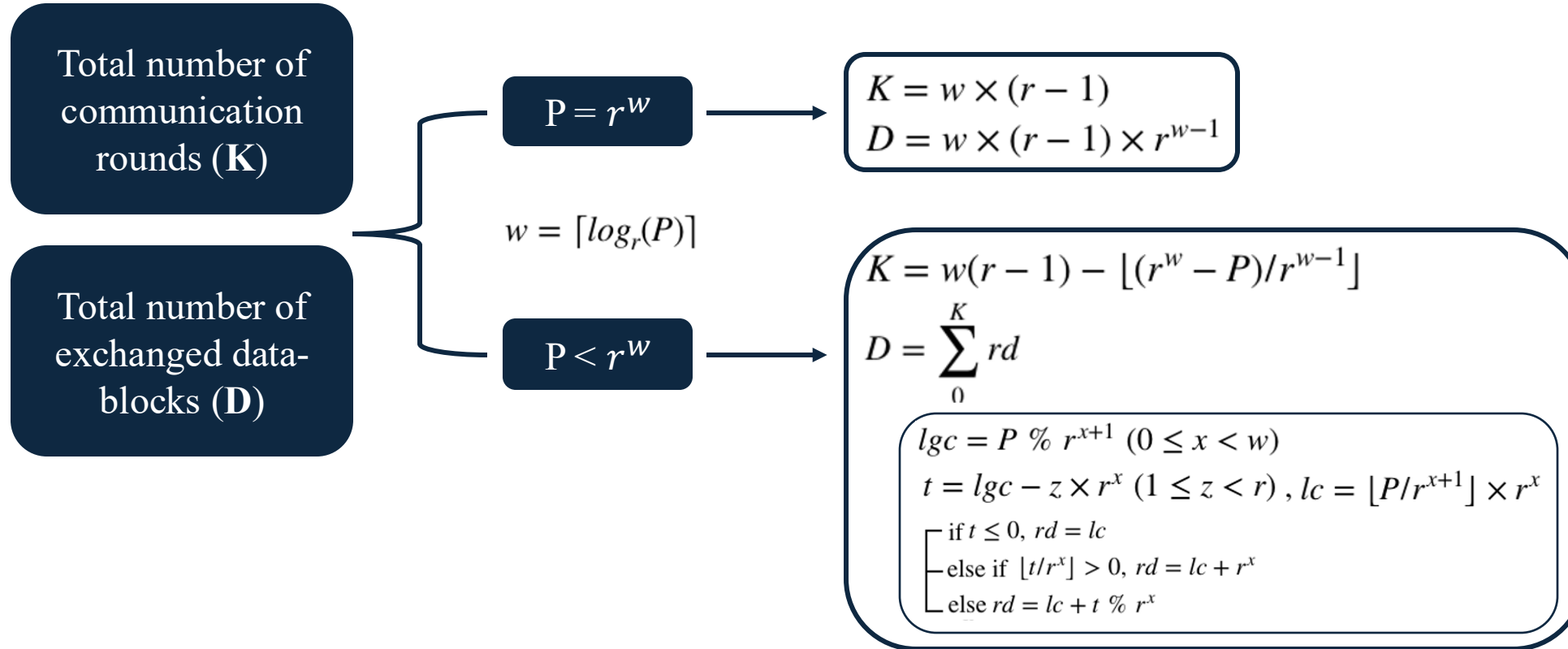
Tuning Radices Enables Performance Tuning



Standard MPI All-to-all Implementation Leaves a Large Unexplored Space

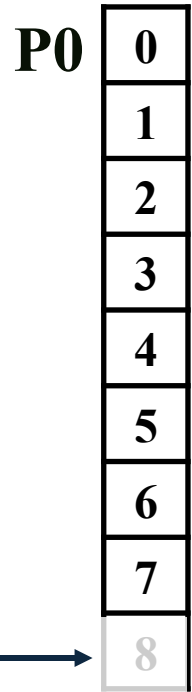


Mathematics of Tunable Radix All-to-all



Tunable Radix All-to-all Example

$$P = 8, r = 3$$



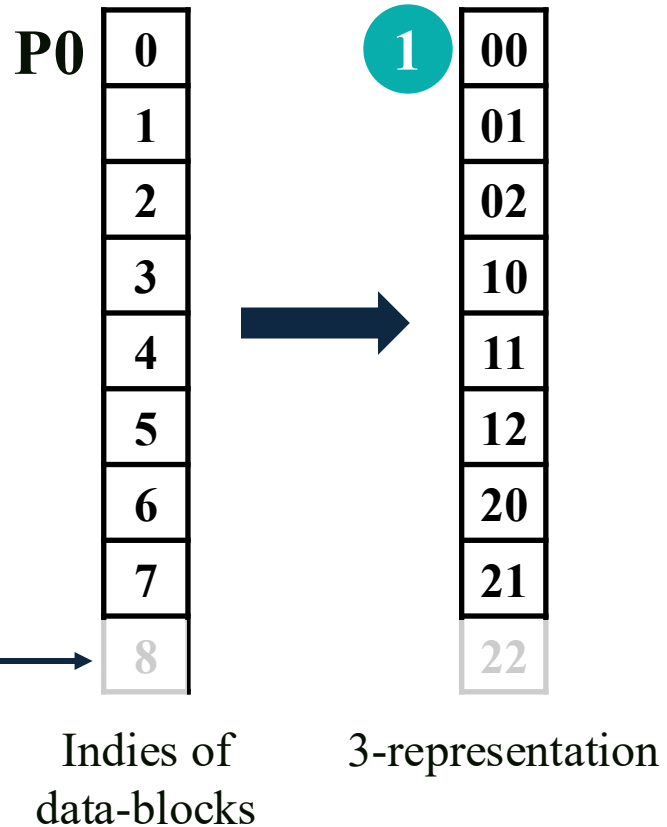
if $\log_r(P)$
is integer



Indices of
data-blocks

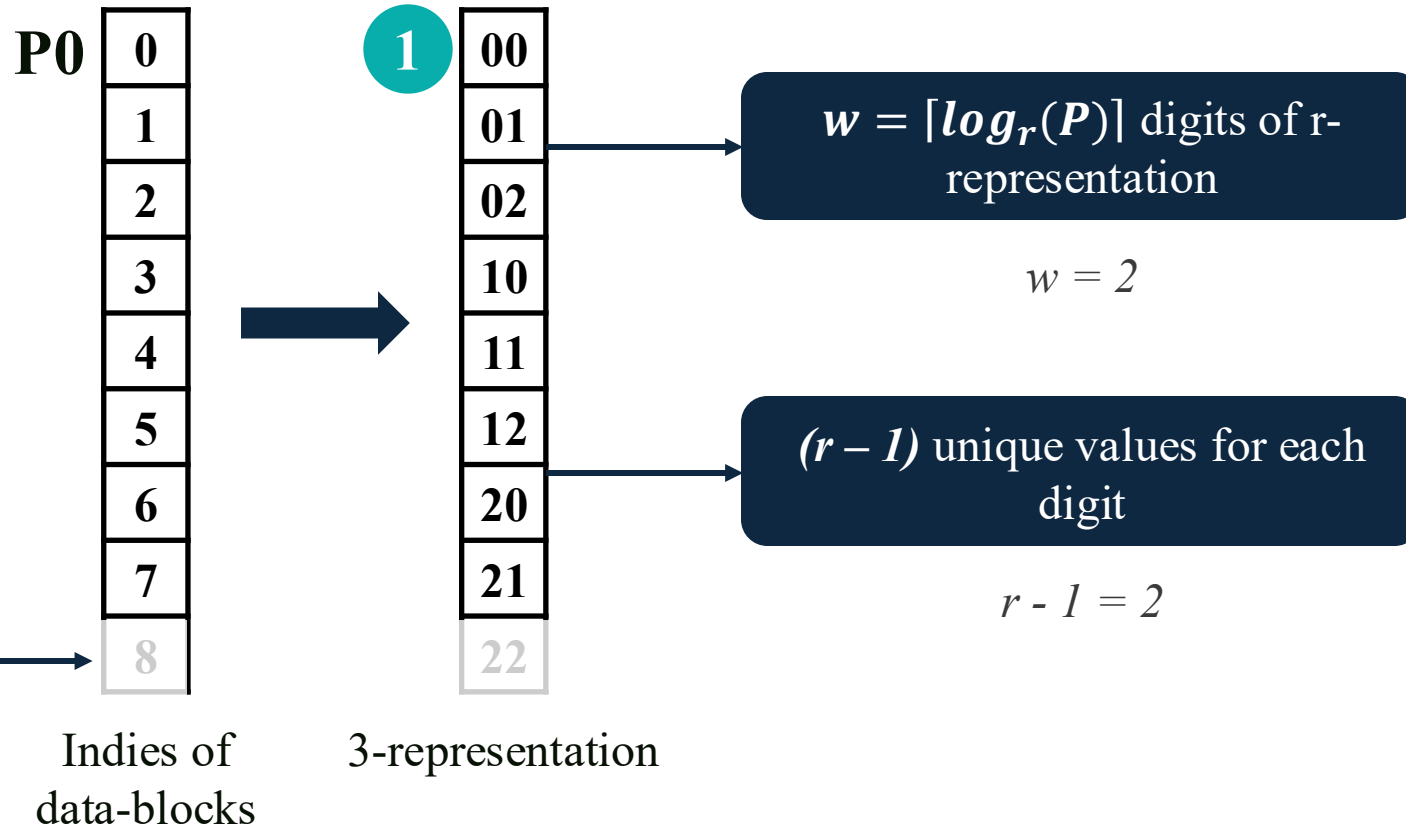
Tunable Radix All-to-all Example: r-representation

$P = 8, r = 3$



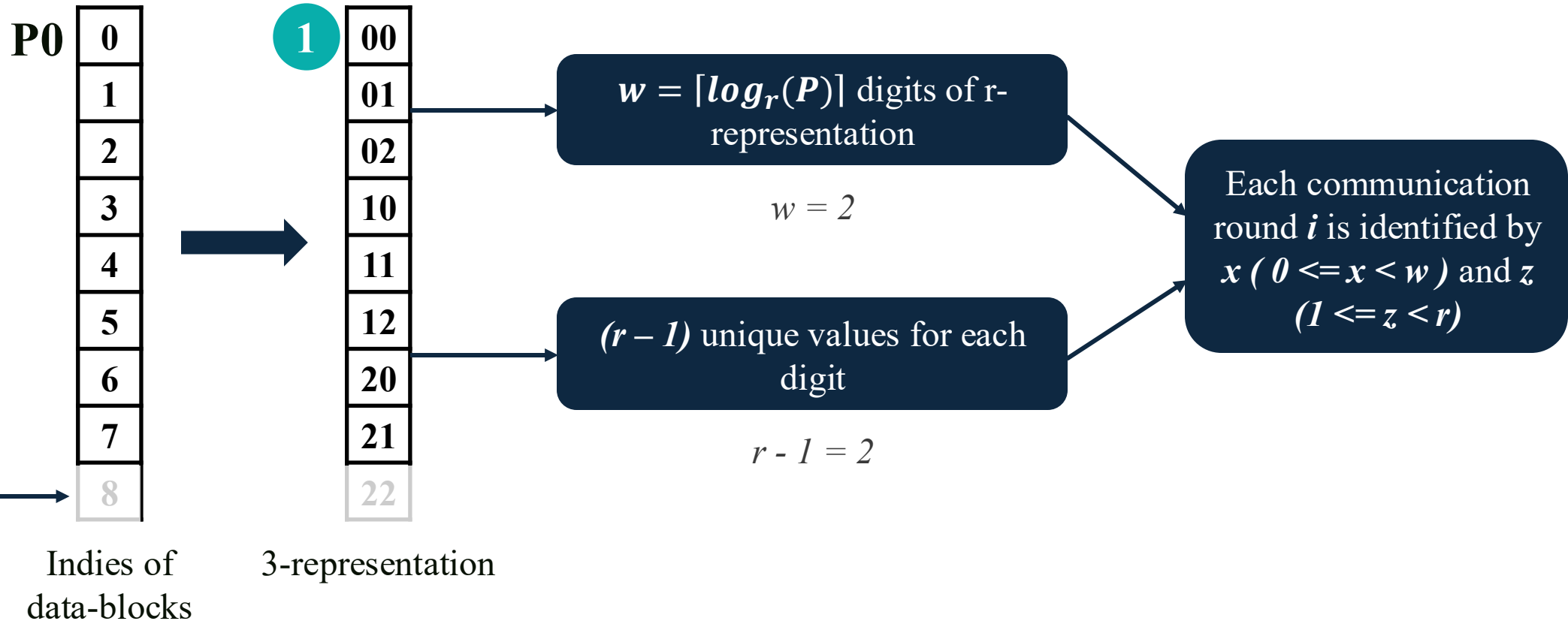
Tunable Radix All-to-all Example: r-representation

$P = 8, r = 3$



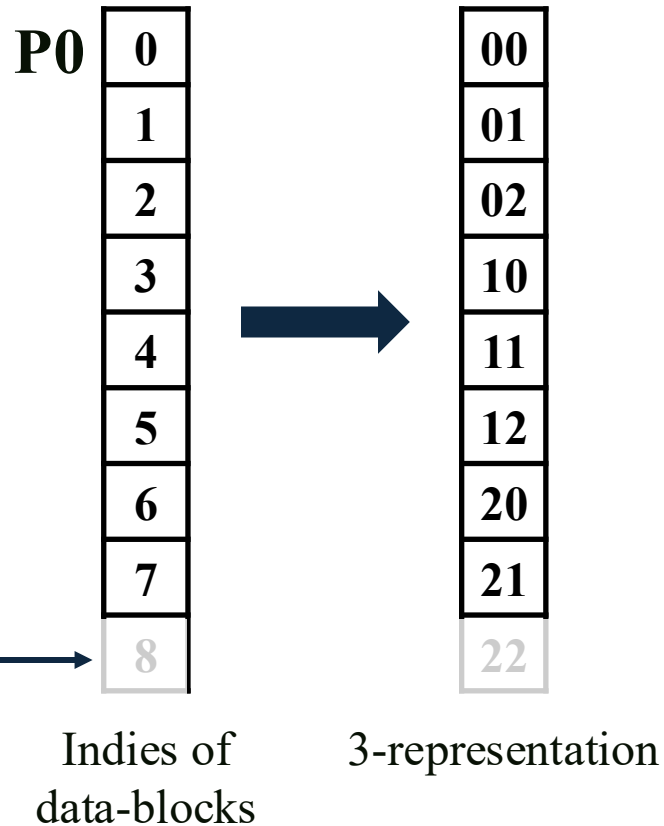
Tunable Radix All-to-all Example: r-representation

$P = 8, r = 3$



Tunable Radix All-to-all Example: Communication

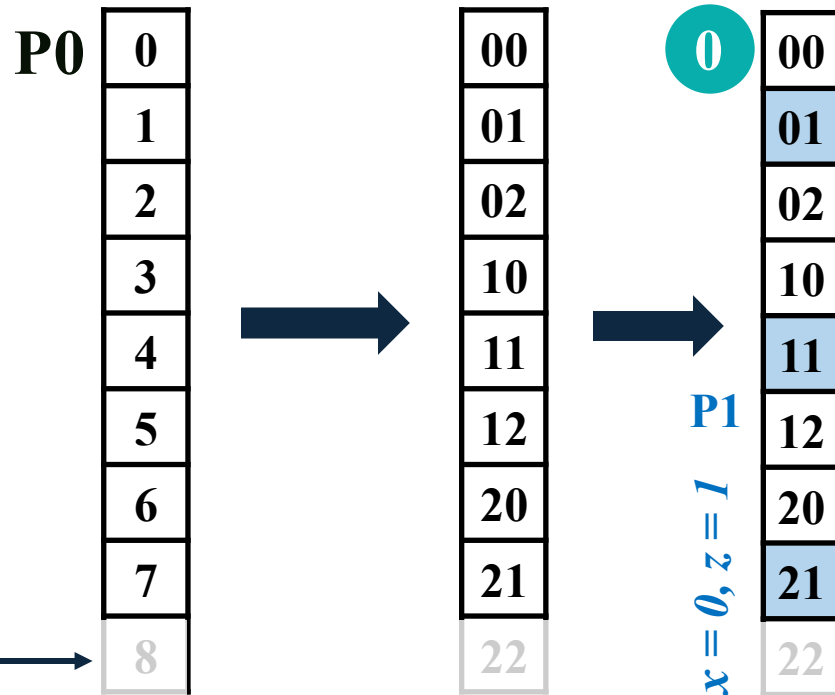
$P = 8, r = 3$



P0 sends data-blocks whose
x digit equals to z

Tunable Radix All-to-all Example: Comm-round-0

$P = 8, r = 3$



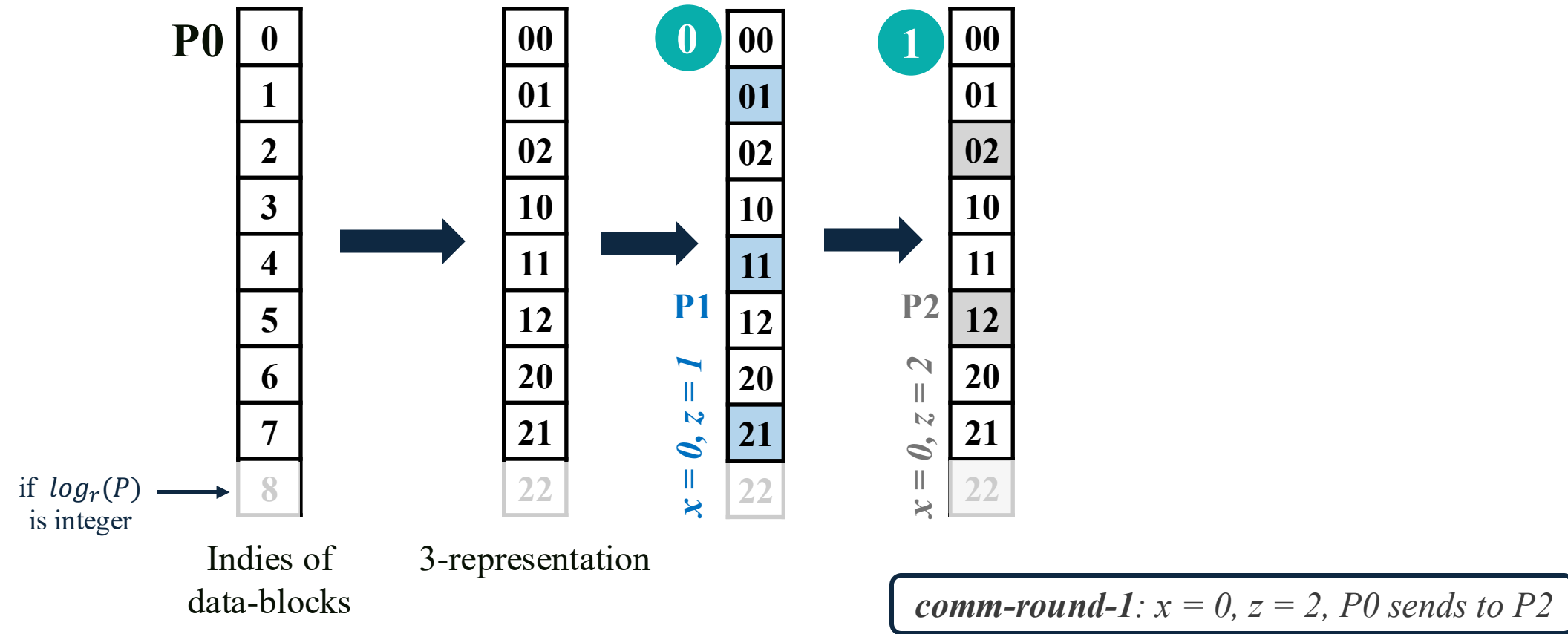
Indices of
data-blocks

3-representation

comm-round-0: $x = 0, z = 1, P_0$ sends to P_1

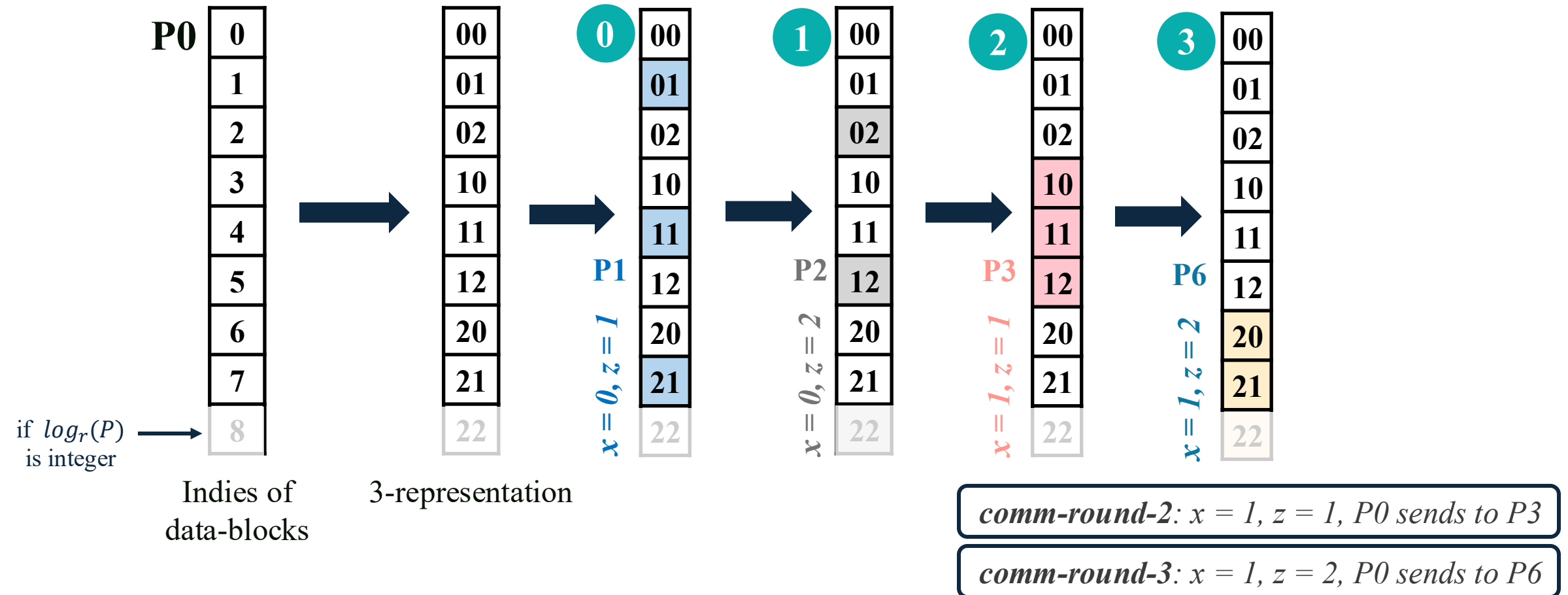
Tunable Radix All-to-all Example: Comm-round-1

$P = 8, r = 3$



Tunable Radix All-to-all Example: Comm-round-2,3

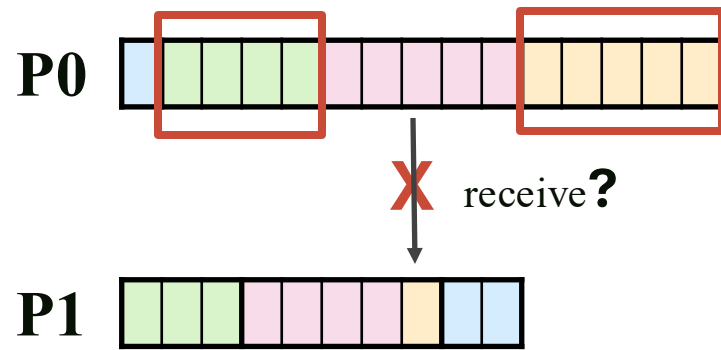
$P = 8, r = 3$



Agenda

- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- **Challenges and Solutions**
 - Tunability of Performance
 - **Logarithmic Non-uniform All-to-all**
- Parameterized Logarithmic Algorithm
- Parameterized Linear Algorithm
- Evaluation
- Application
- Conclusion

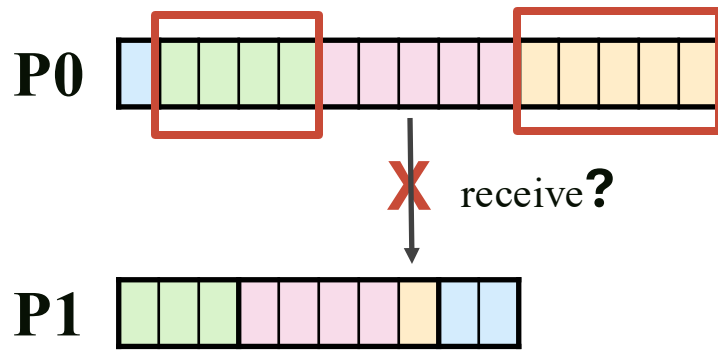
Challenges of Applying the Logarithmic Algorithm to Non-uniform All-to-all Data Exchange



Each process is unaware of how much data to expect during each intermediate communication round.

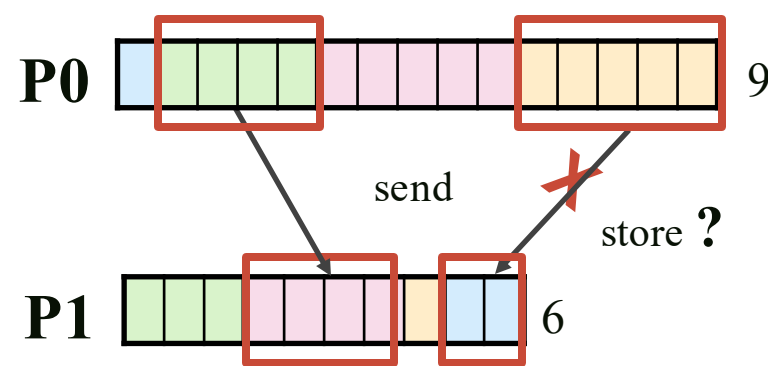
Non-uniform
workloads

Challenges of Applying the Logarithmic Algorithm to Non-uniform All-to-all Data Exchange



Each process is unaware of how much data to expect during each intermediate communication round.

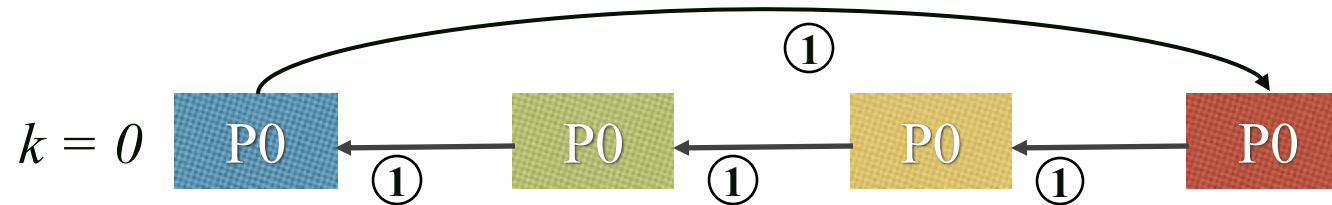
Non-uniform
workloads



The received data elements could be too large to fit into the segment in the send buffer.

Store-and-
forward

Two-phase Communication Scheme

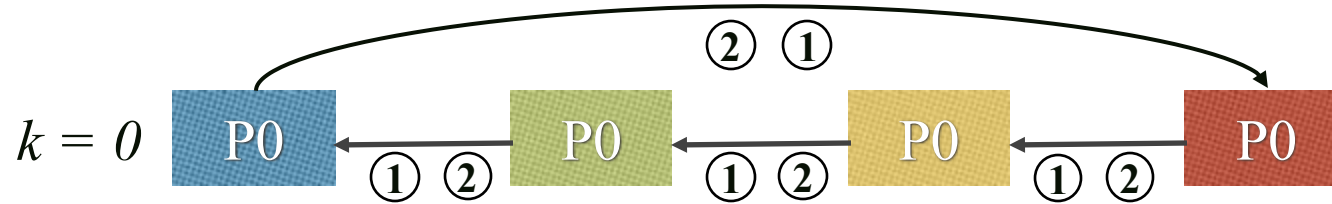


① *Metadata Exchange*

a	b
---	---

 Size of each block

Two-phase Communication Scheme



- ① *Metadata Exchange*

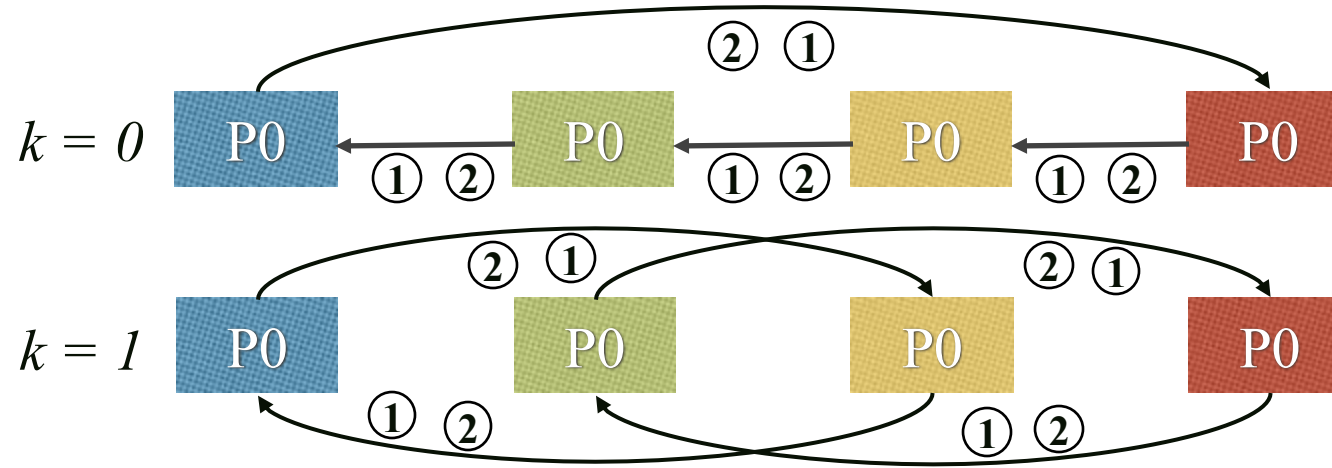
a	b
---	---

 Size of each block
- ② *Data Exchange*

Block 1	Block 3
---------	---------

 Actual data-blocks

Two-phase Communication Scheme



① *Metadata Exchange*

a	b
---	---

Size of each block

② *Data Exchange*

Block 1	Block 3
---------	---------

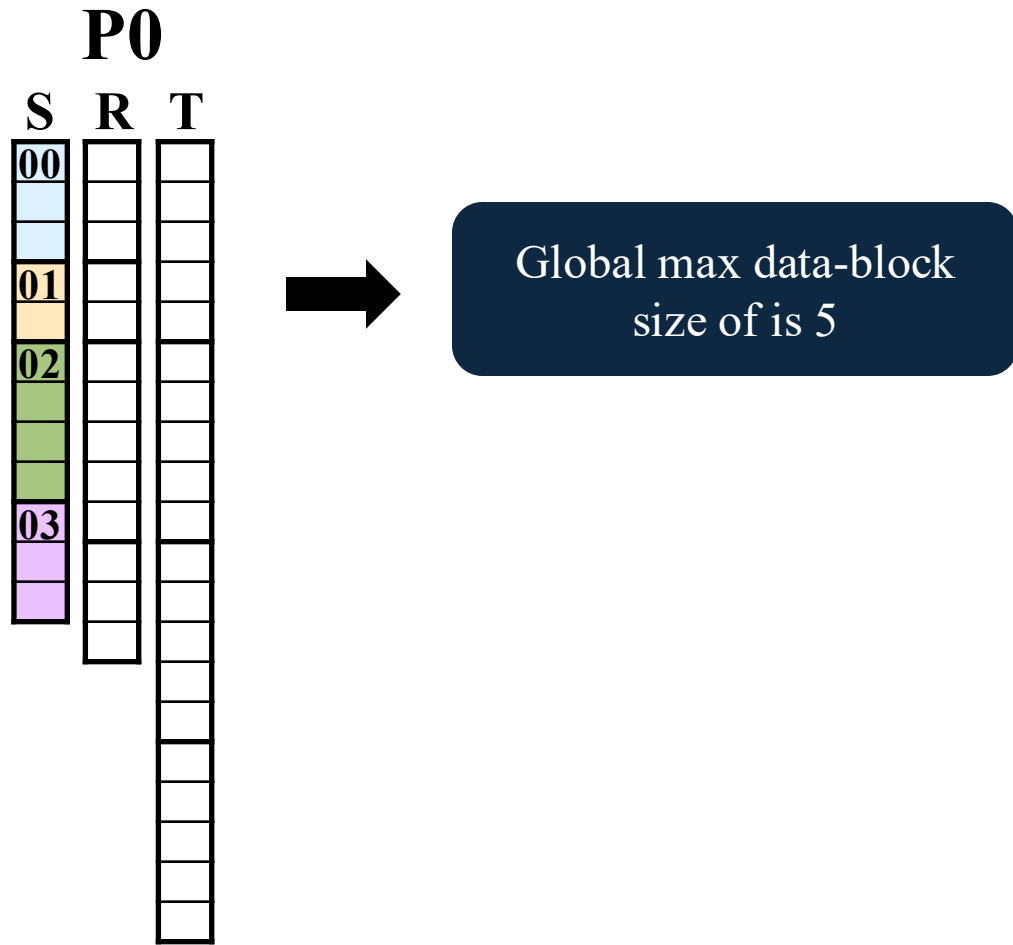
Actual data-blocks

Using Temporary Buffer to Solve Challenge 2

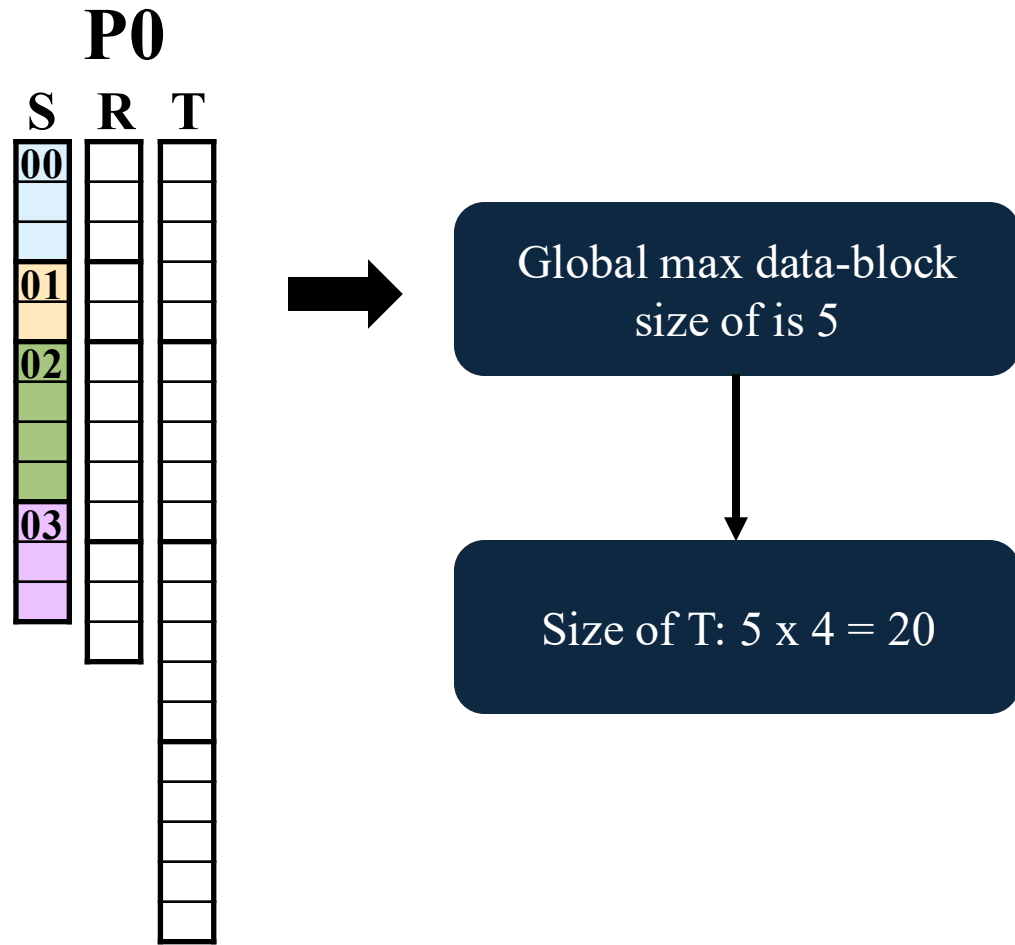
P0

[illegible]

Estimating Temporary Buffer Size



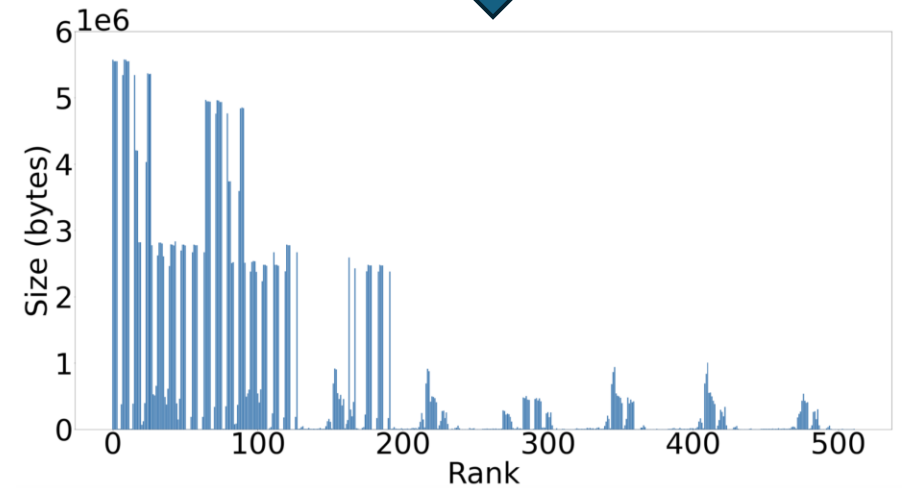
Estimating Temporary Buffer Size



The diagram shows a process flow for calculating the size of T. It starts with a table labeled 'P0' with columns 'S', 'R', and 'T'. The 'S' column contains values 00, 01, 02, 03, and 04. The 'R' column contains values 00, 01, 02, 03, and 04. The 'T' column contains values 00, 01, 02, 03, and 04. An arrow points from the table to a box stating 'Global max data-block size of is 5'. Another arrow points from this box to a box stating 'Size of T: 5 x 4 = 20'.



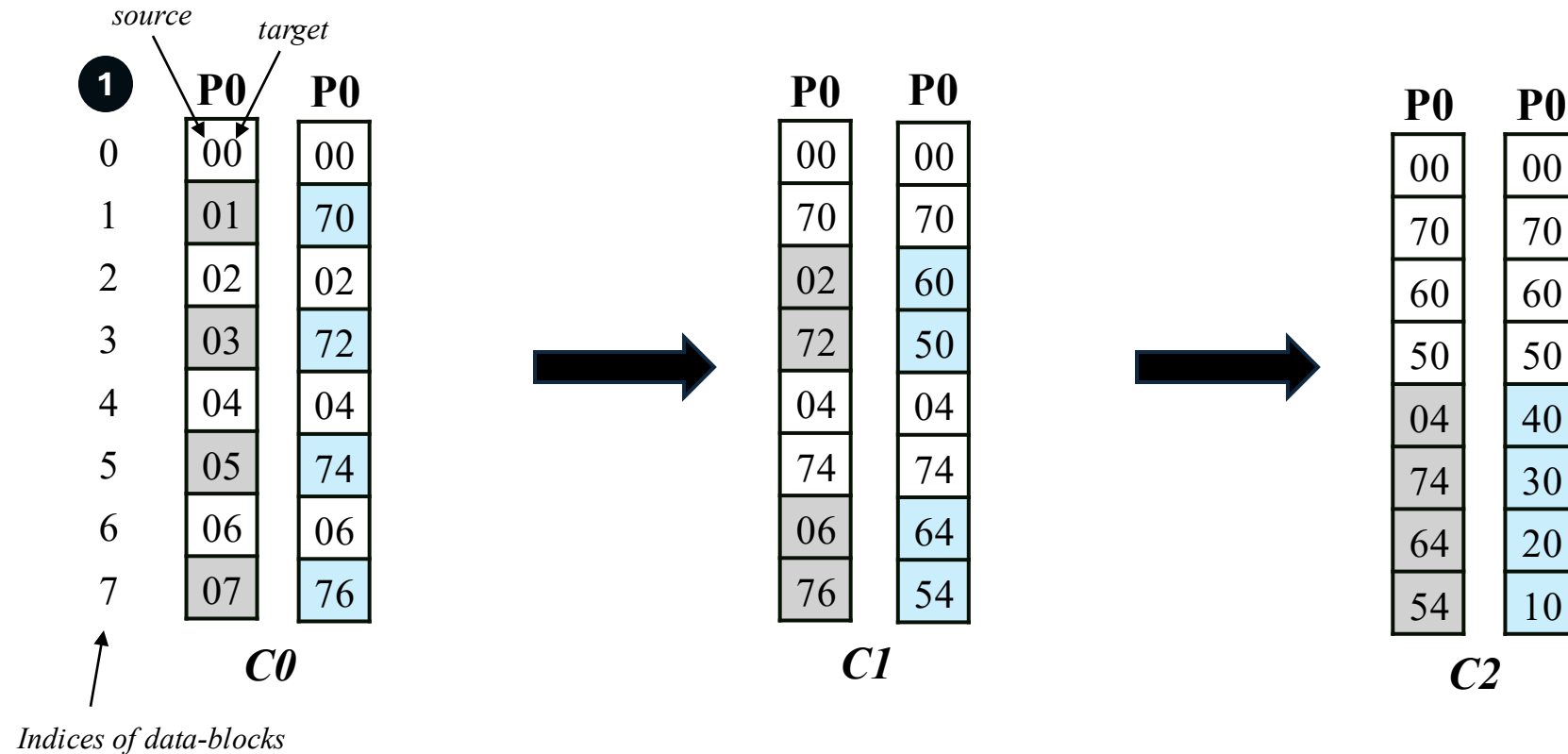
Size of T ?



Sparse workloads

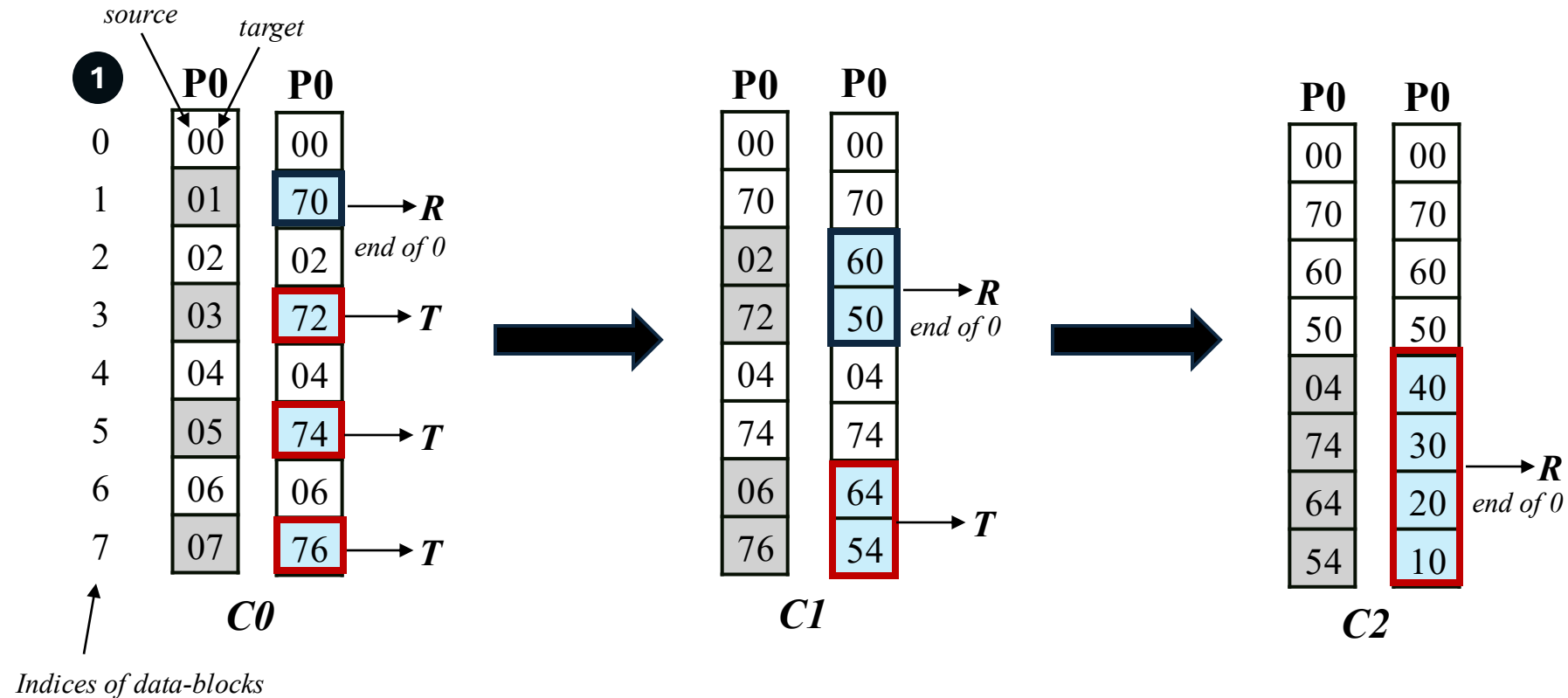
Temporary Buffer Size Analysis

$$P = 8, r = 2$$



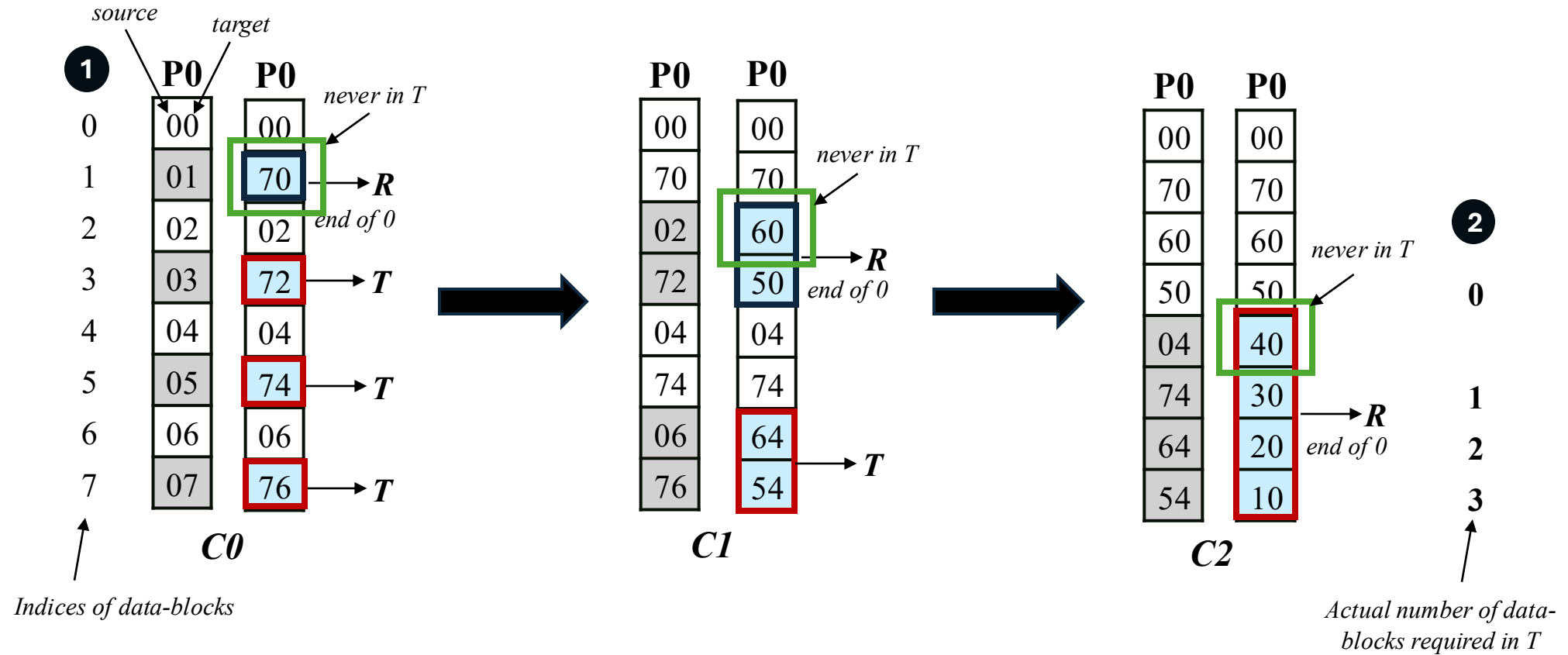
Temporary Buffer Size Analysis

$$P = 8, r = 2$$

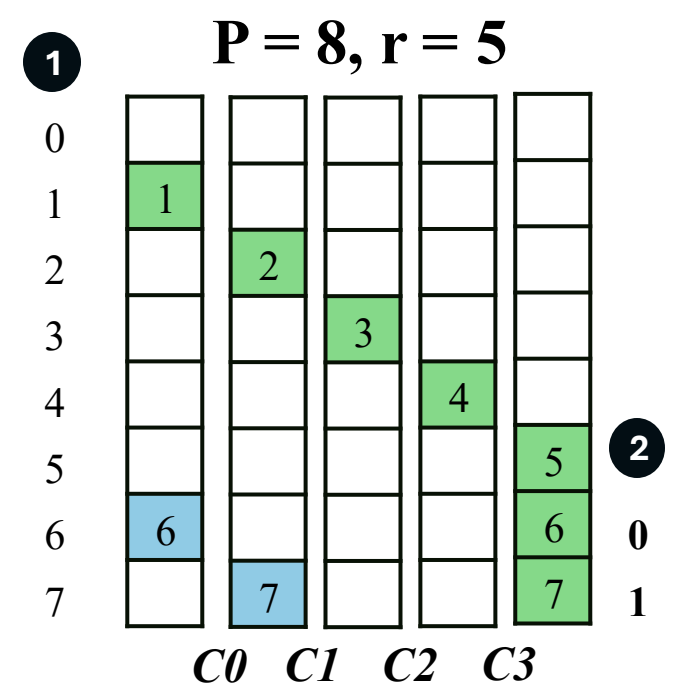
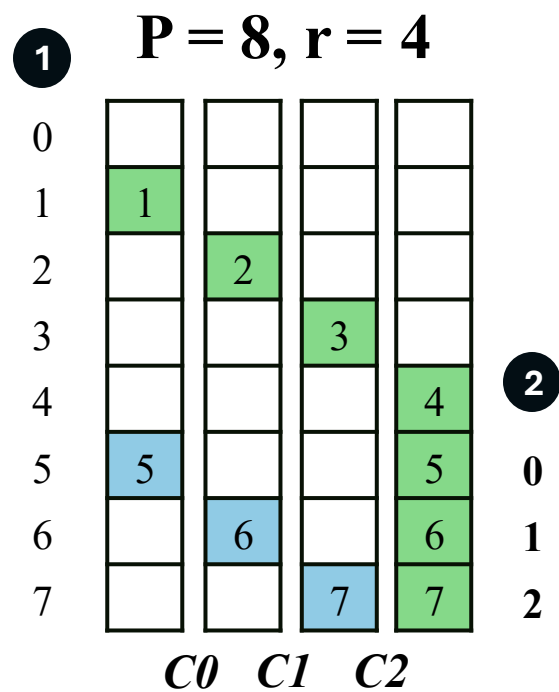
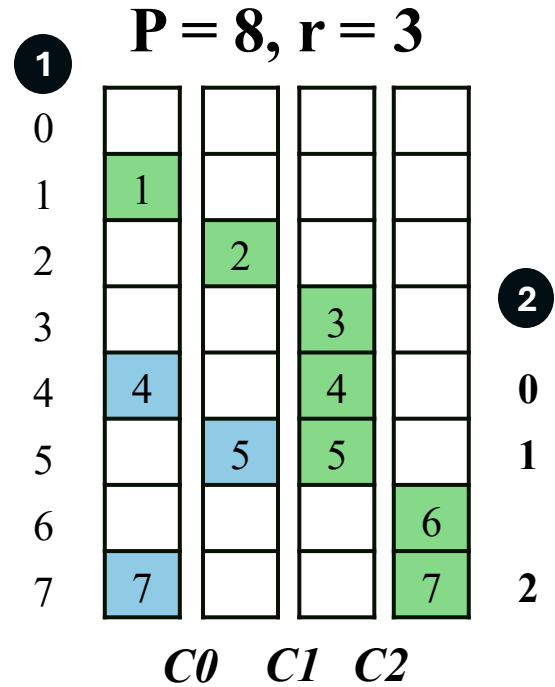


Temporary Buffer Size Analysis

$$P = 8, r = 2$$



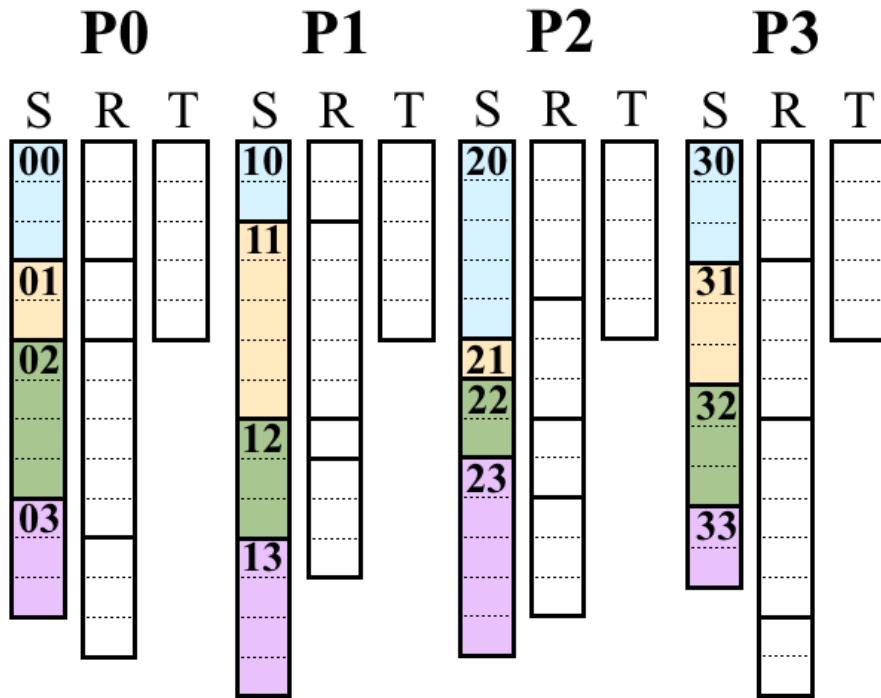
Temporary Buffer Size Analysis



Agenda

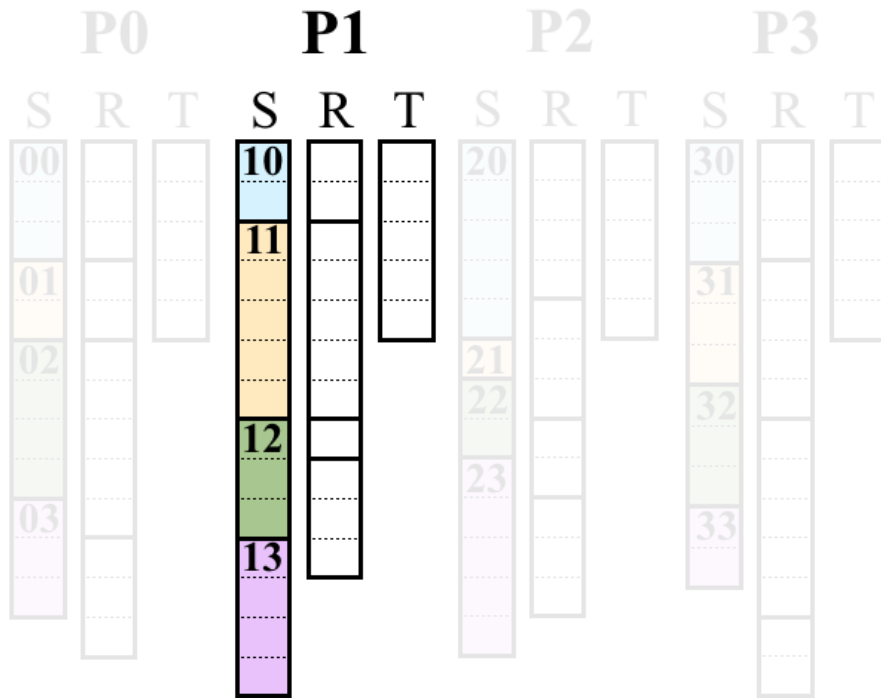
- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- Challenges and Solutions
 - Tunability of Performance
 - Logarithmic Non-uniform All-to-all
- **Parameterized Logarithmic Algorithm**
- Parameterized Linear Algorithm
- Evaluation
- Application
- Conclusion

Example of the ParLogNa with $P = 4$ and $r = 2$.



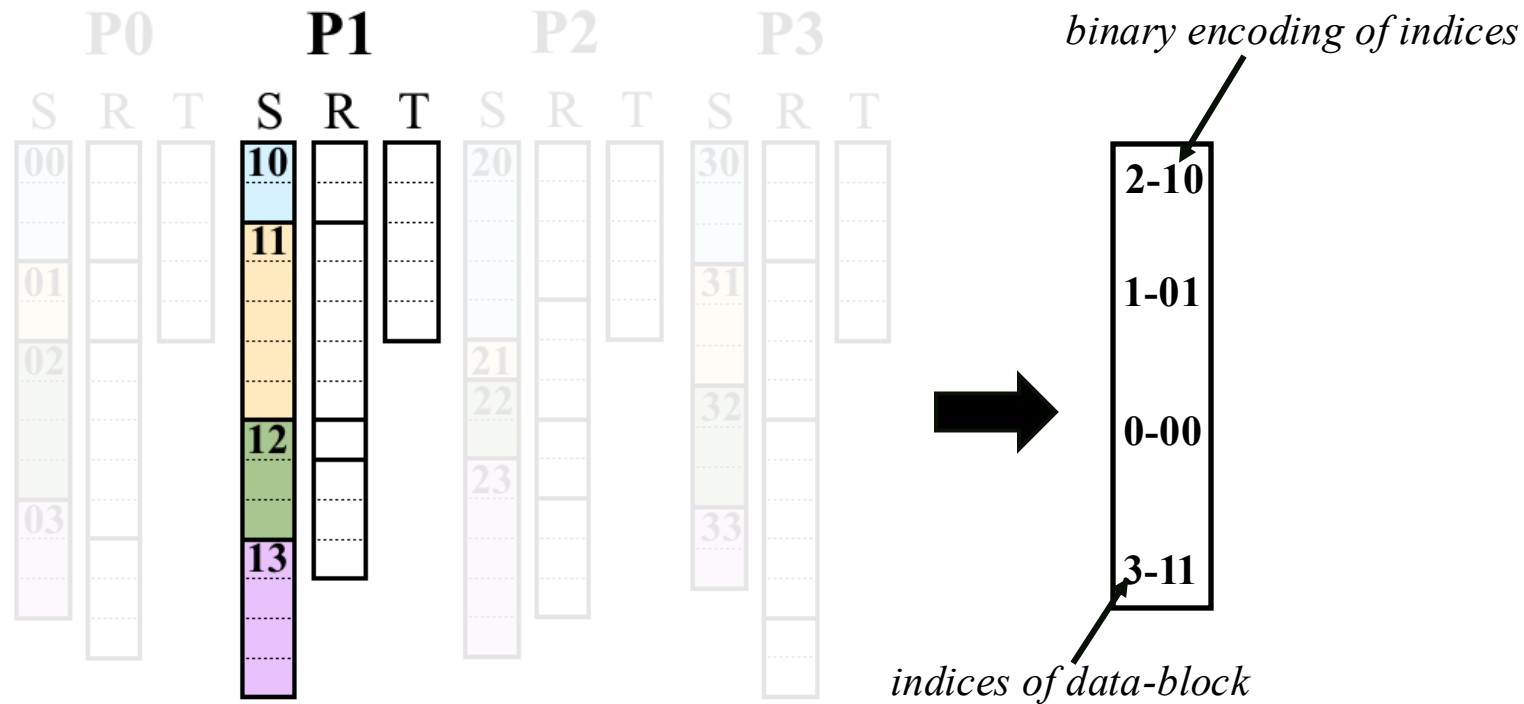
Initial data for 4 process

Example of the ParLogNa with $P = 4$ and $r = 2$.



Taking P1 for example

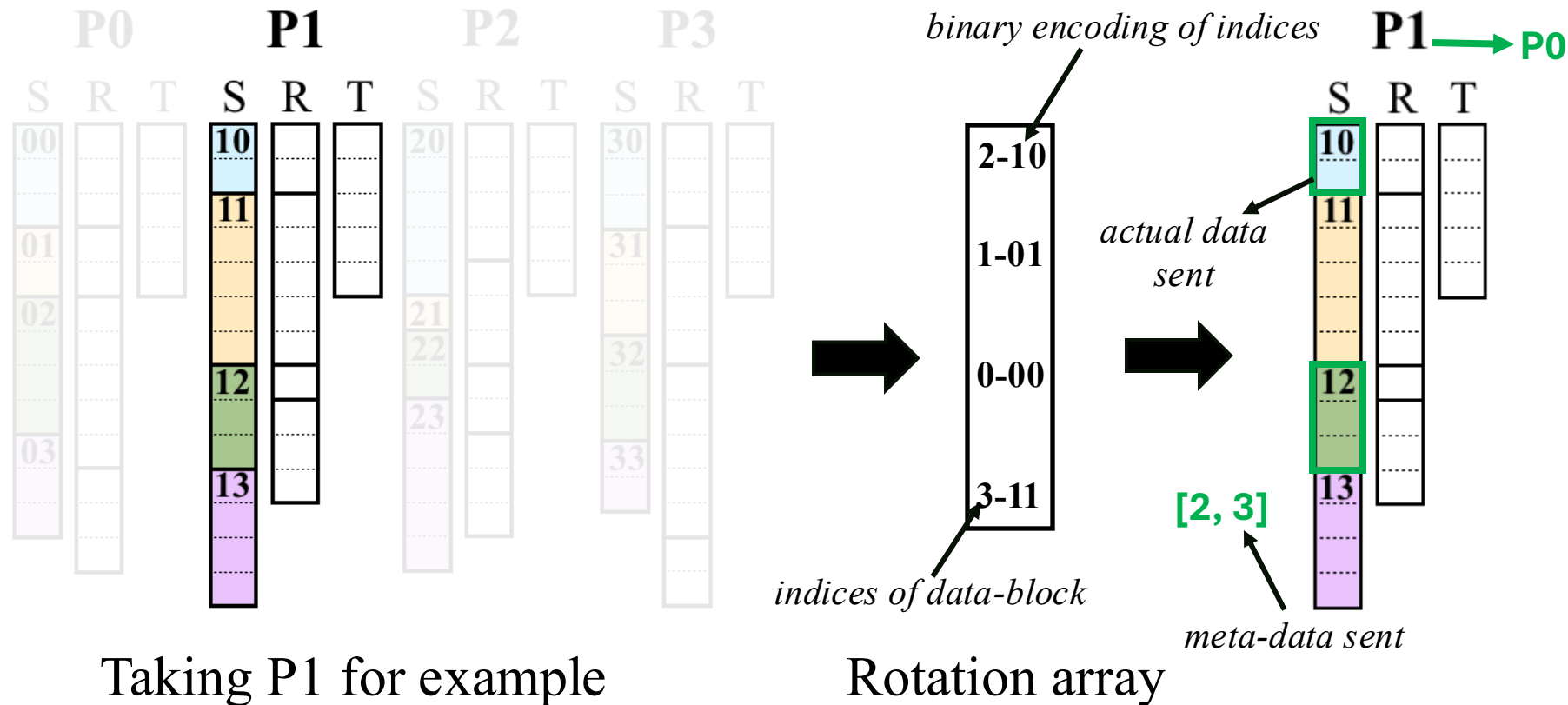
Example of the ParLogNa: Rotation Array and Binary Encoding



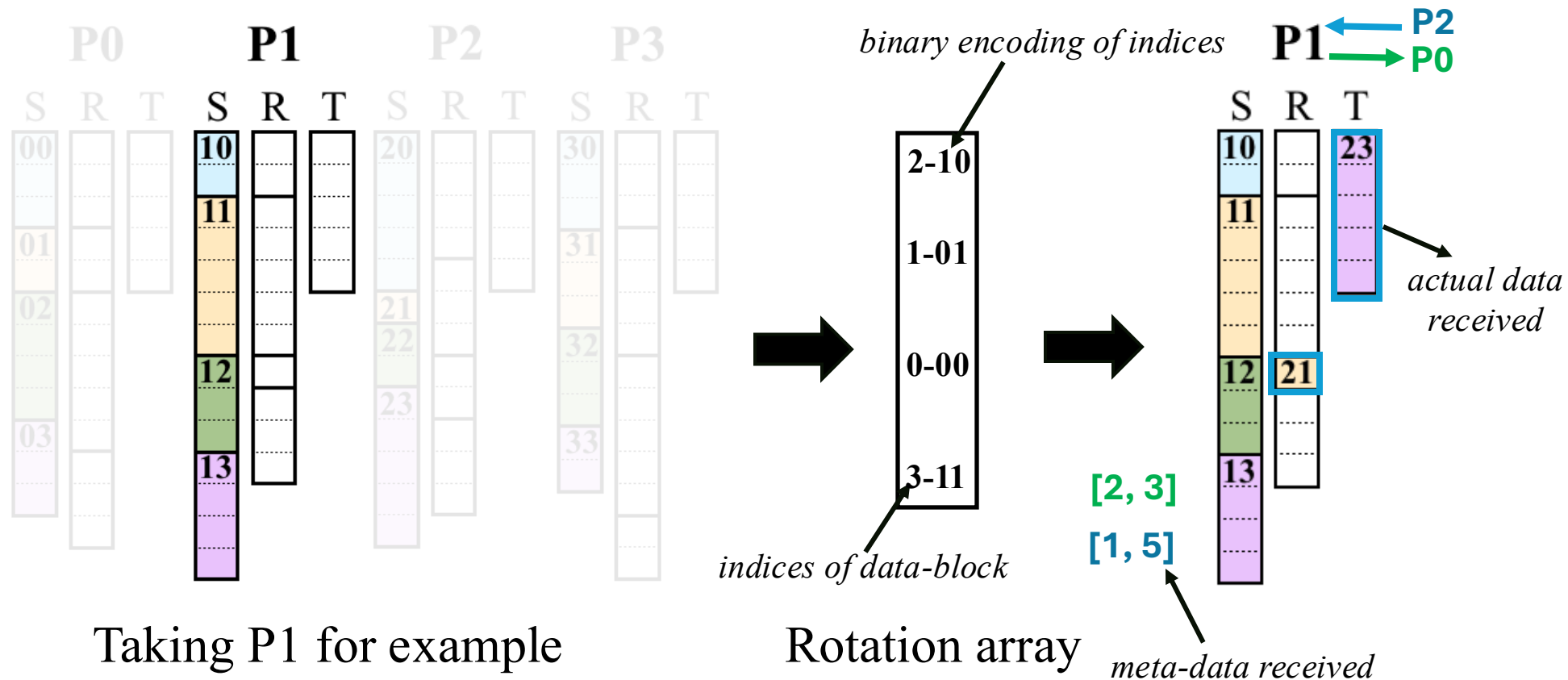
Taking P1 for example

Rotation array

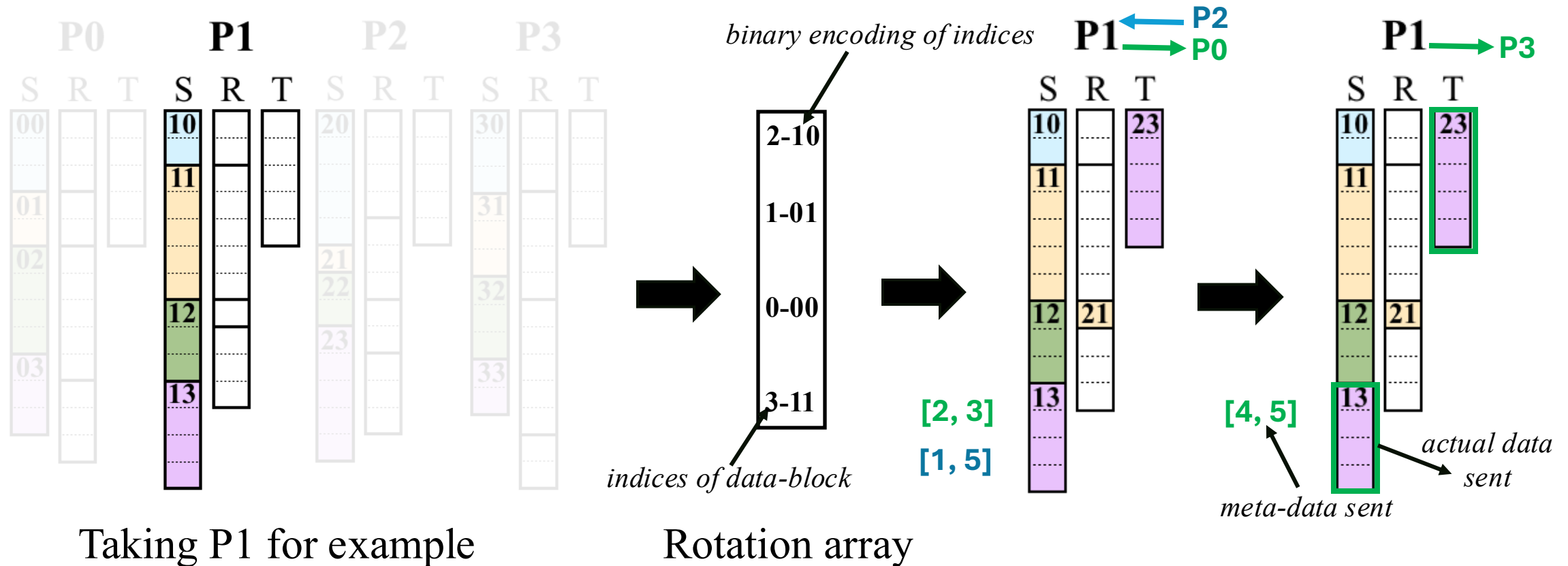
Example of the ParLogNa: Comm Round 0



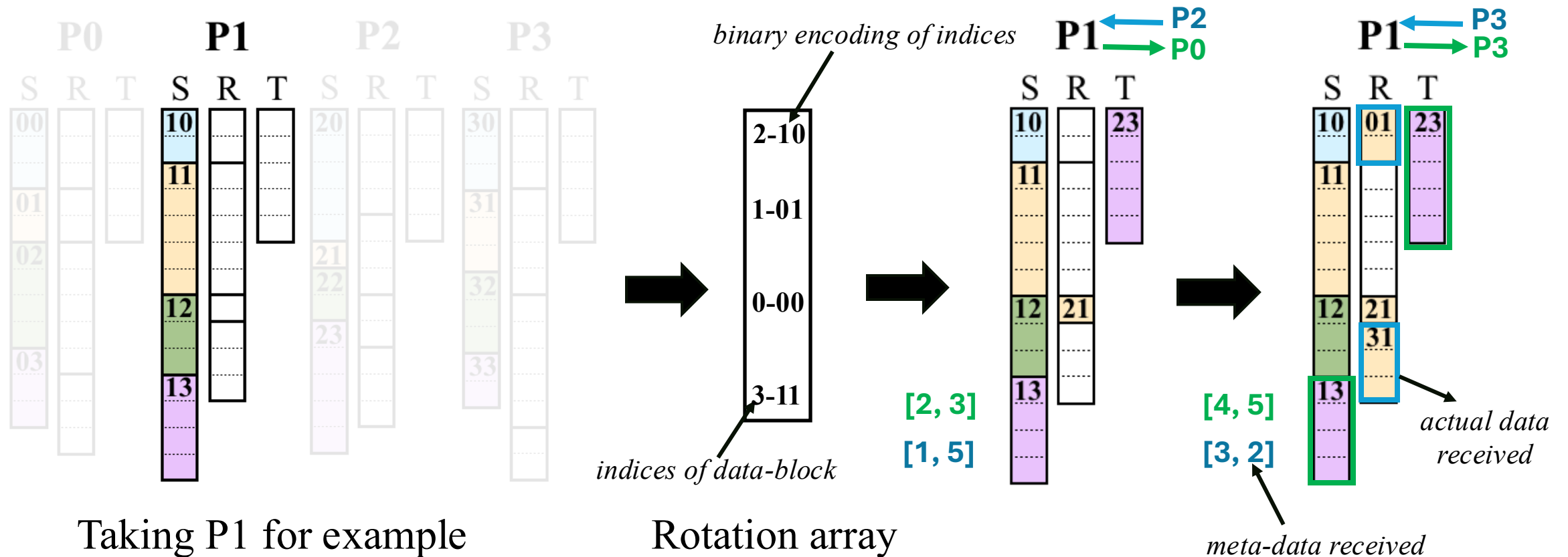
Example of the ParLogNa: Comm Round 0



Example of the ParLogNa: Comm Round 1



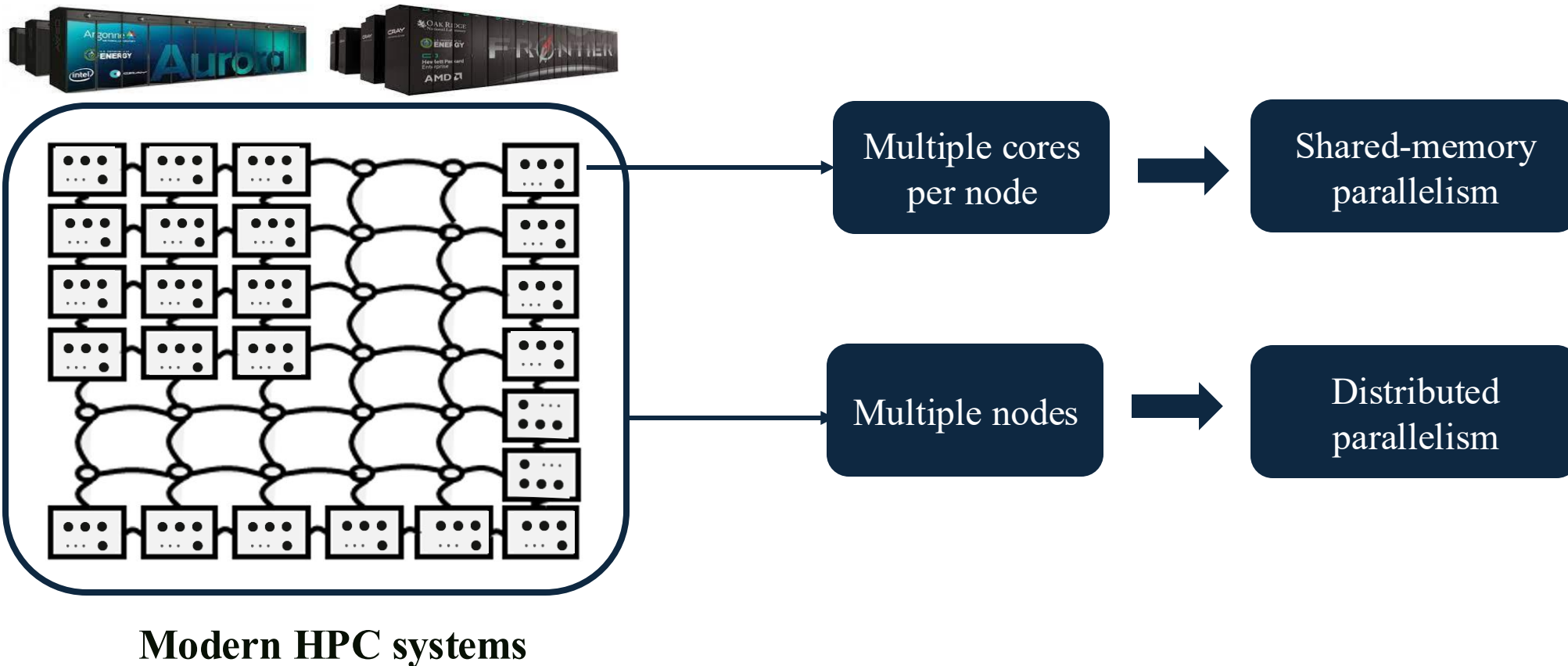
Example of the ParLogNa: Comm Round 1



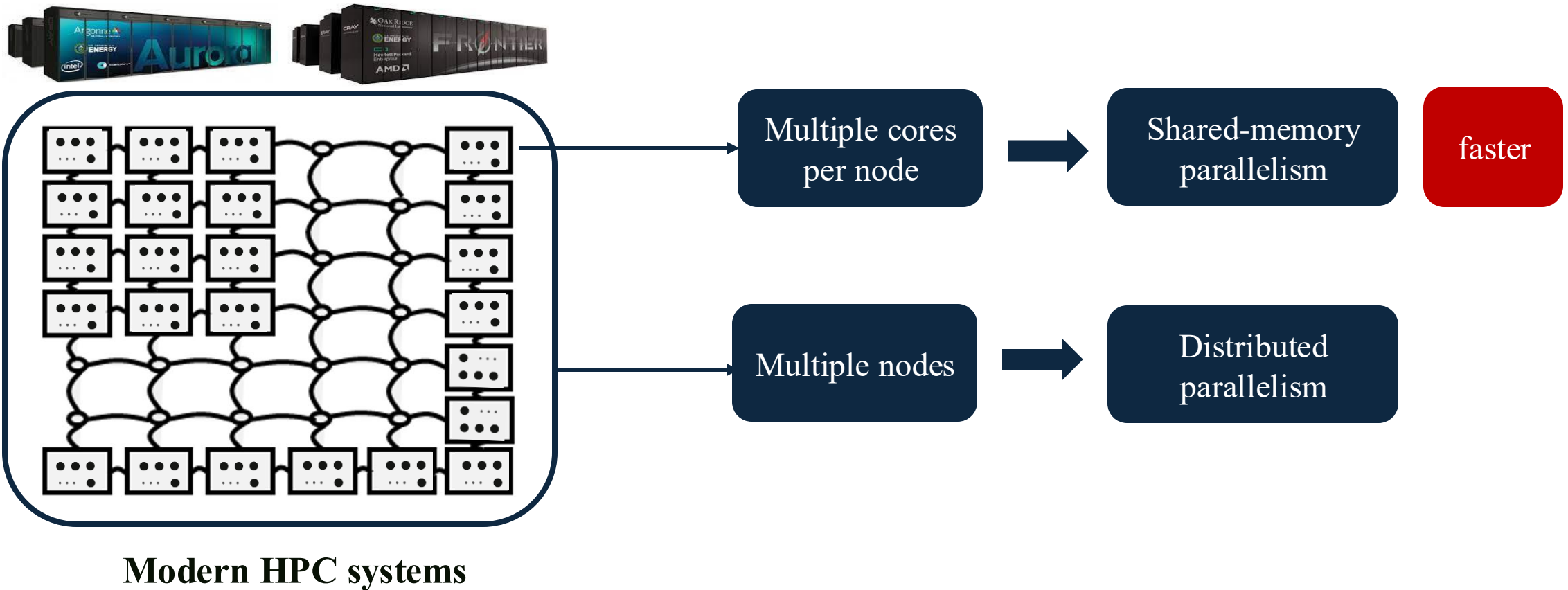
Agenda

- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- Challenges and Solutions
 - Tunability of Performance
 - Logarithmic Non-uniform All-to-all
- Parameterized Logarithmic Algorithm
- **Parameterized Linear Algorithm**
- Evaluation
- Application
- Conclusion

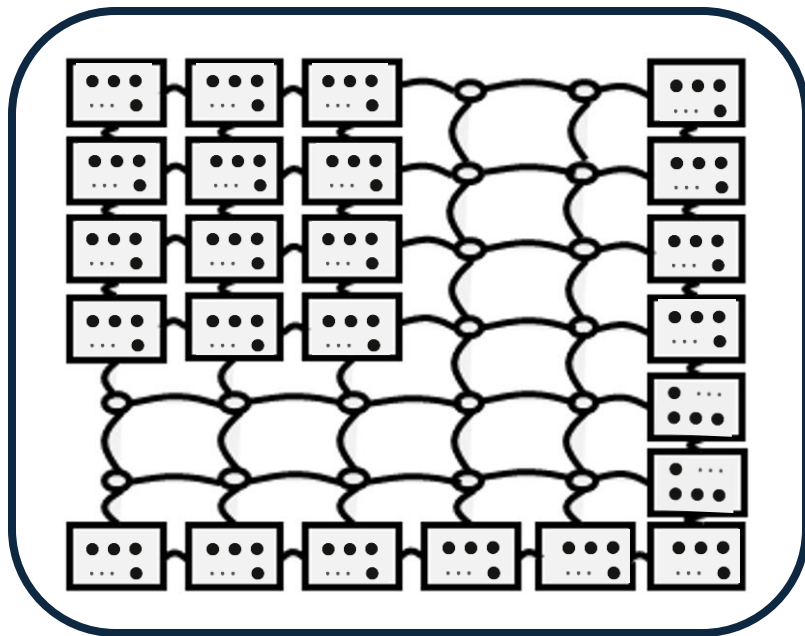
Architecture of Modern HPC Systems



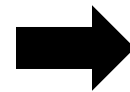
Architecture of Modern HPC Systems



Parameterized Linear Non-uniform All-to-all (ParLinNa)



Modern HPC systems



Latency-bound ←



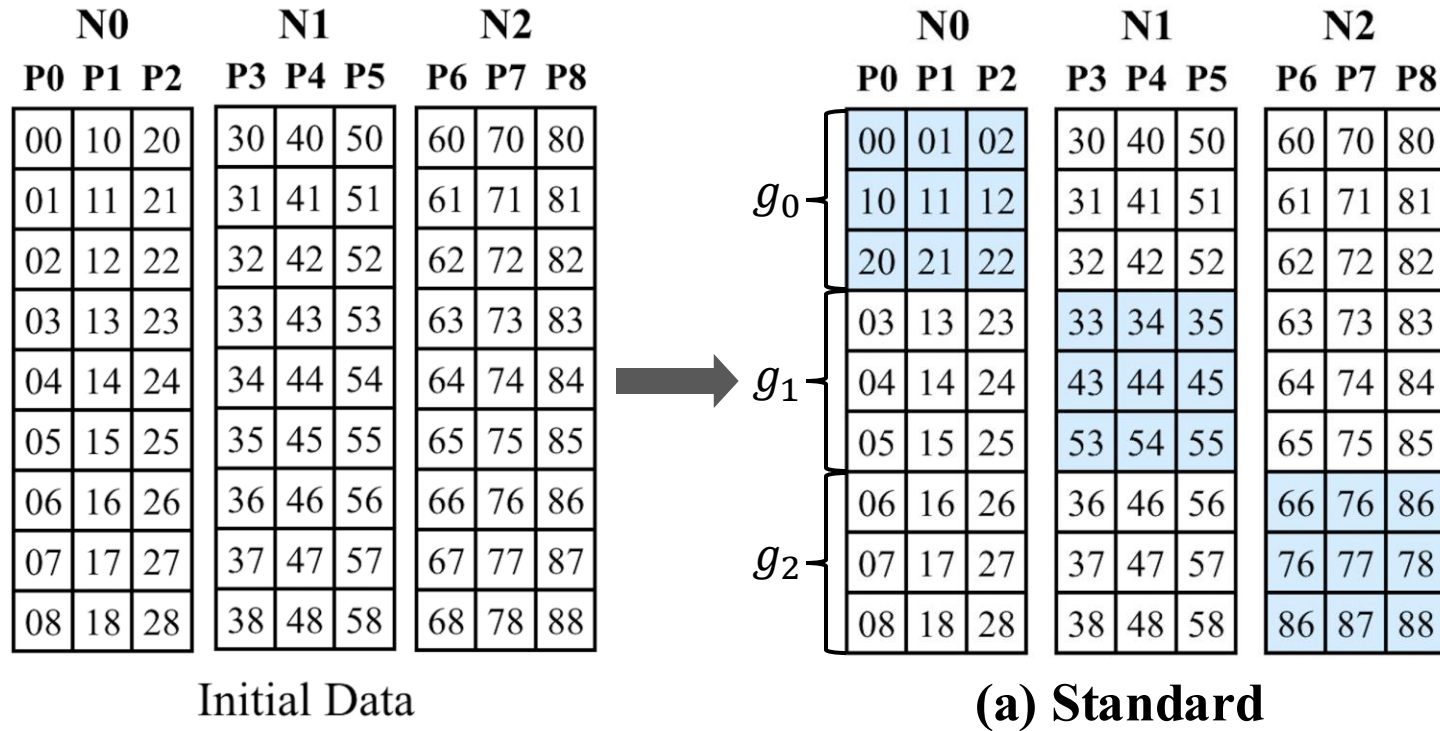
tunable radix



Bandwidth-bound ←

tunable batch_size

Intra-node Communication: Two Strategies



Intra-node Communication: Two Strategies

N0			N1			N2		
P0	P1	P2	P3	P4	P5	P6	P7	P8
00	10	20	30	40	50	60	70	80
01	11	21	31	41	51	61	71	81
02	12	22	32	42	52	62	72	82
03	13	23	33	43	53	63	73	83
04	14	24	34	44	54	64	74	84
05	15	25	35	45	55	65	75	85
06	16	26	36	46	56	66	76	86
07	17	27	37	47	57	67	77	87
08	18	28	38	48	58	68	78	88

Initial Data

N0			N1			N2		
P0	P1	P2	P3	P4	P5	P6	P7	P8
00	01	02	30	40	50	60	70	80
10	11	12	31	41	51	61	71	81
20	21	22	32	42	52	62	72	82
03	13	23	33	34	35	63	73	83
04	14	24	43	44	45	64	74	84
05	15	25	53	54	55	65	75	85
06	16	26	36	46	56	66	76	86
07	17	27	37	47	57	76	77	78
08	18	28	38	48	58	86	87	88

g_0

g_1

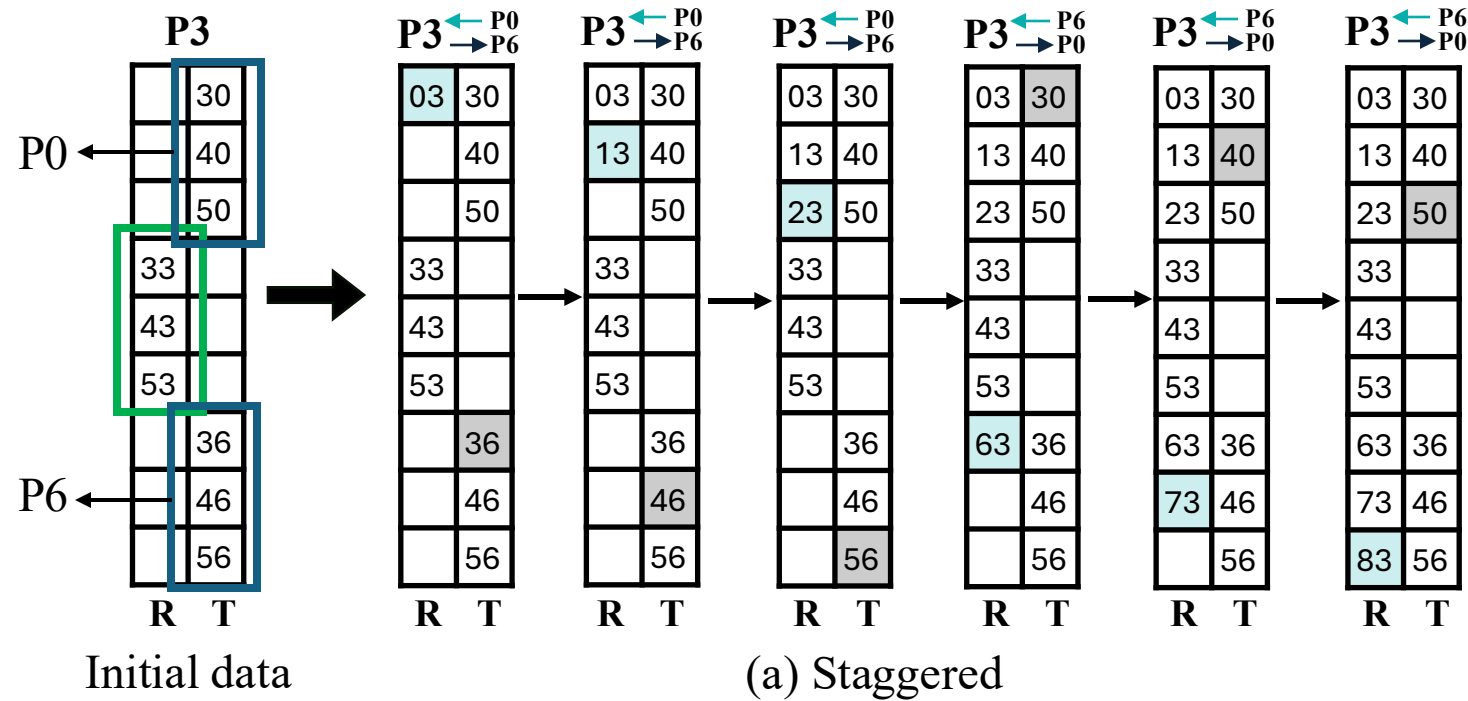
g_2

(a) Standard

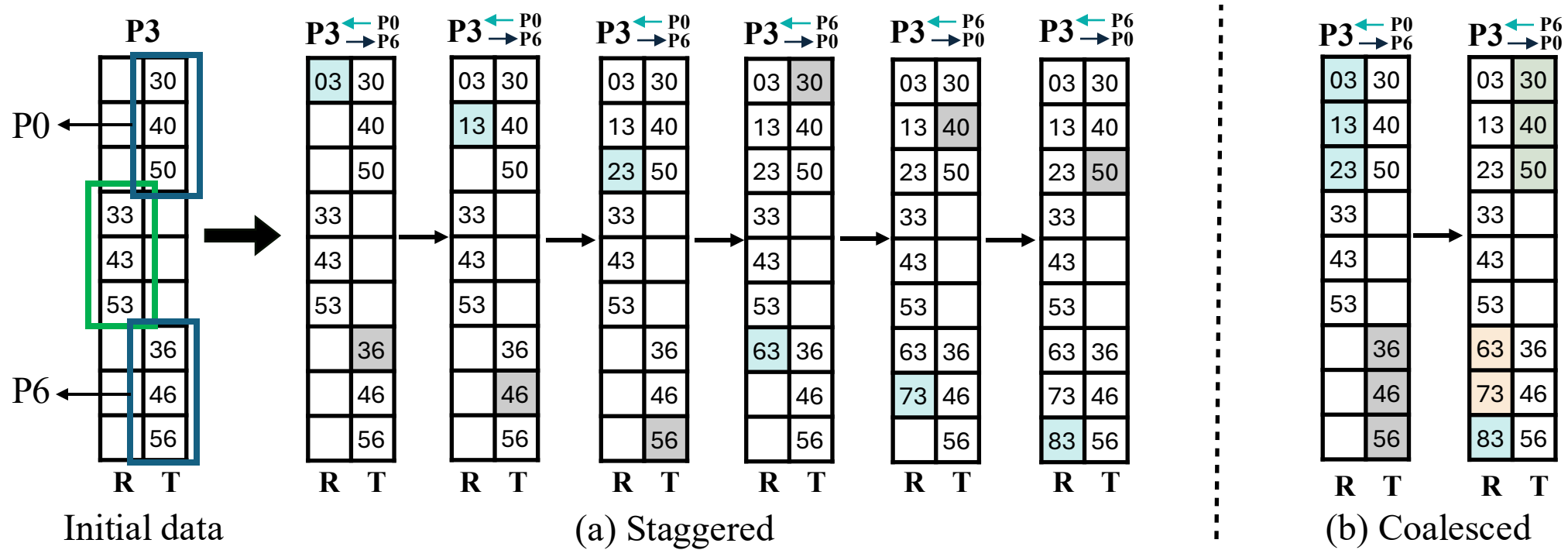
N0			N1			N2		
P0	P1	P2	P3	P4	P5	P6	P7	P8
00	01	02	30	31	32	60	61	62
10	11	12	40	41	42	70	71	72
20	21	22	50	51	52	80	81	82
03	04	05	33	34	35	63	64	65
13	14	15	43	44	45	73	74	75
23	24	25	53	54	55	83	84	85
06	07	08	36	37	38	66	67	68
16	17	18	46	47	48	76	77	78
26	27	28	56	57	58	86	87	88

(b) Ours

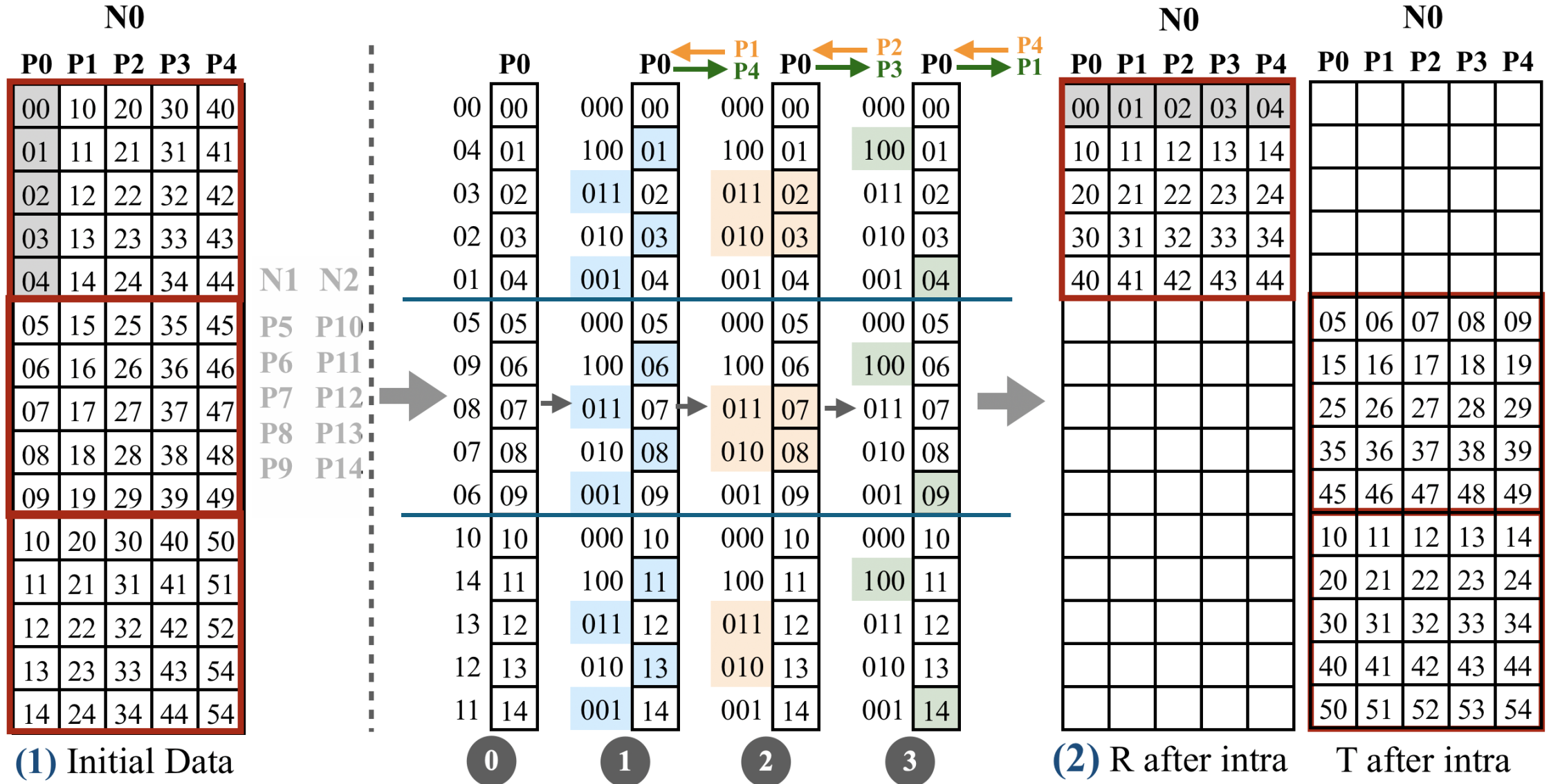
Inter-node Communication: Two Patterns



Inter-node Communication: Two Patterns



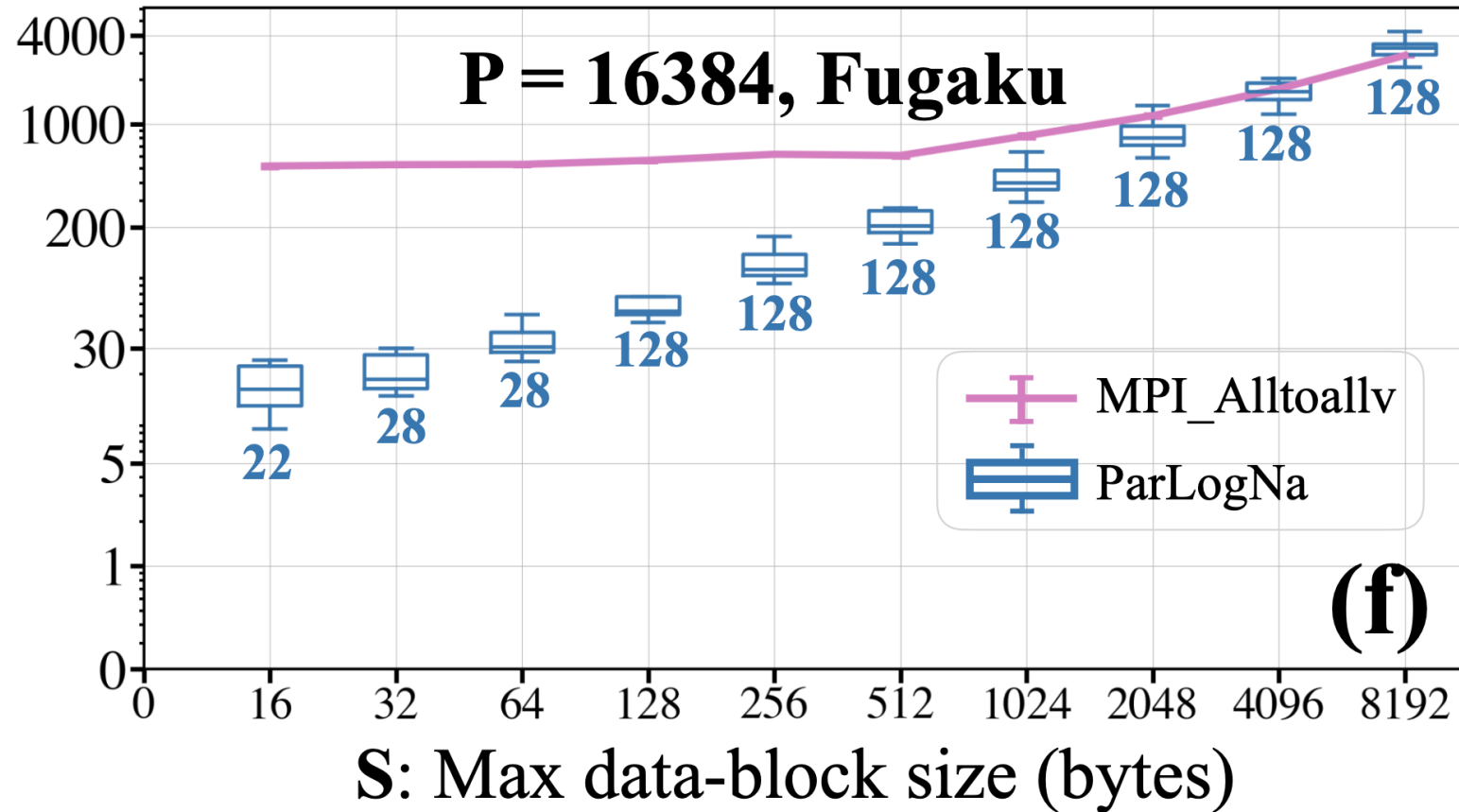
Intra-node Communication Example



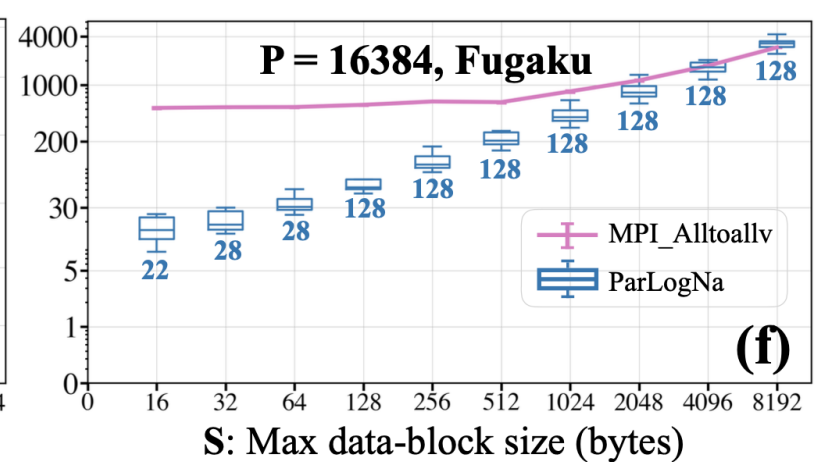
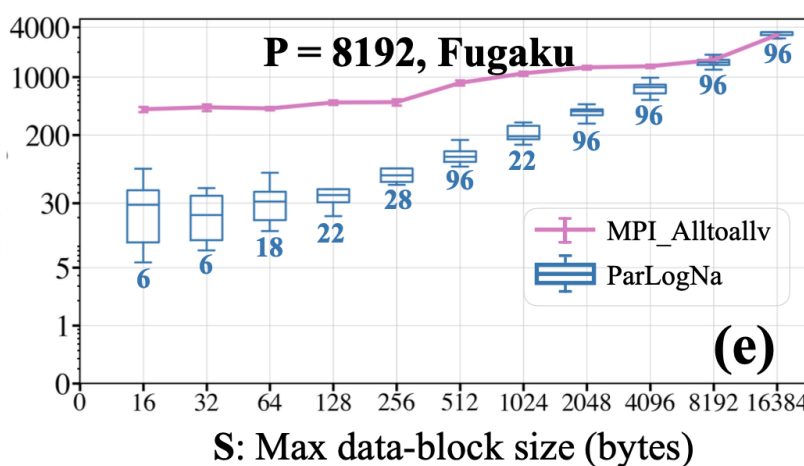
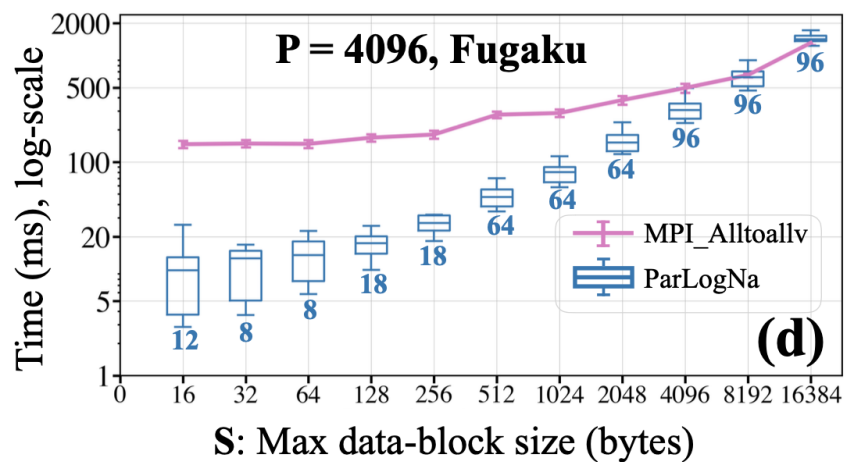
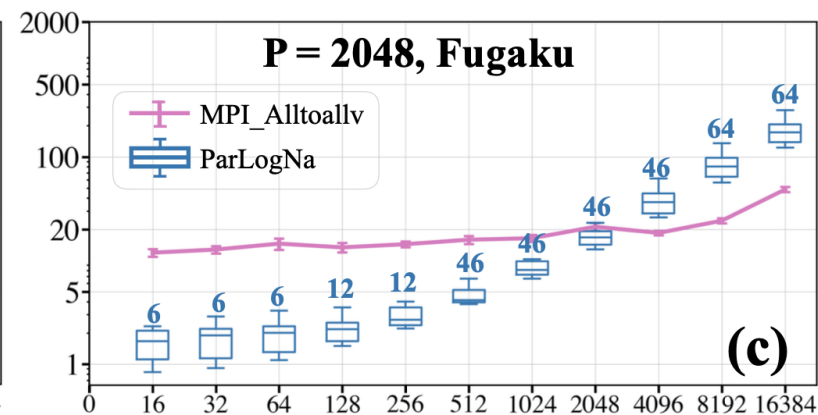
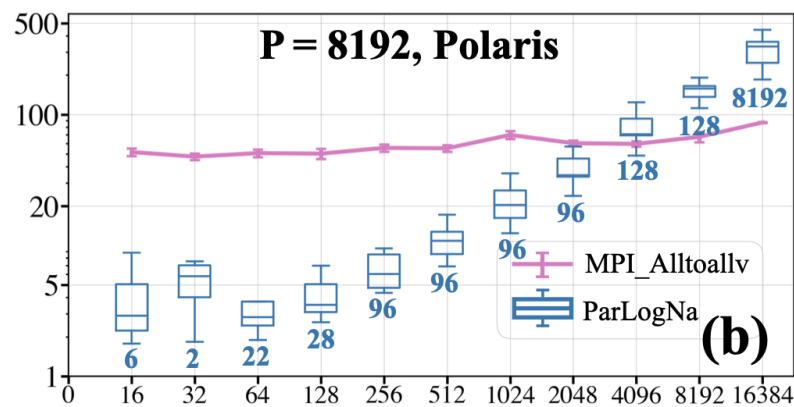
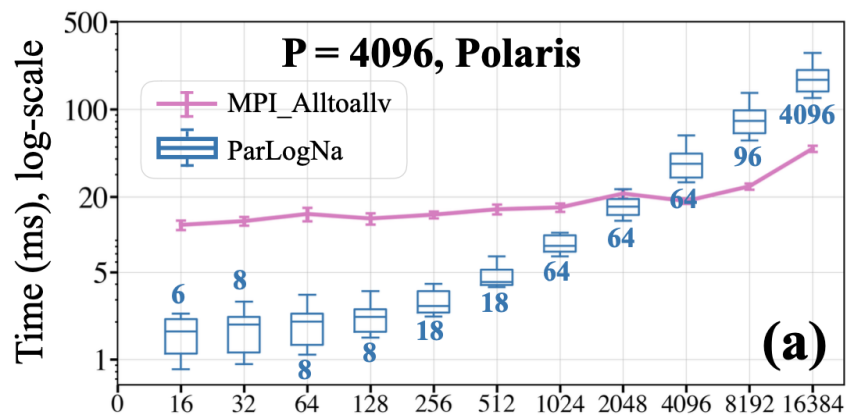
Agenda

- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- Challenges and Solutions
 - Tunability of Performance
 - Logarithmic Non-uniform All-to-all
- Parameterized Logarithmic Algorithm
- Parameterized Linear Algorithm
- **Evaluation**
- Application
- Conclusion

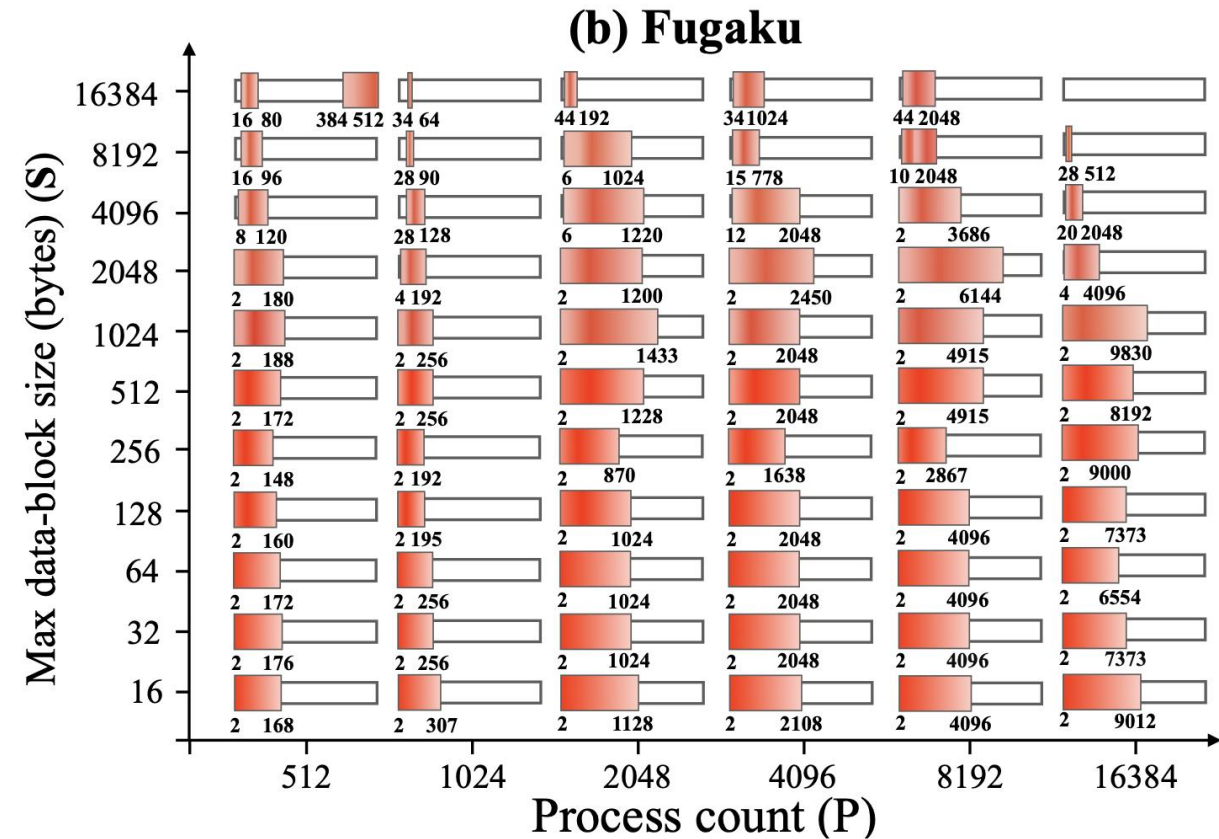
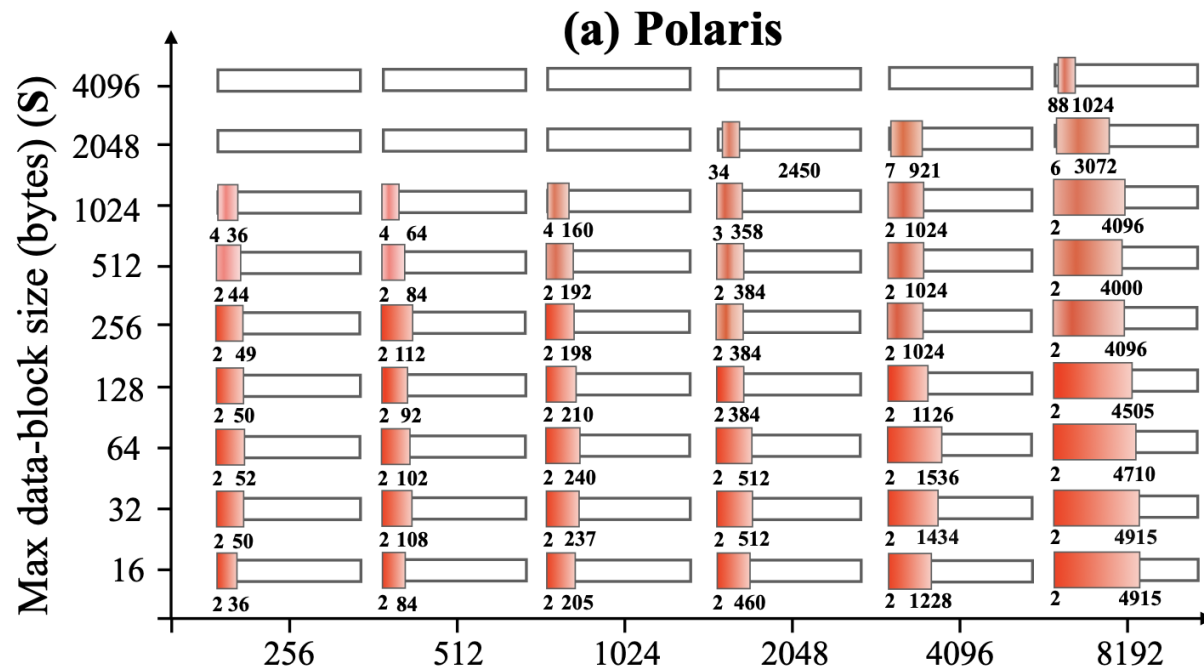
Comparing ParLogNa with MPI_Alltoallv



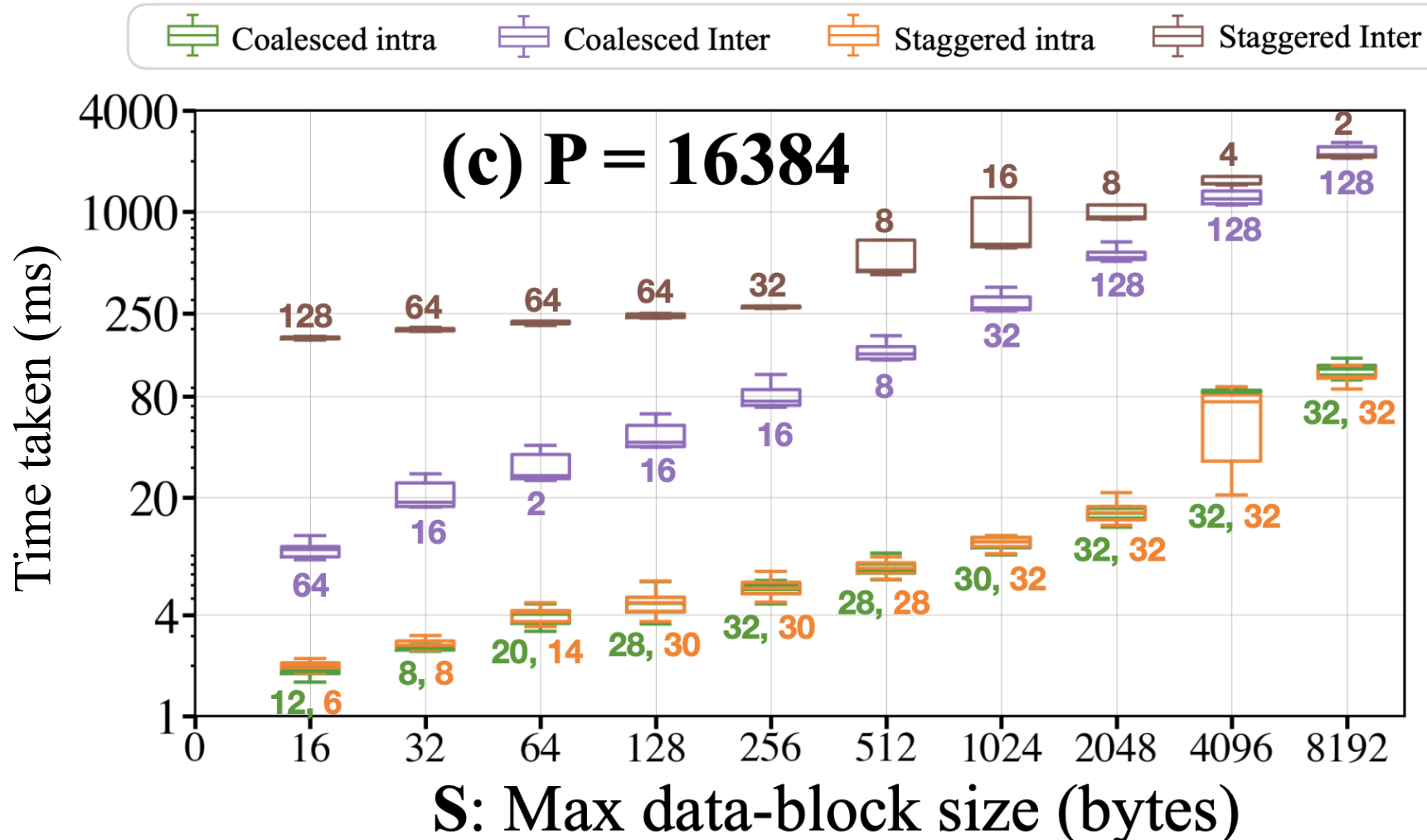
Comparing ParLogNa with MPI_Alltoallv



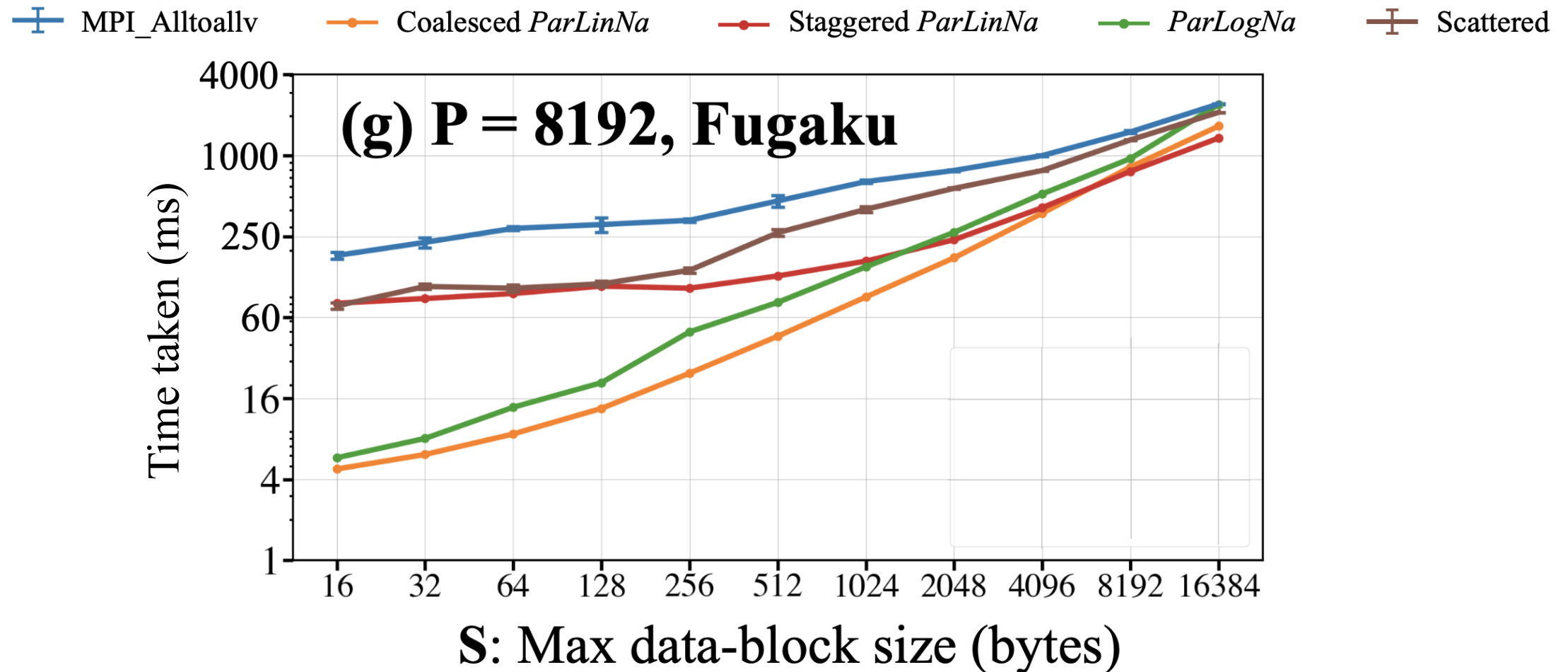
Ranges of radix where ParLogNa outperforms MPI_Alltoallv



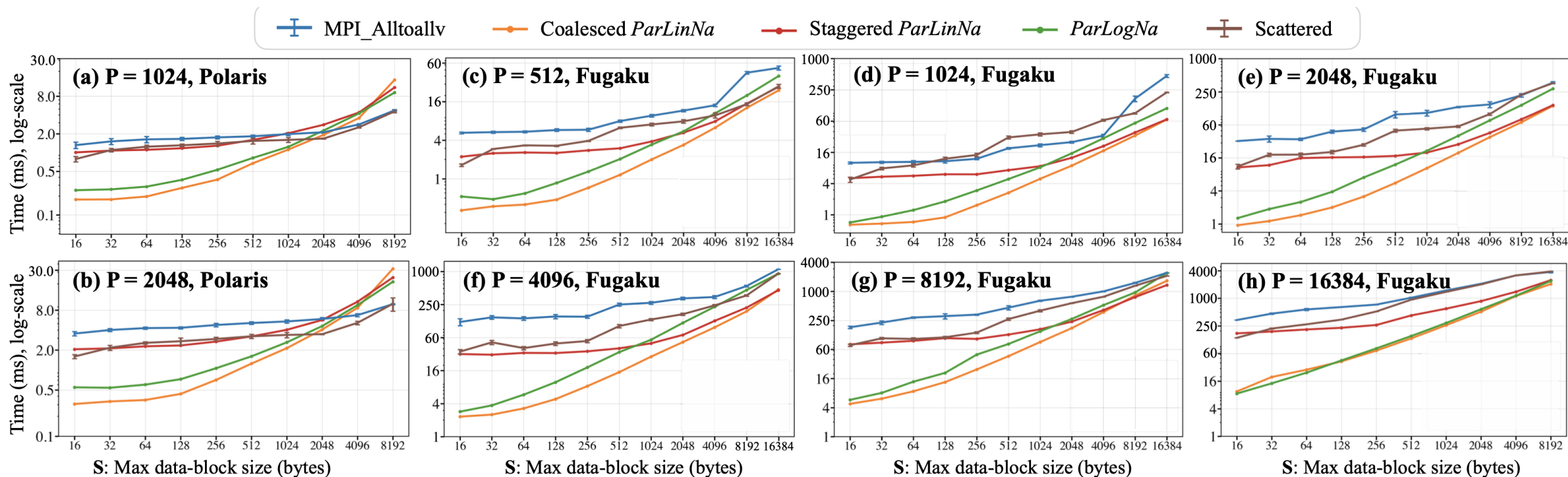
Comparing Coalesced and Staggered ParLinNa



Comparing Proposed Algorithms with the Top-performing Benchmark



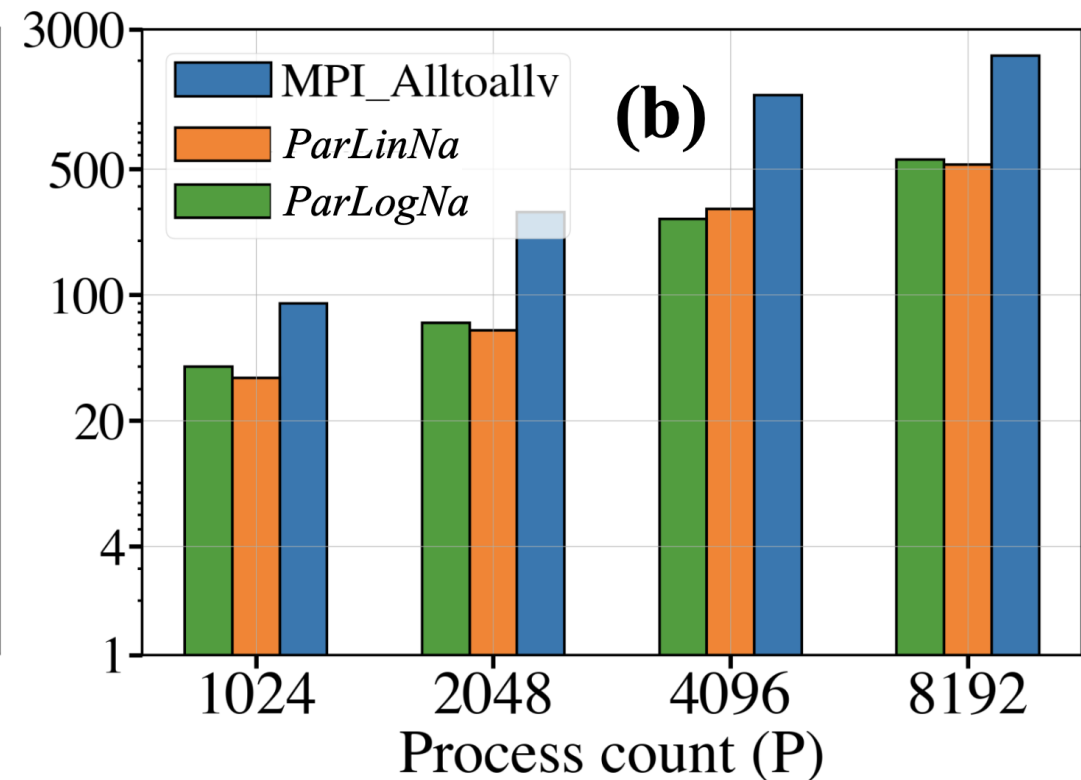
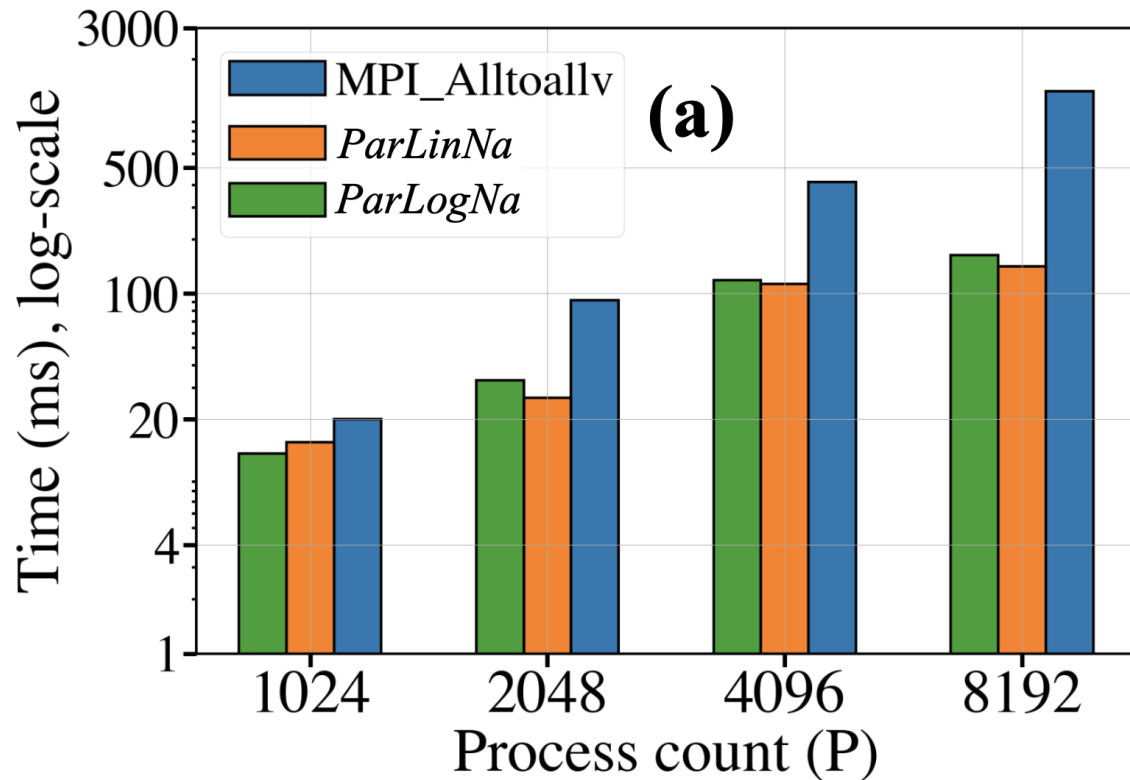
Comparing Proposed Algorithms with the Top-performing Benchmark



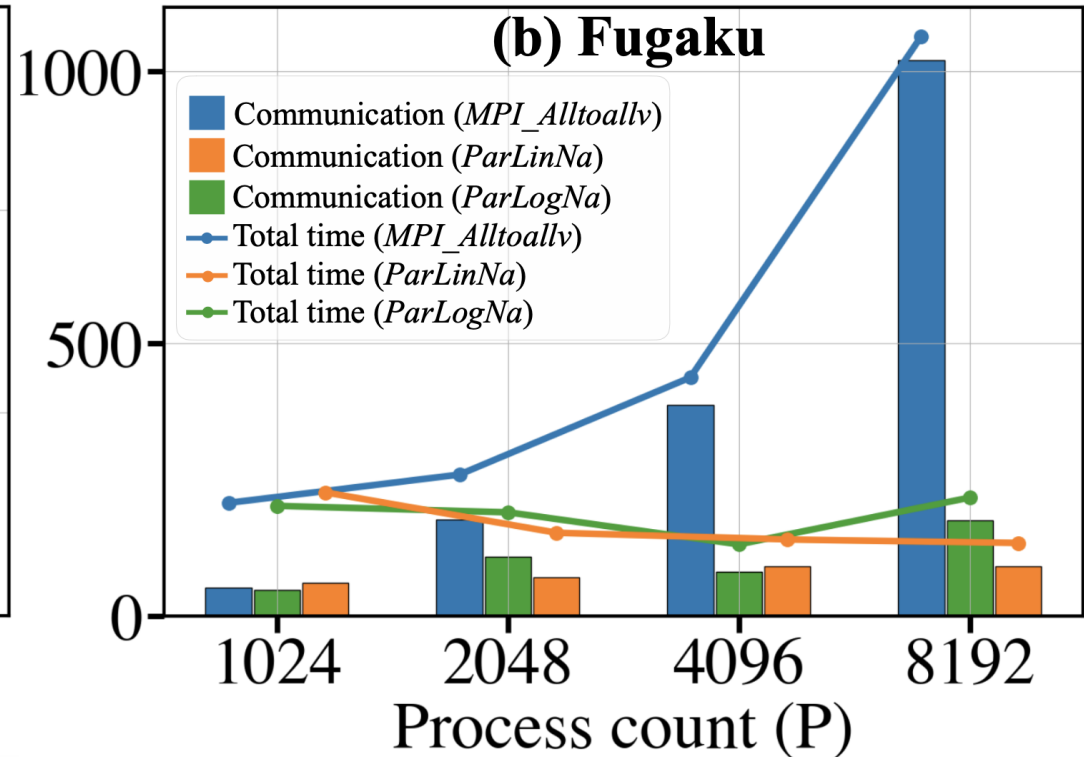
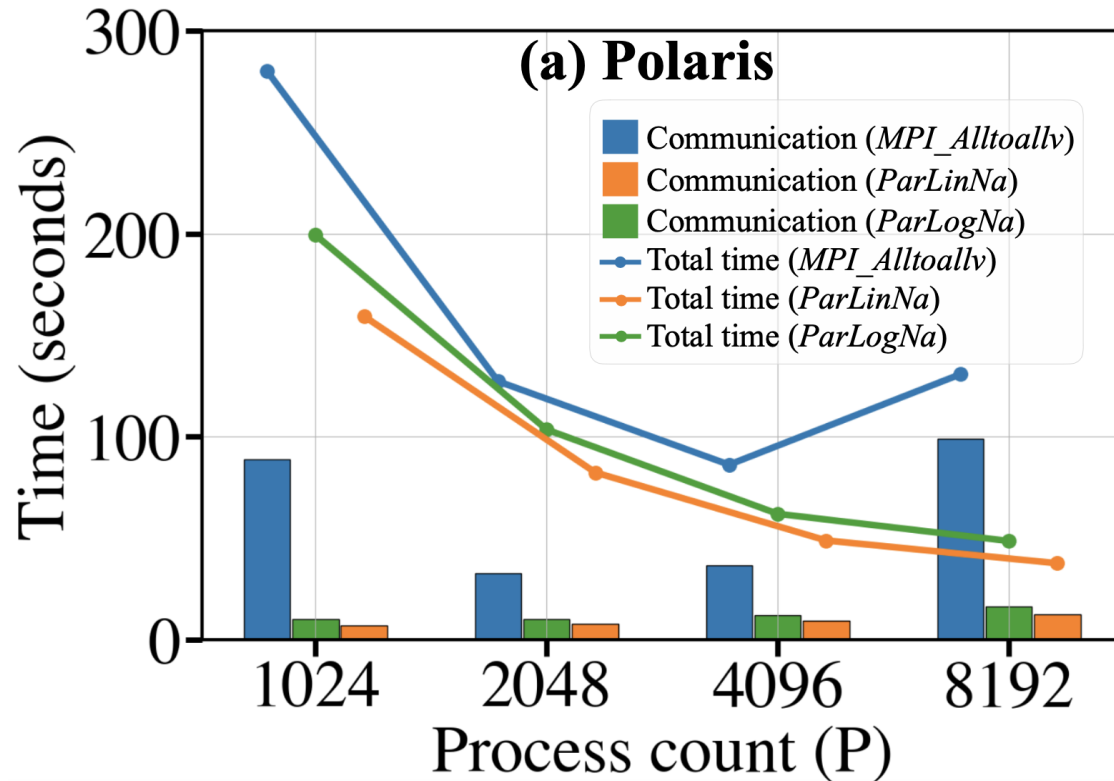
Agenda

- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- Challenges and Solutions
 - Tunability of Performance
 - Logarithmic Non-uniform All-to-all
- Parameterized Logarithmic Algorithm
- Parameterized Linear Algorithm
- Evaluation
- **Application**
- Conclusion

Performance of Applying Our Algorithms to FFT



Performance of Applying Our Algorithms to Path Finding



Agenda

- Introduction and Background
 - Inter-process Communication
 - All-to-all Data Exchange
 - Standard MPI All-to-all Implementations
- Challenges and Solutions
 - Tunability of Performance
 - Logarithmic Non-uniform All-to-all
- Parameterized Logarithmic Algorithm
- Parameterized Linear Algorithm
- Evaluation
- Application
- Conclusion

Conclusion

- We have presented algorithms (ParLogNa and ParLinNa) that can improve an important class of collective functions.
- The work can help vendors further optimize their collective routines.
- The work can have a direct impact on a range of applications that uses all to all communication.

Thank you!