ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE
ASP-DAC 2024
January 22-25, 2024
Incheon Songdo Convensia, South Korea
29th Asia and South Pacific Design Automation Conference

Tokyo Tech

ARTIC

Sparse-Sparse Matrix Multiplication Accelerator on FPGA featuring Distribute-Merge Product Dataflow

Yuta Nagahara, Jiale Yan, Kazushi Kawamura,
Masato Motomura, and Thiem Van Chu
Tokyo Institute of Technology, Japan

Incheon, January 25, 2024

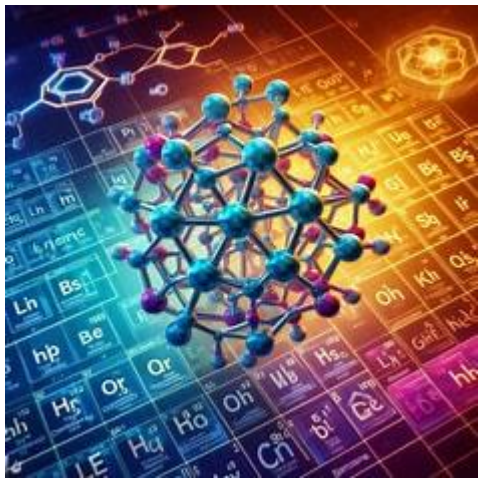
SpMSpM (Sparse-Sparse Matrix Multiplication)

SpMSpM is applied to various domains

Big data analysis

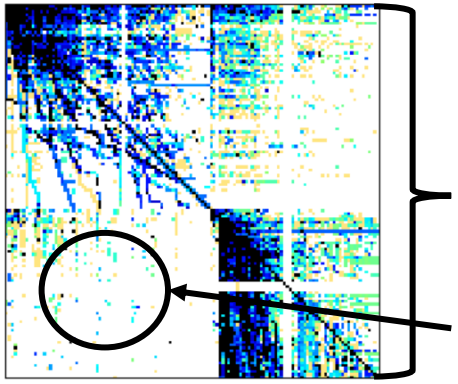
Graph processing

Computational Simulation



Sparse Matrix

Big data size and randomly distributed



# of rows: several millions

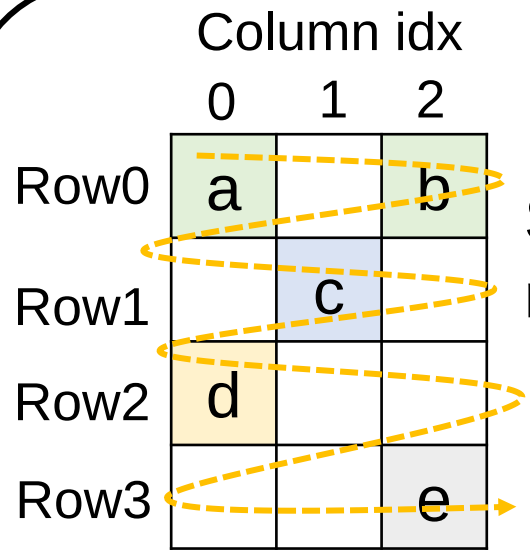
sparse

SpMSpM Processing

(**C**ompressed **S**parse **R**ow/**C**olumn)

- Computed in a compressed format (CSR, CSC, etc.)

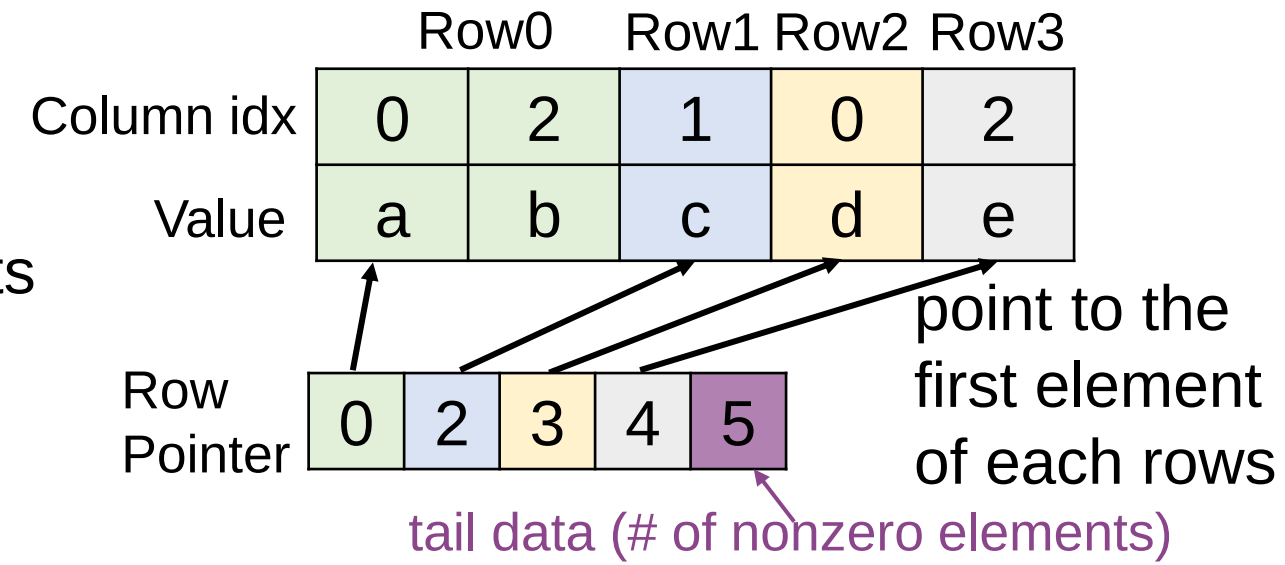
CSR format



Store only nonzero elements

Row direction compression

sparse matrix

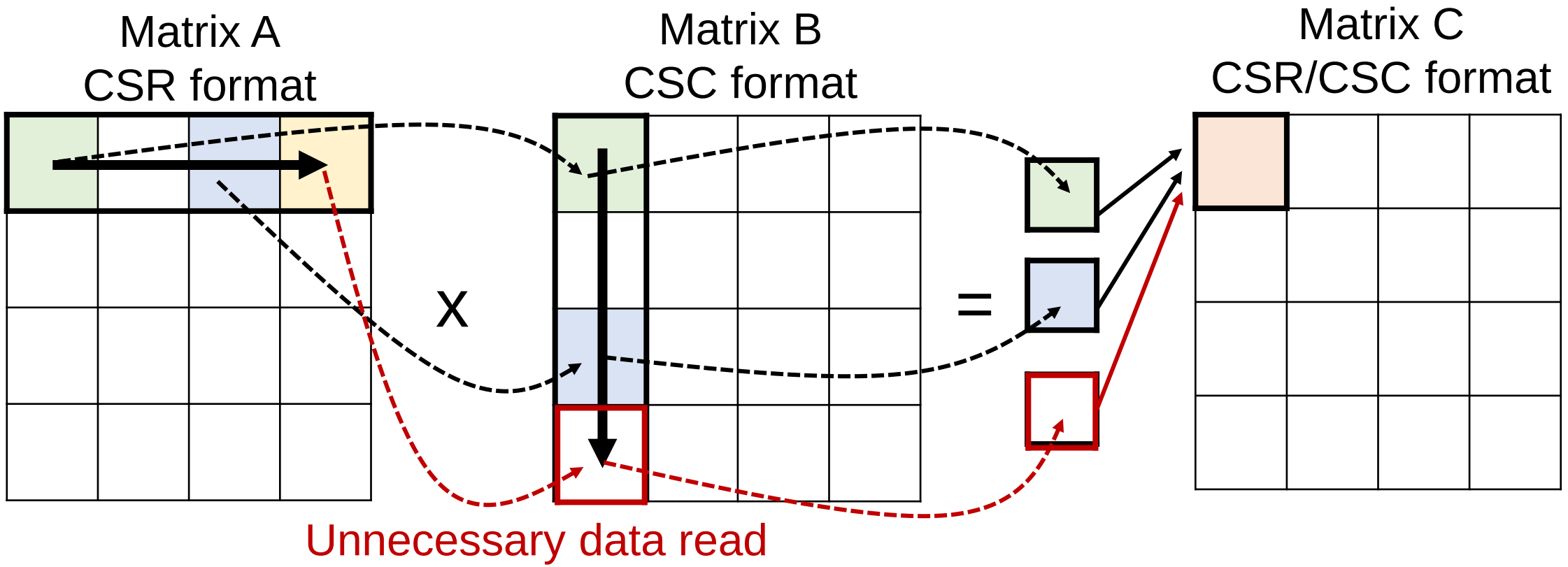


- Indirect access using pointers
- Random access is difficult

- Basic dataflows
  - Inner Product (IP), Outer Product (OP) and Gustavson's Product (GP)

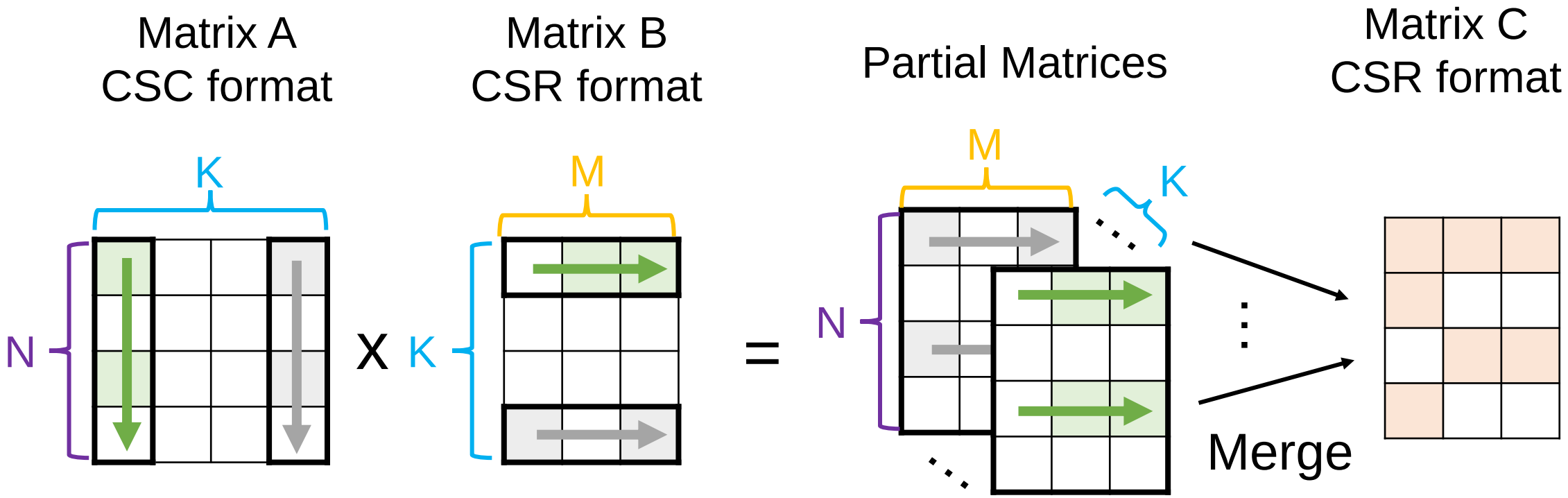
Inner Product (SIGMA [HPCA'20])

- Sequential data access → index matching



Performance degrades as matrix density becomes sparser

Outer Product (OUTERSPACE [HPCA'18])

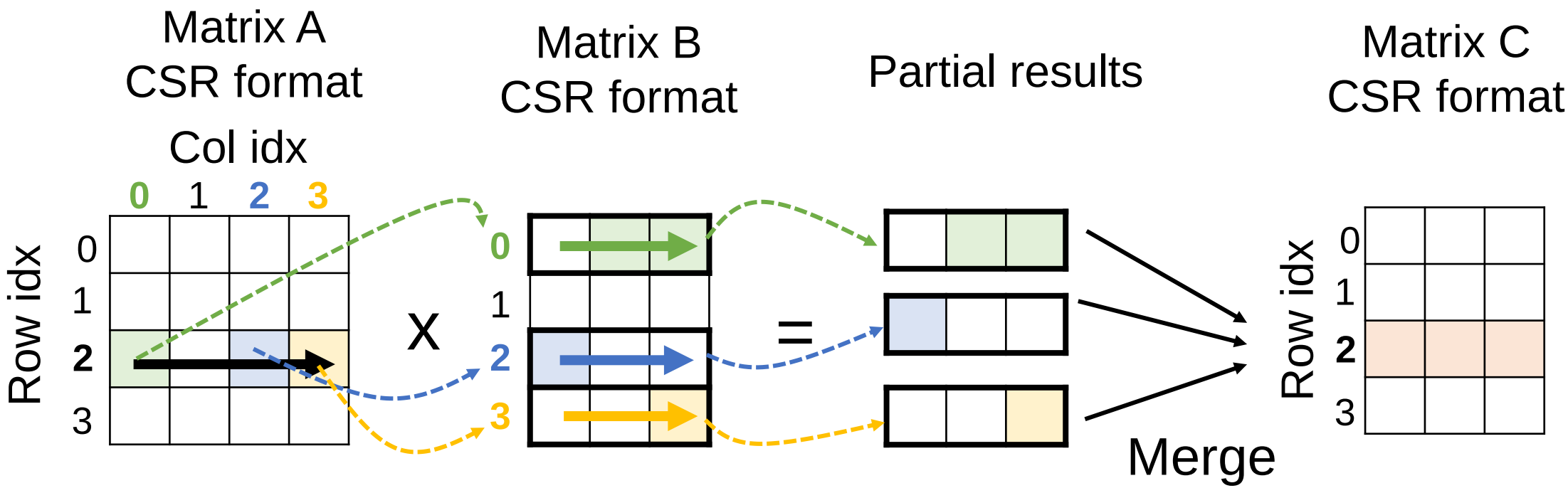


No redundant data read

Larger matrix leads to heavier merge process

Gustavson's Product (Gamma [ASPLOS'21])

- Compute Matrix C row-by-row

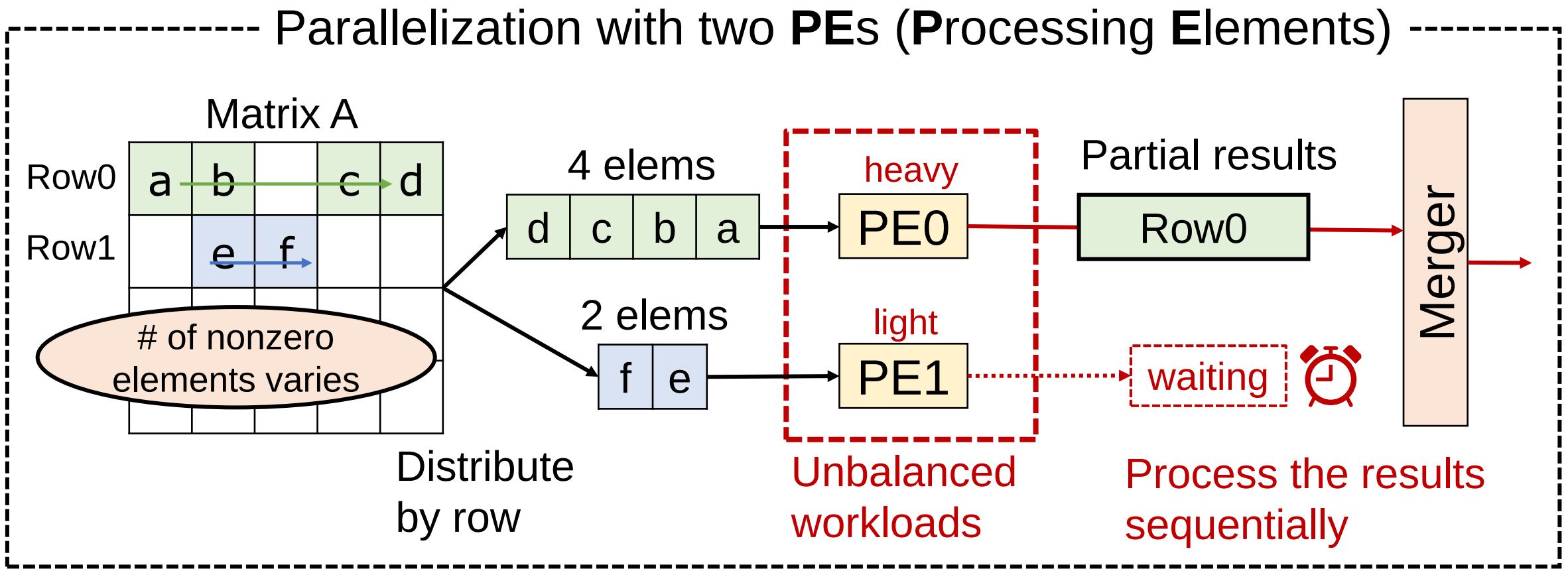


No redundant data read

Merge of row vectors
→ Reduction of merge workload

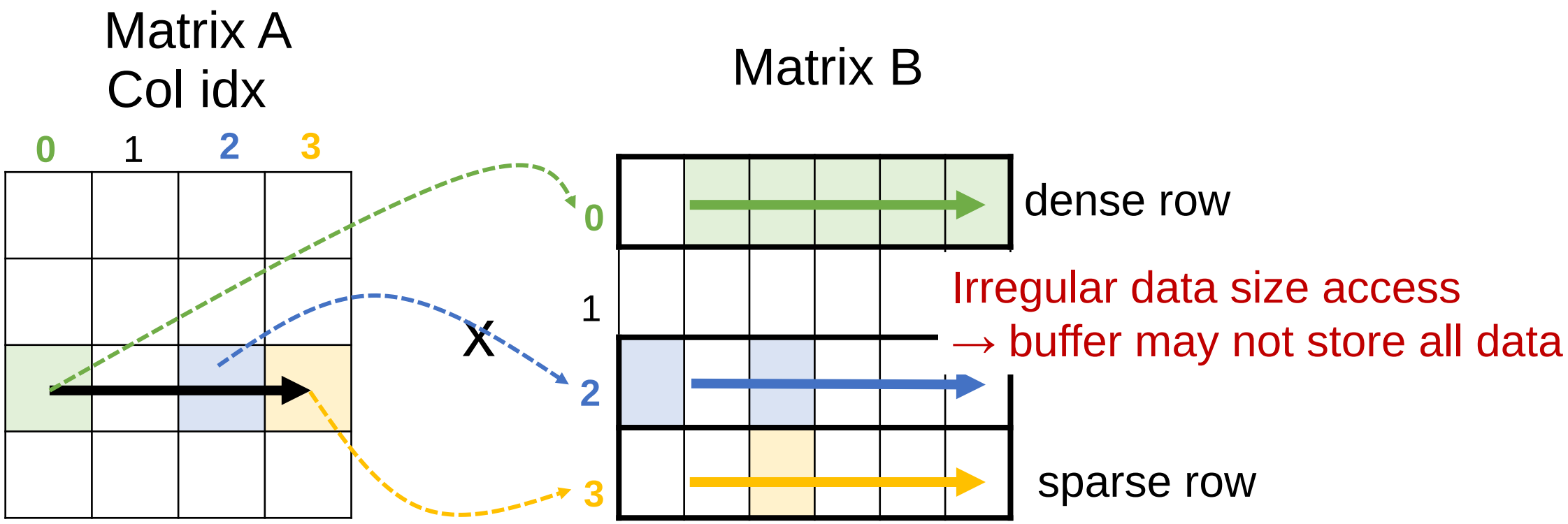
Gustavson's Product: Challenge 1

- Difficulty in efficiently parallelizing the computation
  - due to the variability of the number of nonzero elements from row to row



Gustavson's Product: Challenge 2

- Frequent random and irregular accesses to the rows of Matrix B



The access order depends on the col idx (0, 2, 3, ...)

Random access
→ increase data traffic and long access latency

Conventional works and Our Proposal

Inner Product
- massive unnecessary data read (index matching)

No redundant data read

Outer Product
- heavy merge process

Row-by-Row merge

Gustavson Product
- unbalanced workload per row
- random and irregular access to Matrix B data

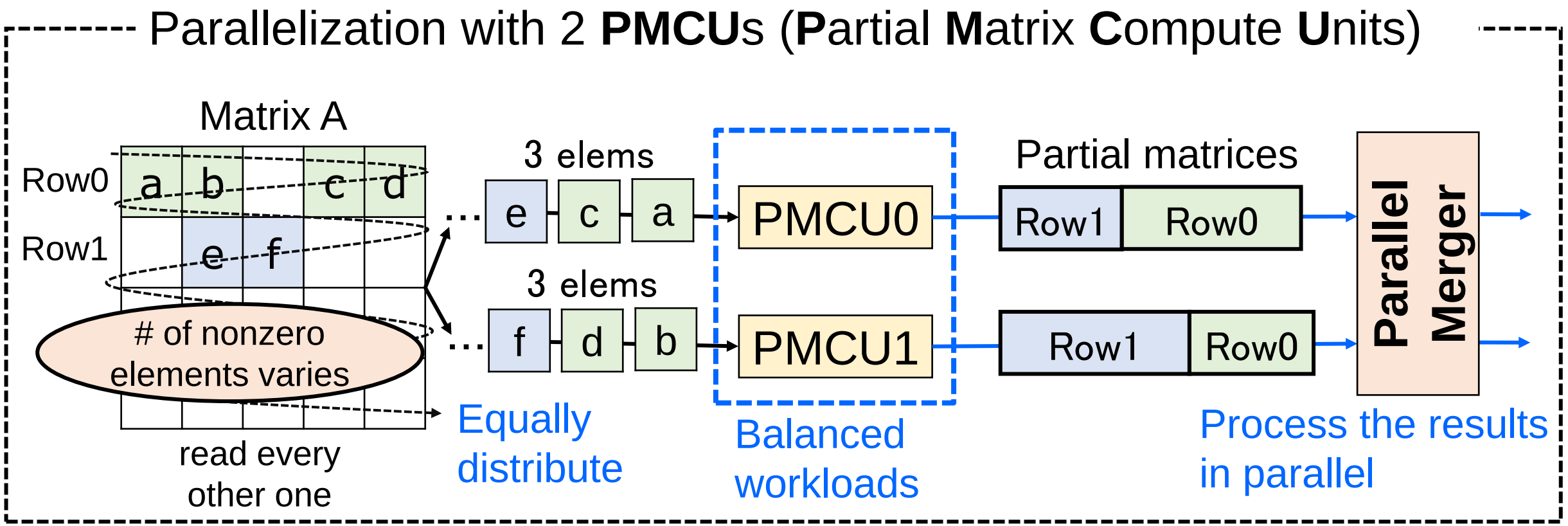
Our Proposal

Idea 1 : Distribute-Merge based SpMSpM dataflow
- **Equally distribute workloads**
- **Parallel merge method**

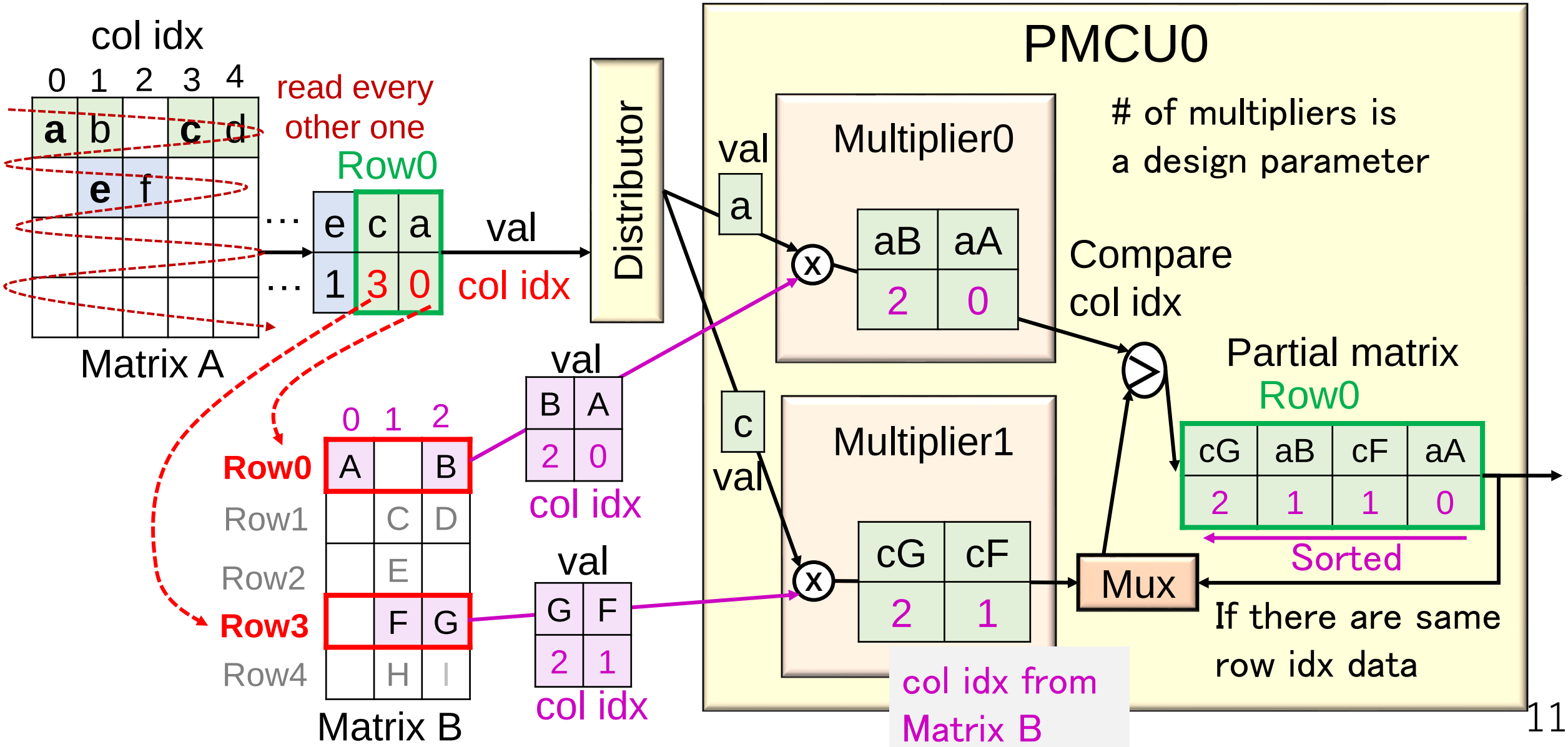
Idea 2 : Efficiently designed input data loader
- **Reuse Matrix B data to reduce memory traffic**
- **Separate read requests for irregular data size**

Idea 1: Distribute-Merge based SpMSpM dataflow

- DMP (Distribute-Merge Product) dataflow
  - **Equally distribute Matrix A elements to PEs**

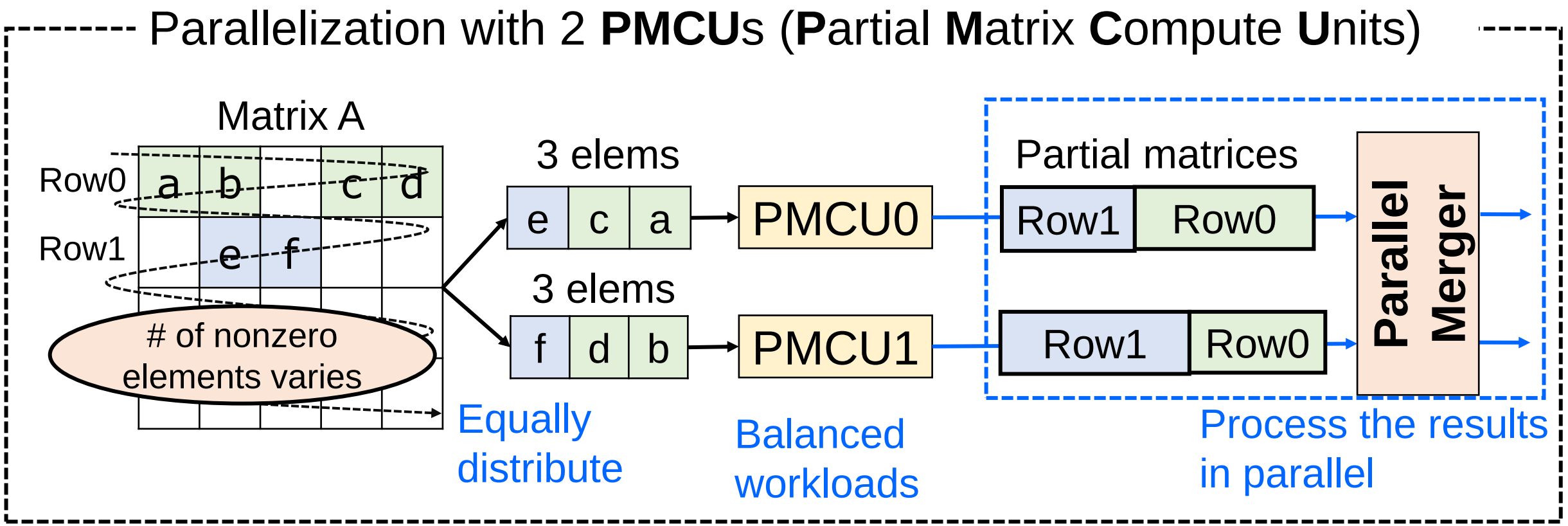


PMCU (Partial Matrix Compute Unit)



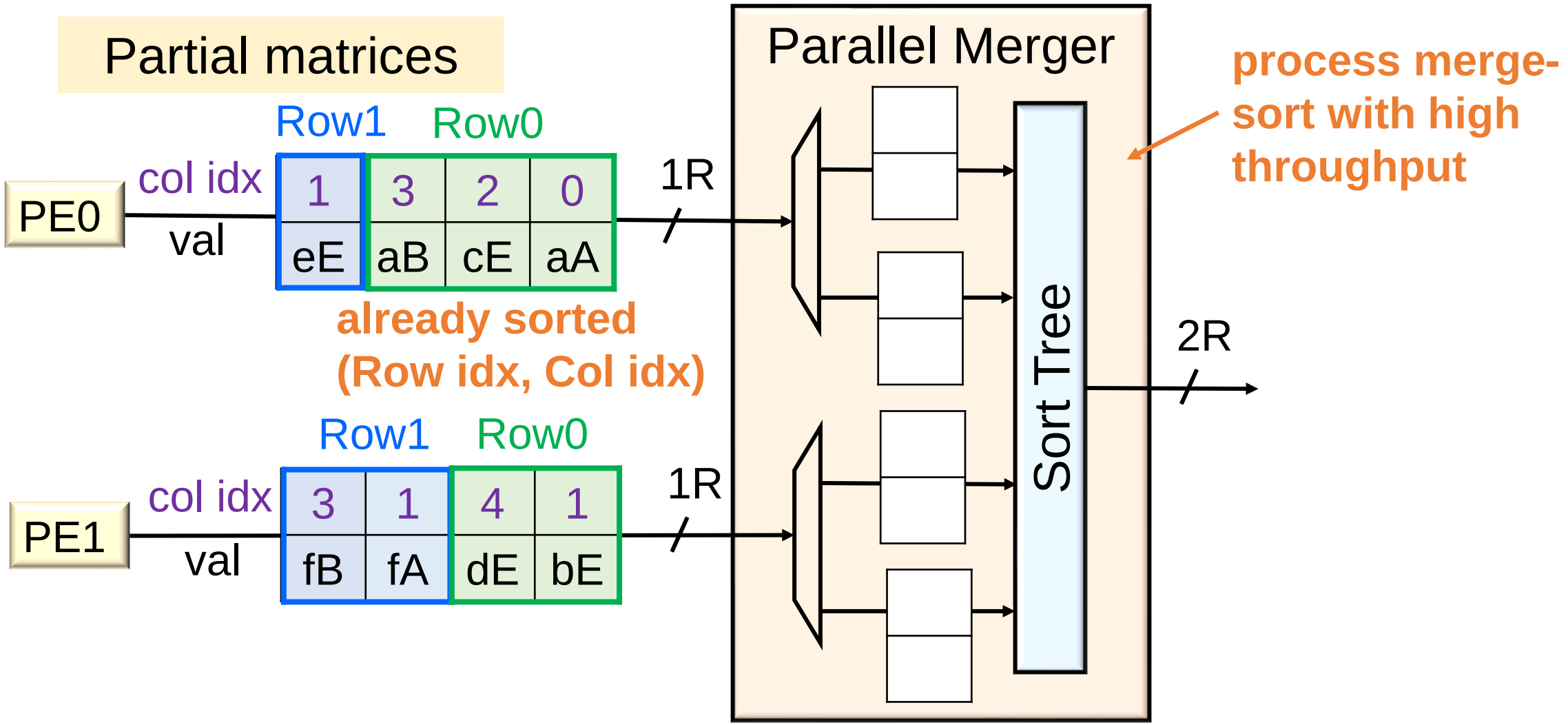
Idea 1: Distribute-Merge based SpMSpM dataflow

- DMP (Distribute-Merge Product) dataflow
  - **Equally distribute Matrix A elements to PEs**



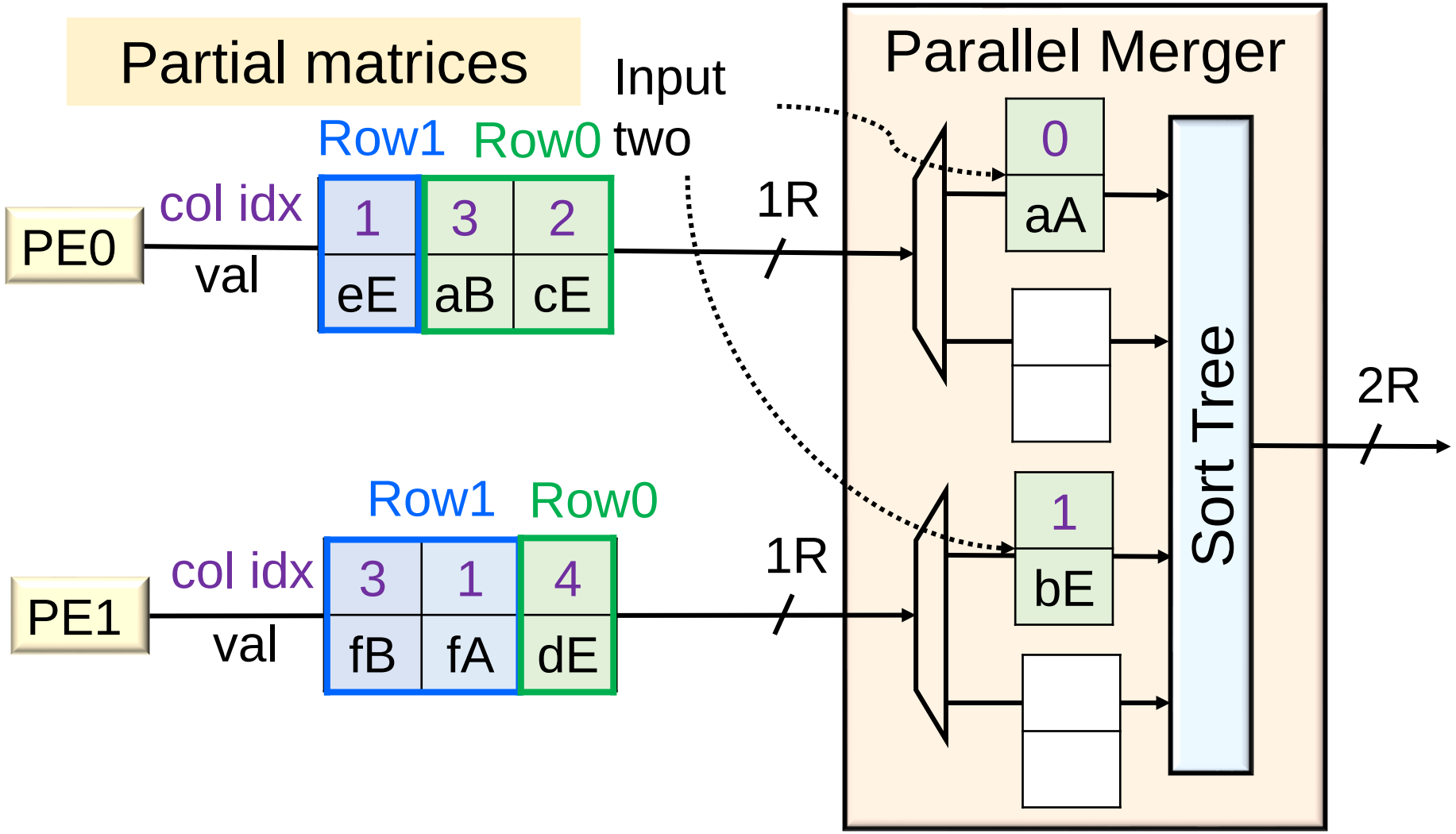
Parallel Merger

- Parallel Merger merges 2 or more partial matrices



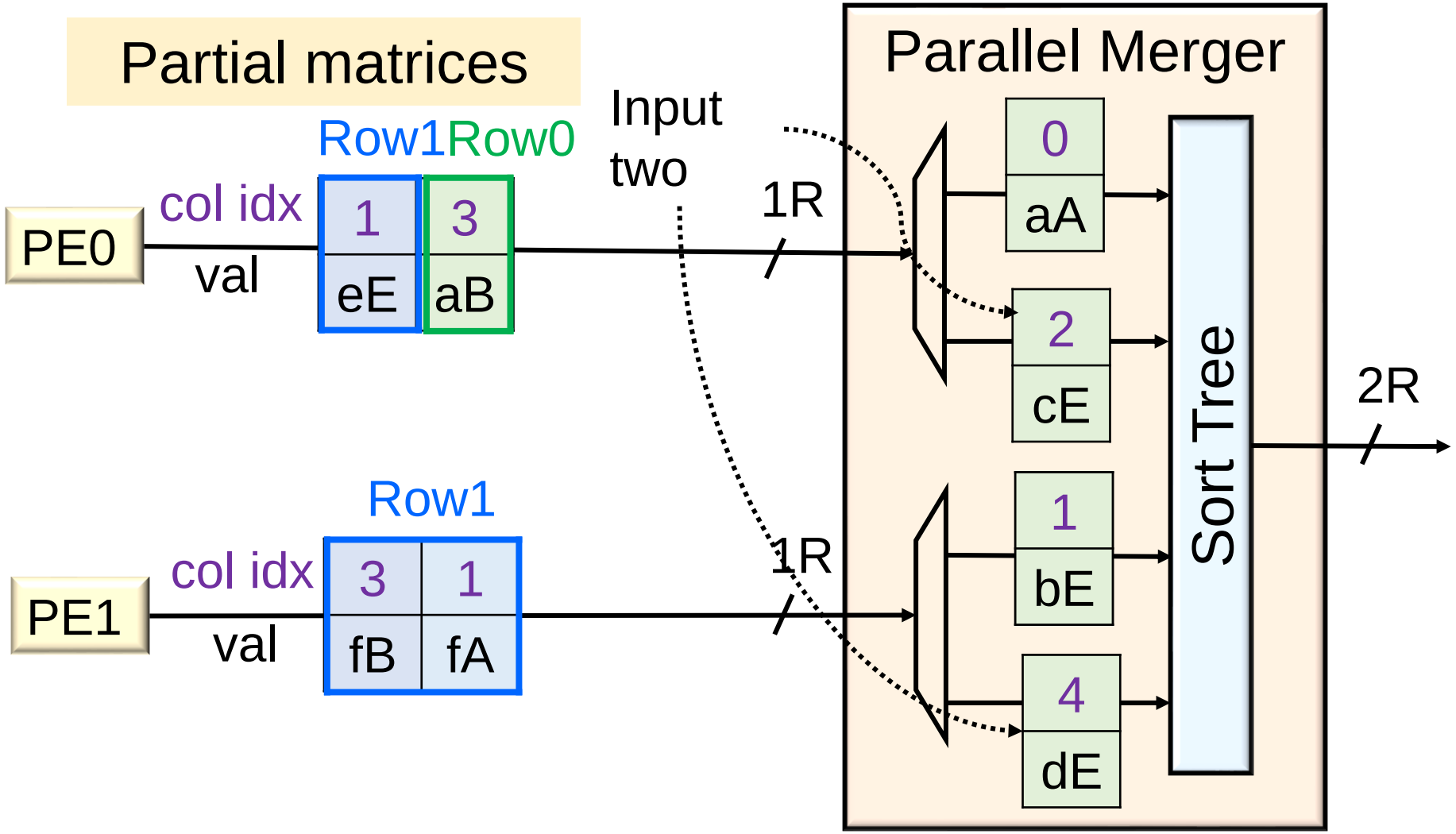
Parallel Merger

- Parallel Merger merges 2 or more partial matrices



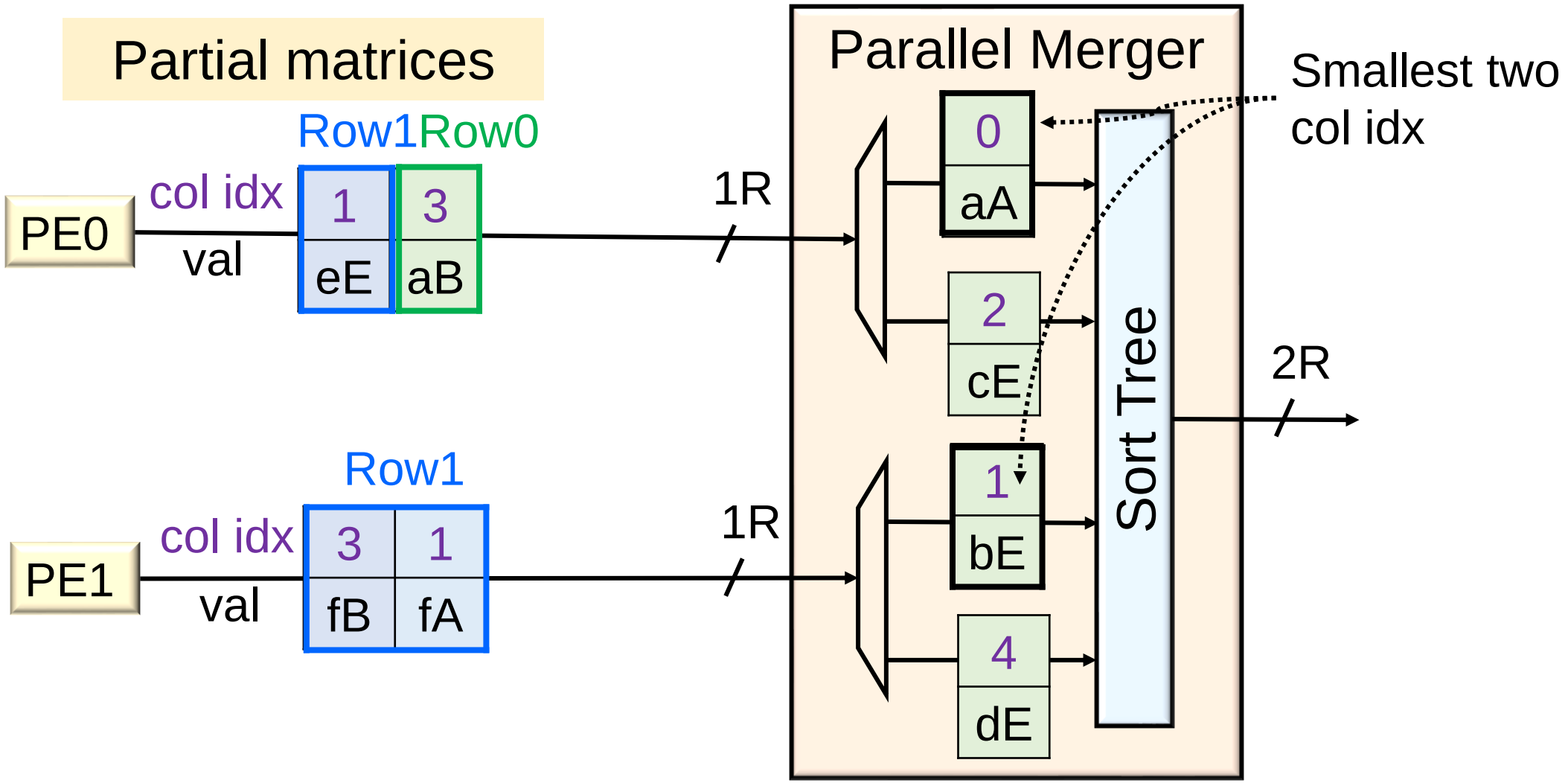
Parallel Merger

- Parallel Merger merges 2 or more partial matrices



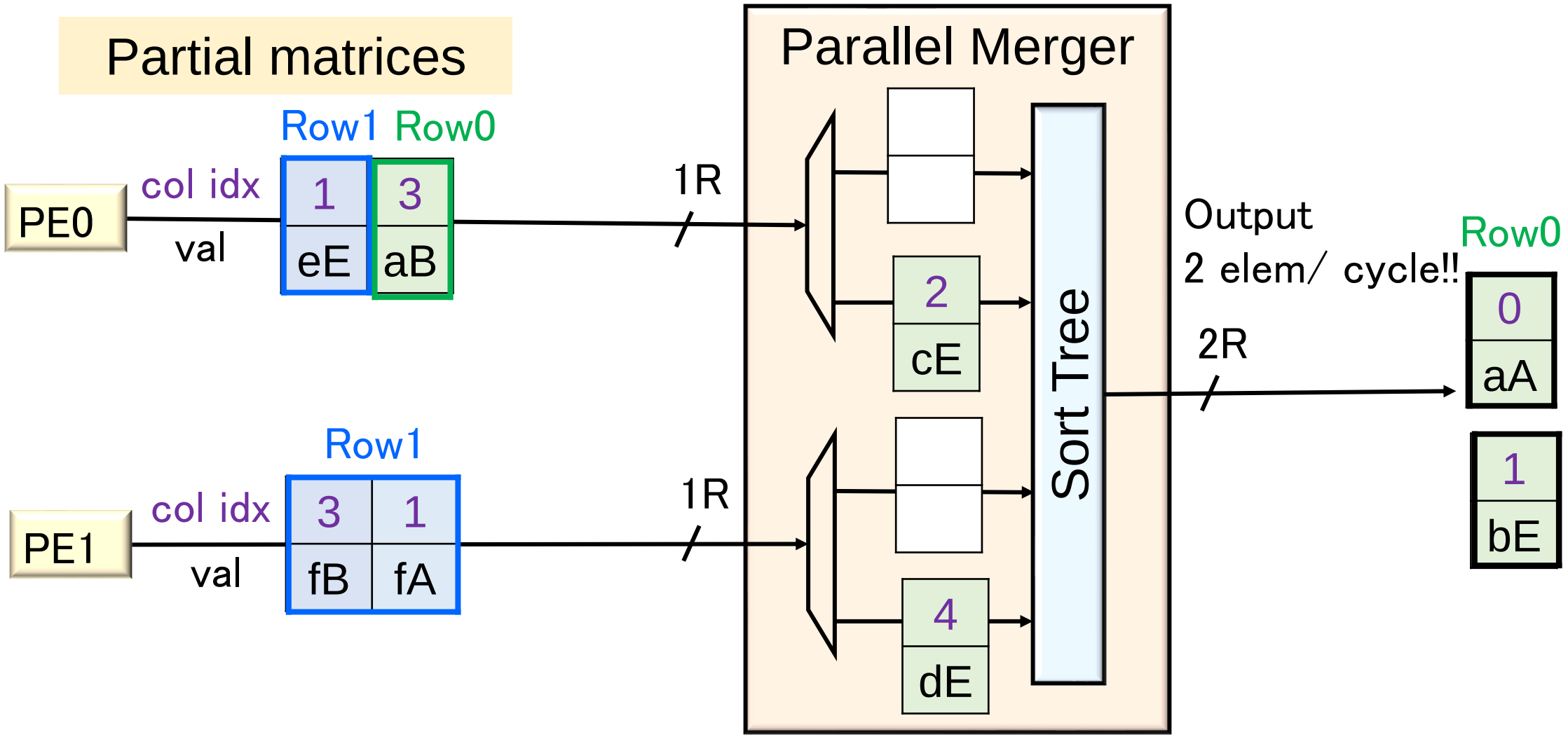
Parallel Merger

- Parallel Merger merges 2 or more partial matrices

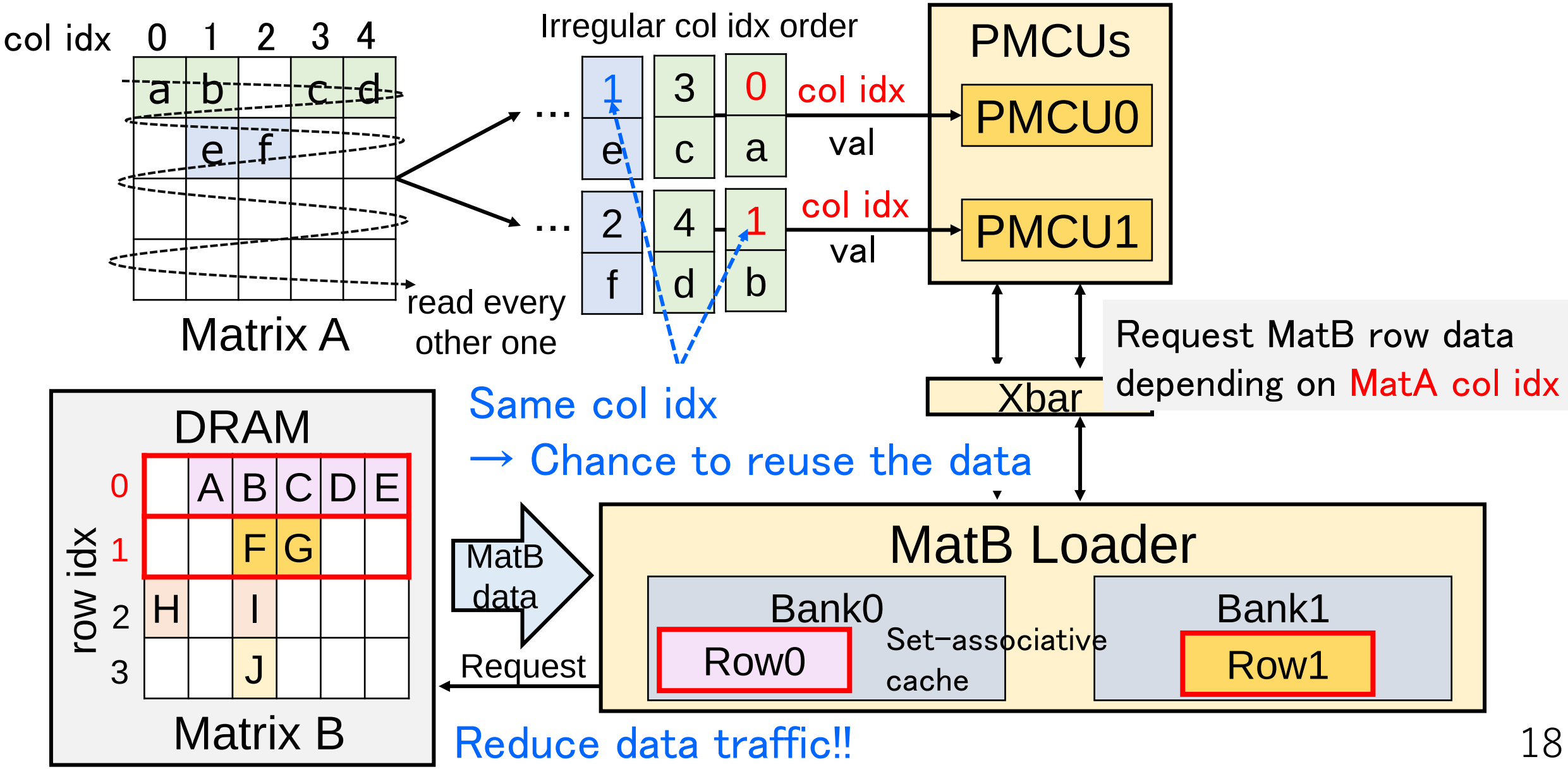


Parallel Merger

- Parallel Merger merges 2 or more partial matrices

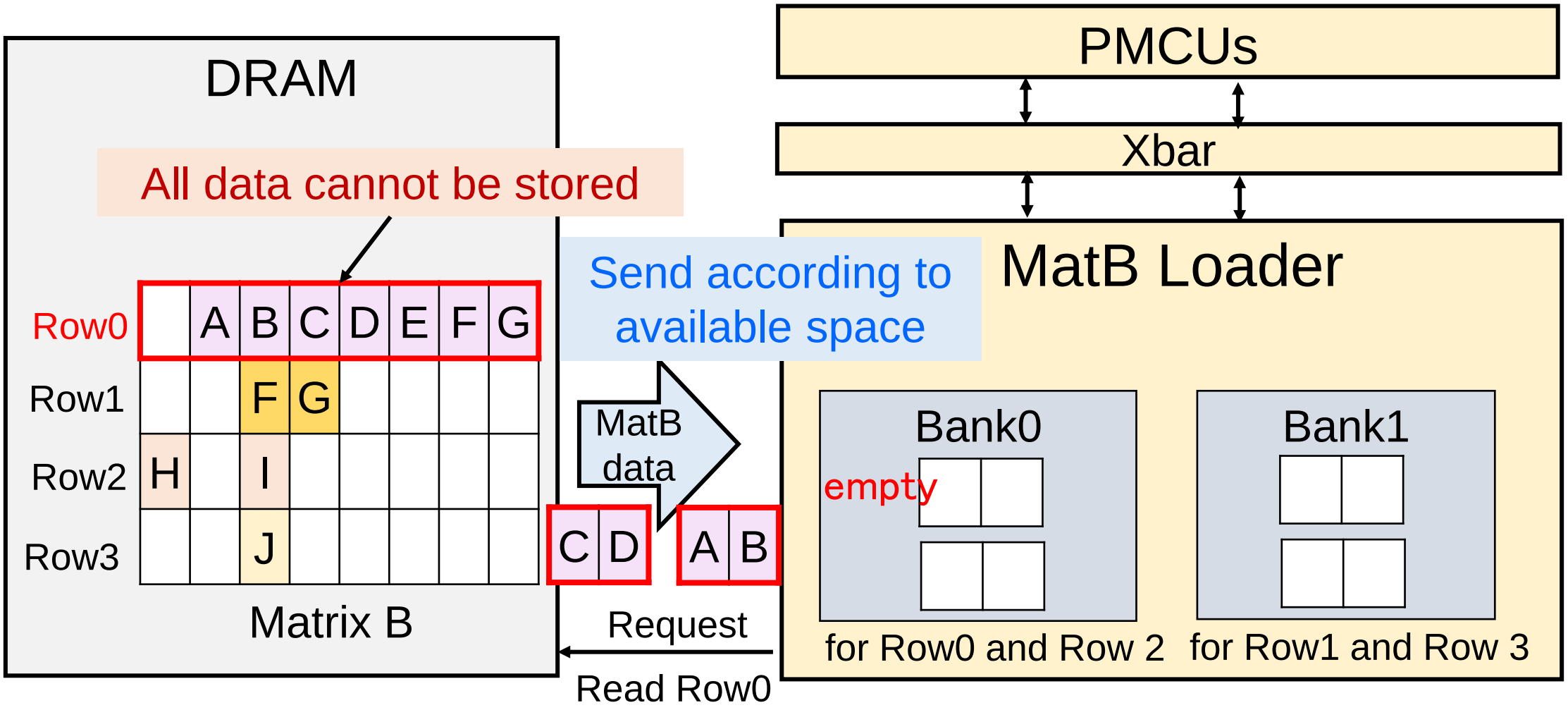


Idea 2.1: Reuse data for Mat B random access



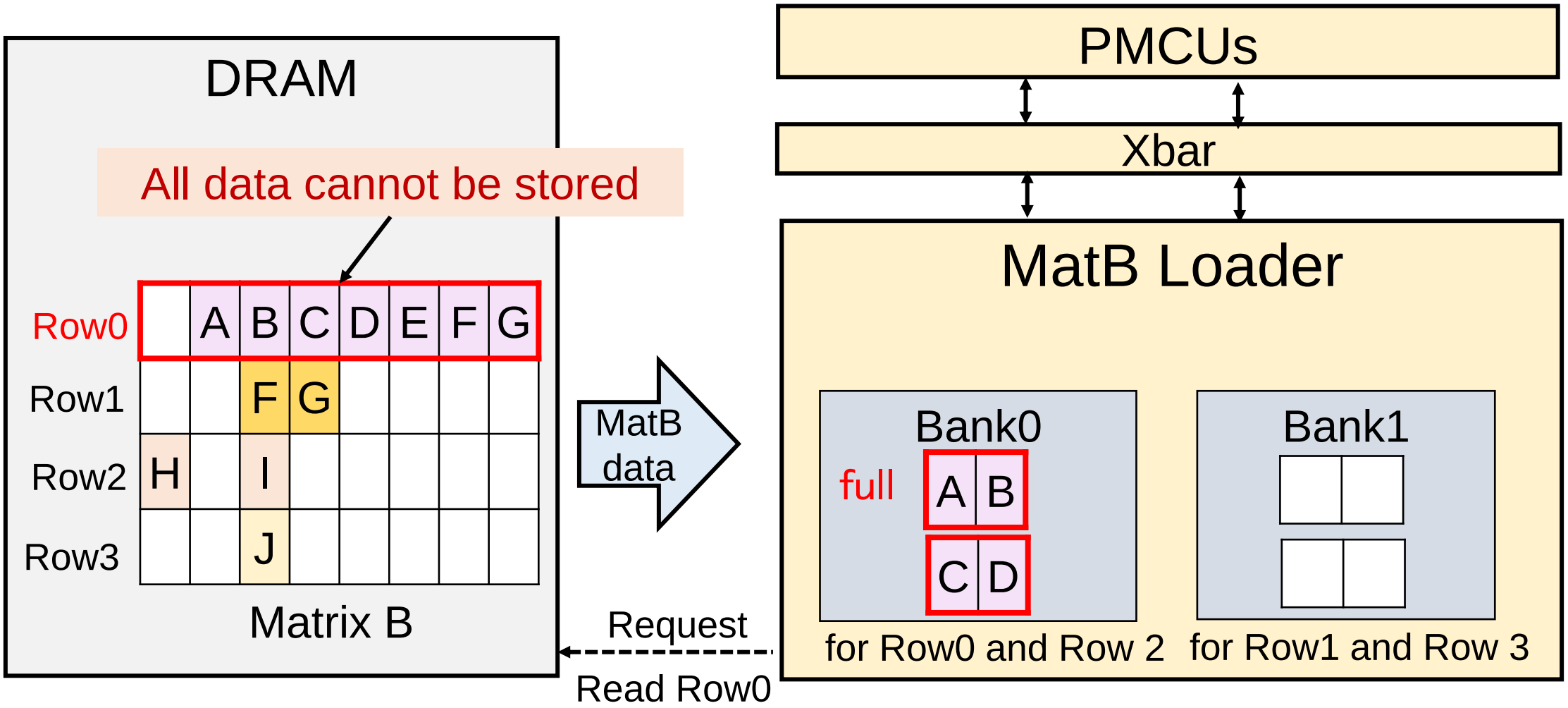
Idea 2.2: Split Matrix B row data into multiple segments

- Can efficiently handle Matrix B even with long rows



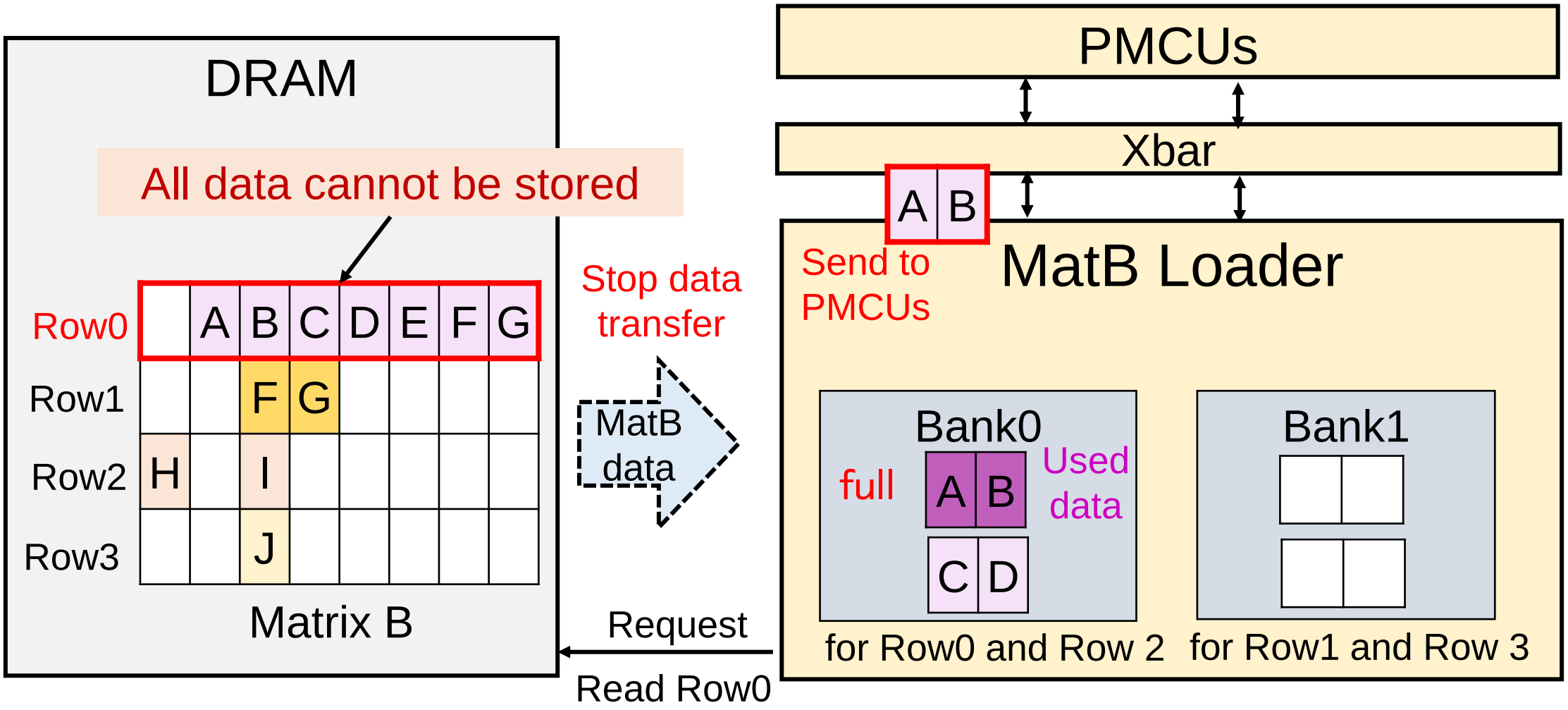
Idea 2.2: Split Matrix B row data into multiple segments

- Can efficiently handle Matrix B even with long rows



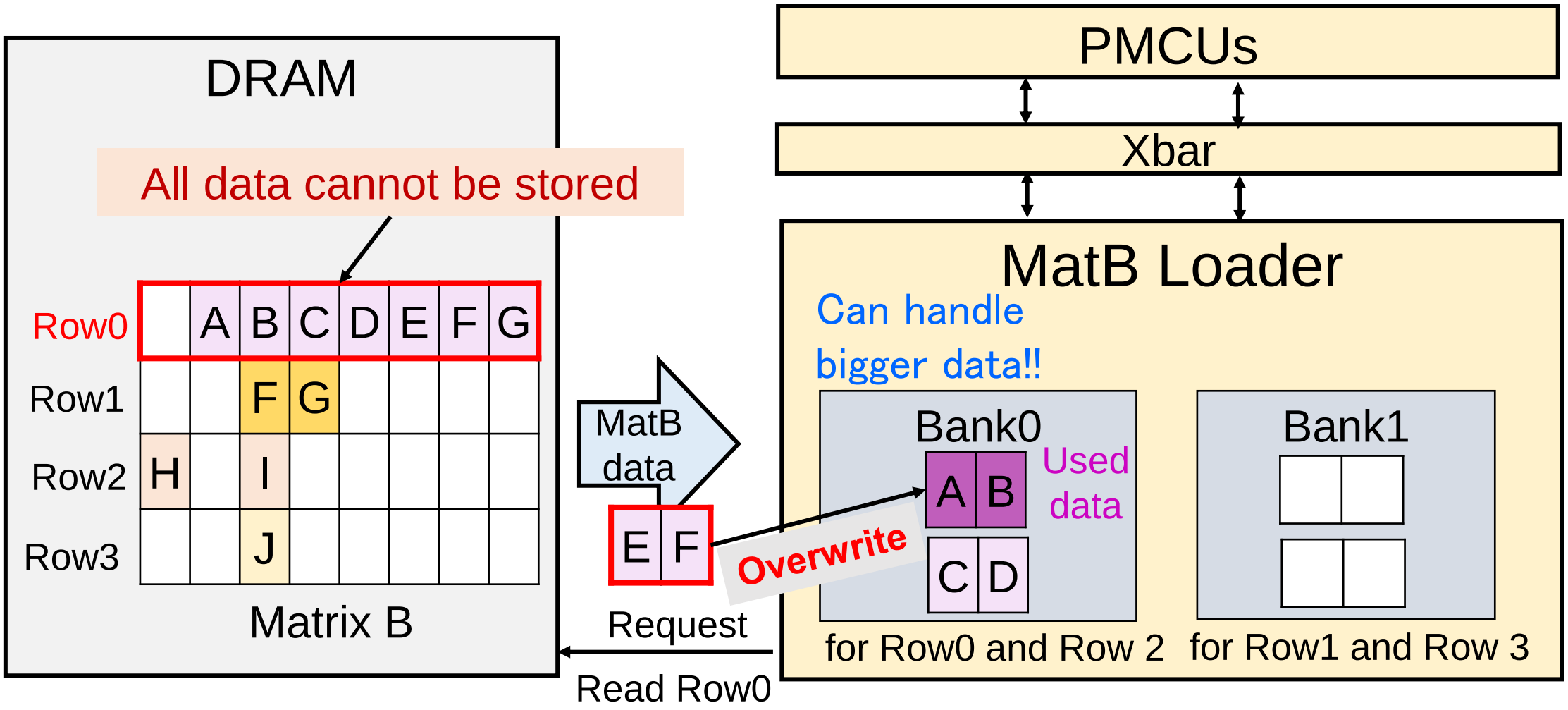
Idea 2.2: Split Matrix B row data into multiple segments

- Can efficiently handle Matrix B even with long rows

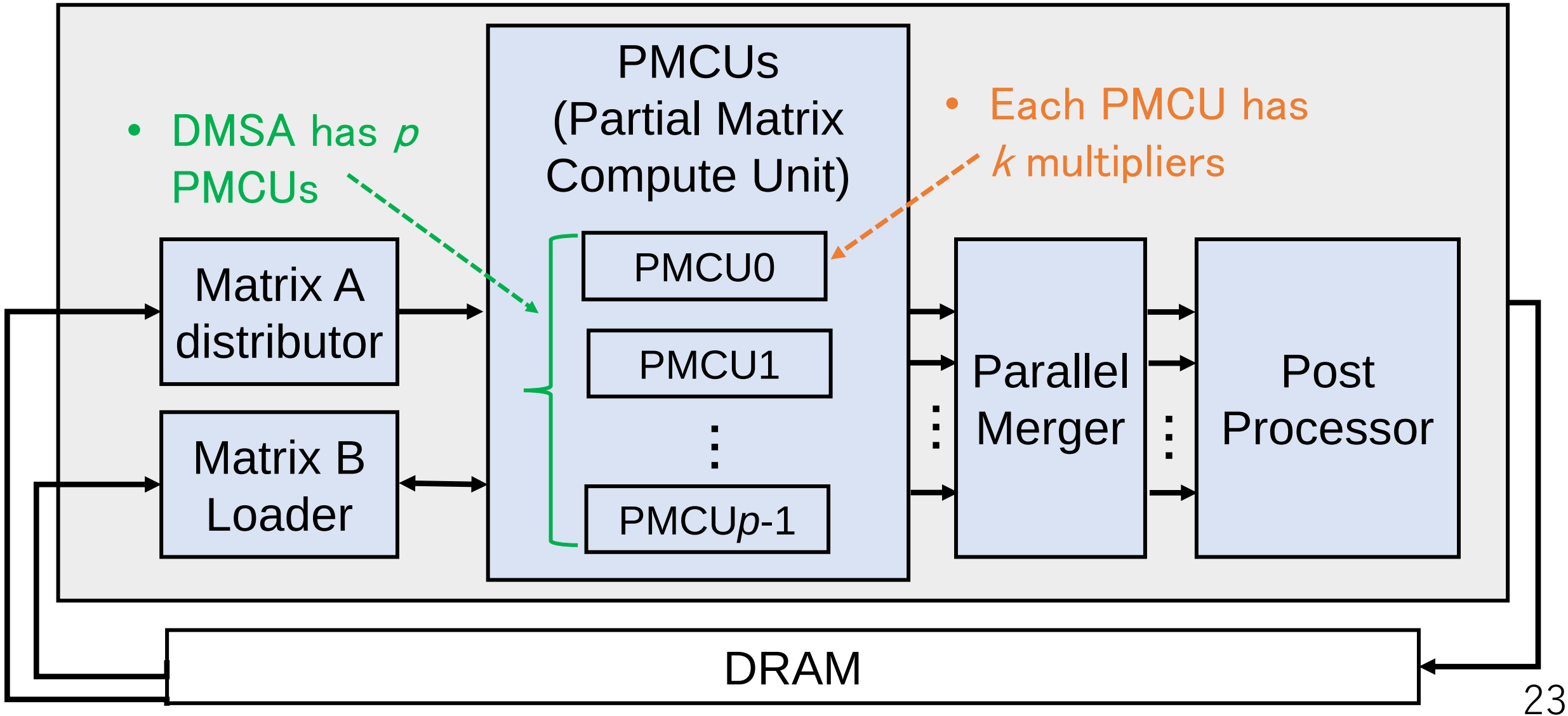


Idea 2.2: Split Matrix B row data into multiple segments

- Can efficiently handle Matrix B even with long rows



Our Architecture (DMSA: DMP-based SpMSpM Architecture)



Evaluation on FPGA（Xilinx ZCU106）

- Comparison metrics
  - Operating frequency
  - Hardware resource consumption
  - # of elapsed cycles
- Comparison target (Li et al. [1])
  - GP-based SpMSpM accelerator

[1] Li et al., IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023

2 configurations

DMSA(p,k)	
DMSA(4, 8)	DMSA(8, 4)

p: # of PMCUs

k: # of multipliers in each PMCU

Both 32 = (4x8) multipliers

10 benchmark matrices from SuiteSparse

Data name	# of rows (=# of cols)	Density	
raefsky1	3,342	2.64E-3	← Densest
af23560	23,560	8.72E-4	← Sparsest
10 matrices average	7,649	4.11E-3	

Result 1

- Operating frequency

Li et al.	DMSA(4,8)	DMSA(8,4)
100MHz	**185MHz**	**185MHz**

Our design can achieve higher frequency

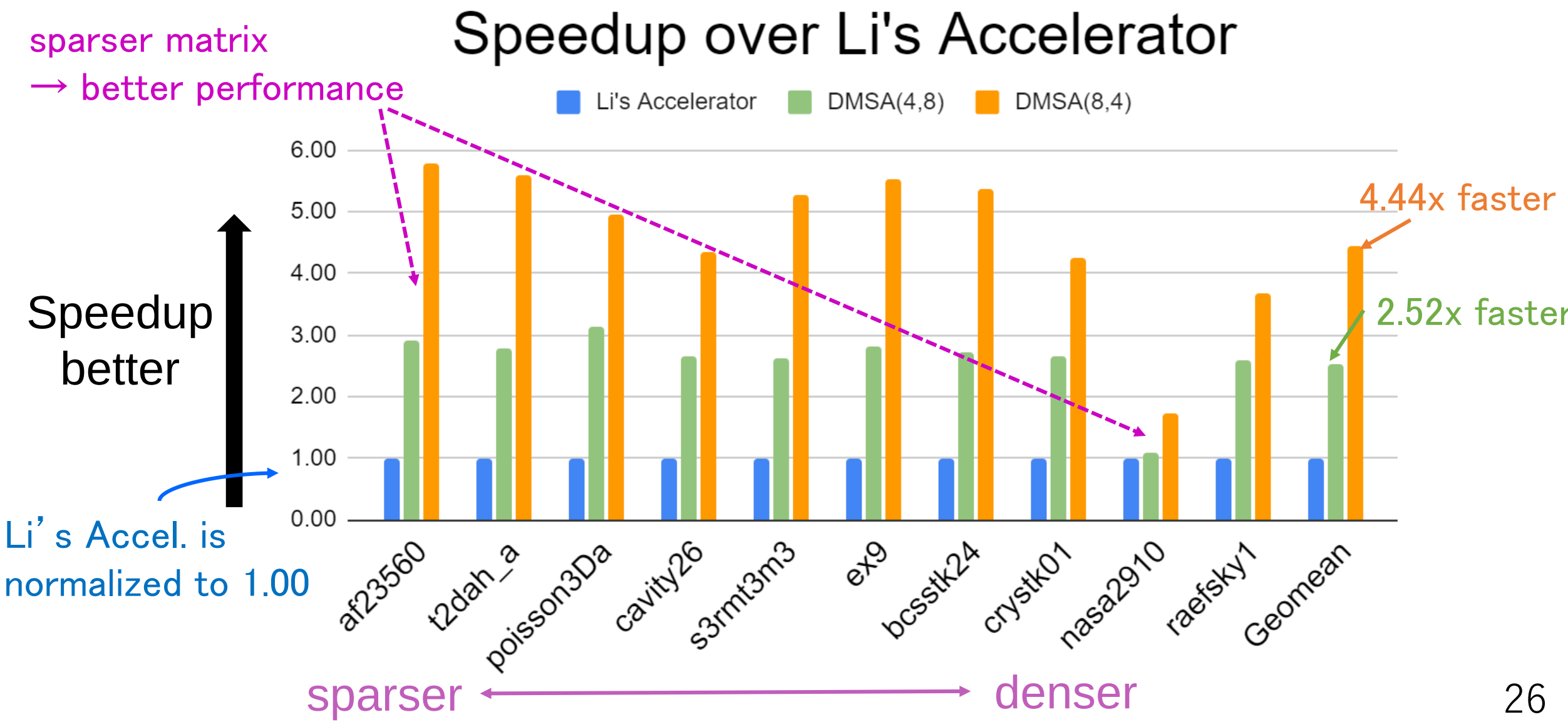
- Hardware resource consumption

	LUTs	BRAMs	URAMs	DSPs
Li et al.	66.29%	64.42%	62.50%	2.66%
DMSA(**4**,8)	61.82%	71.15%	66.67%	6.60%
DMSA(**8**,4)	83.29%	79.49%	66.67%	11.23%

comparable

1.36x larger
More PMCUs need larger Parallel Merger and larger Post Processor!!

Result 2 : # of elapsed cycles (set to same operating frequency)



Conclusion

- Analyzed conventional SpMSpM dataflows
  - Inner Product (IP), Outer Product (OP), Gustavson's Product (GP)
- Improved GP-based architectures with two key ideas
  - Idea 1: DMP (Distribute-Merge based SpMSpM) dataflow
  - Idea 2: Efficiently designed input data loader
- Designed DMSA (DMP-based SpMSpM Accelerator)
- Key results vs the state-of-the-art
  - Operating frequency: 185 MHz vs 100 MHz
  - DMSA(4, 8) is 2.52x faster with comparable resources
  - DMSA(8, 4) is 4.44x faster with about 1.36x larger resources than DMSA(4, 8)

Thank you for listening