Balancing Computation and Communication in Distributed Sparse Matrix-Vector Multiplication

Hongli Mi, Xiangrui Yu, Xiaosong Yu, Shuangyuan Wu and Weifeng Liu

Super Scientific Software Laboratory, China University of Petroleum-Beijing, China

3 May 2023

https://github.com/HongliMi/DistSpMV_Balanced

Outline

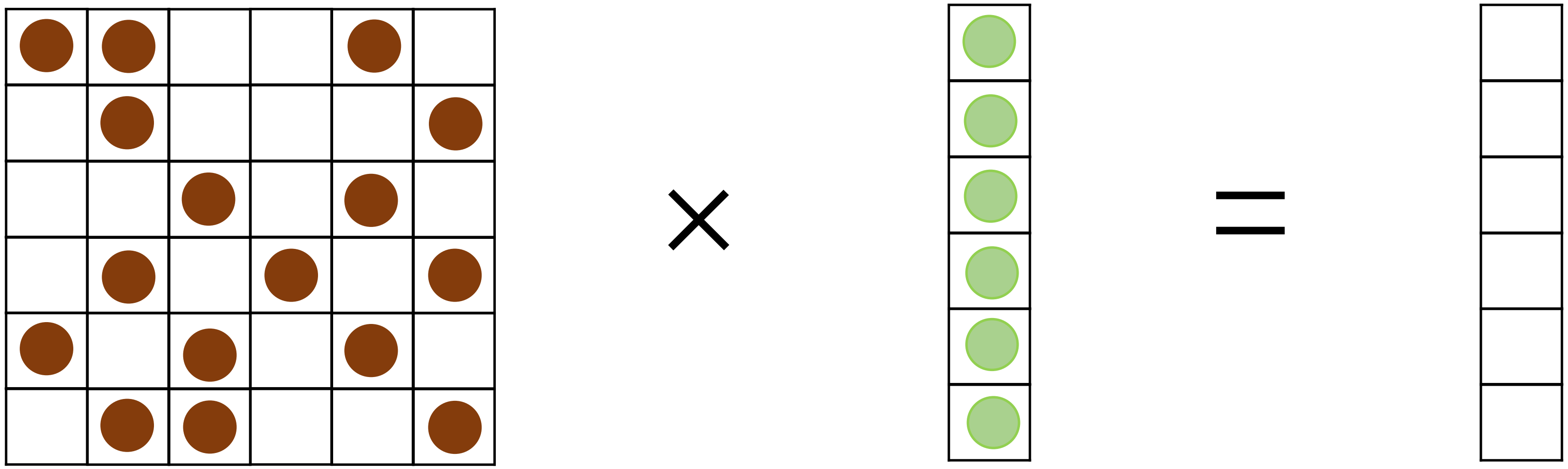
- Introduction
- Motivation
- Algorithm
- Experiment
- Conclusion

Outline

- Introduction
- Motivation
- Algorithm
- Experiment
- Conclusion

Introduction

- General Sparse Matrix-Vector Multiplication (SpMV) computes $y = Ax$, where A is a sparse matrix, x and y are both vectors.



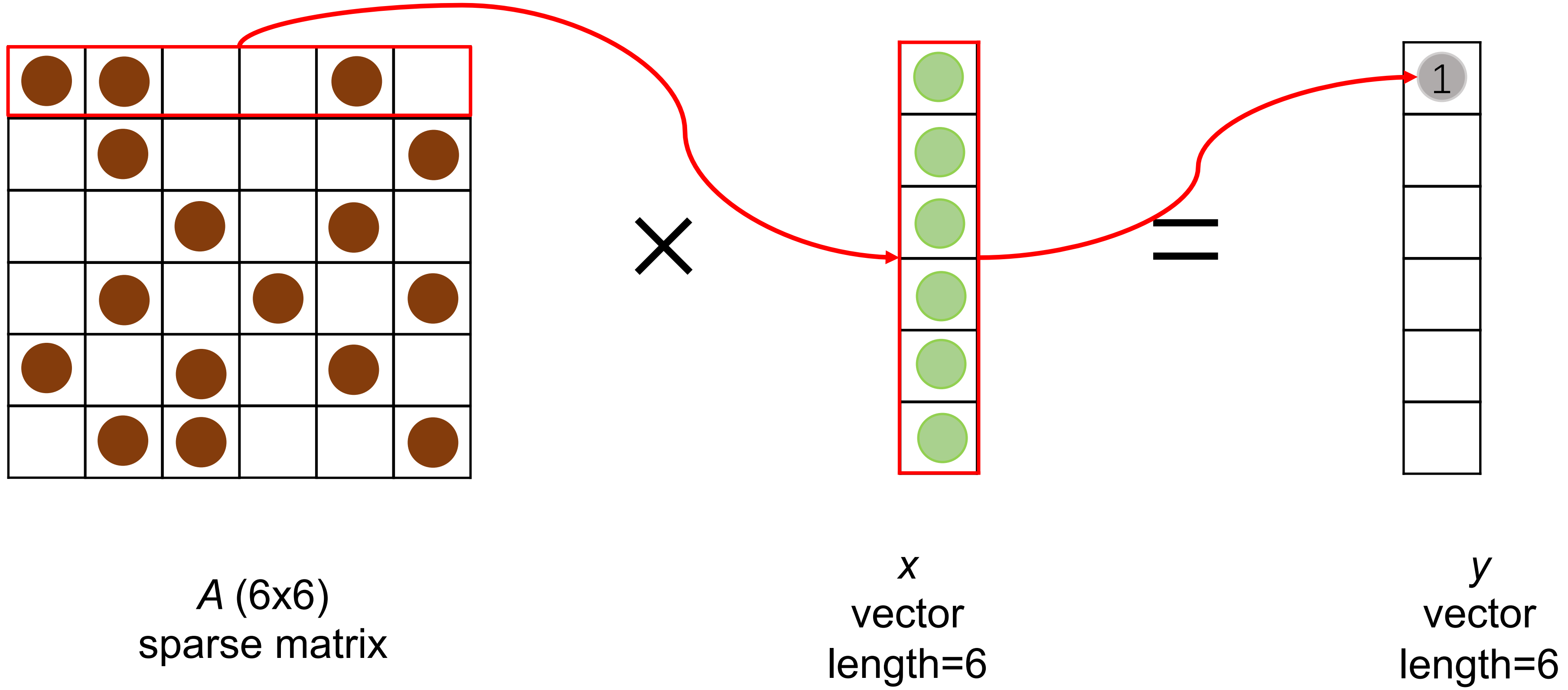
A (6x6)
sparse matrix

x
vector
length=6

y
vector
length=6
to be computed

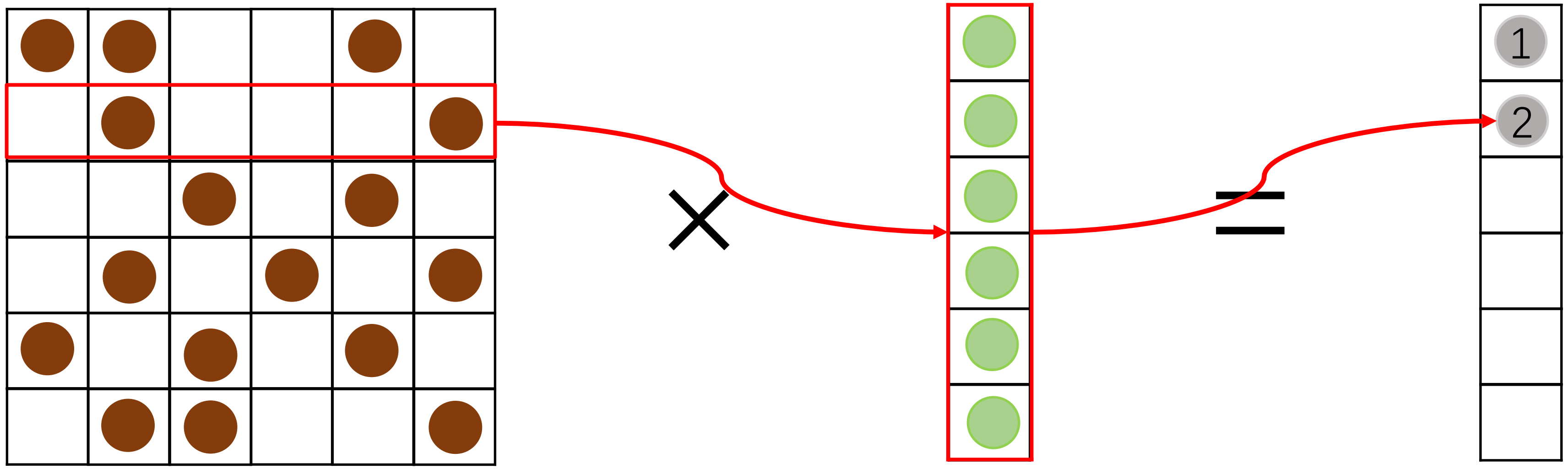
Introduction

- General Sparse Matrix-Vector Multiplication (SpMV) computes $y = Ax$, where A is a sparse matrix, x and y are both vectors.



Introduction

- General Sparse Matrix-Vector Multiplication (SpMV) computes $y = Ax$, where A is a sparse matrix, x and y are both vectors.



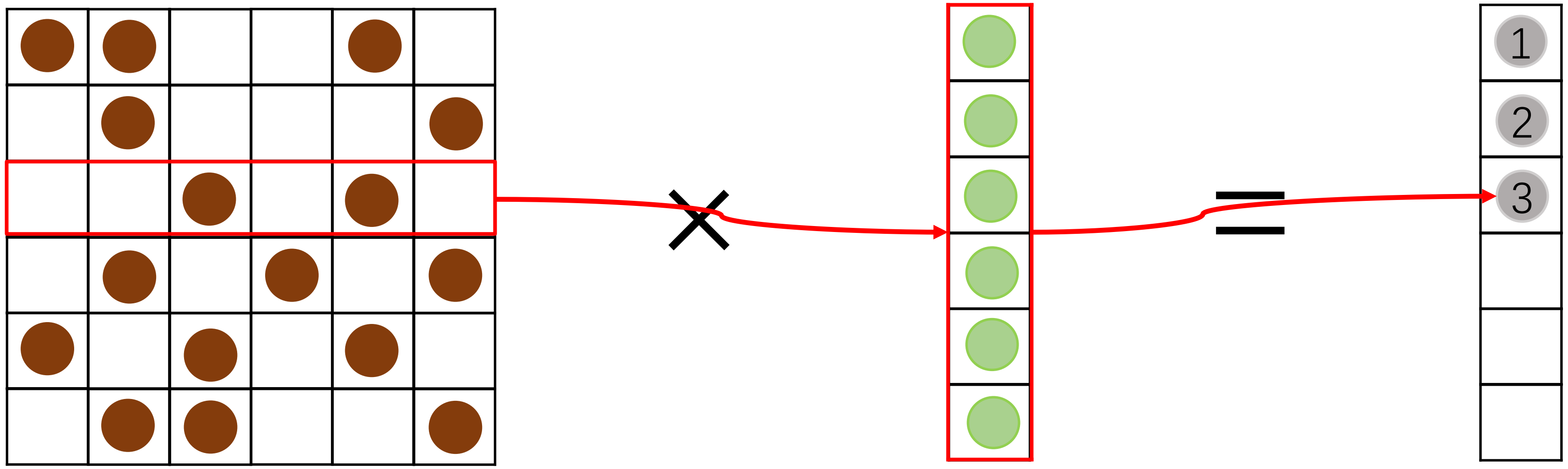
A (6x6)
sparse matrix

x
vector
length=6

y
vector
length=6

Introduction

- General Sparse Matrix-Vector Multiplication (SpMV) computes $y = Ax$, where A is a sparse matrix, x and y are both vectors.



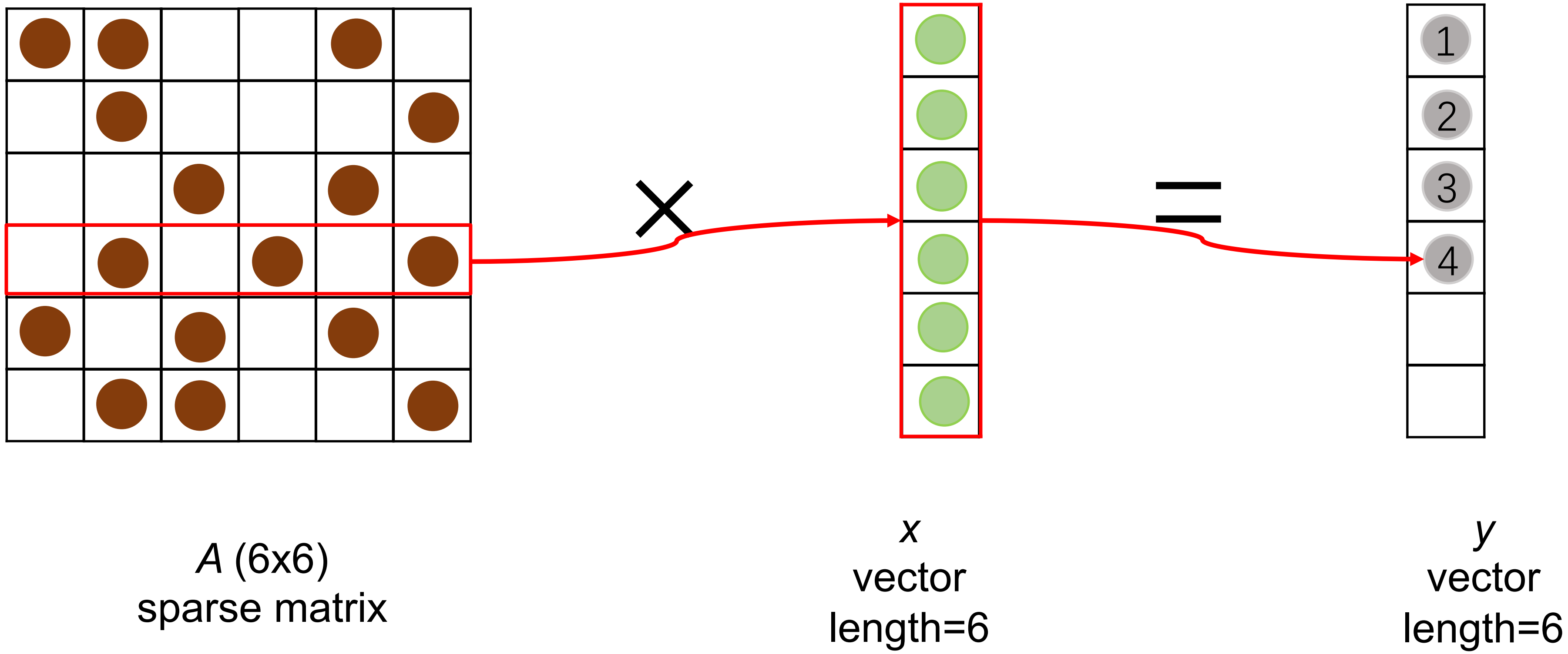
A (6x6)
sparse matrix

x
vector
length=6

y
vector
length=6

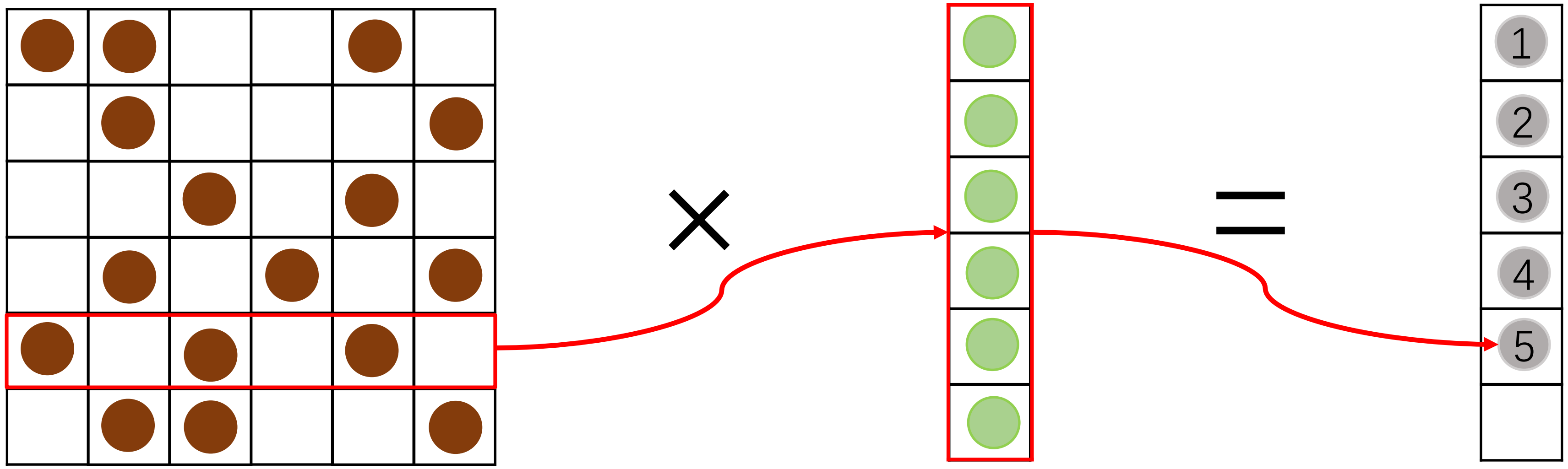
Introduction

- General Sparse Matrix-Vector Multiplication (SpMV) computes $y = Ax$, where A is a sparse matrix, x and y are both vectors.



Introduction

- General Sparse Matrix-Vector Multiplication (SpMV) computes $y = Ax$, where A is a sparse matrix, x and y are both vectors.



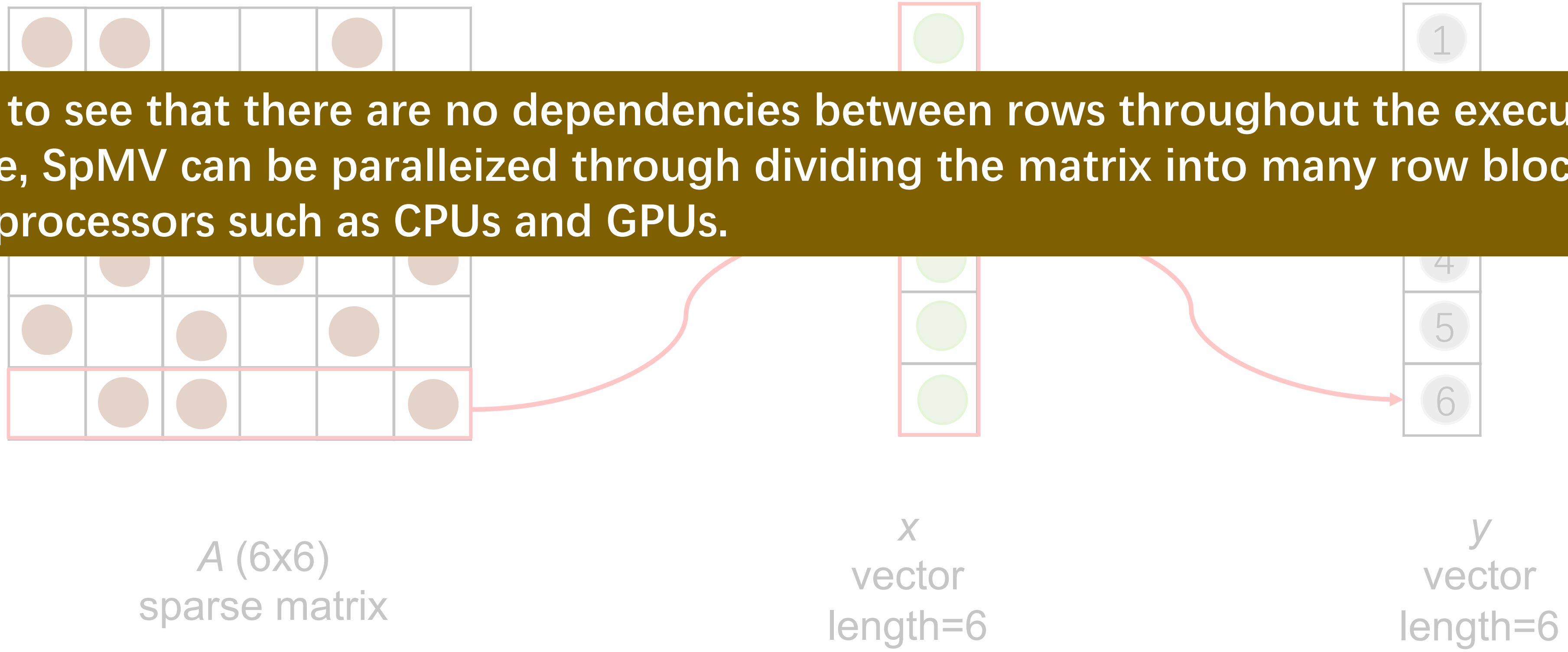
A (6x6)
sparse matrix

x
vector
length=6

y
vector
length=6

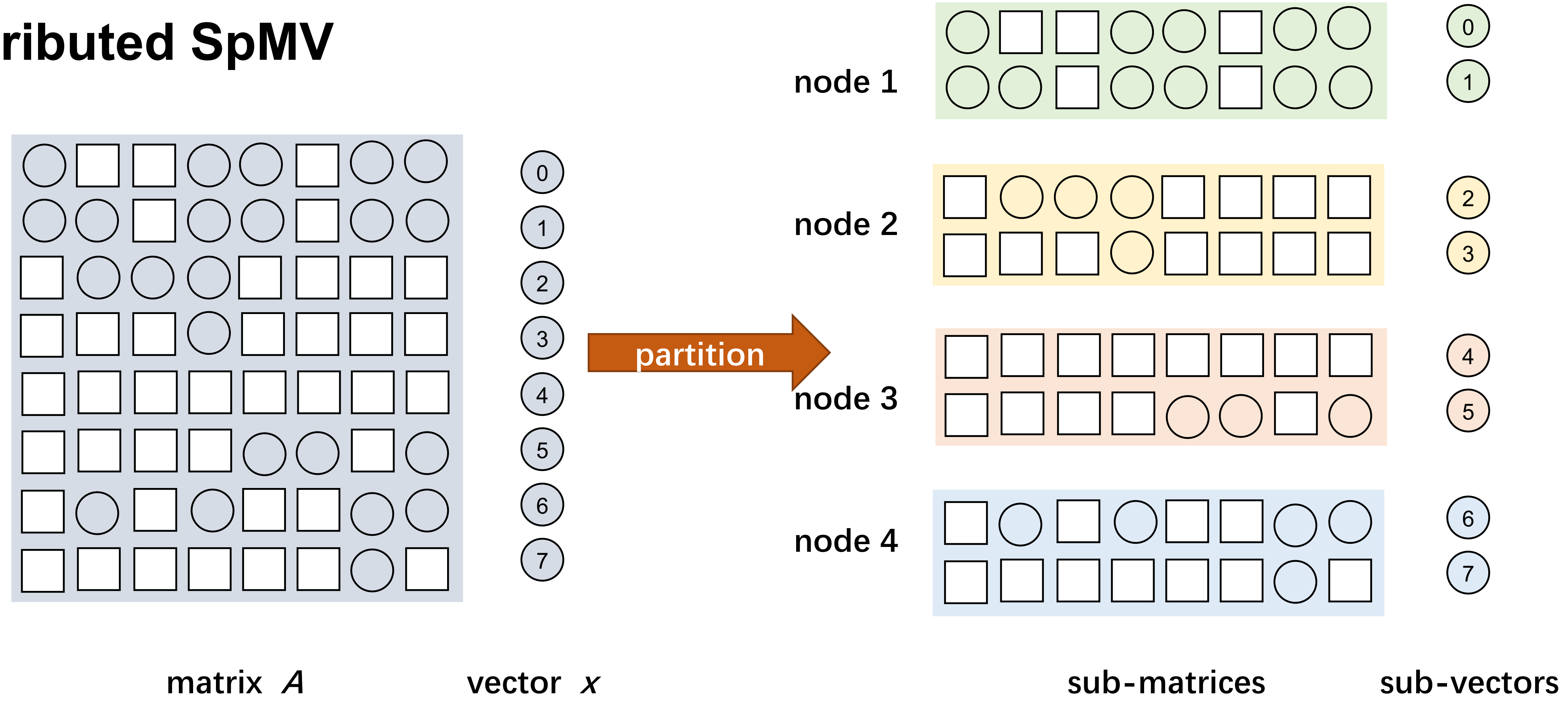
Introduction

- General Sparse Matrix-Vector Multiplication (SpMV) computes $y = Ax$, where A is a sparse matrix, x and y are both vectors.

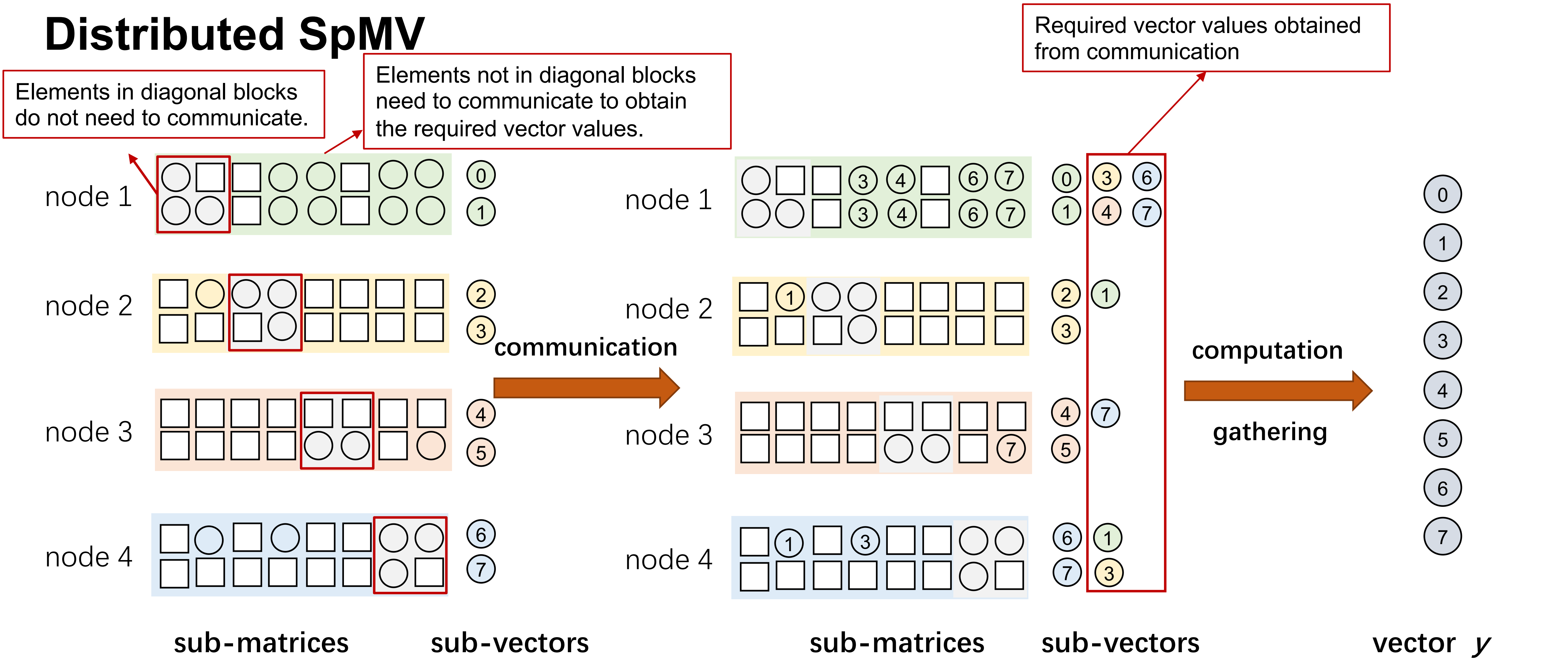


It is easy to see that there are no dependencies between rows throughout the execution. Therefore, SpMV can be paralleized through dividing the matrix into many row blocks on modern processors such as CPUs and GPUs.

Distributed SpMV



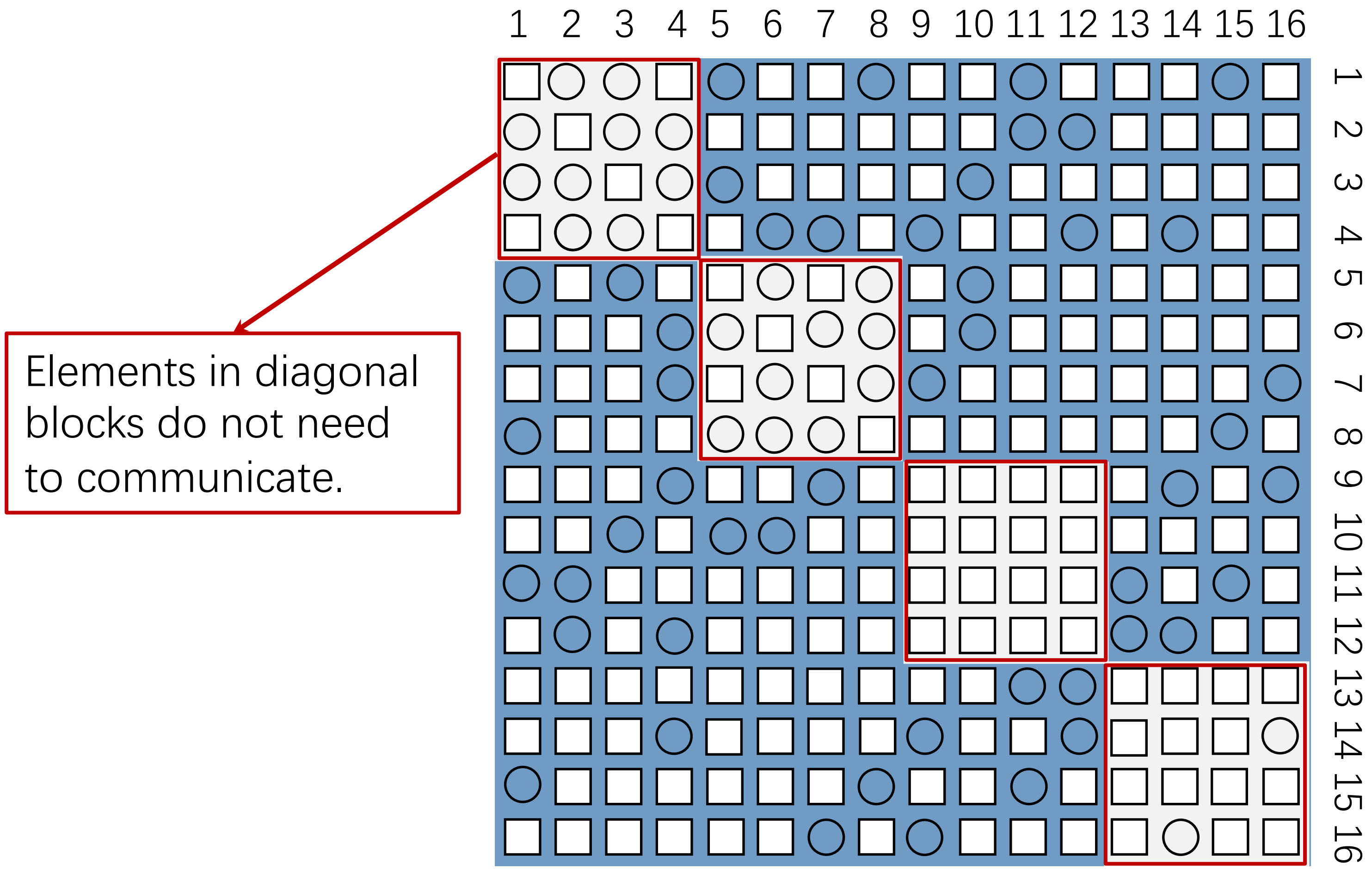
Distributed SpMV



Outline

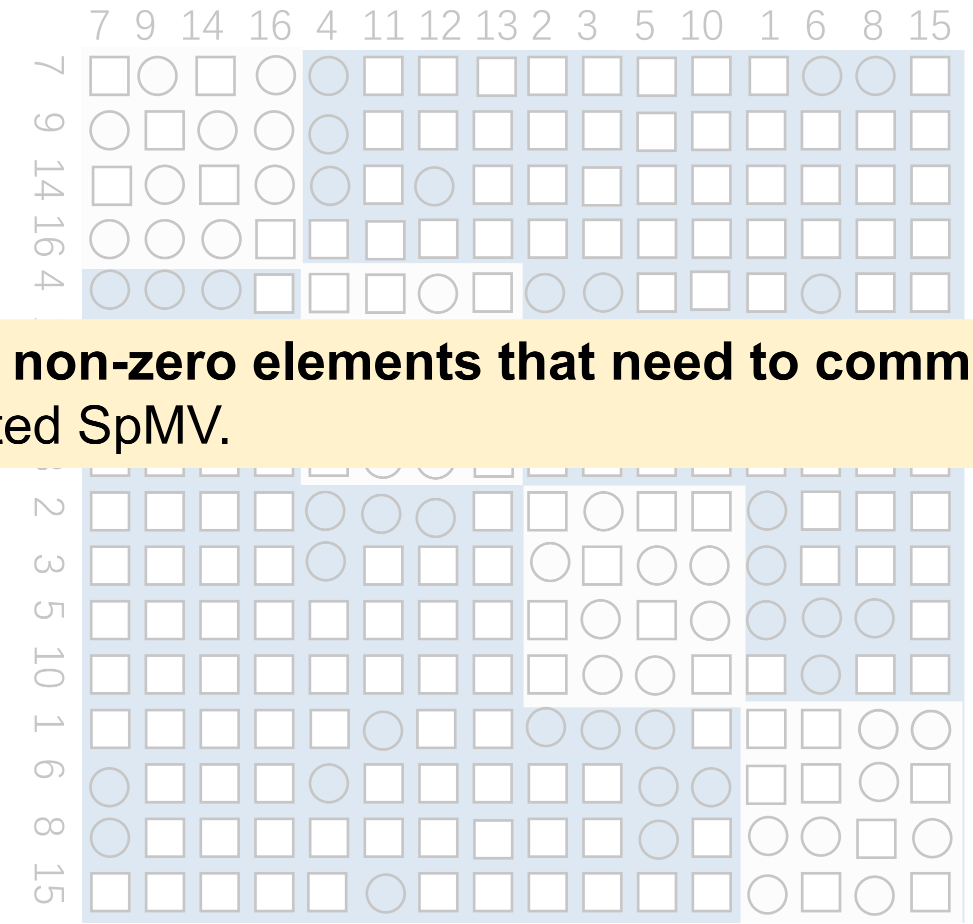
- Introduction
- Motivation
- Algorithm
- Experiment
- Conclusion

Motivation 1: Large amount of communication



Large number of non-zero elements outside the diagonal block, which results in large mount of communication.

Motivation 1: Large amount of communication

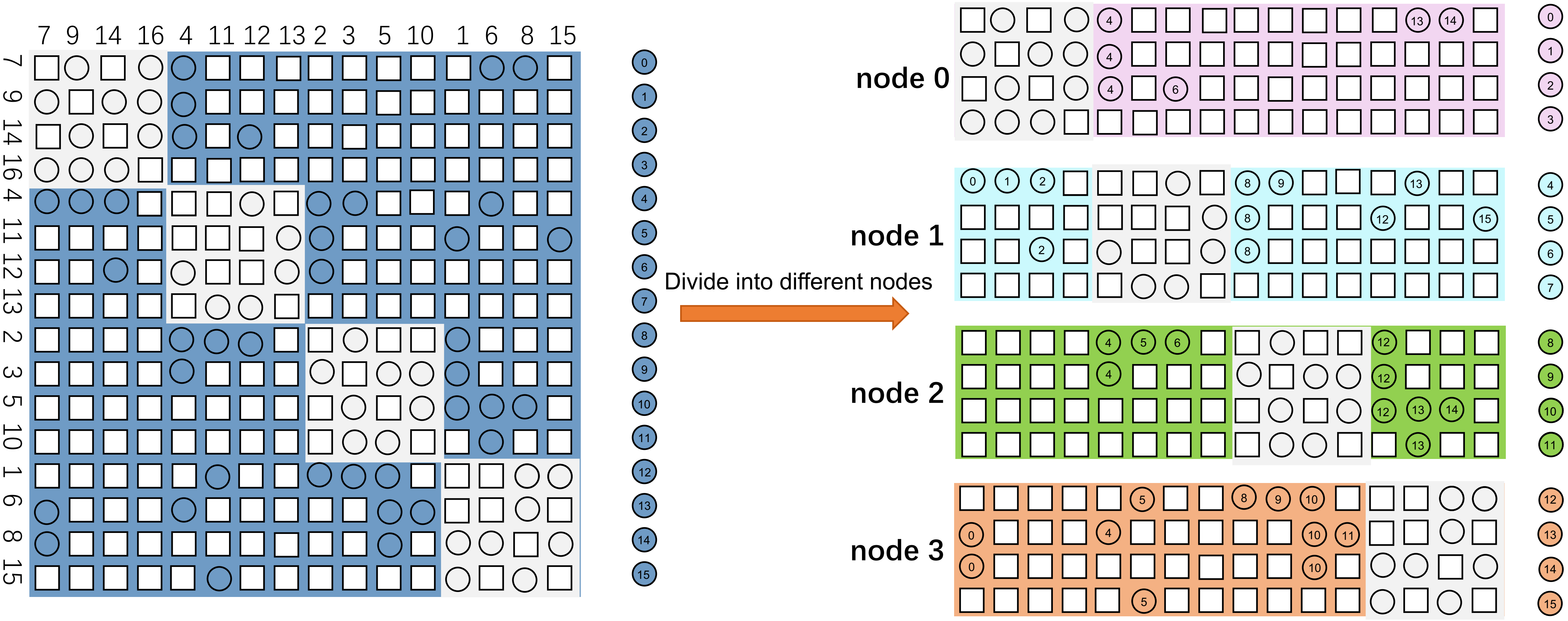


But there are still **many non-zero elements that need to communicate**, which limits the performance of distributed SpMV.

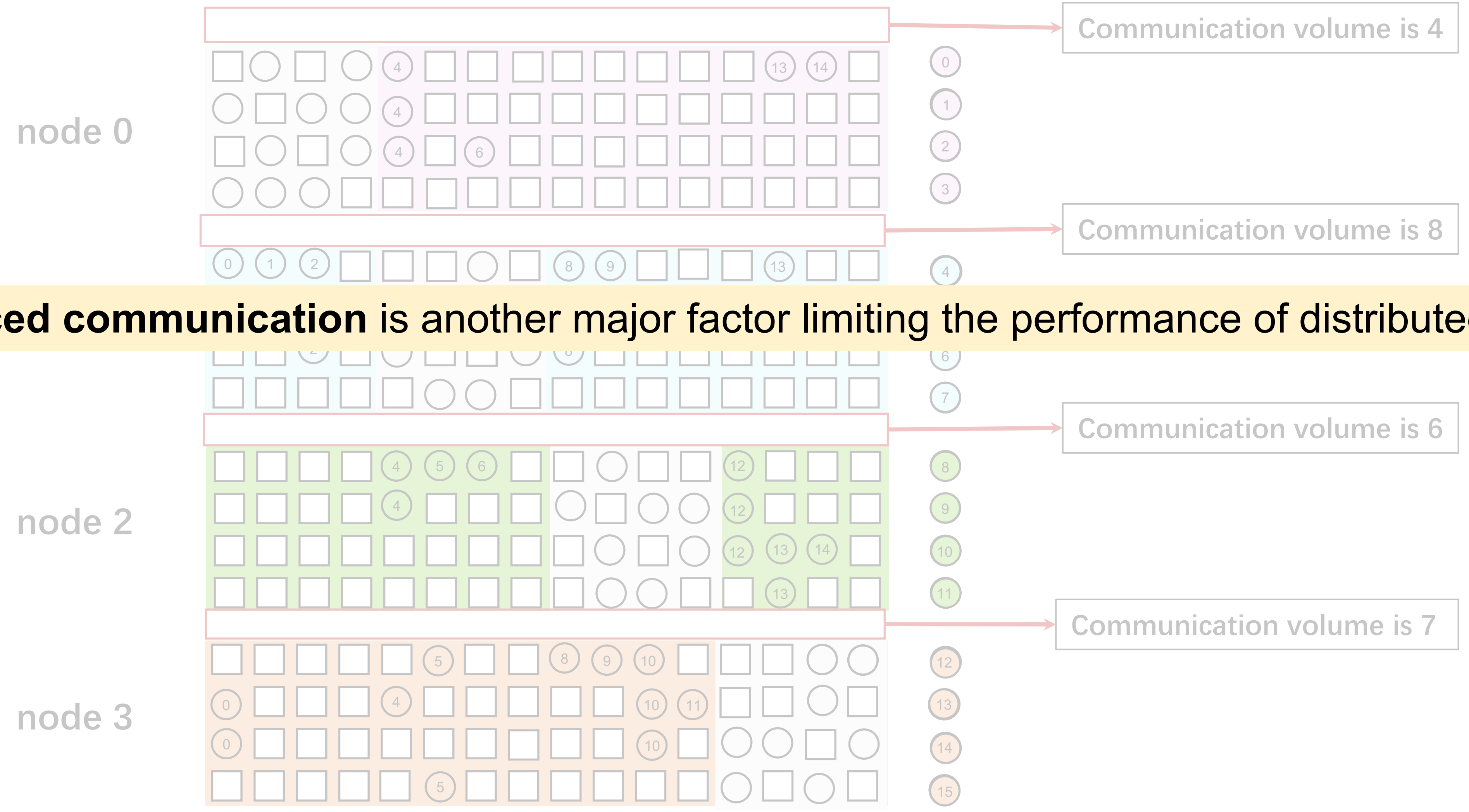
rearrangement, the number of non-zero elements of the diagonal blocks increases.

[1]G. Karypis and V. Kumar, “Analysis of multilevel graph partitioning,” in SC ’95, 1995, p. 29–es.

Motivation 2:Imbalanced communication

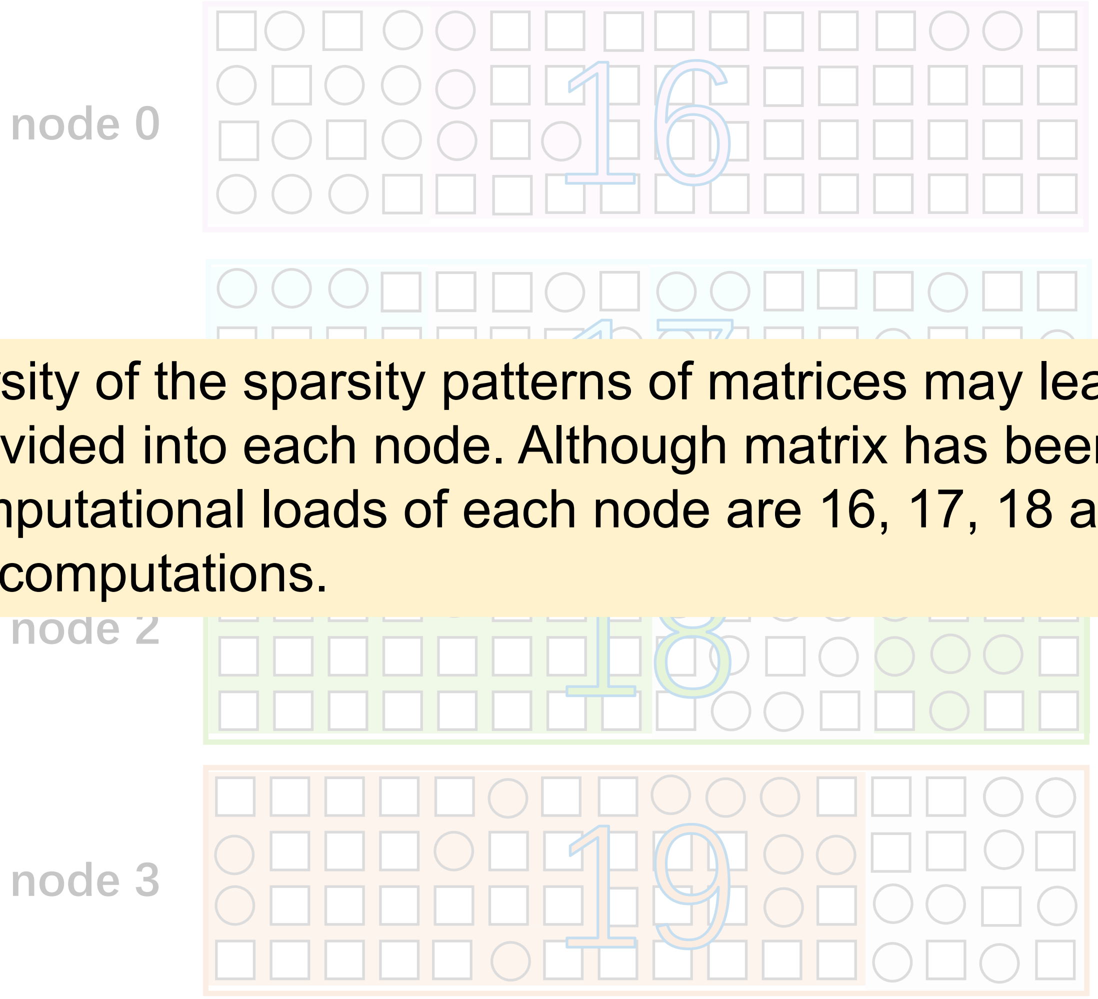


Motivation 2:Imbalanced communication



Imbalanced communication is another major factor limiting the performance of distributed SpMV.

Motivation 3:Imbalanced computation



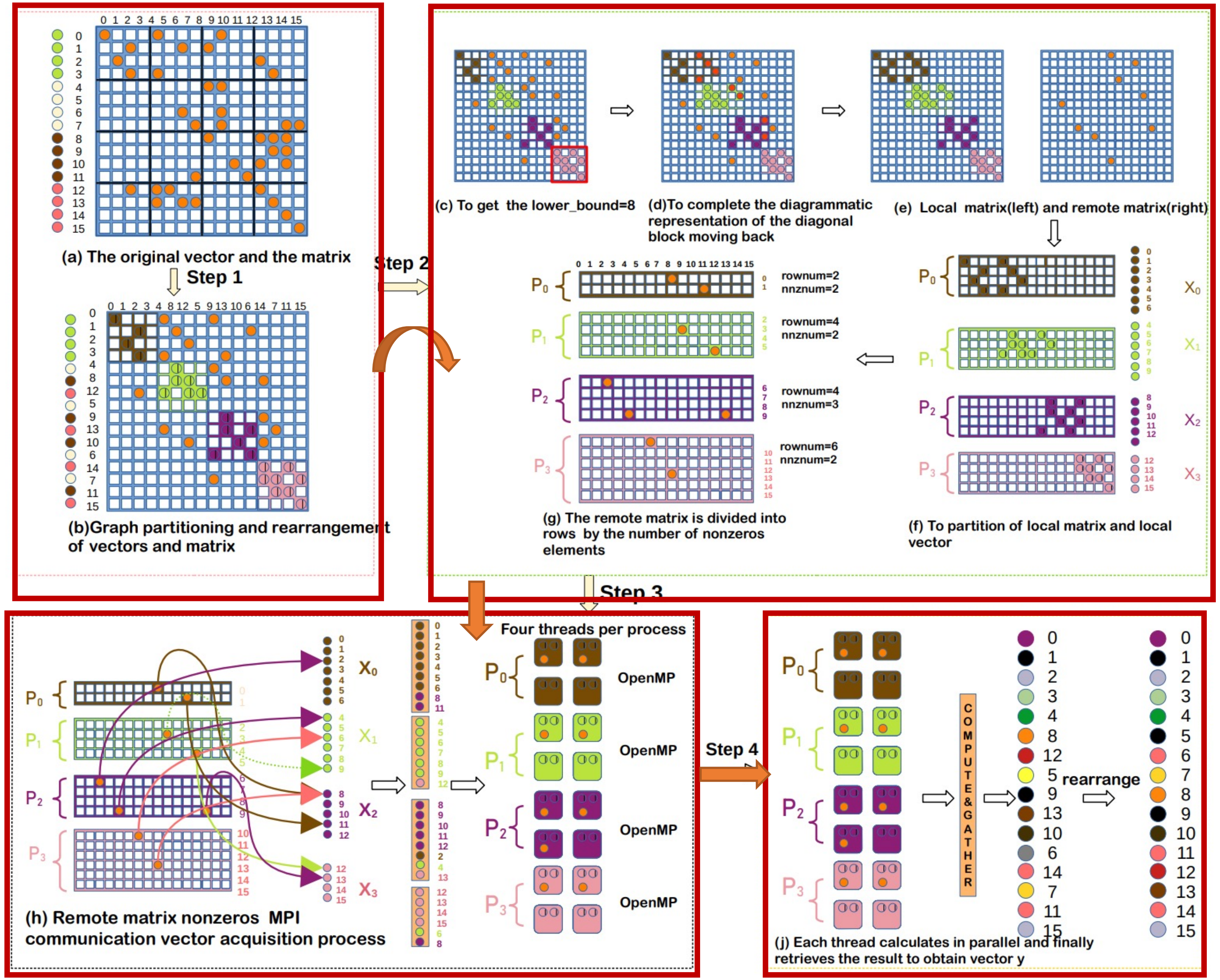
In addition,the diversity of the sparsity patterns of matrices may lead to **imbalanced calculation** after the matrix is divided into each node. Although matrix has been reorganized after graph partitioning, the computational loads of each node are 16, 17, 18 and 19, respectively, and thus lead to imbalanced computations.

Outline

- Introduction
- Motivation
- Algorithm
- Experiment
- Conclusion

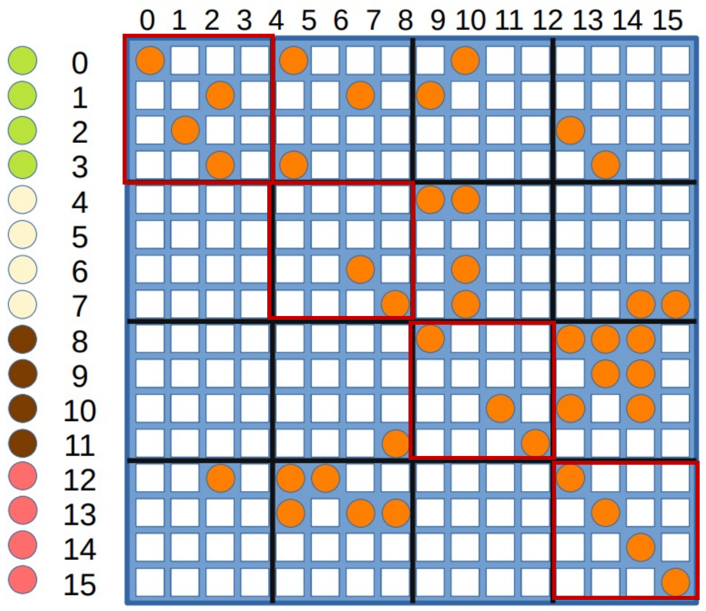
DistSpMV_Balanced

- step 1: Preprocessing stage: Graph partitioning and Matrix rearrangement
- step 2: Adjust the number of columns of the diagonal block and partition matrix
- step 3: Communication
- step 4: Computation and gather results



DistSpMV_Balanced

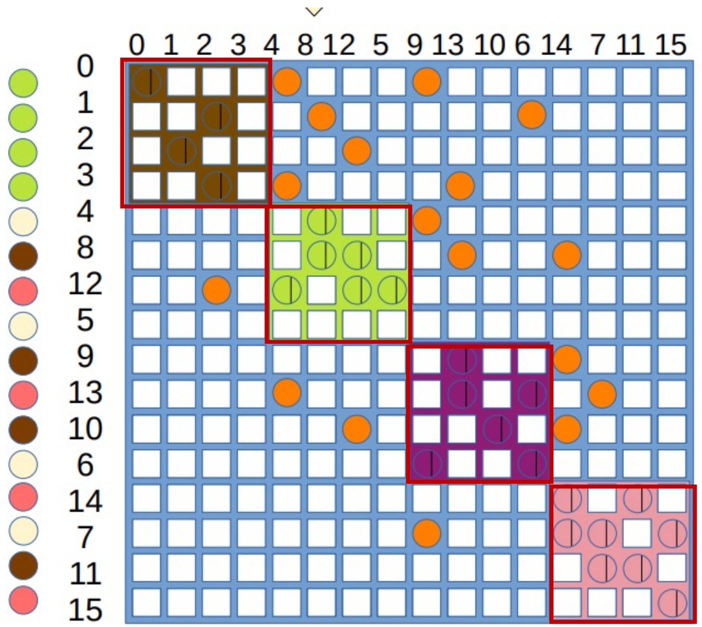
step 1: Preprocessing stage: Graph partitioning and Matrix rearrangement



(a) The original vector and the matrix

1. Use graph partitioning tool ***METIS*** [1] to partition matrix

2. Reorder vector and matrix based on partitioning results



(b)Graph partitioning and rearrangement of vectors and matrix

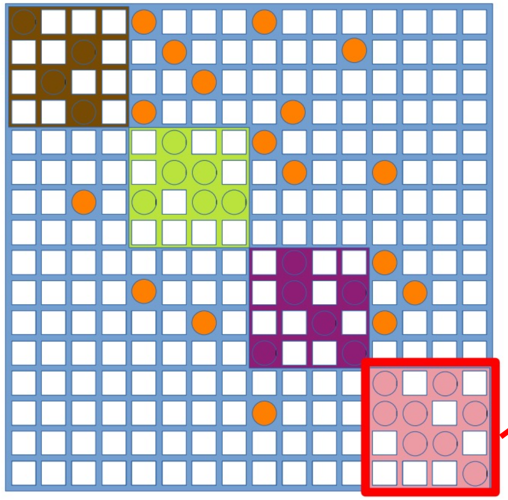
- After preprocessing, the number of non zero elements within the diagonal block increases, which reduces communication volume to a certain extent.

[1]G. Karypis and V. Kumar, “Analysis of multilevel graph partitioning,” in SC ’95, 1995, p. 29–es.

DistSpMV_Balanced

step 2: Adjust the number of columns of the diagonal block and partition matrix

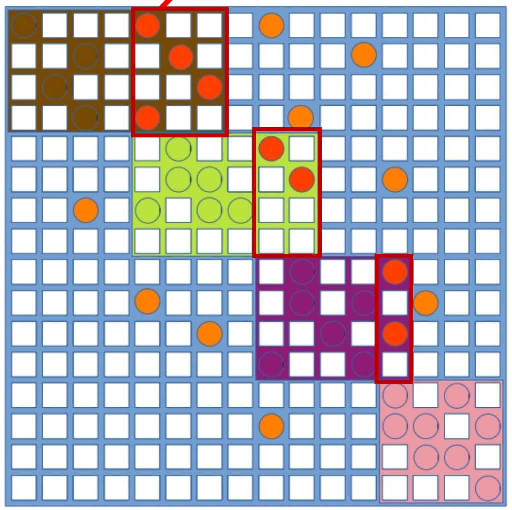
Strategy 1 ： **Expand the number of non zero elements in diagonal blocks to further reduce communication volume.**



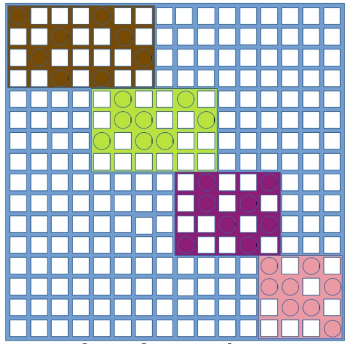
lower_bound = 8

Firstly, take the **maximum** number of non-zero elements in diagonal blocks as the **threshold(we call it "*lower_bound*")**

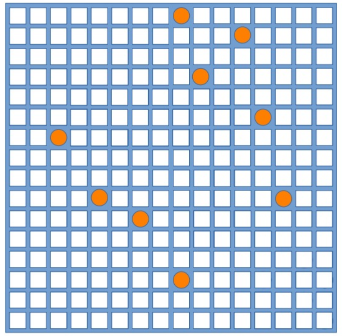
Non zero elements added to diagonal blocks



Secondly, **move the boundary of each diagonal block** until ① the non zero elements within the diagonal block **are greater than or equal** to lower_ Bound or ② move to the right boundary of the original matrix.



local matrix



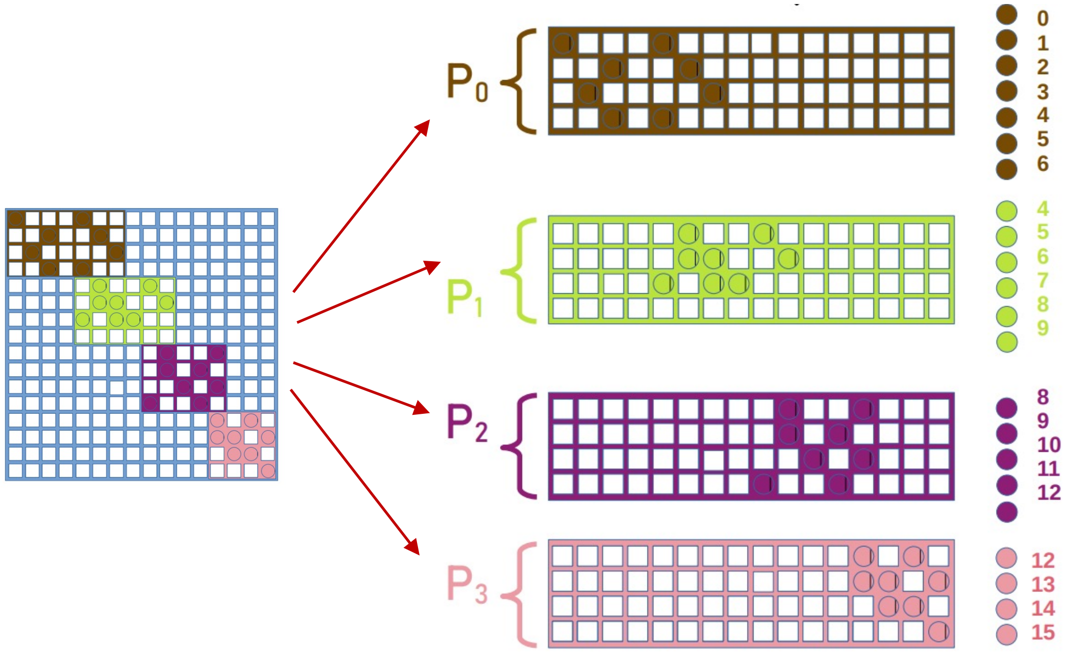
remote matrix

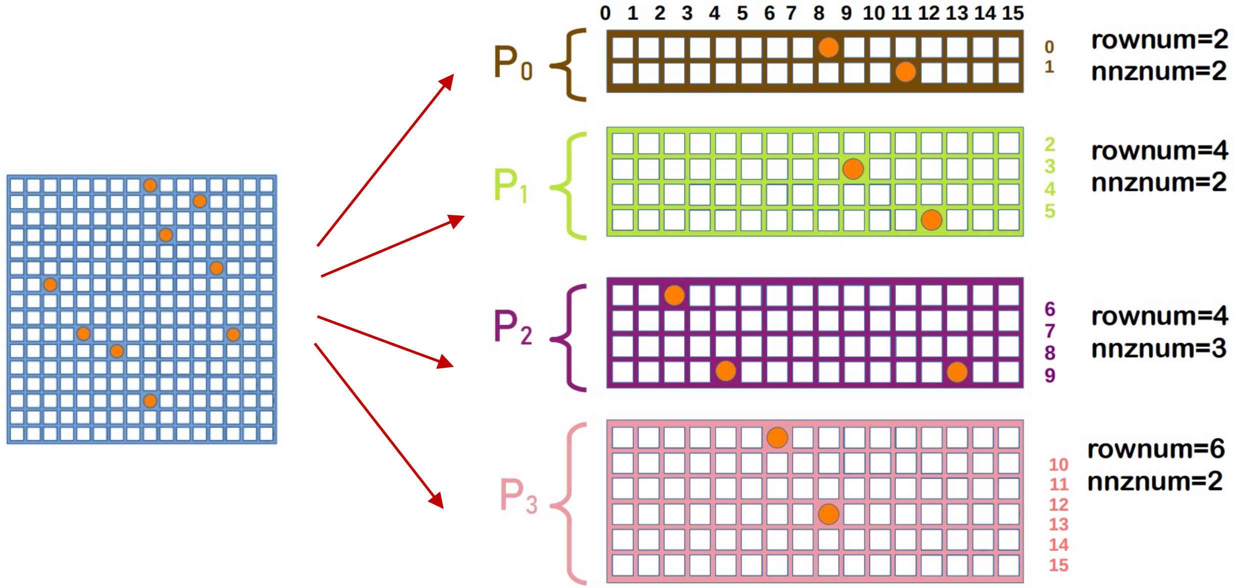
Finally, the matrix is divided into **local matrix** and **remote matrix** based on whether the non-zero elements are within the diagonal block

DistSpMV_Balanced

step 2: Adjust the number of columns of the diagonal block and partition matrix

Strategy 2 **: Local matrix and remote matrix are processed separately.**



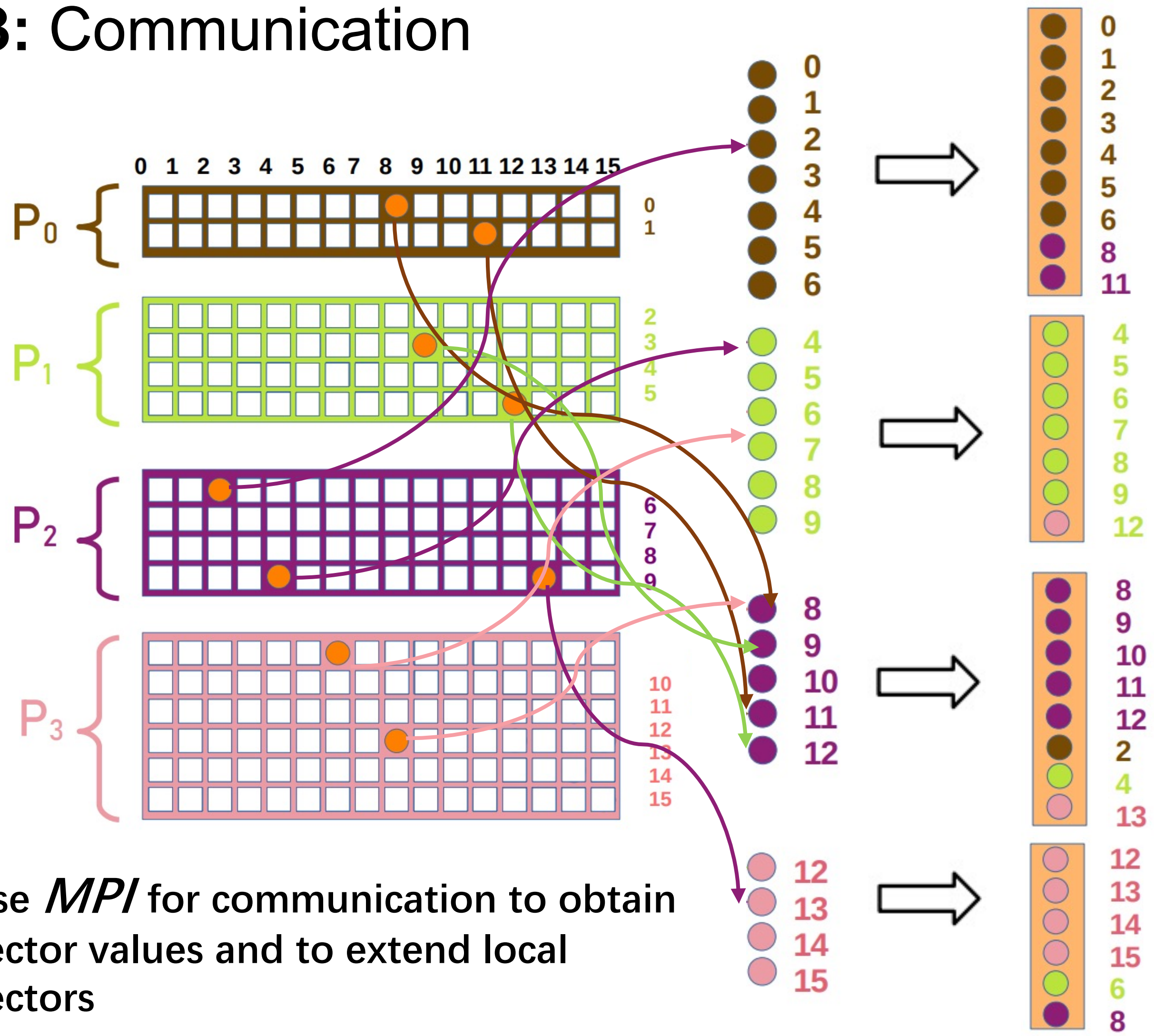


The local matrix can be directly divided based on the row equalization strategy.

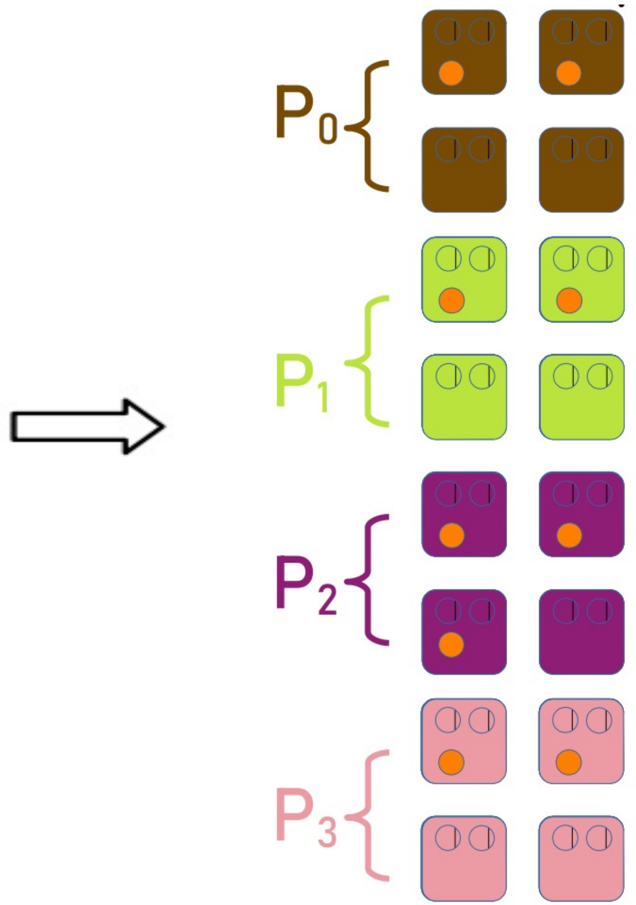
The remote matrix is divided based on **the principle of non zero element averaging**, further achieving **communication and computational load balancing.**

DistSpMV_Balanced

step 3: Communication



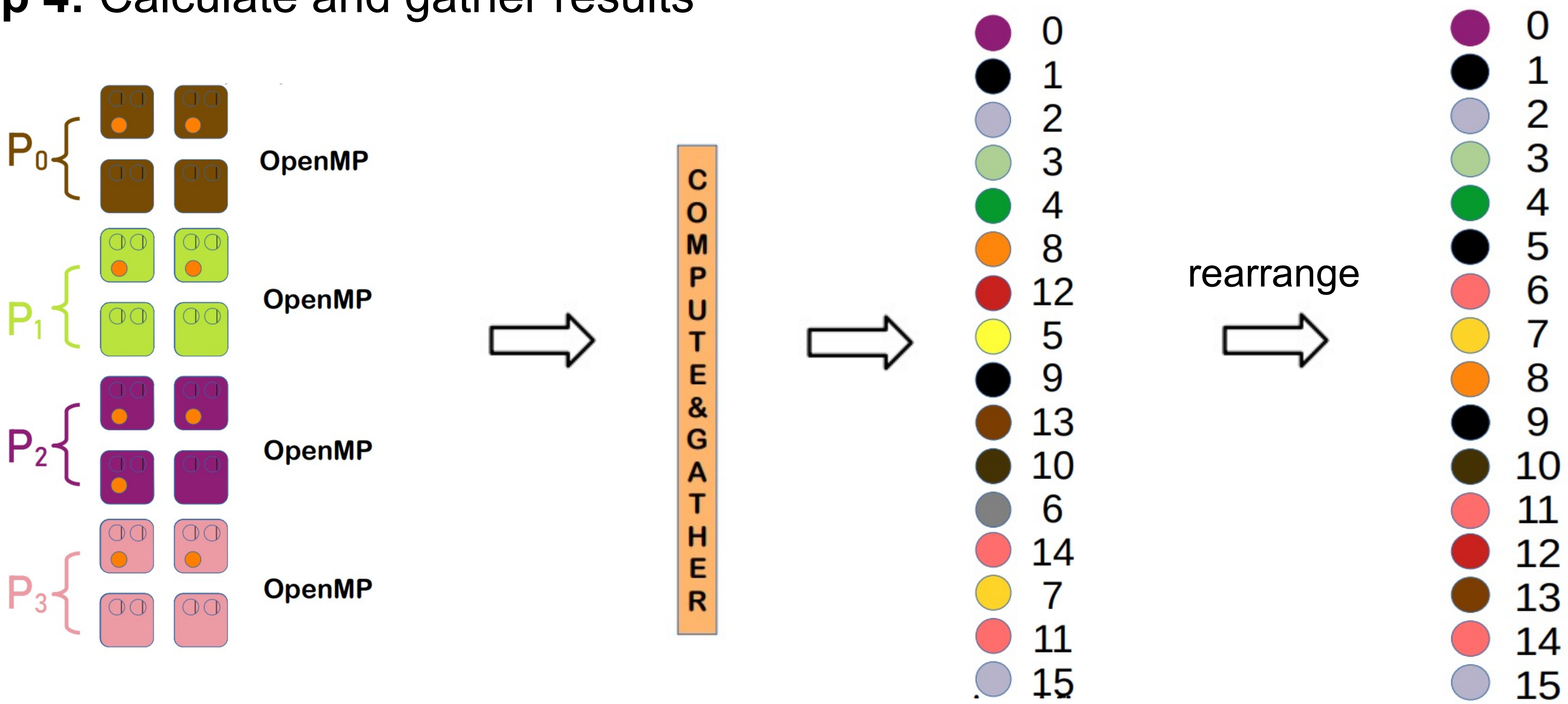
Use *MPI* for communication to obtain vector values and to extend local vectors



Divide nonzero elements equally into each thread.

DistSpMV_Balanced

step 4: Calculate and gather results



Calculation completed!

Outline

- Introduction
- Motivation
- Algorithm
- Experiment
- Conclusion

Experiment

Experimental platform
AMD 32-core EPYC 7551 CPU and 128GB DRAM

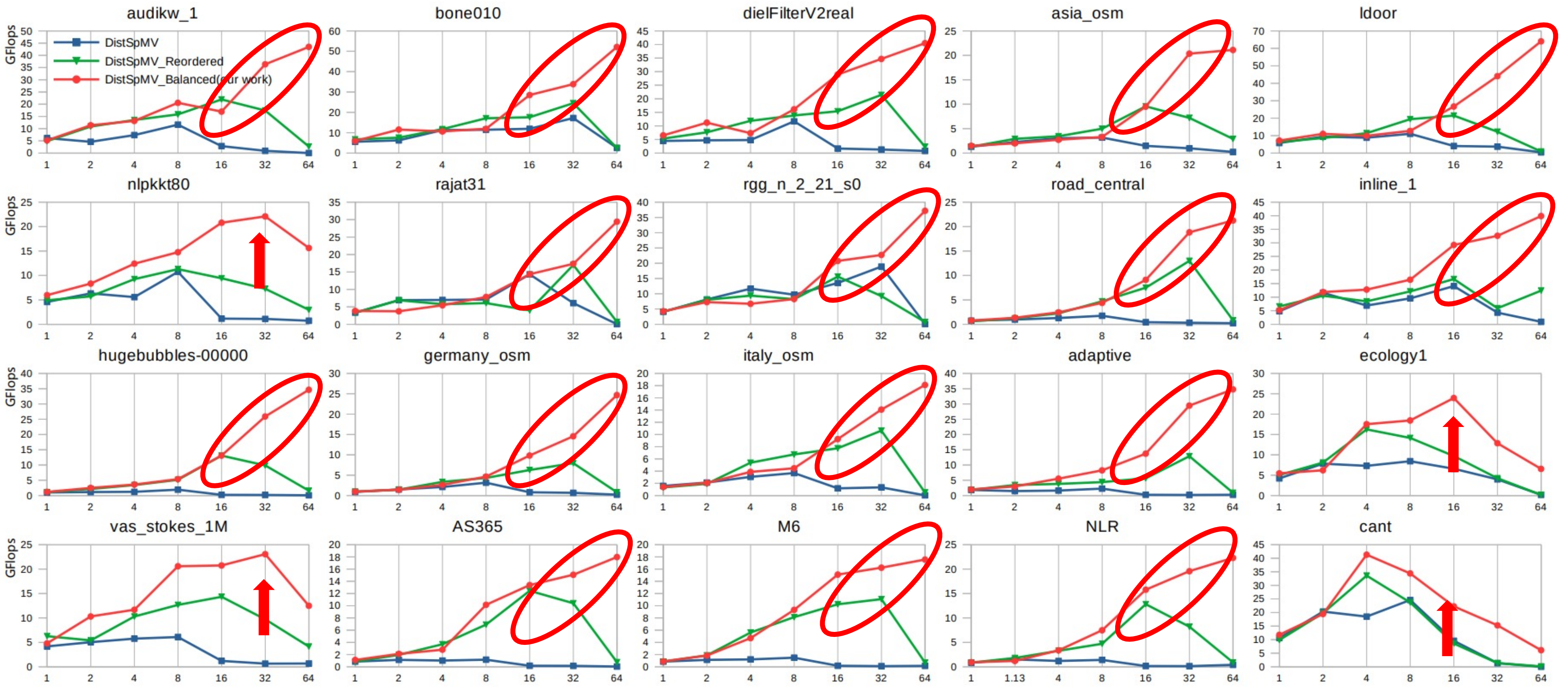
Dataset (20 representative matrices in SuiteSparse Matrix Collection[2])

Matrix	Plot	Size	*nnz*	Matrix	Plot	Size	*nnz*
cant		62.4K×62.4K	4M	inline_1		503.7K×503.7K	36.8M
bone010		986.7K×986.7K	47.8M	hugebubbles00000		18.3×18.3M	54.9M
rajat31		4.6M×4.6M	20.3M	germany_osm		11.5M×11.5M	24.7M
ecology1		1M×1M	4.9M	italy_osm		6.6M×6.6M	14M
asia_osm		11.9M×11.9M	25.4M	adaptive		6.8M×6.8M	27.2M
ldoor		852.2K×952.2K	42.4M	vas_stokes_1M		1M×1M	34.7M
nlpkkt80		1M×1M	28.1M	AS365		3.7M×3.7M	22.7M
dielFilterV2real		1.1M×1.1M	48.5M	M6		3.5M×3.5M	21M
rgg_n_2_21_s0		2M×2M	28.9M	NLR		4.1M×4.1M	24.9M
road_central		14M×14M	33.8M	audikw_1		943.6K×943.6K	77.6M

[2] T. A. Davis and Y. Hu, “The university of florida sparse matrix collection,” ACM Transactions on Mathematical Software (TOMS), vol. 38, no. 1, pp. 1–25, 2011.

Experiment

Performance Comparison of Three Algorithms



1. **DistSpMV:**
   pure distributed SpMV.
2. **DistSpMV_Reordered:**
   distributed SpMV using graph partition tool ***METIS***[1].
3. **DistSpMV_Balanced:**
   our work.

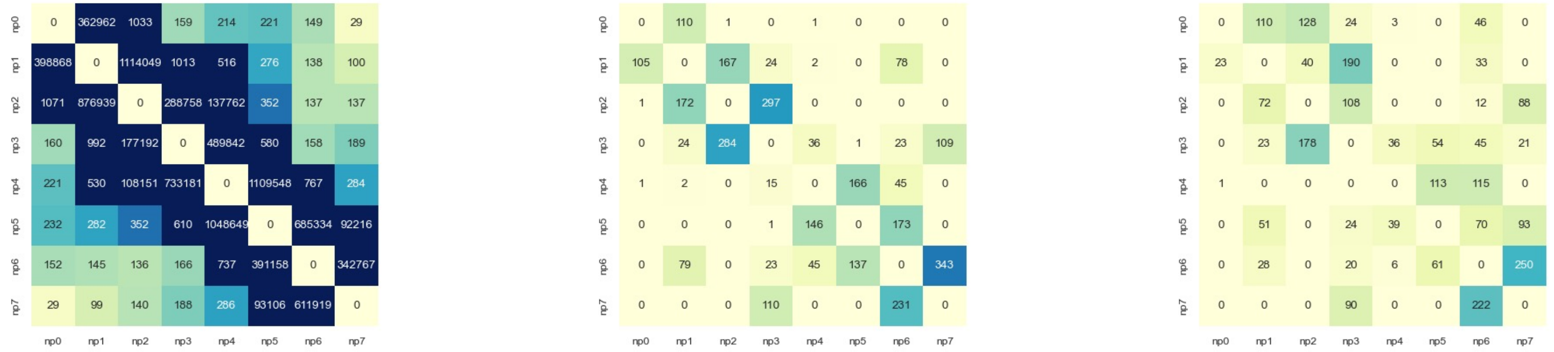
- With the expansion of the number of processes, the performance of most matrices has been improved.
- Although the performance of the other four matrices has decreased, the overall performance compared with the other two algorithms still greatly improved.

- Compared with DistSpMV and DistSpMV_Reordered, our algorithm achieves on average **77.20x** and **5.18x** (up to **460.52x** and **27.50x**) speedups, respectively.

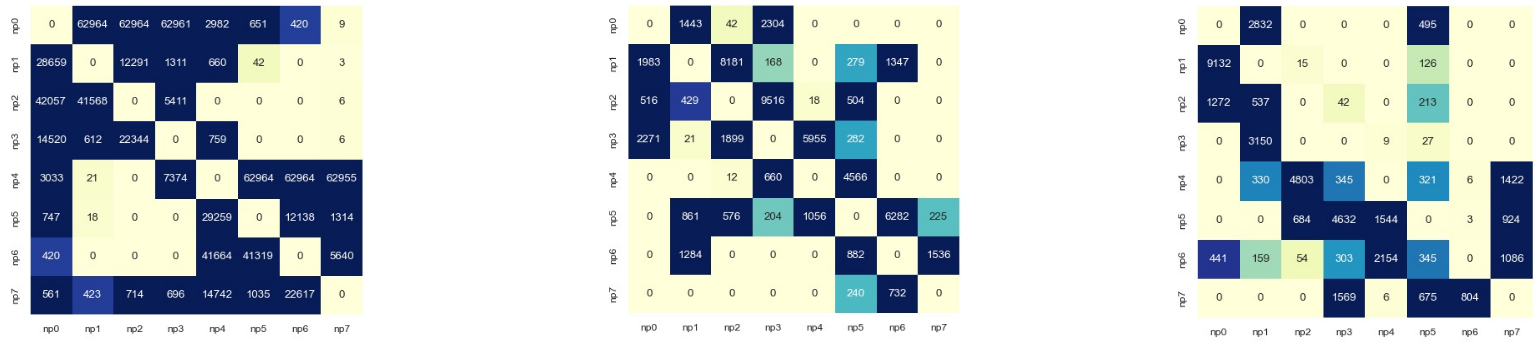
[1] G. Karypis and V. Kumar, “Analysis of multilevel graph partitioning,” in SC ’95, 1995, p. 29–es

Experiment

Analysis 1: Communication volume Comparison of Three Algorithms



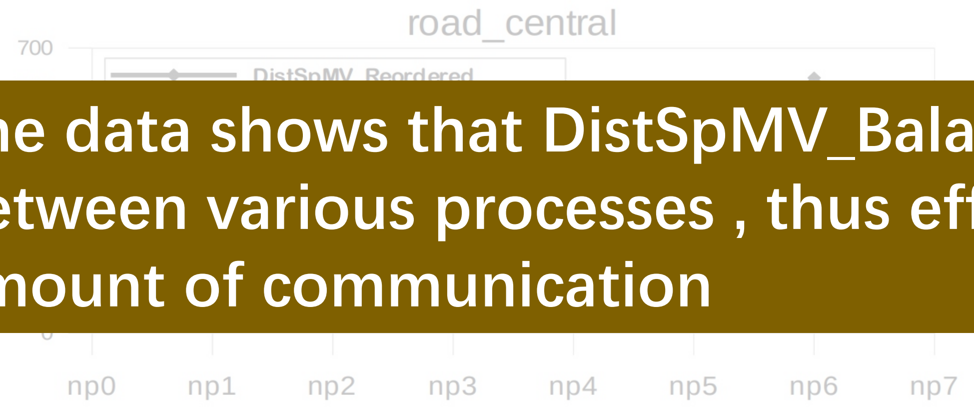
From left to right is three heat maps of the traffic between various processes in ***DistSpMV***, ***DistSpMV_Reordered***, and ***DistSpMV_Balanced*** for matrix **road_central**

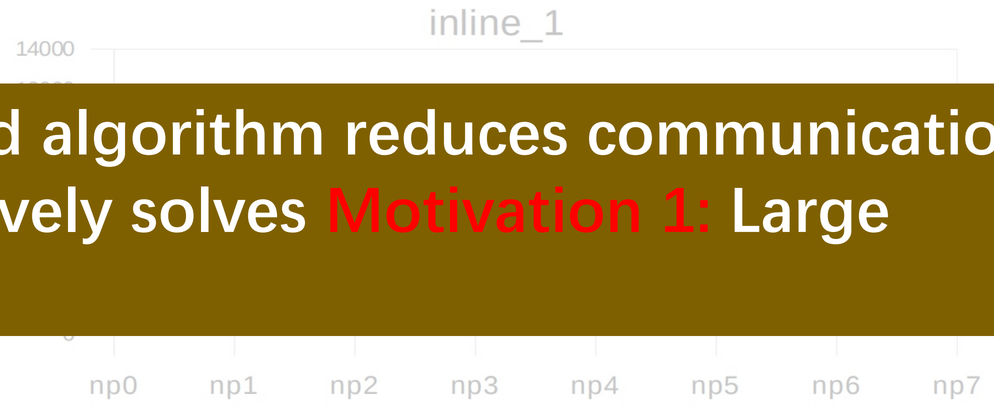


From left to right is three heat maps of the traffic between various processes in ***DistSpMV***, ***DistSpMV_Reordered***, and ***DistSpMV_Balanced*** for matrix **inline_1**

Experiment

Analysis 1: Further comparison of communication volume between *DistSpMV_Reordered* and *DistSpMV_Balanced.*





The data shows that DistSpMV_Balanced algorithm reduces communication between various processes , thus effectively solves Motivation 1: Large amount of communication

Experiment

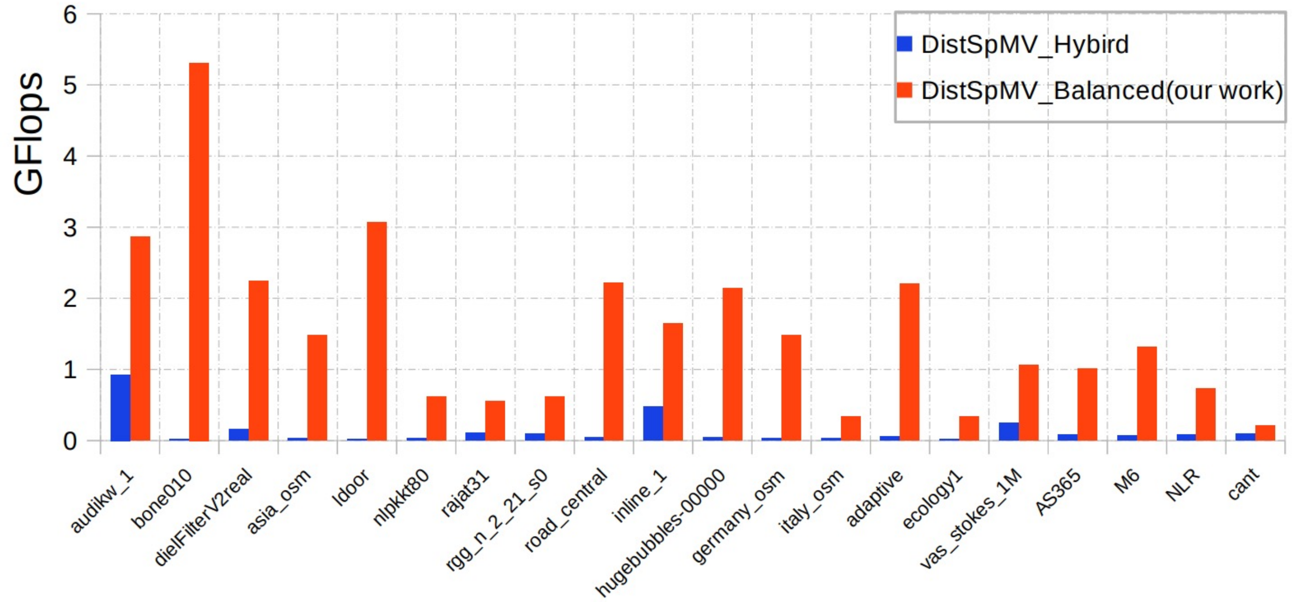
Analysis 2: Computation volume of the remote matrix comparison of *DistSpMV_Reordered* and *DistSpMV_Balanced*

Thus DistSpMV_Balanced algorithm effectively solves Motivation 2: Imbalanced communication and Motivation 3: Imbalanced computation

- In our algorithm, the computational load of each remote matrix tends to be **straight**, indicating that the algorithm has largely achieved communication load balancing and computational load balancing !

Experiment

Comparison with Existing Work *DistSpMV_Hybrid* developed by Page and Kogge [2]



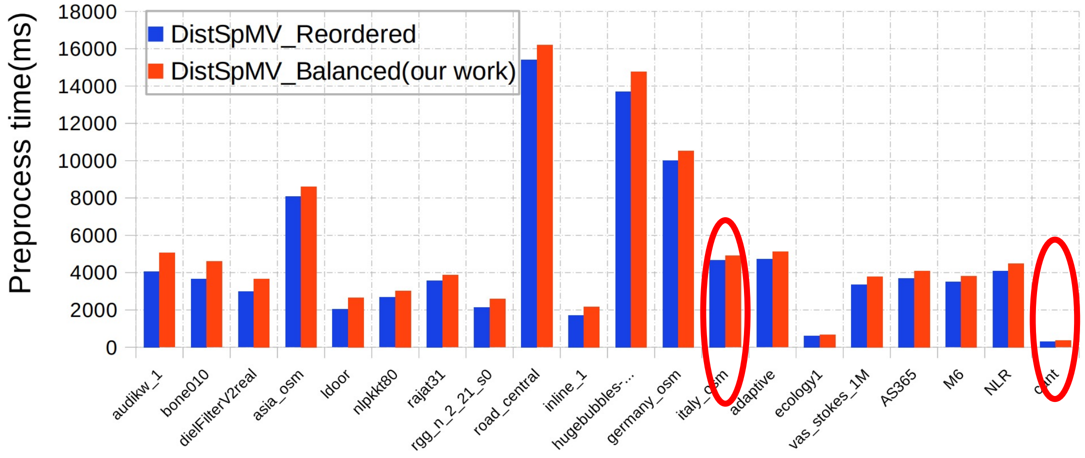
Performance comparison between DistSpMV_Balanced and DistSpMV _Hybrid with 256 cores (64 processes ×4 threads).

- DistSpMV_Balanced achieves an average acceleration ratio of **19.56x** (up to **48.49x**). The performance of **all matrices** has been greatly improved.

[2] B. A. Page and P. M. Kogge, "Scalability of hybrid sparse matrix dense vector (spmv) multiplication," in 2018 International Conference on High Performance Computing & Simulation (HPCS). IEEE, 2018, pp. 406– 414

Experiment

Comparison of Preprocessing Overhead *between DistSpMV_Reordered* and *DistSpMV_Balanced.*



- the nonzero element distribution diversity of different matrices leads to different preprocessing time cost, and the cost changes of the two algorithms are roughly the same.
- At the same time, overall, our optimization on top of the graph partitioning does not cost much additional overhead.
- Among these 20 matrices, the maximum preprocessing cost is **1.31x** that of ***DistSpMV_Reordered*** algorithm (at matrix 'cant'), and the minimum preprocessing cost is only **1.05x** (at matrix 'italy osm')

Outline

- Introduction
- Motivation
- Algorithm
- Experiment
- Conclusion

Conclusion

- We identify that matrix reordering techniques are not adequate to achieve good computation and communication balancing, and thus more schemes are required.

- We design an algorithm called ***DistSpMV_Balanced*** that reorganizes the distribution of sparse matrix on compute nodes for balanced computation and communication.

- We evaluate the new algorithm by using 20 representative sparse matrices on a 256-core cluster, and bring significant speedups over existing work.

Thanks for your time!

Balancing Computation and Communication in Distributed Sparse Matrix-Vector Multiplication

Hongli Mi, Xiangrui Yu, Xiaosong Yu, Shuangyuan Wu and Weifeng Liu

Super Scientific Software Laboratory, China University of Petroleum-Beijing, China

3 May 2023

https://github.com/HongliMi/DistSpMV_Balanced