21st Summer School of PARALLEL COMPUTING

July 2 - 13, 2012 (Italian)
September 3 - 14, 2012 (English)

MPI Exercises

SuperComputing, Applications and Innovation Department

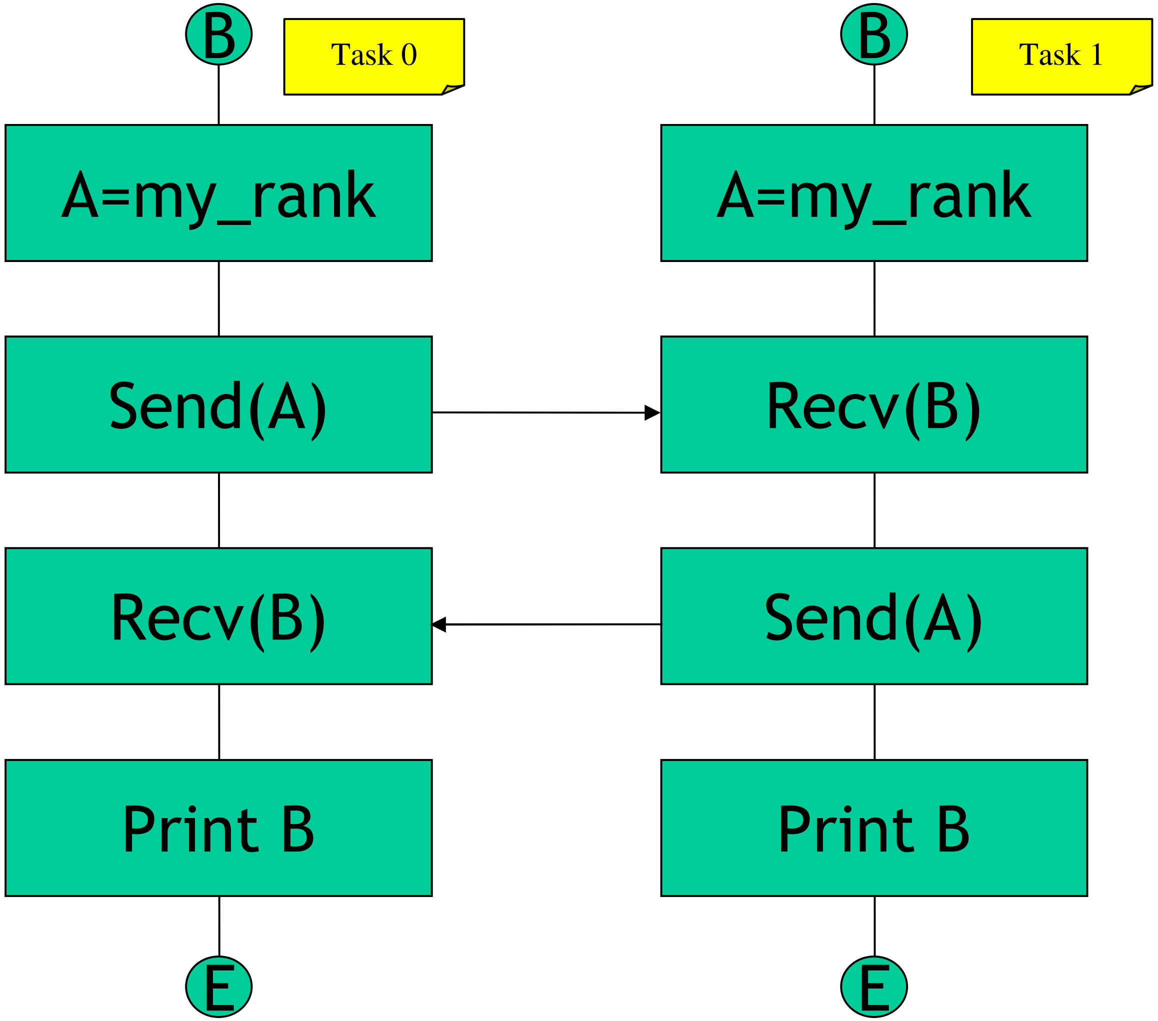
CINECA

Exercise 1

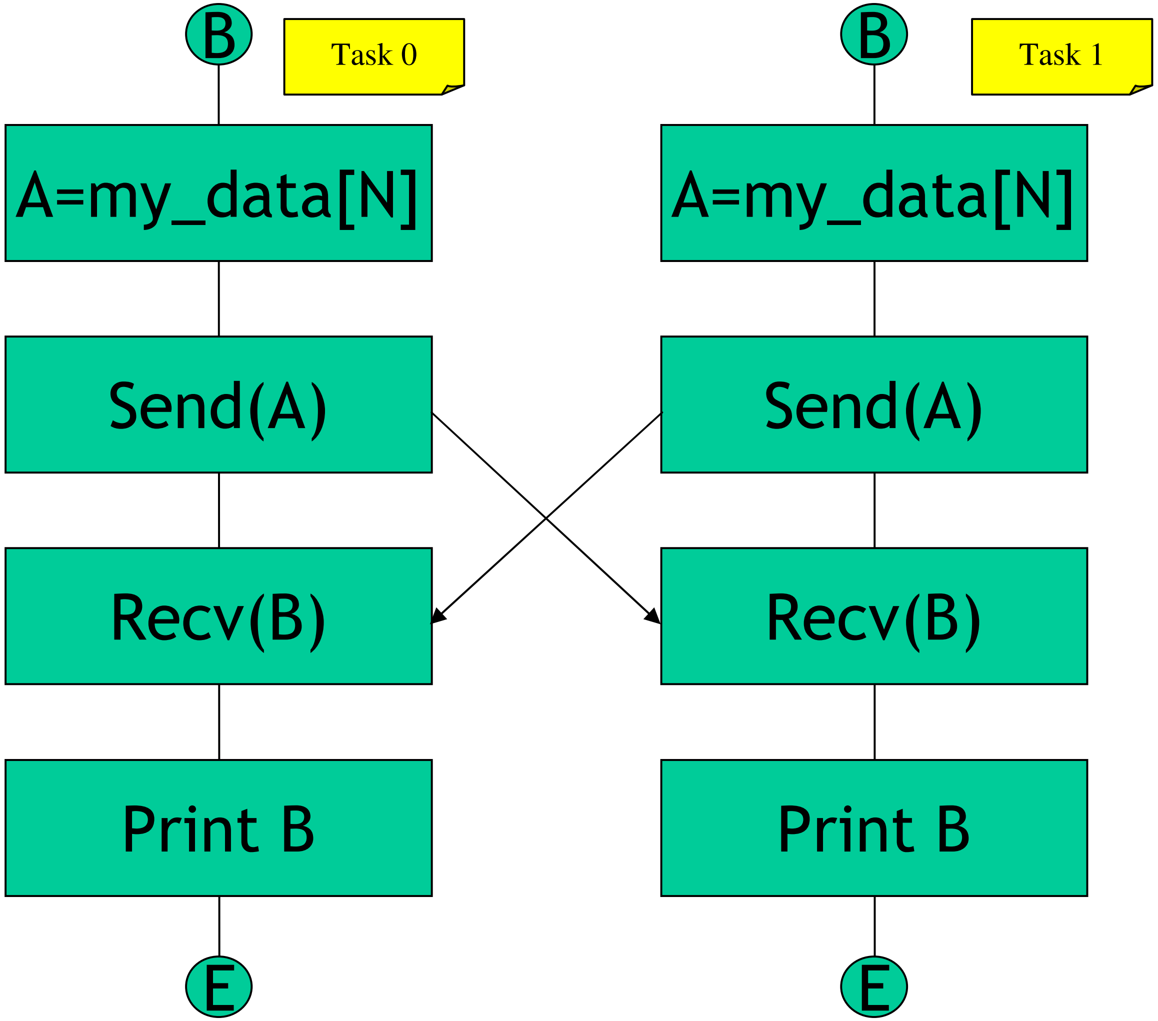
Using MPI, print:

1. Hello world
2. Hello world, I am proc X of total Y (from 1 to total 4 tasks)
3. Submit it as a batch job

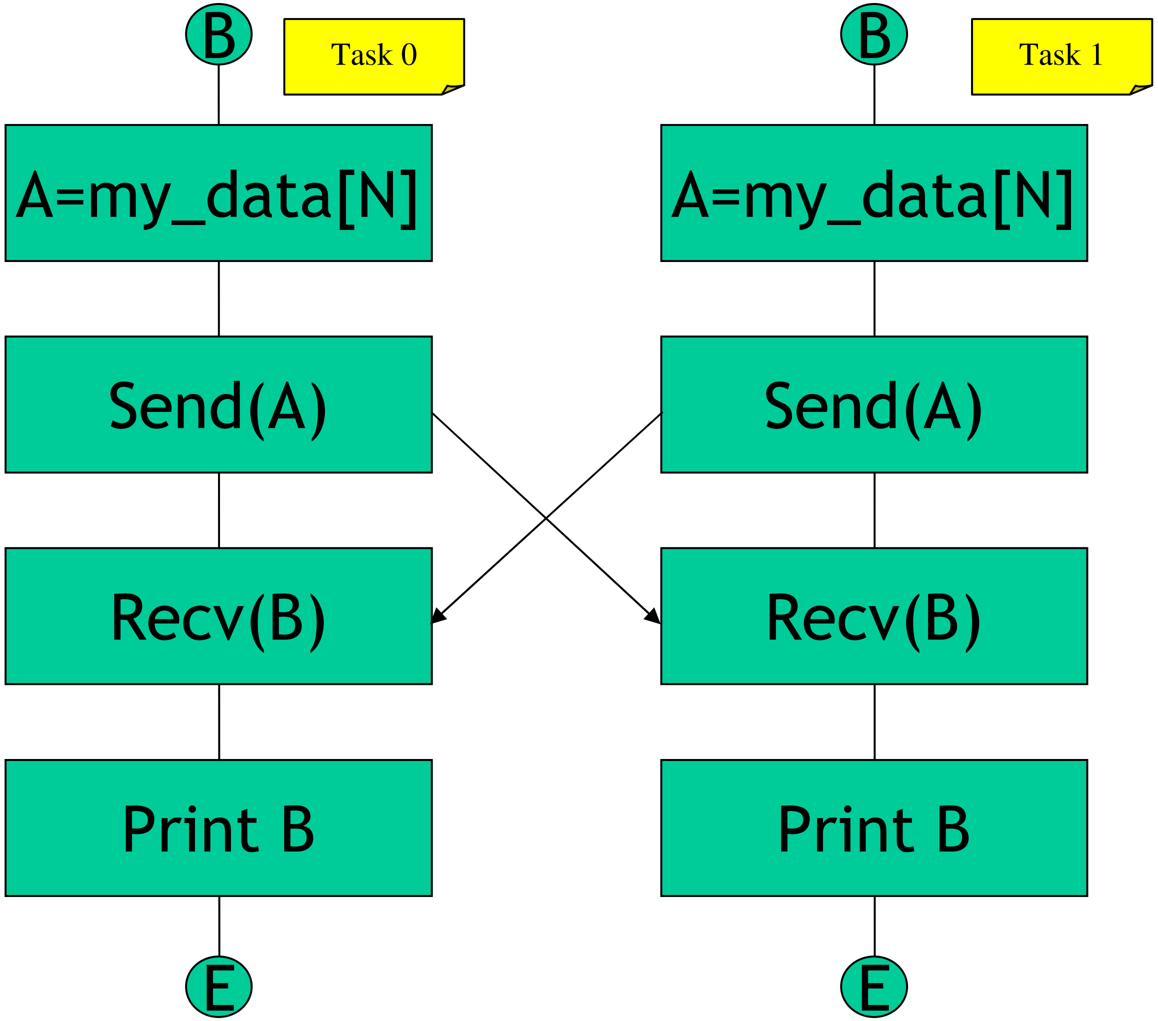
Exercise 2:
a) Data exchange



b) try an array[N] to see a deadlock!

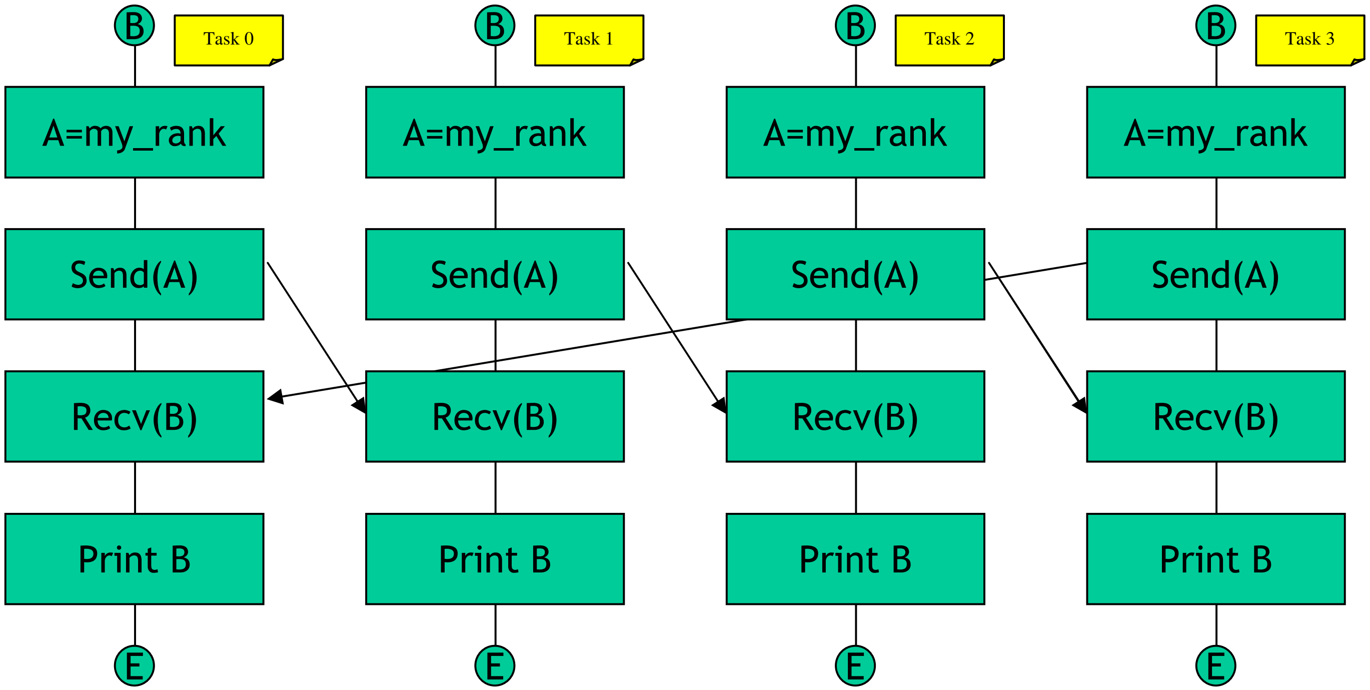


c) without ANY if and without deadlock (using non-blocking send)



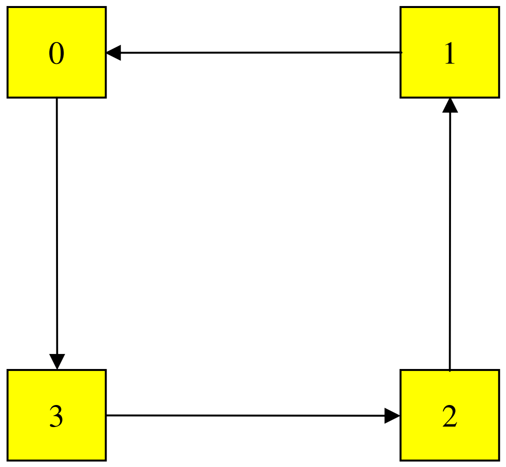
Exercise 3:

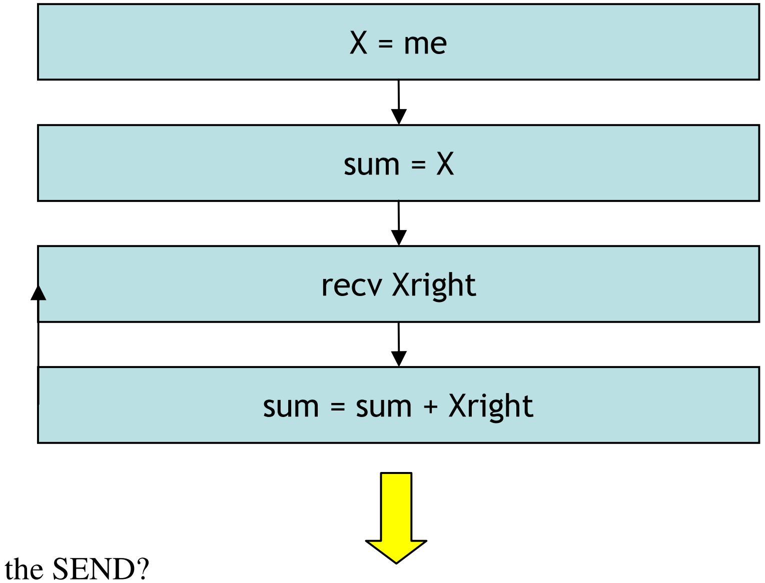
as Exercise 2c, but scalable, circular communications



Exercise 4 (optional):
a) Sum with circular communications

Each process must print the sum of all ranks. Only communication with the left neighbour process is allowed, as showed in the figure.





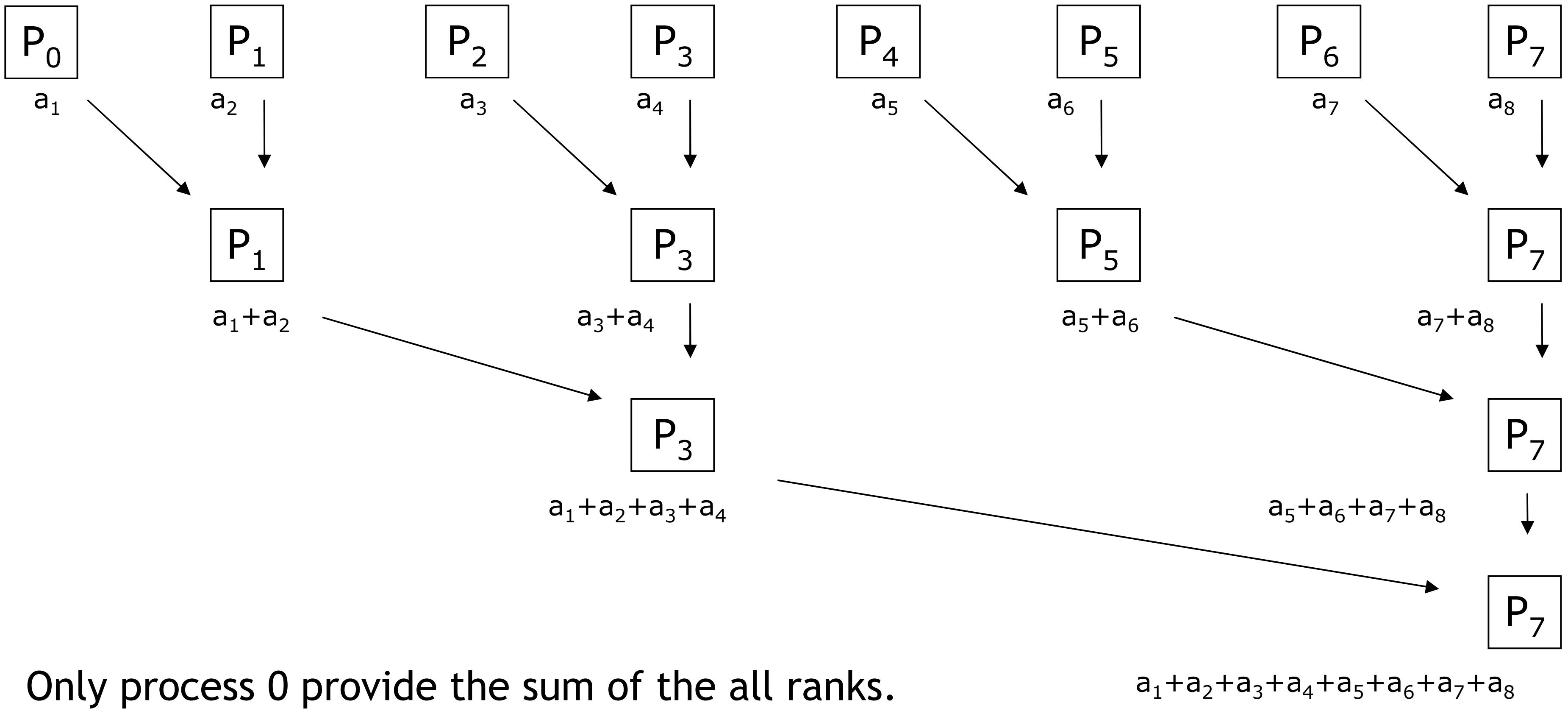
Who/when does the SEND?

What does it send?

How many times?

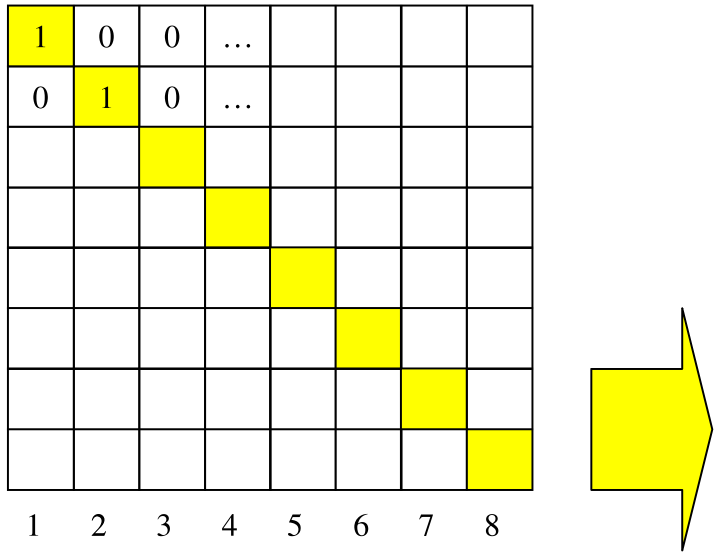
sum = 6 (on 4 processors)

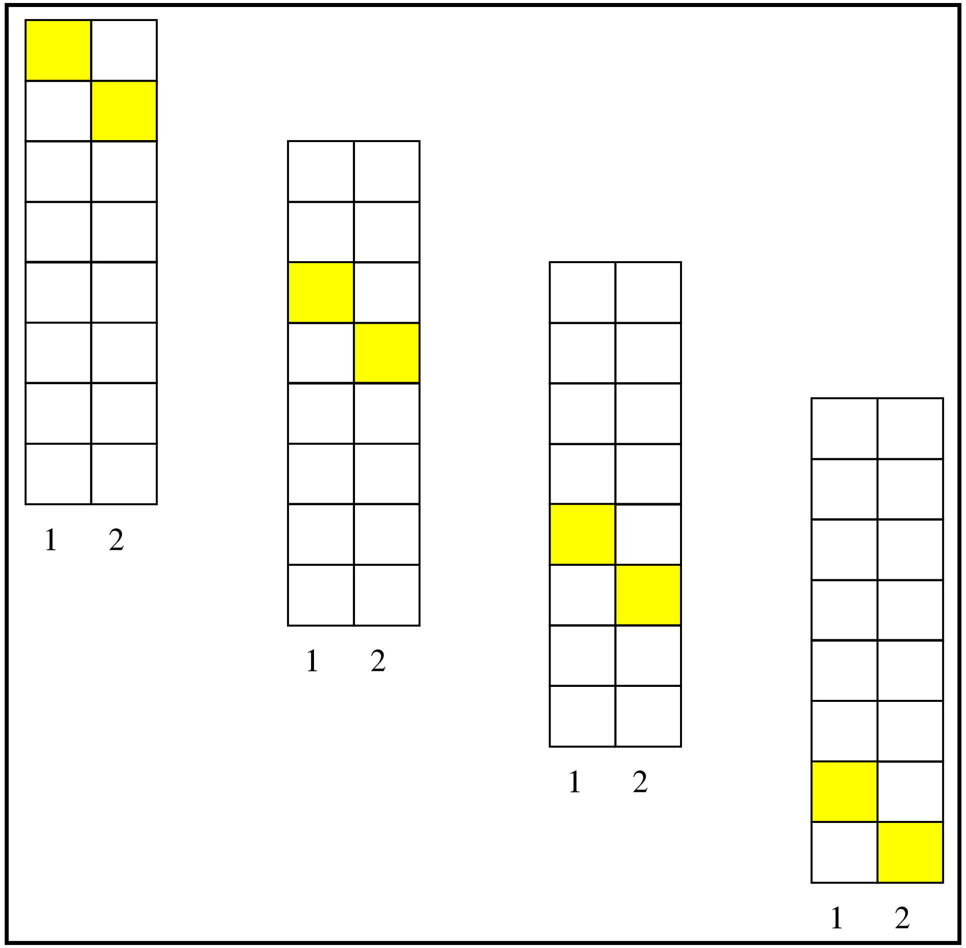
b) Sum with Binary Tree (optional)



Only process 0 provide the sum of the all ranks.
In this case communication works according to the binary tree showed in figure. This algorithm complete in $\log_2 n$ steps.

Exercise 5 (optional): Data distribution of identity





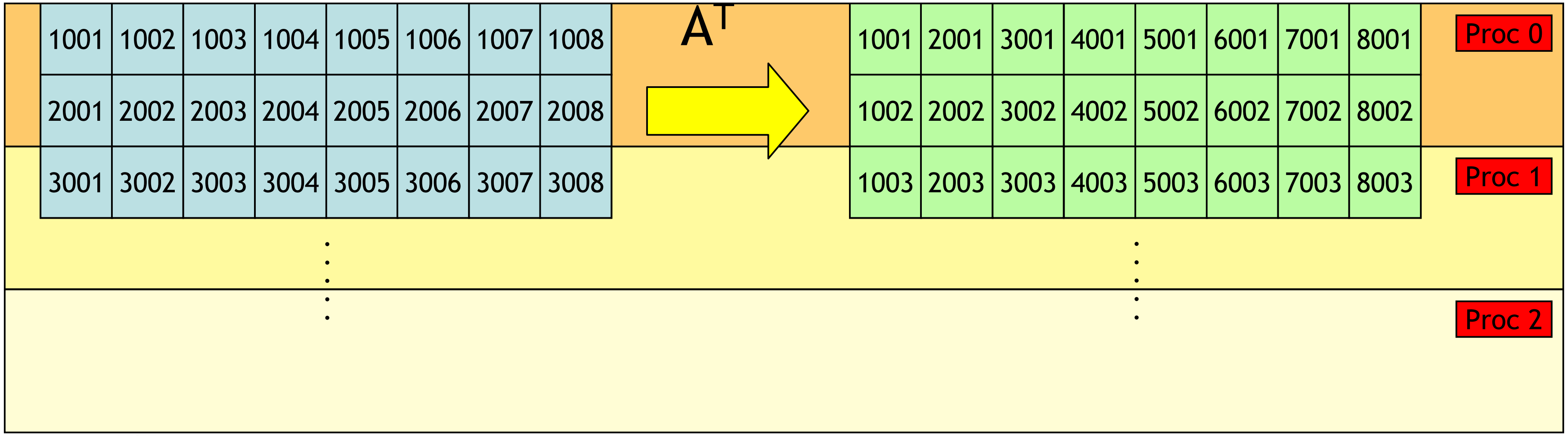
Transform global coordinates to task local ones:

given the number of processes and the dimension of the identity matrix, each process must allocate and set its own portion of the matrix.

Exercise 8: matrix transposition

- Initialize a 8x8 distributed matrix A by setting A_{ij} = 1000*i+j
- Both A and B are distributed by lines over 4 processors
- Print out A
- Evaluate: B = A^T
- Print out B

FORTRAN



1001	1002	1003	1004	1005	1006	1007	1008
2001	2002	2003	2004	2005	2006	2007	2008
3001	3002	3003	3004	3005	3006	3007	3008

⋮

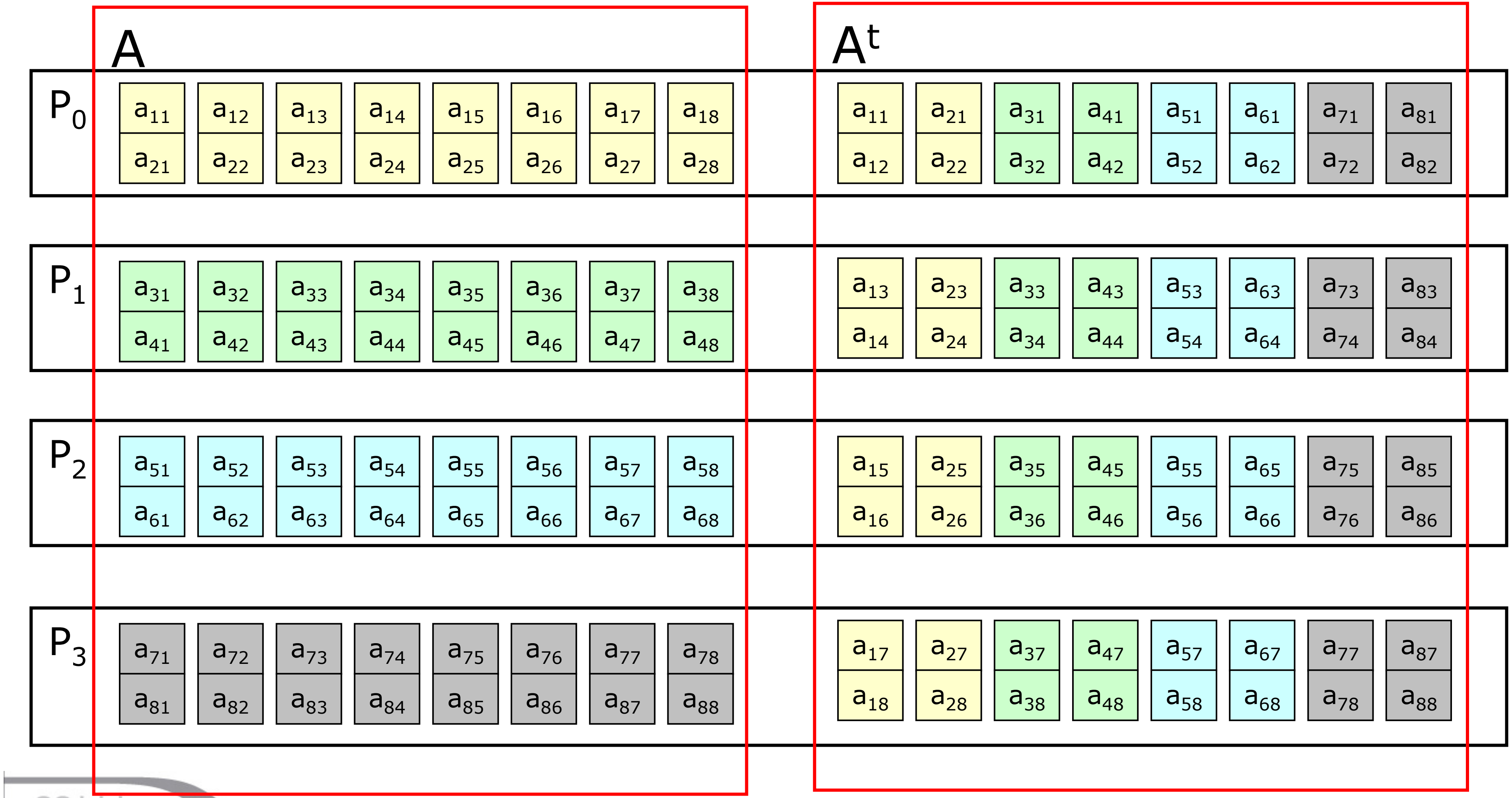
A^T

1001	2001	3001	4001	5001	6001	7001	8001
1002	2002	3002	4002	5002	6002	7002	8002
1003	2003	3003	4003	5003	6003	7003	8003

⋮

MPI Exercises

1. Use a collective communication. Which one?
2. Each block passed to the collective MPI function must contain data stored contiguously in memory!!



A

P_0	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}
	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}
P_1	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}
	a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}	a_{48}
P_2	a_{51}	a_{52}	a_{53}	a_{54}	a_{55}	a_{56}	a_{57}	a_{58}
	a_{61}	a_{62}	a_{63}	a_{64}	a_{65}	a_{66}	a_{67}	a_{68}
P_3	a_{71}	a_{72}	a_{73}	a_{74}	a_{75}	a_{76}	a_{77}	a_{78}
	a_{81}	a_{82}	a_{83}	a_{84}	a_{85}	a_{86}	a_{87}	a_{88}

A^t

P_0	a_{11}	a_{21}	a_{31}	a_{41}	a_{51}	a_{61}	a_{71}	a_{81}
	a_{12}	a_{22}	a_{32}	a_{42}	a_{52}	a_{62}	a_{72}	a_{82}
P_1	a_{13}	a_{23}	a_{33}	a_{43}	a_{53}	a_{63}	a_{73}	a_{83}
	a_{14}	a_{24}	a_{34}	a_{44}	a_{54}	a_{64}	a_{74}	a_{84}
P_2	a_{15}	a_{25}	a_{35}	a_{45}	a_{55}	a_{65}	a_{75}	a_{85}
	a_{16}	a_{26}	a_{36}	a_{46}	a_{56}	a_{66}	a_{76}	a_{86}
P_3	a_{17}	a_{27}	a_{37}	a_{47}	a_{57}	a_{67}	a_{77}	a_{87}
	a_{18}	a_{28}	a_{38}	a_{48}	a_{58}	a_{68}	a_{78}	a_{88}

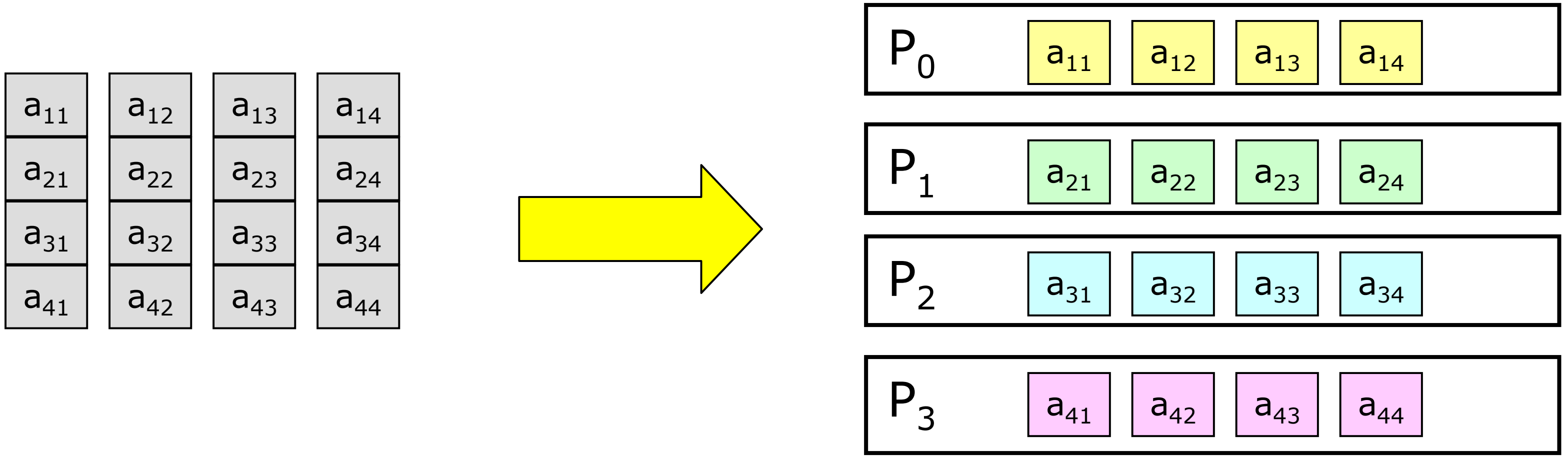
FORTRAN case

Exercise 9: Matrix Multiplication

Write a **subroutine** implementing matrix multiplication and test it.

$$C = A\,B \quad \longrightarrow \quad c_{ij} = \Sigma_k \; a_{ik} b_{kj}$$

A, B and C being NxN matrices distributed by row among processes (at least 8x8).
Inizialize A and B matrices respectively as $a_{ij} = i*j$ and $b_{ij}=1/(i*j)$.
Try to minimize memory allocation and the number of MPI calls.

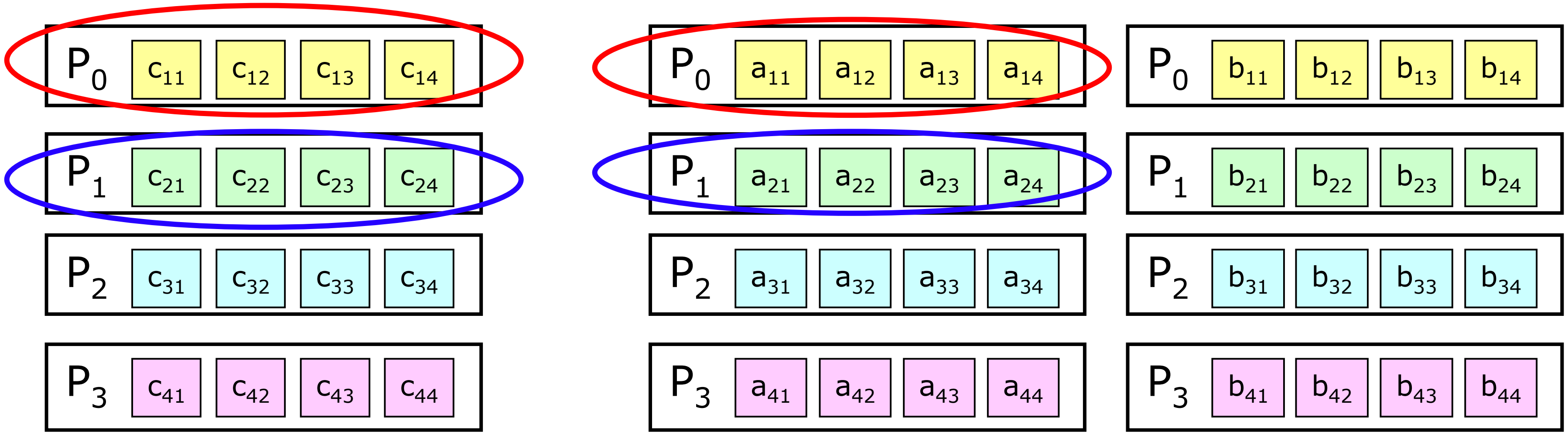


a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

P_0	a_{11}	a_{12}	a_{13}	a_{14}
P_1	a_{21}	a_{22}	a_{23}	a_{24}
P_2	a_{31}	a_{32}	a_{33}	a_{34}
P_3	a_{41}	a_{42}	a_{43}	a_{44}

MPI Exercises

FORTRAN CASE: each process manage a row of elements (or blocks)



P_0	c_{11}	c_{12}	c_{13}	c_{14}
P_1	c_{21}	c_{22}	c_{23}	c_{24}
P_2	c_{31}	c_{32}	c_{33}	c_{34}
P_3	c_{41}	c_{42}	c_{43}	c_{44}

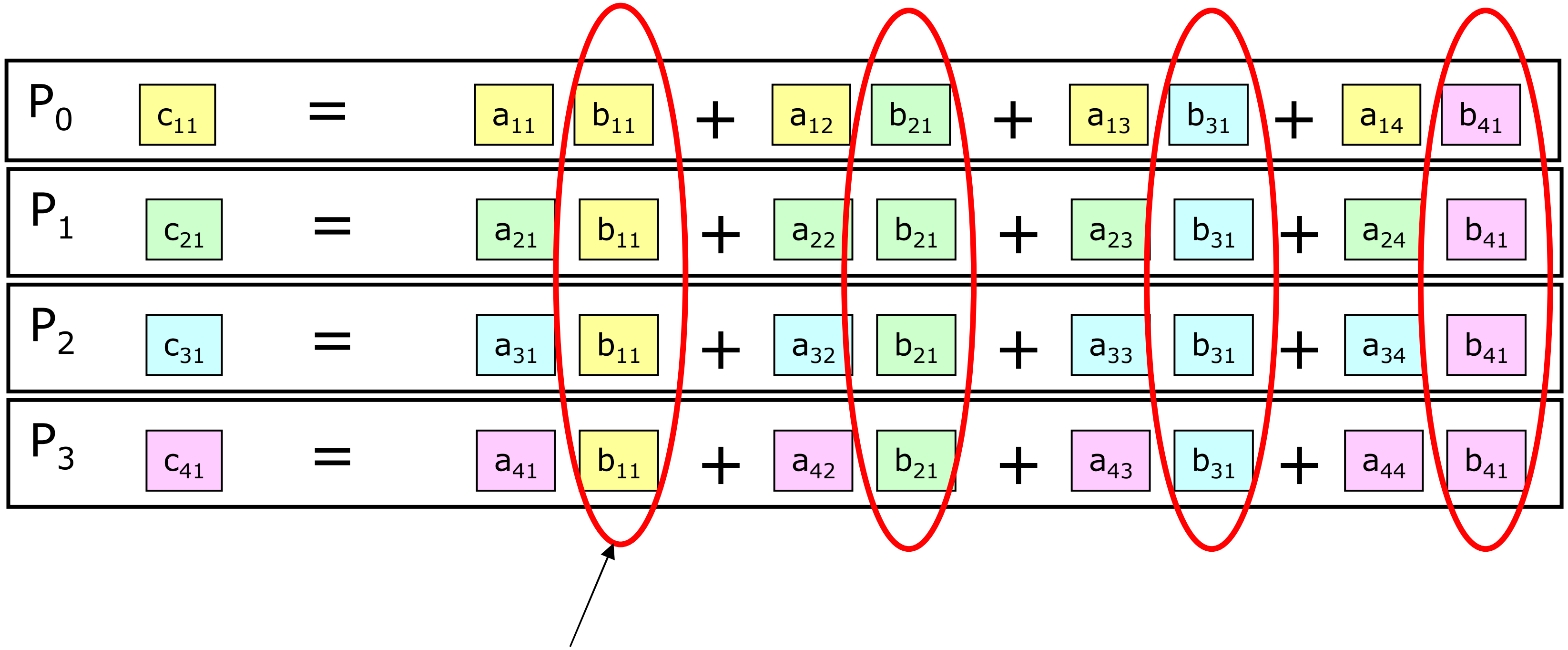
P_0	a_{11}	a_{12}	a_{13}	a_{14}
P_1	a_{21}	a_{22}	a_{23}	a_{24}
P_2	a_{31}	a_{32}	a_{33}	a_{34}
P_3	a_{41}	a_{42}	a_{43}	a_{44}

P_0	b_{11}	b_{12}	b_{13}	b_{14}
P_1	b_{21}	b_{22}	b_{23}	b_{24}
P_2	b_{31}	b_{32}	b_{33}	b_{34}
P_3	b_{41}	b_{42}	b_{43}	b_{44}

$$c_{11} = a_{11}\,b_{11} + a_{12}\,b_{21} + a_{13}\,b_{31} + a_{14}\,b_{41}$$

MPI Exercises

Each process compute the first element (block) of its own row



P_0 $c_{11} = a_{11} b_{11} + a_{12} b_{21} + a_{13} b_{31} + a_{14} b_{41}$

P_1 $c_{21} = a_{21} b_{11} + a_{22} b_{21} + a_{23} b_{31} + a_{24} b_{41}$

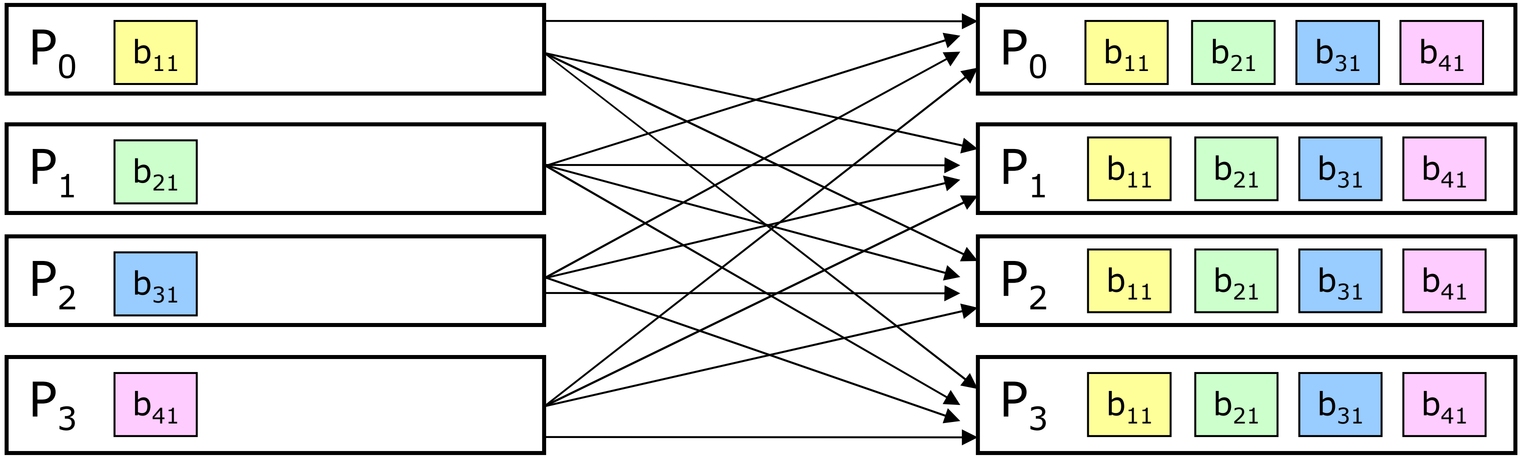
P_2 $c_{31} = a_{31} b_{11} + a_{32} b_{21} + a_{33} b_{31} + a_{34} b_{41}$

P_3 $c_{41} = a_{41} b_{11} + a_{42} b_{21} + a_{43} b_{31} + a_{44} b_{41}$

It’s always the same data for all the tasks! ⇨ Allgather

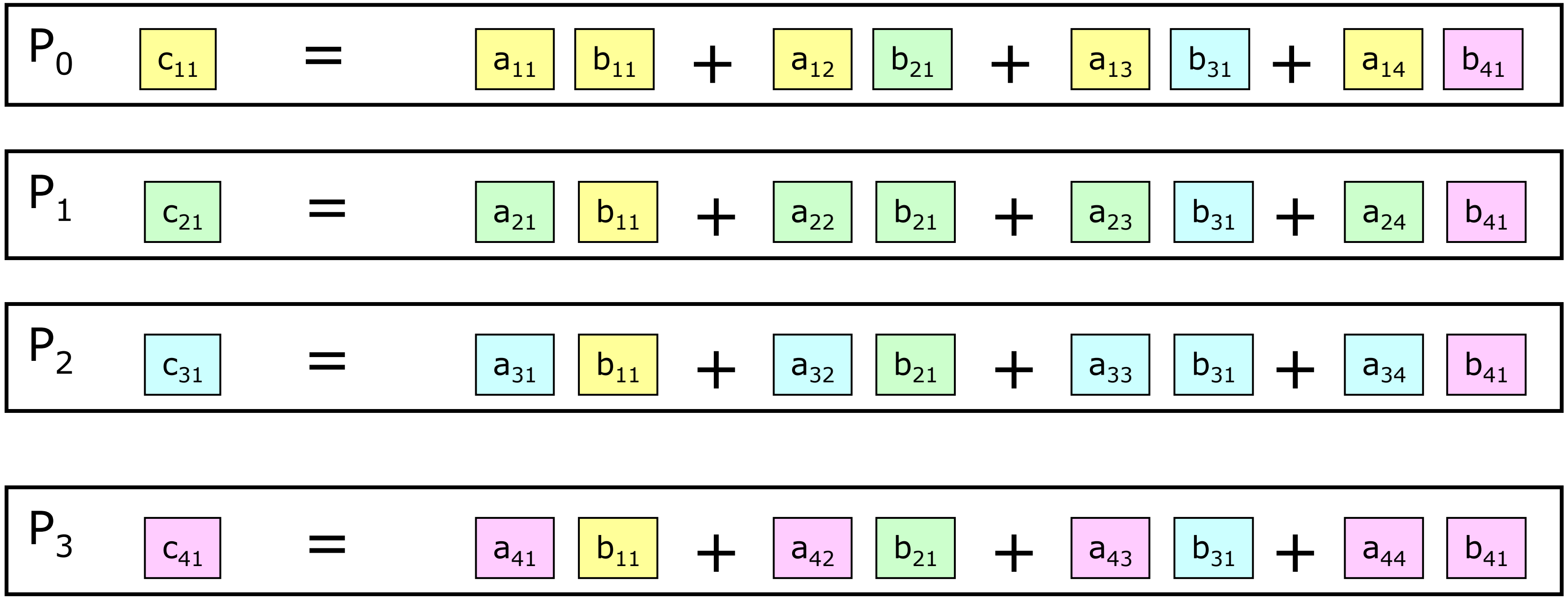
Step 1: allgather

Perform an All gather, of the first colum of blocks


P_0 b_{11}
P_1 b_{21}
P_2 b_{31}
P_3 b_{41}
P_0 b_{11} b_{21} b_{31} b_{41}
P_1 b_{11} b_{21} b_{31} b_{41}
P_2 b_{11} b_{21} b_{31} b_{41}
P_3 b_{11} b_{21} b_{31} b_{41}

Step 2: local computation

Each processor calculate the first column of the matrix C



P_0 $c_{11} = a_{11} \, b_{11} + a_{12} \, b_{21} + a_{13} \, b_{31} + a_{14} \, b_{41}$

P_1 $c_{21} = a_{21} \, b_{11} + a_{22} \, b_{21} + a_{23} \, b_{31} + a_{24} \, b_{41}$

P_2 $c_{31} = a_{31} \, b_{11} + a_{32} \, b_{21} + a_{33} \, b_{31} + a_{34} \, b_{41}$

P_3 $c_{41} = a_{41} \, b_{11} + a_{42} \, b_{21} + a_{43} \, b_{31} + a_{44} \, b_{41}$

Generalize

Repeat Step 1 and Step 2 for each column elements or blocks of matrix C, until matrix C is complete

Exercise 10 (optional): MPI-2 Parallel I/O

Write a code (mpi2io.f90) that reads a binary file (mpi2io.bin) according to the following instructions (see mpi2io_template.f90):

1. read initial data in parallel, using parallel data distribution and standard I/O.
2. write in a "critical" way (one processor after the other)
3. do the same using MPI2 I/O
4. use MPI2 I/O with Views

Exercise 7: Parallelization of a code

transport problem

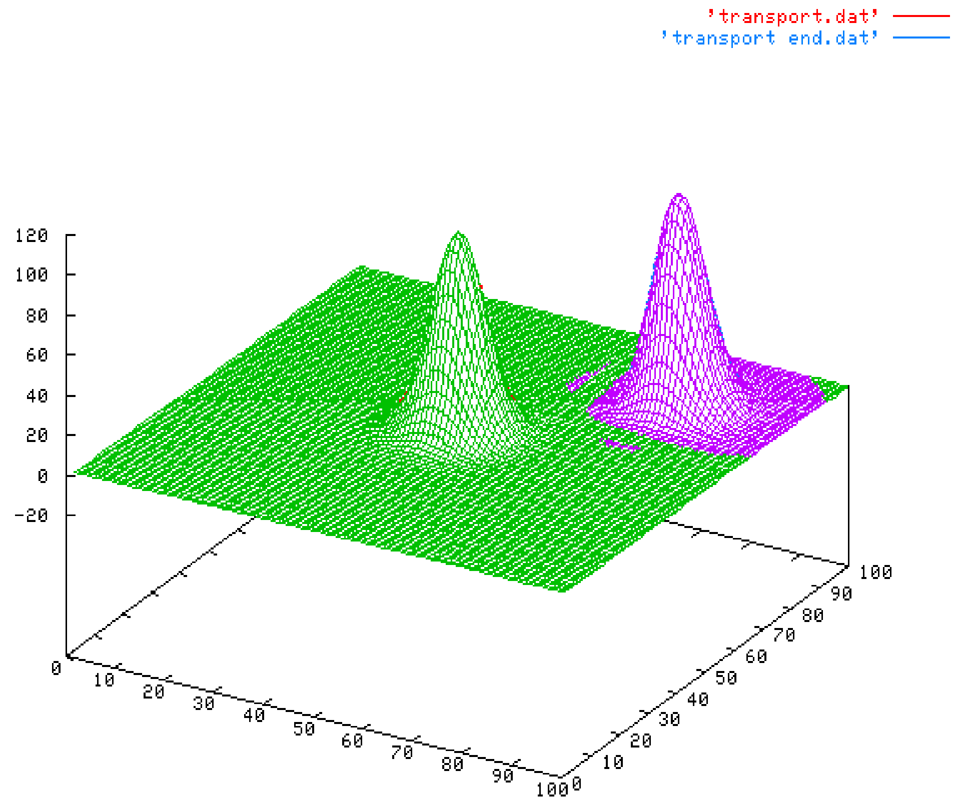
A code evolving the motion equation:

d/dx + d/dy = -d/dt,

is provided.

Original data have a gaussian distribution and must be moved along Y=X direction, i.e. toward the up-right corner of the system.

Execise: MAKE IT PARALLEL using domain decomposition



```
echo "set hidden3d; splot 'transport.dat' w l, 'transport_end.dat' w l" | gnuplot -persist
```

Exercise 7: Data distribution, ghost-cells

- Distribution by "leading dimension":

  -FORTRAN: first coordinate

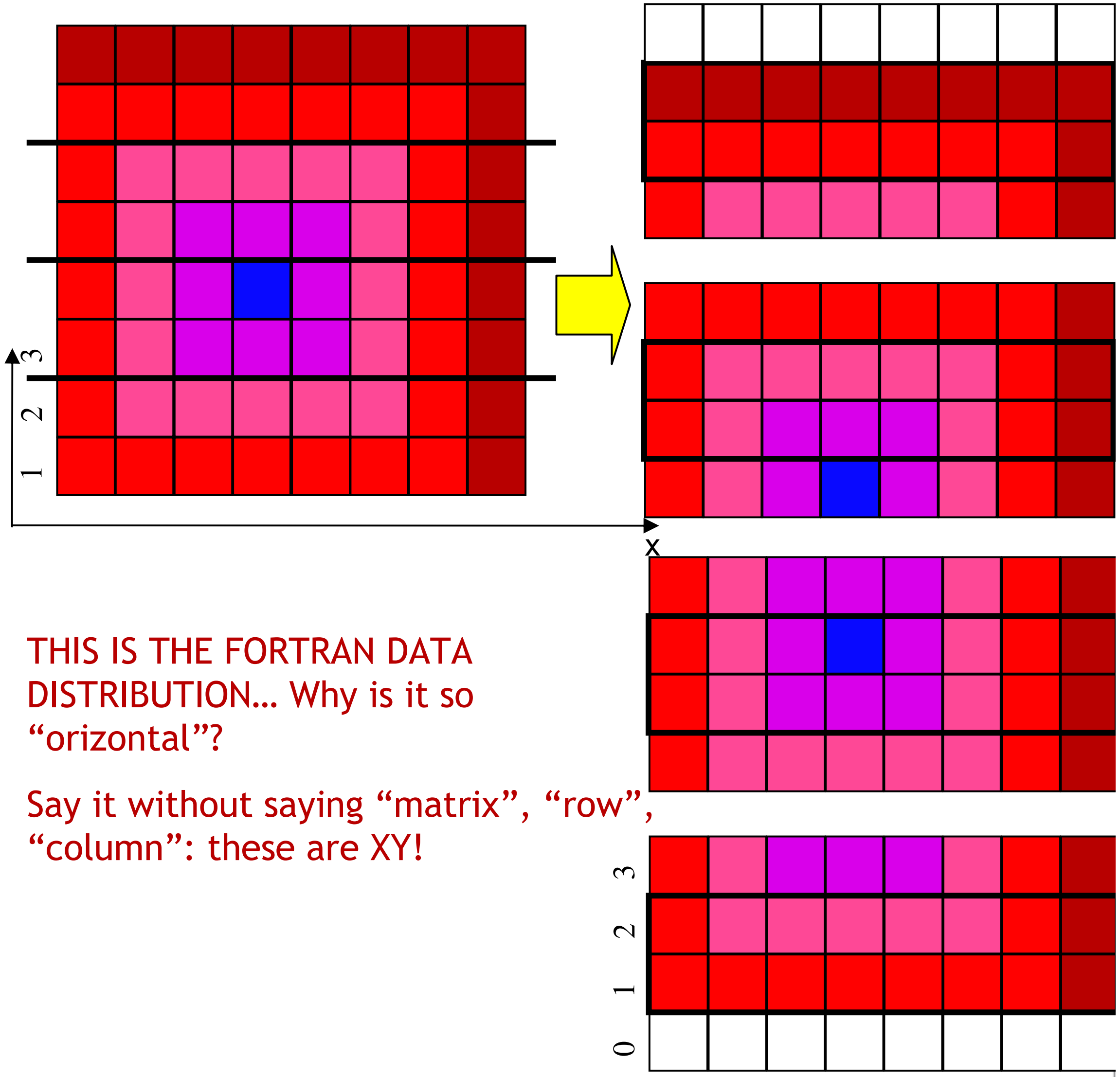
  -C: last coordinate

- The important thing is that the DATA to send/recv are CONTIGUOUS in memory. Otherwise a copy of them to/from a temporary, contiguous buffer is needed.

MORE TASKS: these are easy for the serial version… do it in parallel too.

- Evaluate the average over all the system and check it every 10 steps

- Find the global maximum and print its position every 10 steps



THIS IS THE FORTRAN DATA DISTRIBUTION… Why is it so "orizontal"?

Say it without saying "matrix", "row", "column": these are XY!

Exercise 7: Hybridization

- Add the OpenMP directives to the MPI code to parallelize some loops and to manage the MPI communications.
- Select and check the right MPI level of thread support.
- Print both the process and thread identifiers.
- Compile your code with the OpenMP support.
- Run with different configurations for processes and threads.