PaStiX: Distributed Interface

Supervisors: Mathieu Faverge and Pierre Ramet

Alycia Lisito

Summary

Inria

01

PaStiX

PaStiX = Parallel Sparse Linear Algebra Solver

- Linear Algebra Solver
  - > Solves $Ax = b$
- Sparse
  - > Matrix with a lot of zero elements
- Parallel
  - > Several schedulers:
    - Sequential
    - Static
    - Dynamic
    - StarPU
    - Parsec
  - > MPI

Inria

4 steps

- Analyse
  - > Permutation P
  - > Blocks
- Numerical Factorisation
  - > $\mathbf{A} \rightarrow \mathbf{PAP^T} \rightarrow LU$
- Solve
  - > $\mathbf{b} \rightarrow \mathbf{Pb}$
  - > Solves $Ly = Pb$
  - > Solves $UPx = y$
  - > $\mathbf{Px} \rightarrow \mathbf{x}$
- Refinement
  - > Refines the solution x

Inria

The matrix format: *CSC* format

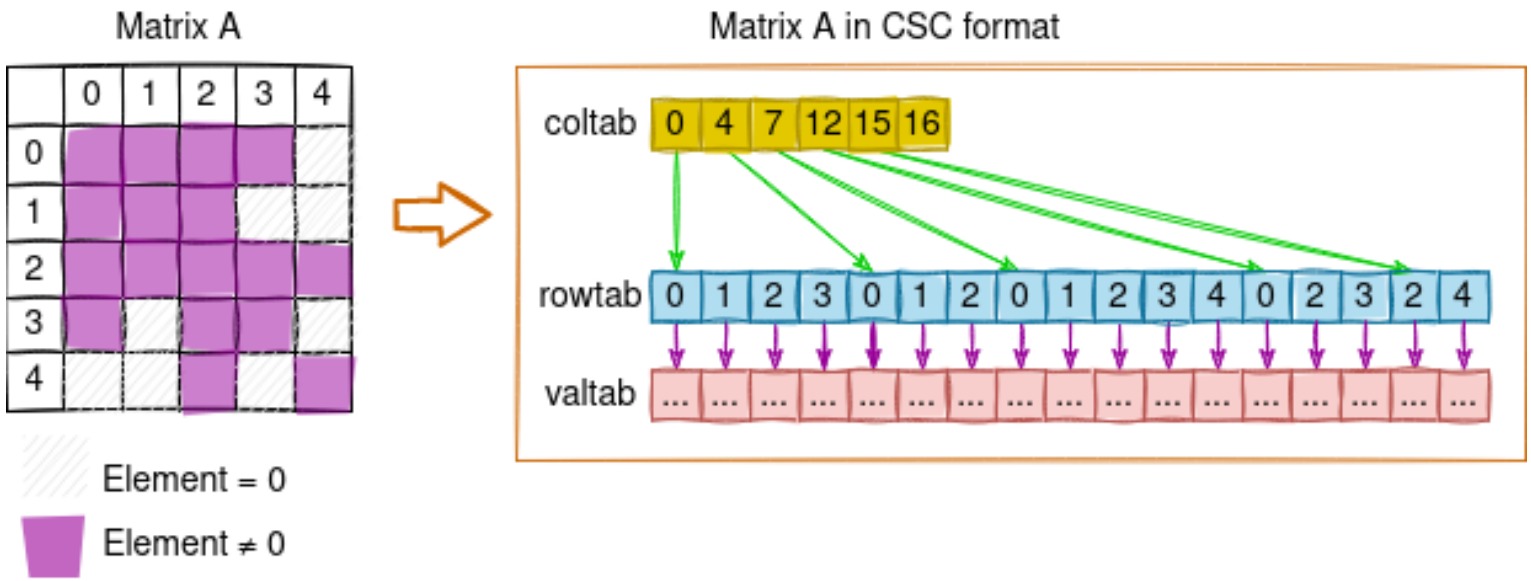


Figure: Example of a matrix A in the *CSC* format (A_{CSC})

The *CSC* format in distributed memory

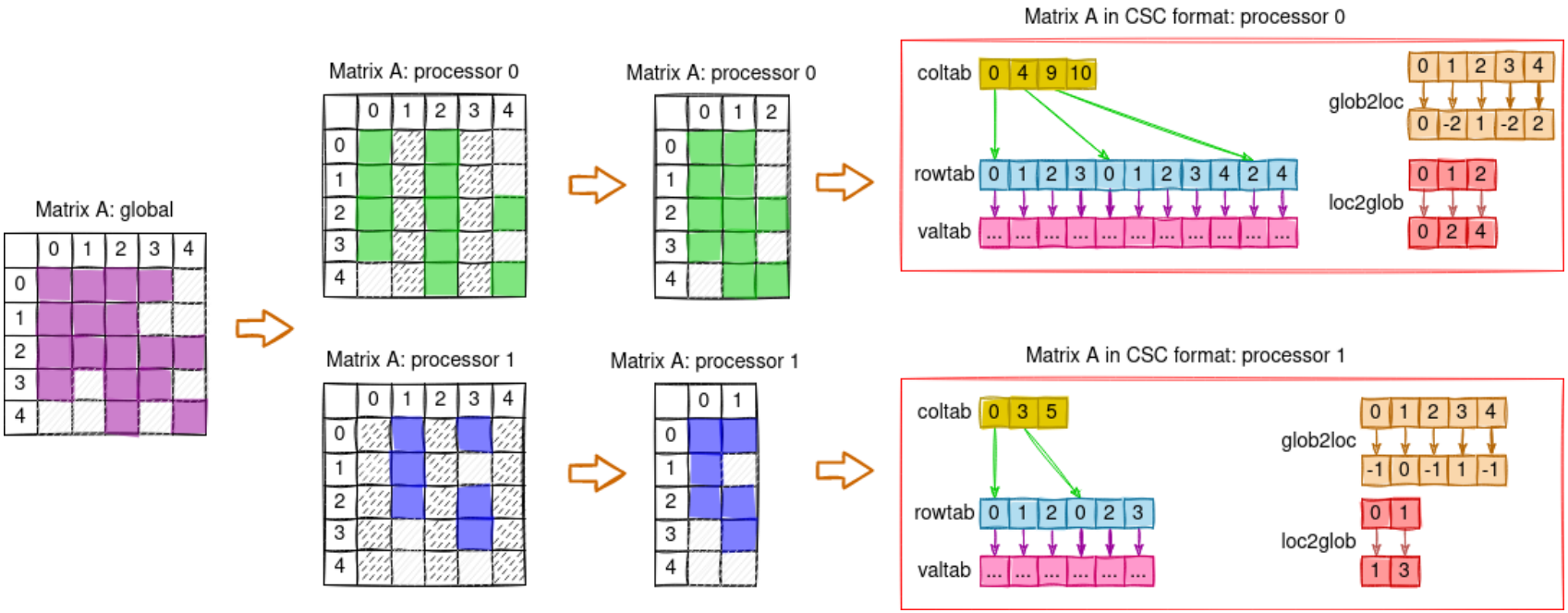


Figure: Example of a **distributed** matrix A in the *CSC* format (A_{CSC})

The block format: *BCSC* format

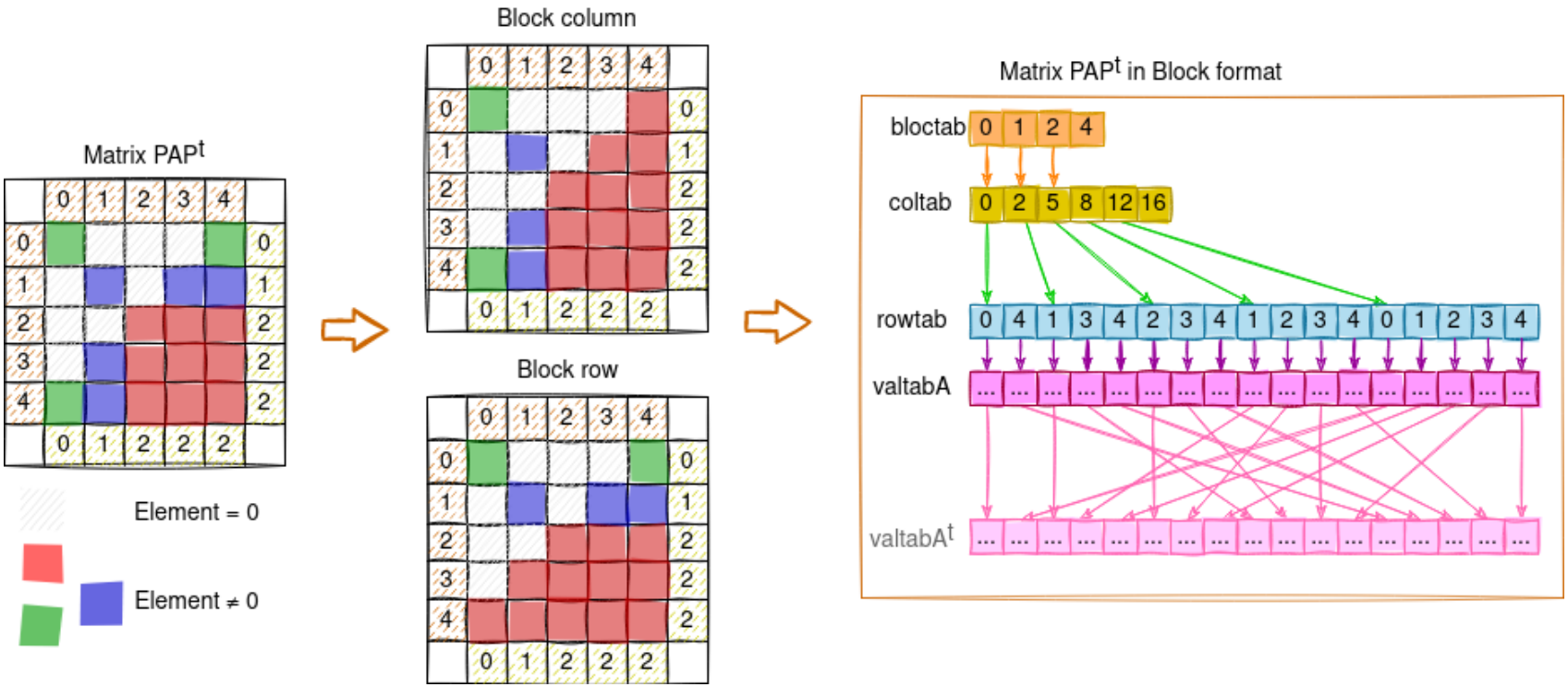


Figure: Example of a matrix PAP^T in the *BCSC* format (PAP^T_{BCSC})

Degree of Freedom: Single

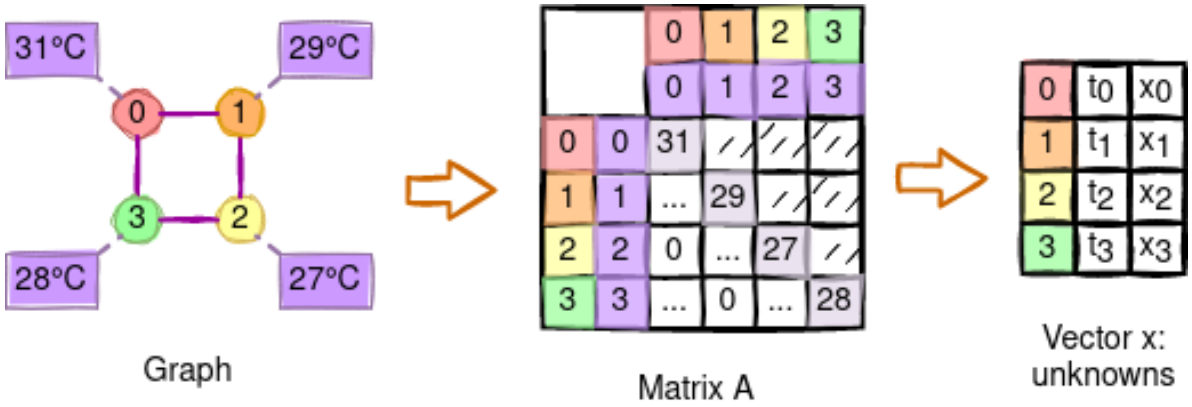


Figure: Example of a matrix A with a Single DoF

Degree of Freedom: Multiple constant

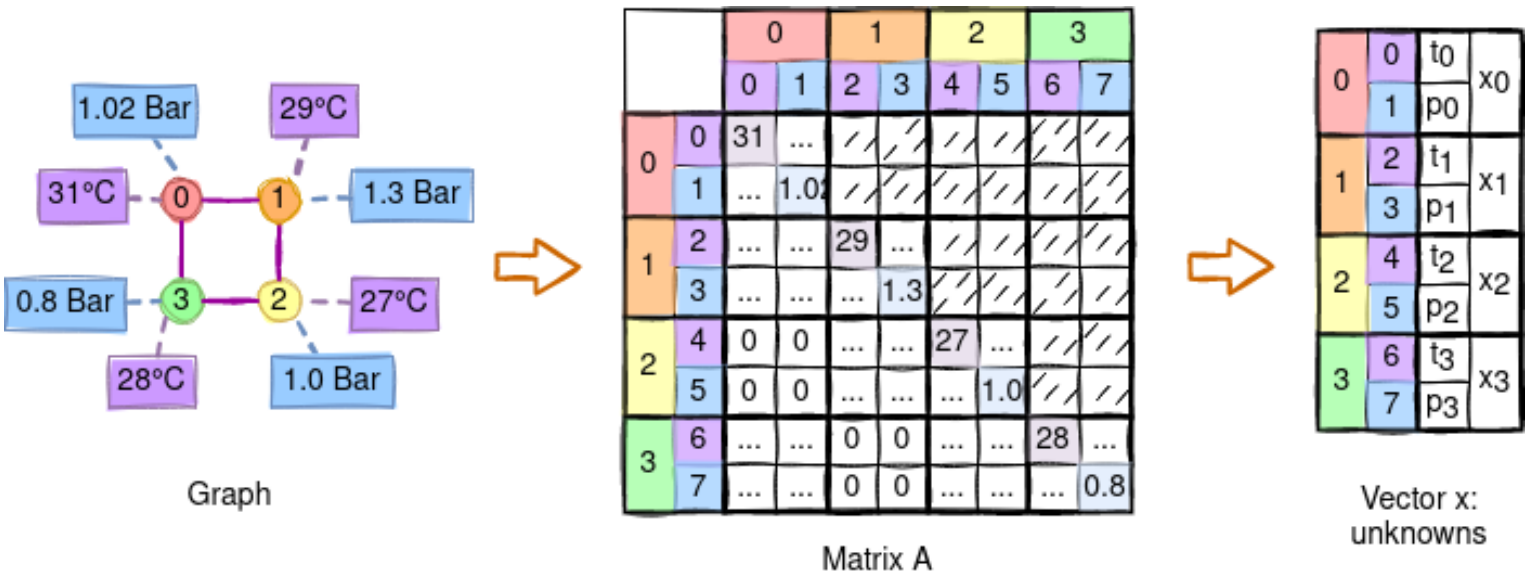
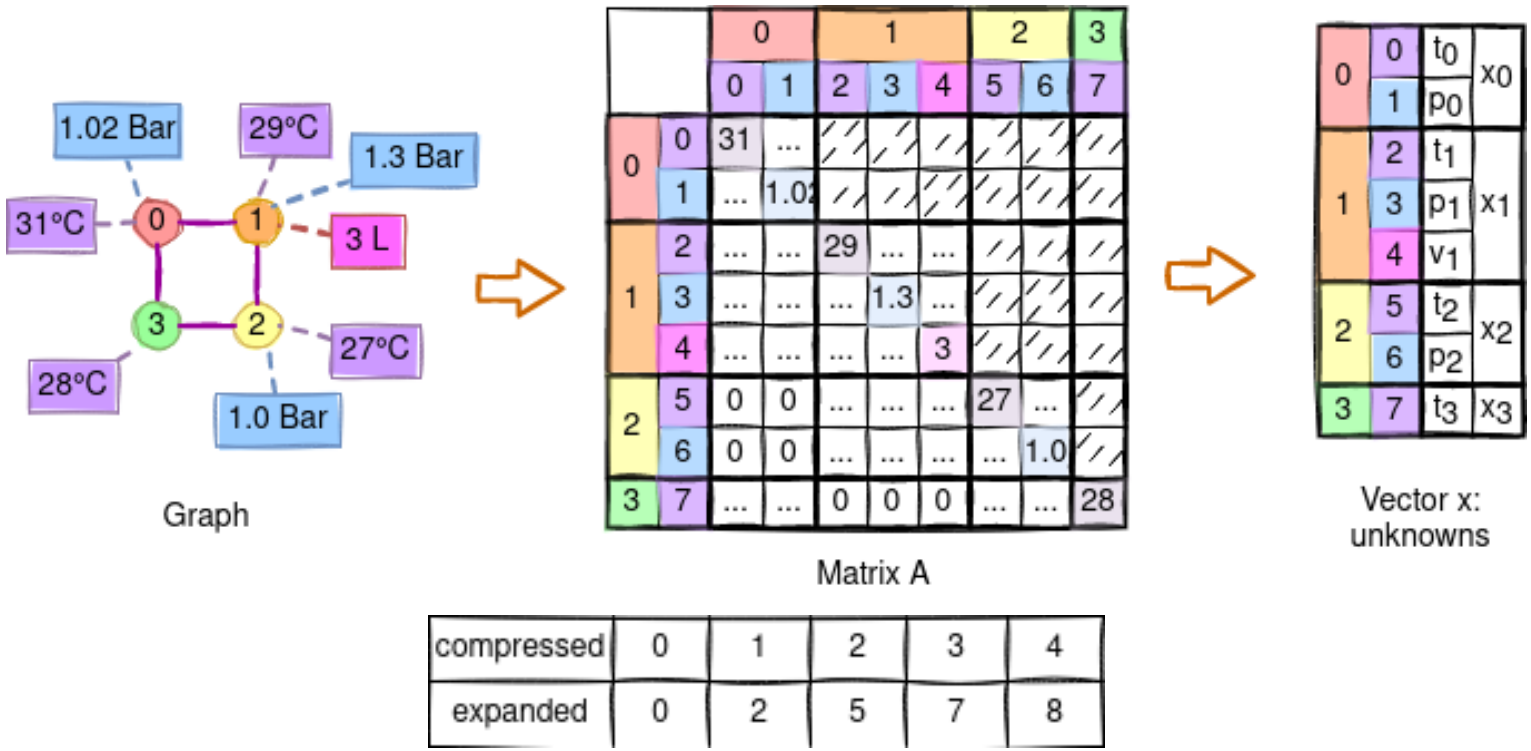


Figure: Example of a matrix A with a Multiple Constant DoF

Degree of Freedom: Multiple variadic



compressed	0	1	2	3	4
expanded	0	2	5	7	8

Figure: Example of a matrix A with a Multiple Variadic DoF

Goal of my work

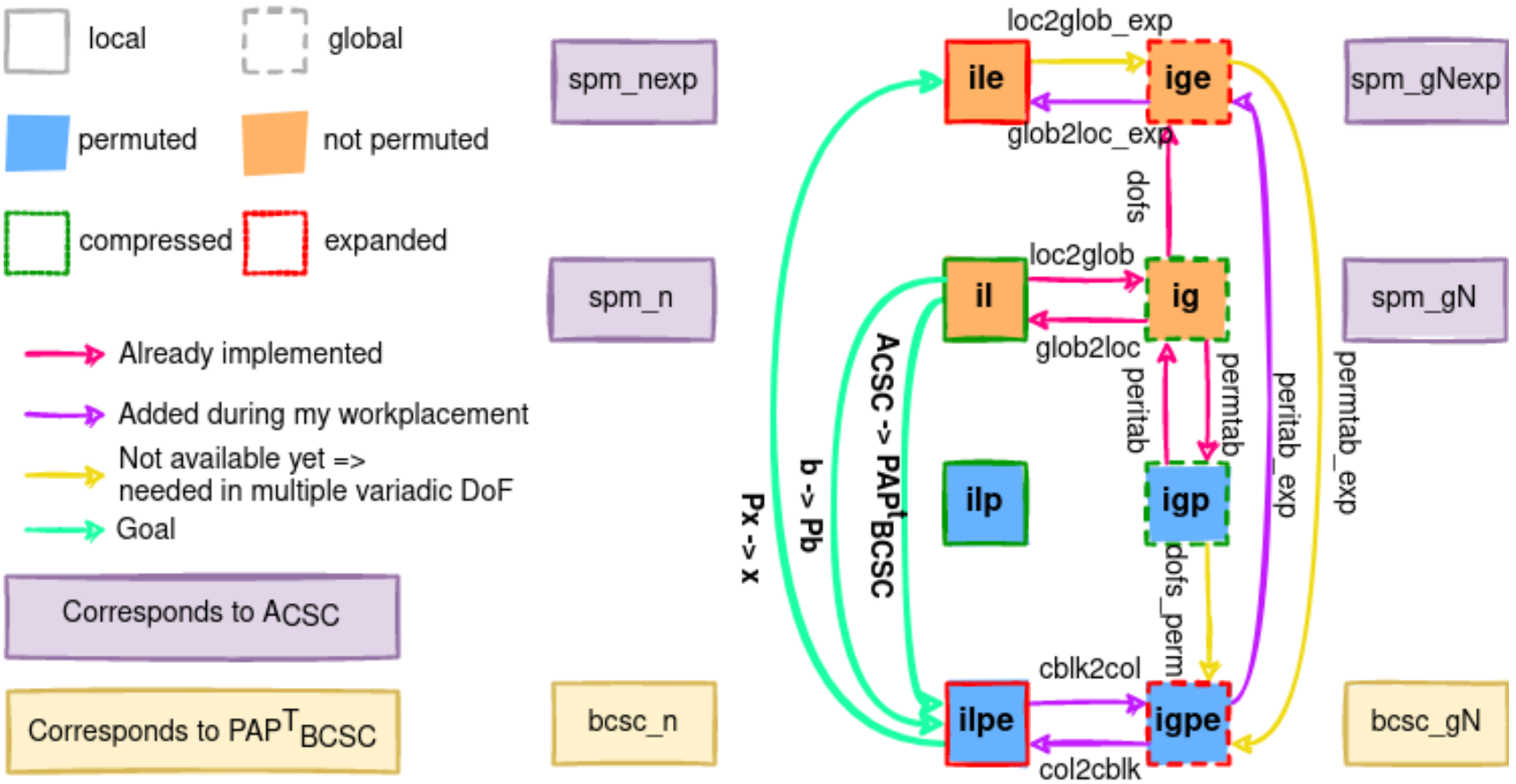


Figure: The different types of **indexes**

02

Matrix permutation

Matrix permutation: $\mathbf{A \rightarrow PAP^T}$

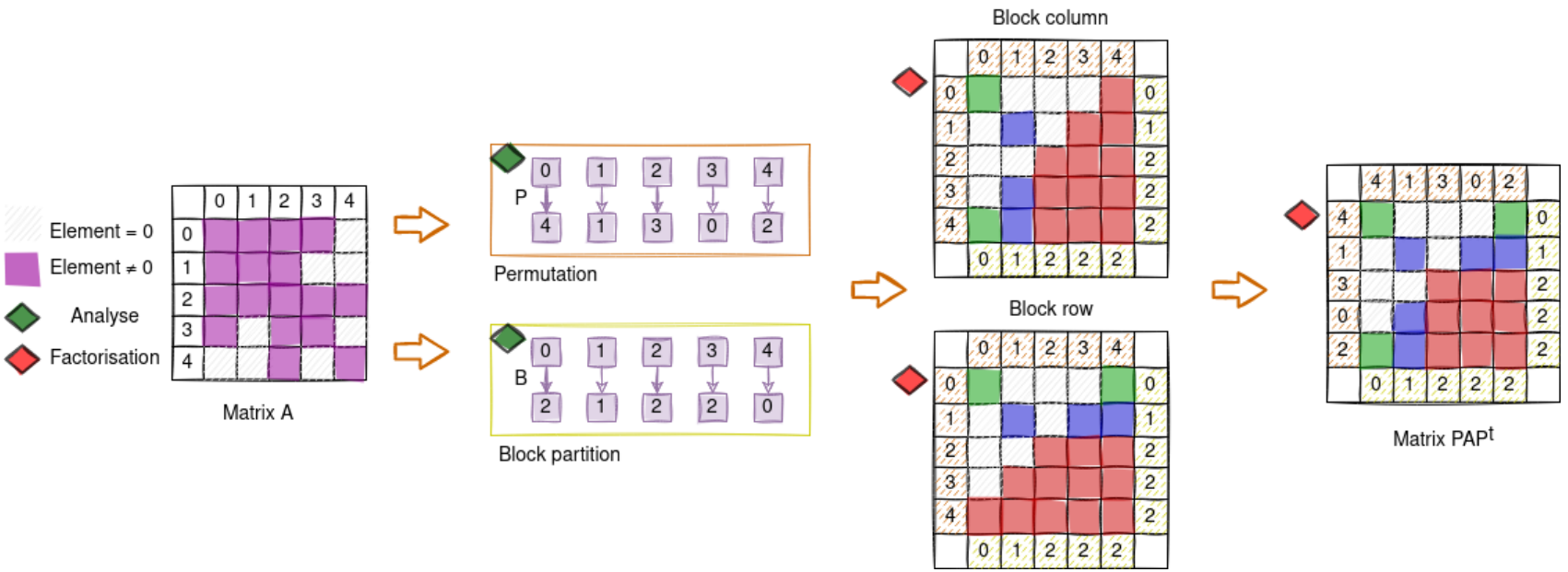


Figure: Goal of the permutation and block storage

From A_{CSC} to PAP^T_{BCSC}

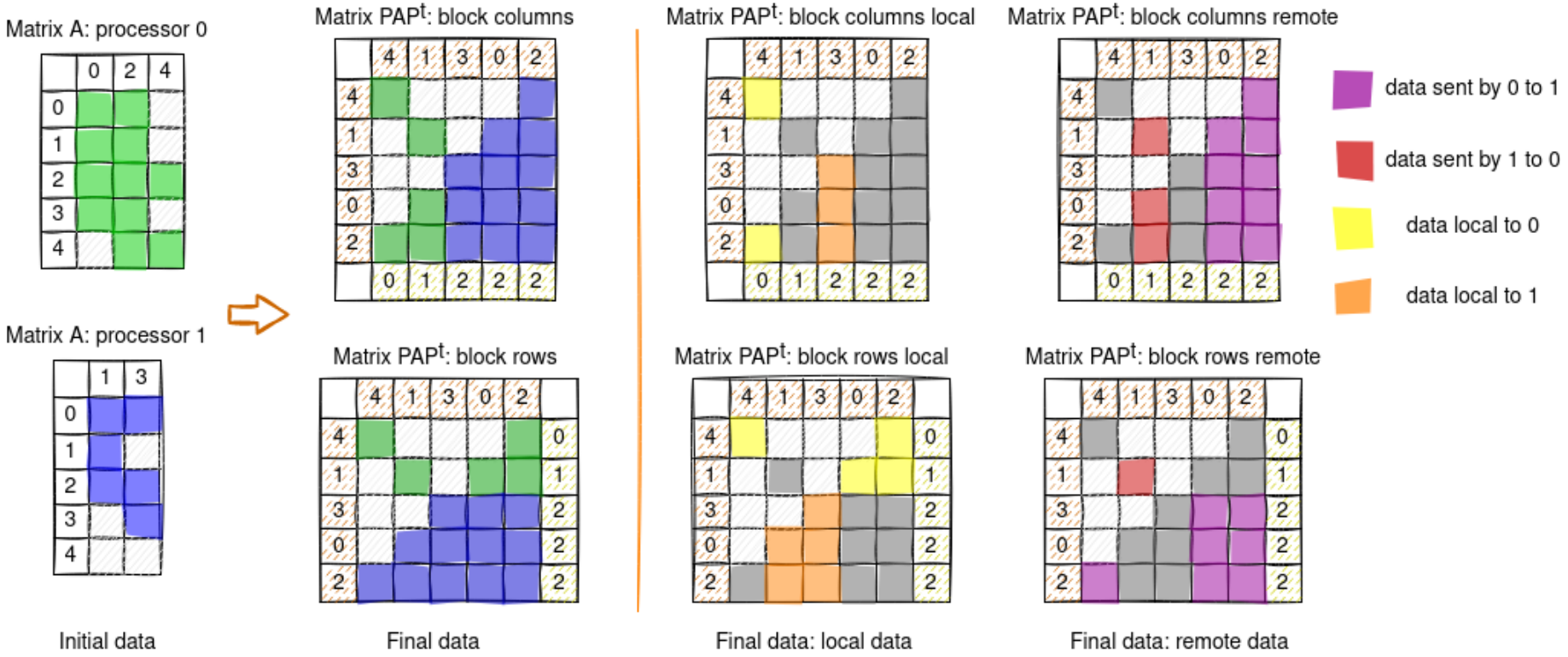


Figure: Data exchanged between the processor in the distributed memory case

Processors communications: the difficulties

- How much data will I send ?
- How much data will I receive ?
- Where can I store the buffers ?

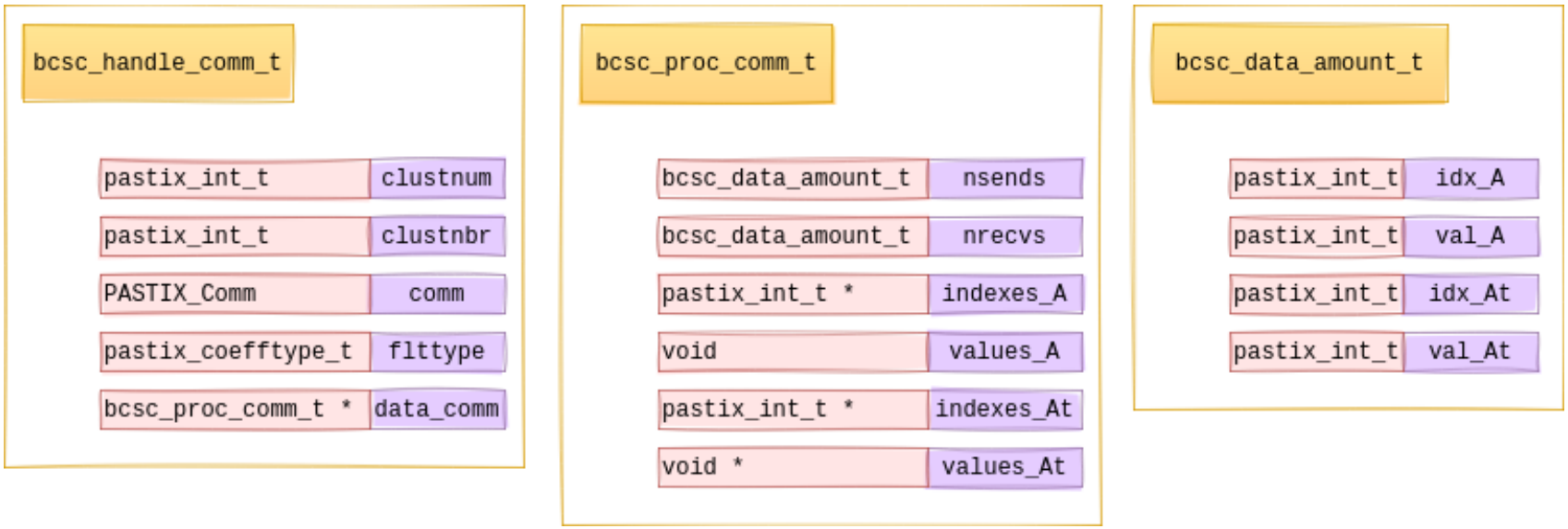


Figure: Structure to handle the processors communications

How are the different indexes handled ?

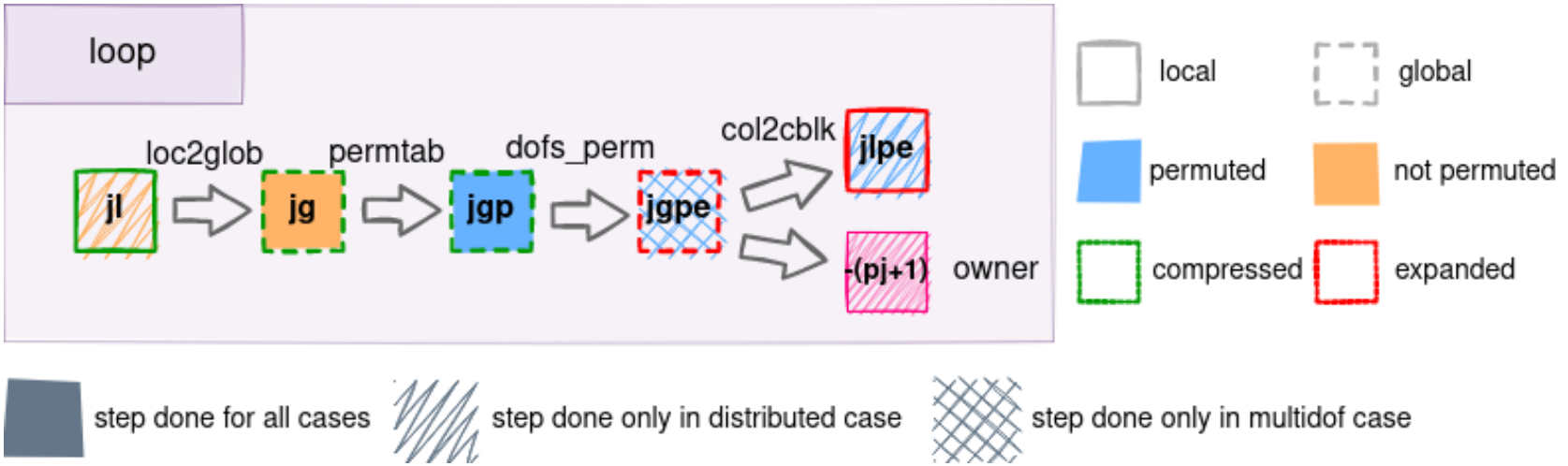


Figure: Algorithm in terms of indexes conversion

03

Vector permutation

Permutation of the vector: the replicated case

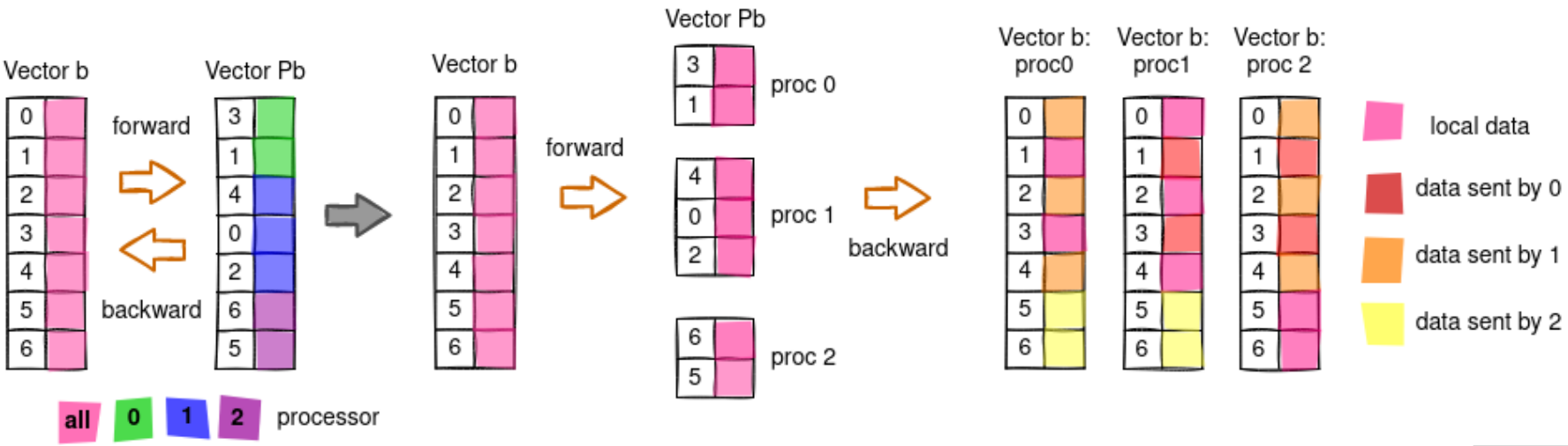


Figure: Data exchanged in the replicated case

Permutation of the vector: the distributed case

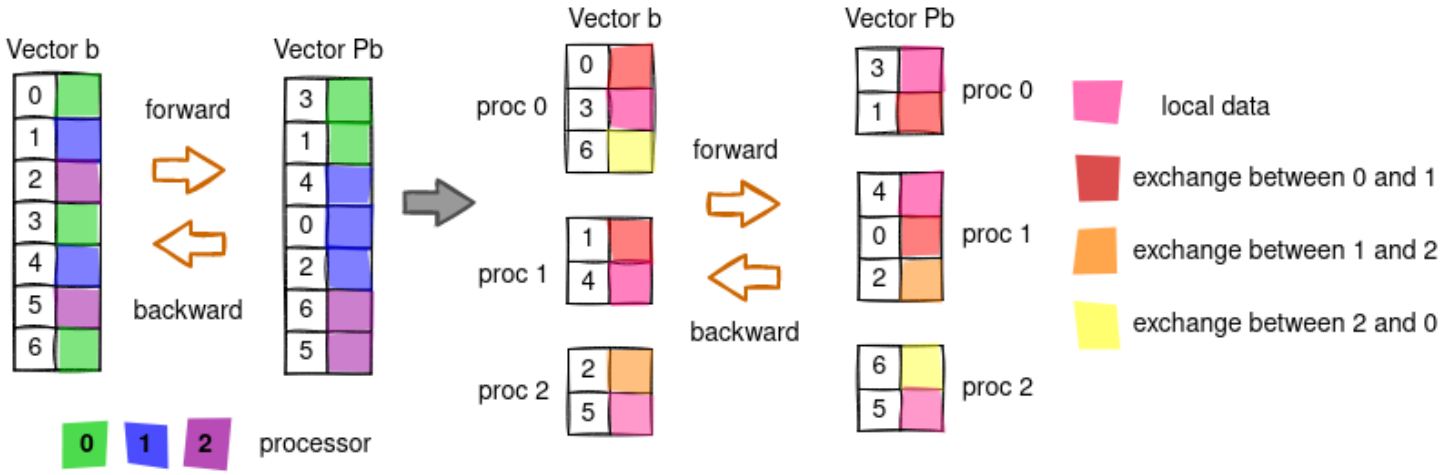


Figure: Data exchanged in the distributed case

04

Performances

Performances of the matrix permutation

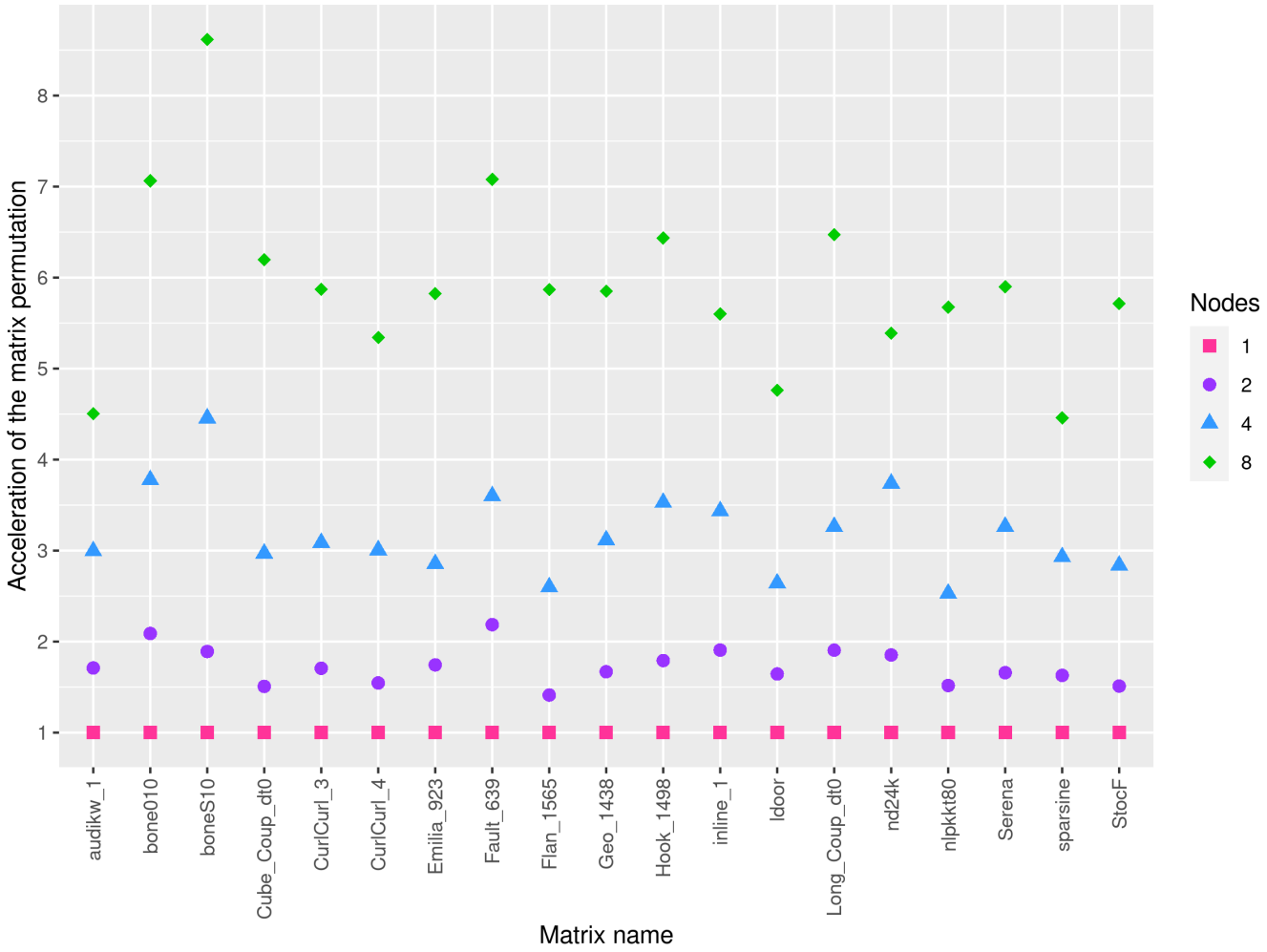


Figure: Acceleration of the matrix permutation on 2, 4, and 8 nodes for different matrices

05

Conclusion

Inria

Conclusion

Goals achieved

- Matrix permutation in distributed memory
- Matrix permutation with multiple constant DoF
- Vector permutation in replicated to distributed case
- Vector permutation in distributed to distributed case
- Distributed sequential solve

Next steps

- Improve the *MPI* communications
- Implement the distributed multi-threaded solve
- Implement the matrix permutation with variadic DoF
- Implement the vector permutation with variadic DoF

Inria