Master in High Performance Computing

3DFFT IN PARALLEL

Ivan Girotto – igirotto@ictp.it

Information & Communication Technology Section (ICTS)

International Centre for Theoretical Physics (ICTP)

ma per seguir virtute e canoscenza

SISSA

Scuola Internazionale Superiore di Studi Avanzati

The Abdus Salam International Centre for Theoretical Physics

FFT in multi-dimentions

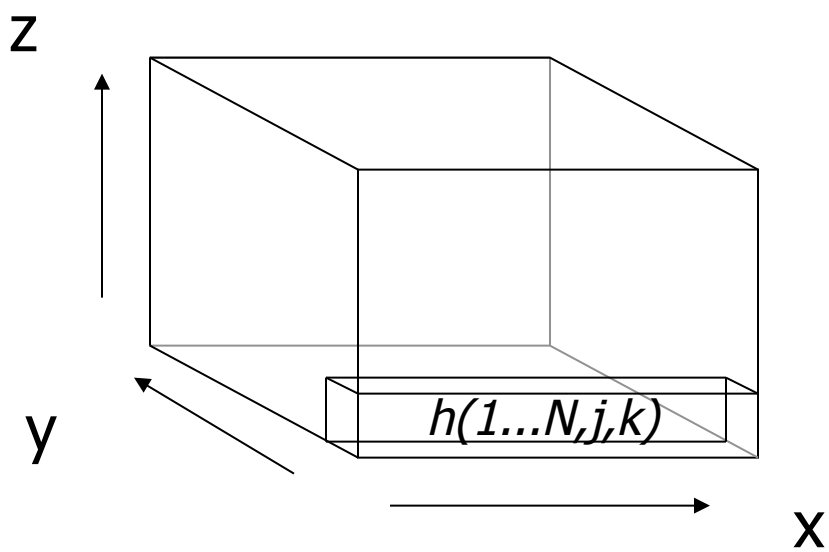
$$f(x,y,z) = \frac{1}{N_z N_y N_x} \underbrace{\sum_{z=0}^{N_z-1} \underbrace{\left(\sum_{y=0}^{N_y-1} \underbrace{\left(\sum_{x=0}^{N_x-1} F(u,v,w)\, e^{-2\pi i \frac{xu}{N_x}} \right)}_{\text{DFT long x-dimension}} e^{-2\pi i \frac{yv}{N_y}} \right)}_{\text{DFT long y-dimension}} e^{-2\pi i \frac{zw}{N_z}}}_{\text{DFT long z-dimension}}$$

For 2D FFT becomes:

$$\begin{aligned} H(n_1, n_2) &= \text{FFT-on-index-1}\left(\text{FFT-on-index-2}\left[h(k_1, k_2)\right]\right) \\ &= \text{FFT-on-index-2}\left(\text{FFT-on-index-1}\left[h(k_1, k_2)\right]\right) \end{aligned}$$

FFT in multi-dimentions

$$H(n_1, n_2) = \text{FFT-on-index-1}\left(\text{FFT-on-index-2}\left[h(k_1, k_2)\right]\right)$$
$$= \text{FFT-on-index-2}\left(\text{FFT-on-index-1}\left[h(k_1, k_2)\right]\right)$$



1) For each **j** and **k**

Transform along **i** h(:,j,k)

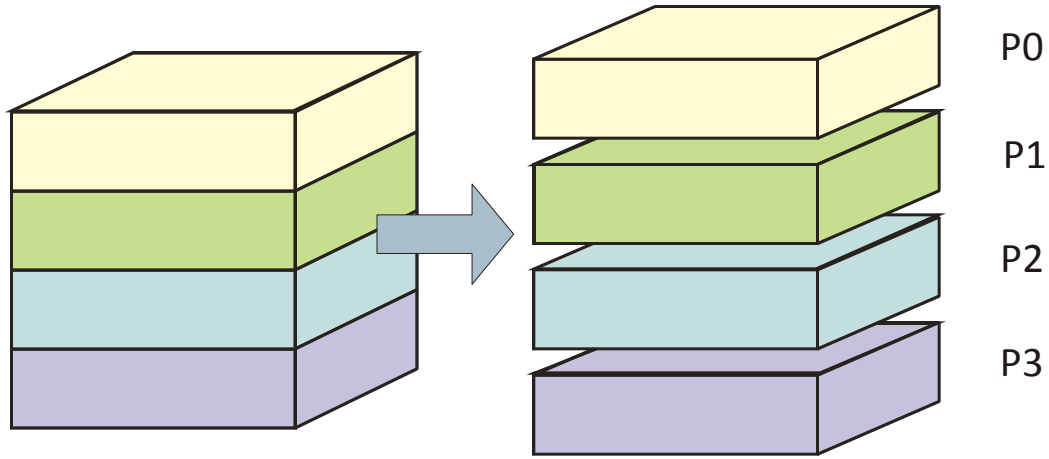
2) For each **i** and **k**

Transform along **j** h(i,:,k)

3) For each **i** and **j**

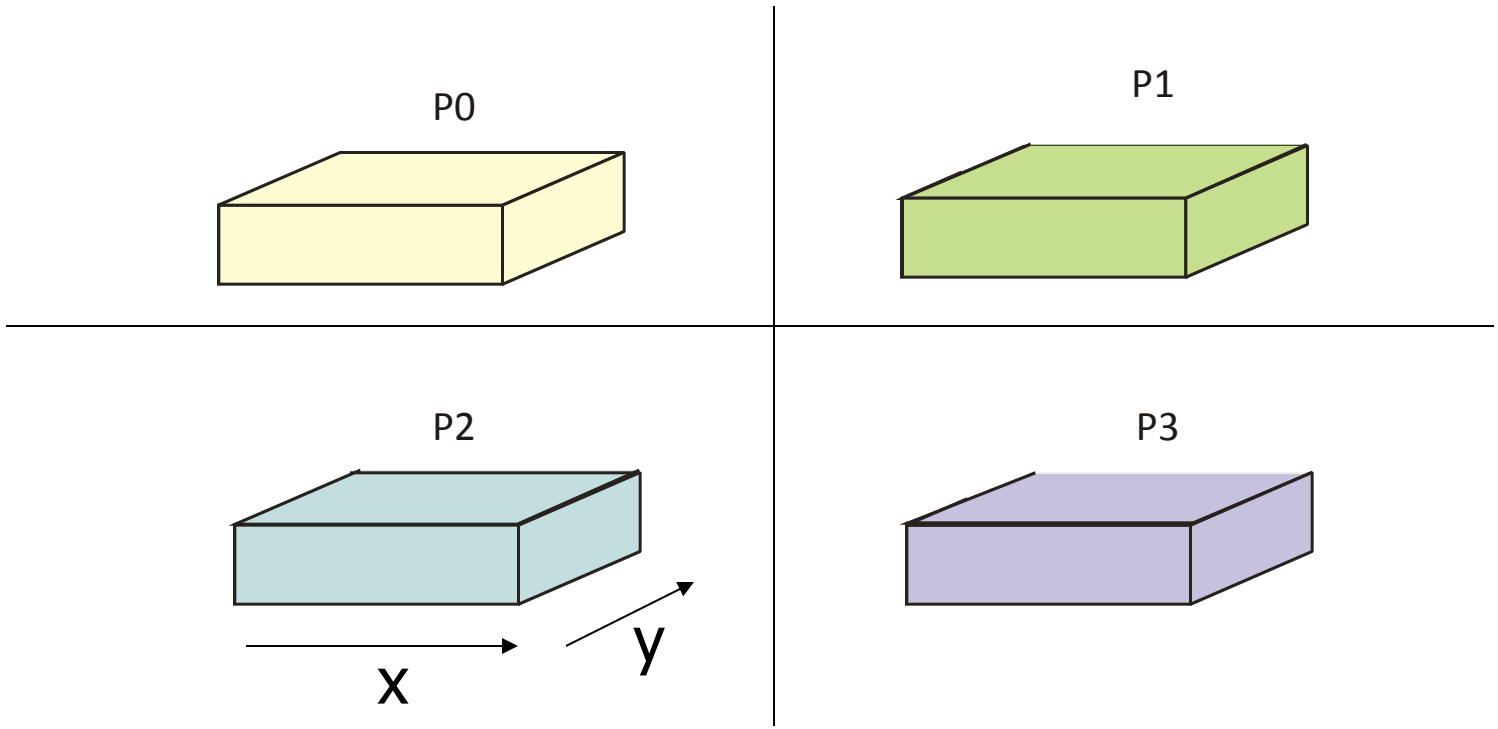
Transform along **k** h(i,j,:)

3D Parallel FFT Data Distribution



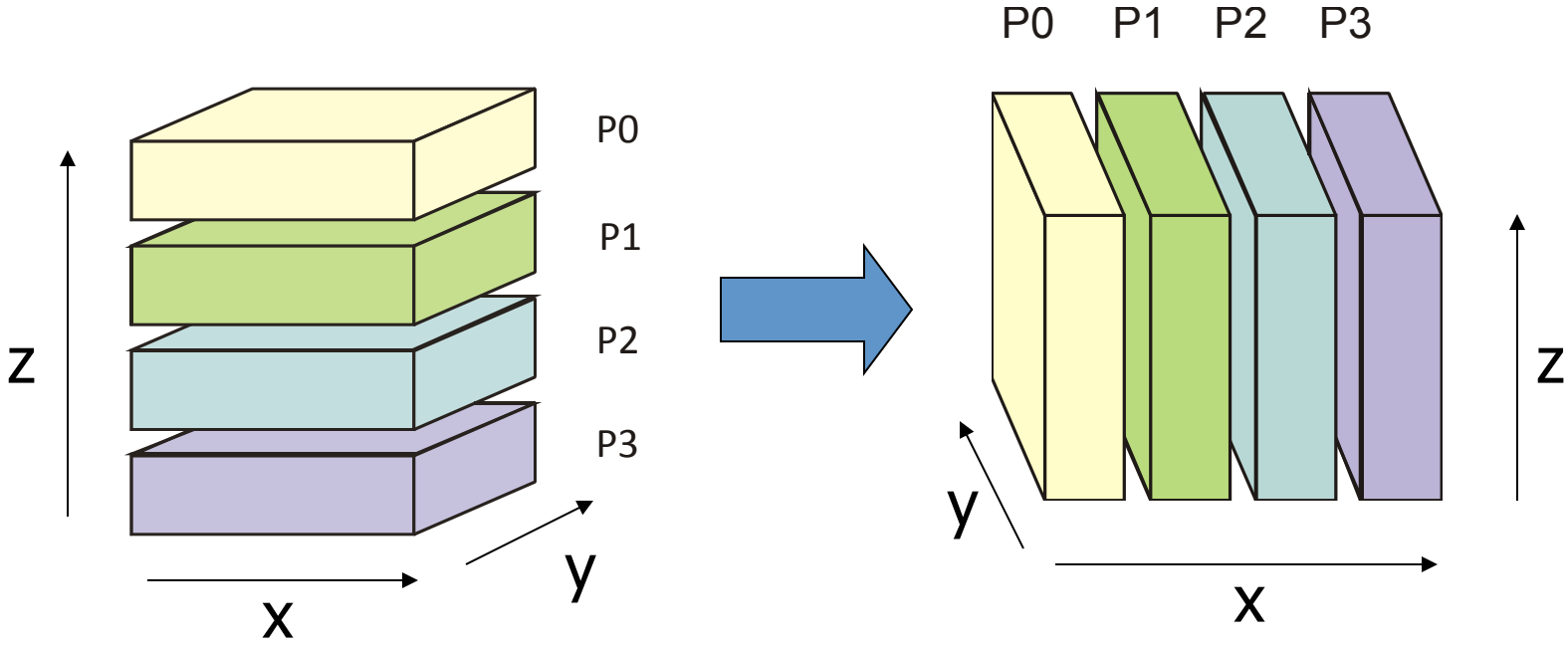
"Slab Decomposition"

Trasformata in x ed y



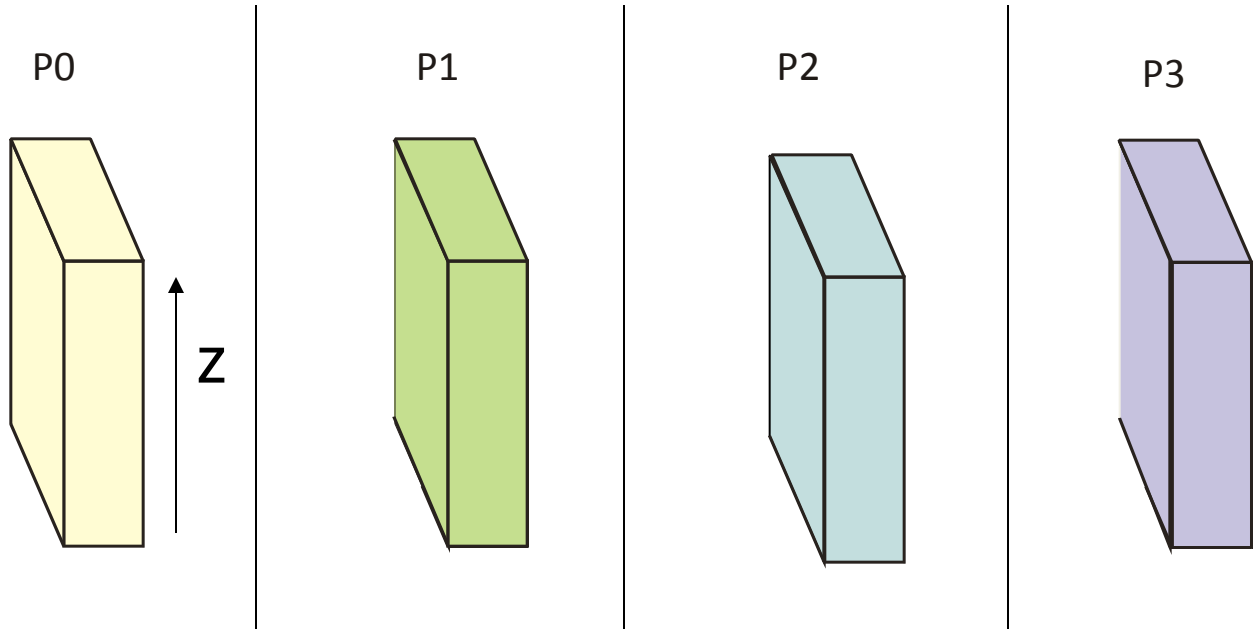
Each processor transform its own local part of the domain along x, y. All 2D FFT transforms are independent.

Change data distribution



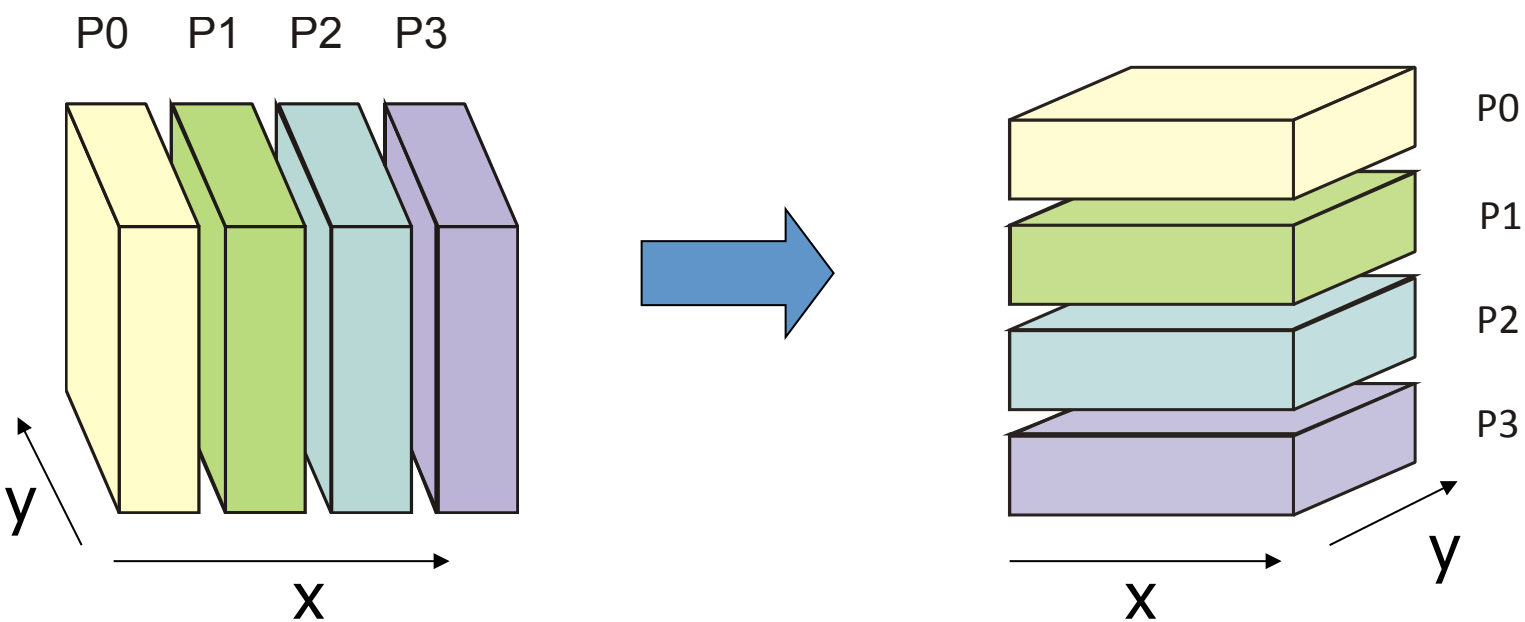
From “Z” slab to “X” slab

Trasform along z



Each PE transform its own part of the domain along z, again all 1D FFT are independent

Back distribution from z to x



This step can be skipped if you accept to have two different Data Distribution in real and reciprocal space

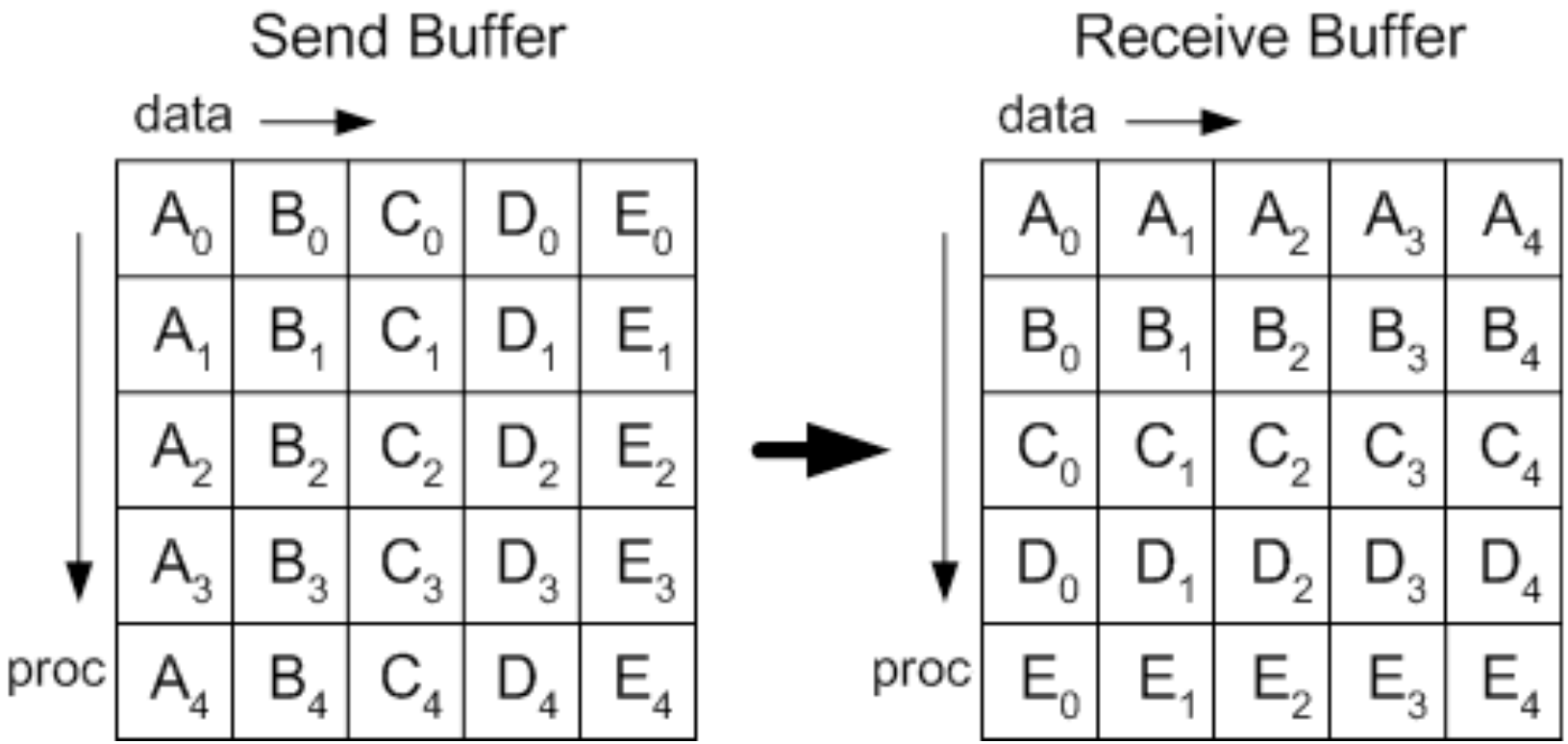
MPI_Alltoall

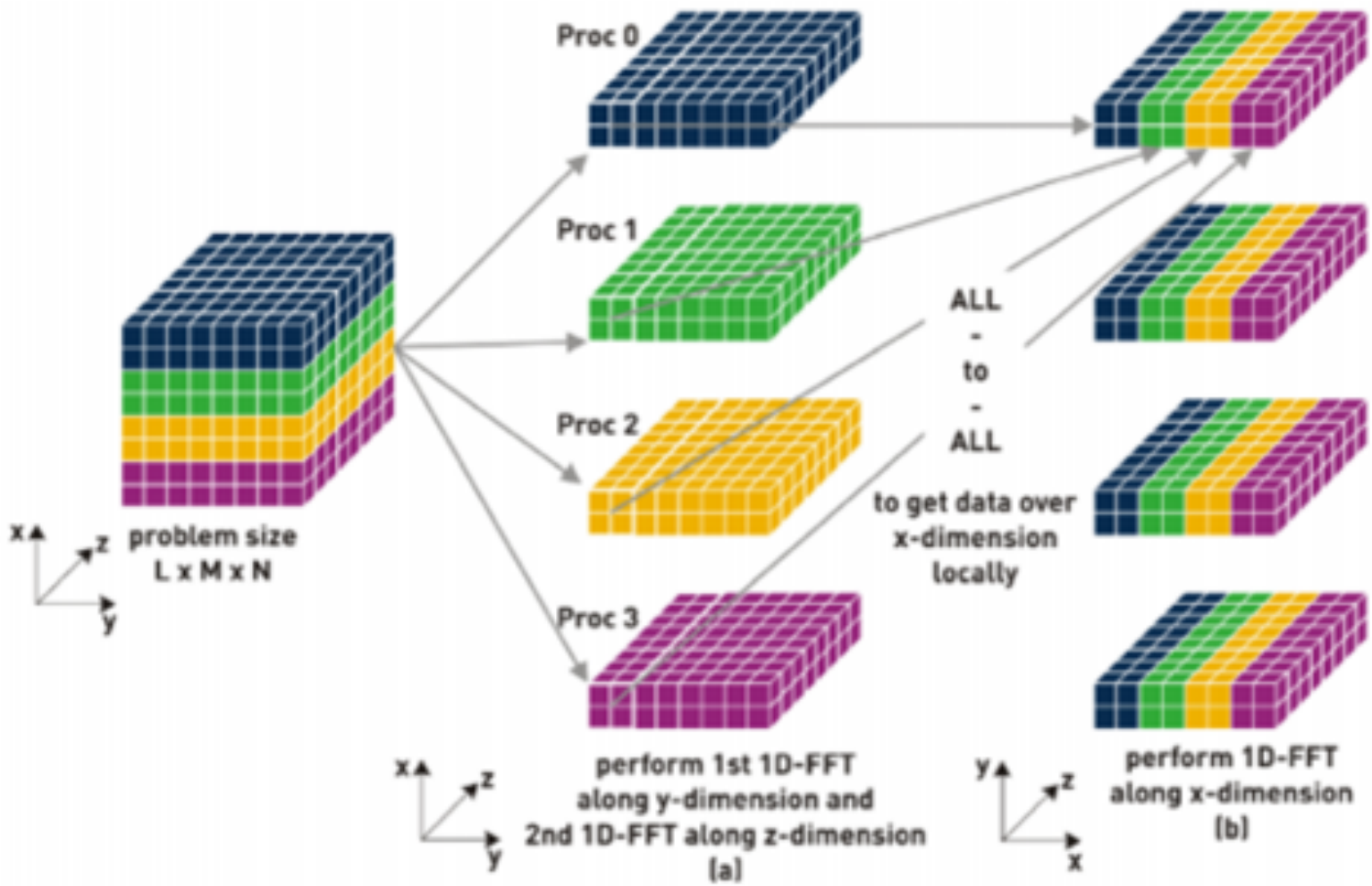
- It implements a All to All communication with ith location in jth processor being sent to jth location in ith processor
- Let's suppose to have 4 processes:
  - **T1 - (a0, a1, a2, a3) (b0, b1, b2, b3) (c0, c1, c2, c3) (d0, d1, d2, d3)**
  - **T2 - (a0, b0, c0, d0) (a1, b1, c1, d1) (a2, b2, c2, d2) (a3, b3, c3, d3)**
- Each process breaks up its local *sendbuf* into n blocks - each containing *sendcount* elements of type *sendtype* - and divides its *recvbuf* similarly according to *recvcount* and *recvtype*. Process j sends the k-th block of its local *sendbuf* to process k, which places the data in the j-th block of its local *recvbuf*. The amount of data sent must be equal to the amount of data received, pairwise, between every pair of processes
- MPI_Alltoallv handles non-contiguous data and different buffer size among processes

MPI_Alltoall

- *MPI_Alltoall* (void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm)
  - IN sendbuf (starting address of send buffer)
  - IN sendcount (number of elements sent to each proc)
  - IN sendtype (type)
  - OUT recvbuf (address of receive bufer) ← From Each PE!!!
  - IN recvcount (n-elements in receive buffer)
  - IN recvtype (data type of receive elements)
  - IN comm (communicator)

MPI_Alltoall




Proc 0
Proc 1
Proc 2
Proc 3
problem size
L x M x N
ALL
-
to
-
ALL
to get data over
x-dimension
locally
perform 1st 1D-FFT
along y-dimension and
2nd 1D-FFT along z-dimension
(a)
perform 1D-FFT
along x-dimension
(b)