TRACK II

ADVANCED DISTRIBUTED-MEMORY PROGRAMMING

HANDS ON EXERCISES

2024 IHPCSS

EXERCISE.1

A[4][4]

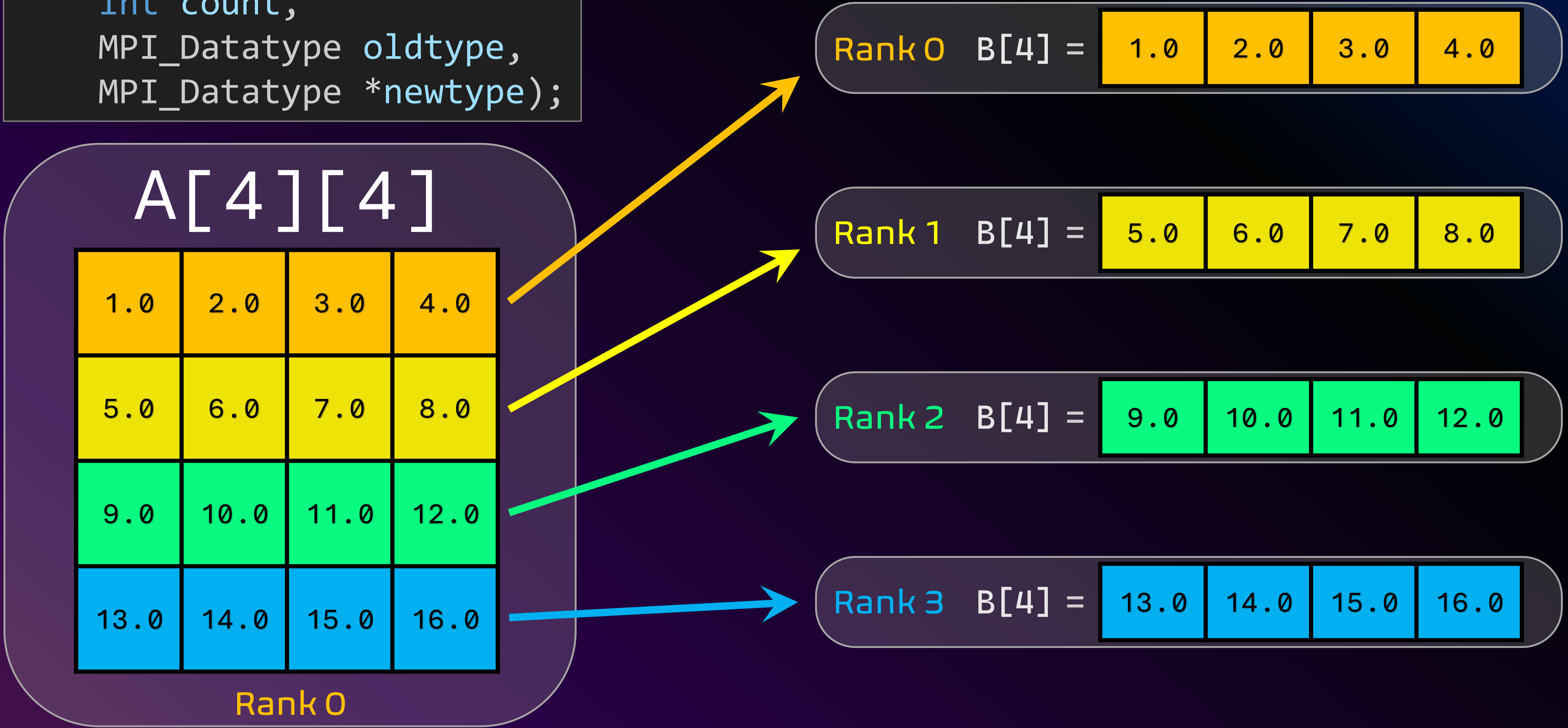
1.0	2.0	3.0	4.0
5.0	6.0	7.0	8.0
9.0	10.0	11.0	12.0
13.0	14.0	15.0	16.0

Four Programs

- ✓ ddt.1.contiguous
- ✓ ddt.2.vector
- ✓ ddt.3.indexed
- ✓ ddt.4.struct

EXERCISE.1 ddt.1.contiguous DERIVED DATA TYPES

```
int MPI_Type_contiguous(
    int count,
    MPI_Datatype oldtype,
    MPI_Datatype *newtype);
```



EXERCISE.1 ddt.2.vector

DERIVED DATA TYPES

```c
int MPI_Type_vector(
    int count,
    int blocklen,
    int stride,
    MPI_Datatype oldtype,
    MPI_Datatype *newtype);
```

A[4][4]

1.0	2.0	3.0	4.0
5.0	6.0	7.0	8.0
9.0	10.0	11.0	12.0
13.0	14.0	15.0	16.0

Rank 0

Rank 0 B[4] =	1.0	5.0	9.0	13.0
Rank 1 B[4] =	2.0	6.0	10.0	14.0
Rank 2 B[4] =	3.0	7.0	11.0	15.0
Rank 3 B[4] =	4.0	8.0	12.0	16.0

EXERCISE.1 ddt.3.indexed DERIVED DATA TYPES

```c
int MPI_Type_indexed(
 int count,
 int *array_of_blocklens,
 int *array_of_displacements,
 MPI_Datatype oldtype,
 MPI_Datatype *newtype);
```

Rank 0

A[16]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

- Block Count
- Block Lengths
- Displacements

Rank 0 B[6] =

6	7	8	9	13	14

Rank 2 B[6] =

6	7	8	9	13	14

Rank 1 B[6] =

6	7	8	9	13	14

Rank 3 B[6] =

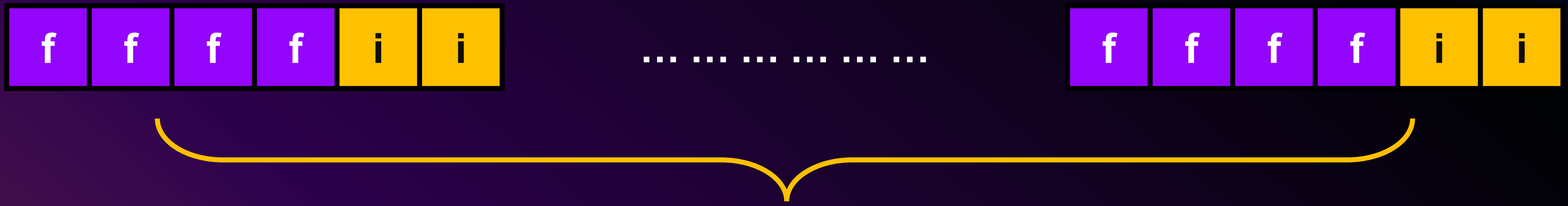
6	7	8	9	13	14

EXERCISE.1 ddt.4.struct DERIVED DATA TYPES

```c
int MPI_Type_create_struct(
     int count,
     int *array_of_blocklens,
MPI_Aint *array_of_displacements,
MPI_Datatype *array_of_types,
MPI_Datatype *newtype);
```

```c
typedef struct {
    float x, y, z, velocity;
    int n, type;
} Particle;

Particle particles[NELEM];
```




```c
particles[NELEM];
```

EXERCISE.1 DERIVED DATA TYPES

Four Programs

✓ ddt.1.contiguous

✓ ddt.2.vector

✓ ddt.3.indexed

✓ ddt.4.struct

Step 1: `cp -r /jet/home/akirby/IHPCSS2024-mpi/exercises ~/.`

Step 2: `cd exercises/Exercise.1-DerivedDataTypes/{c or f90}`

Step 3: Complete the "TODO" tasks in each of the programs.

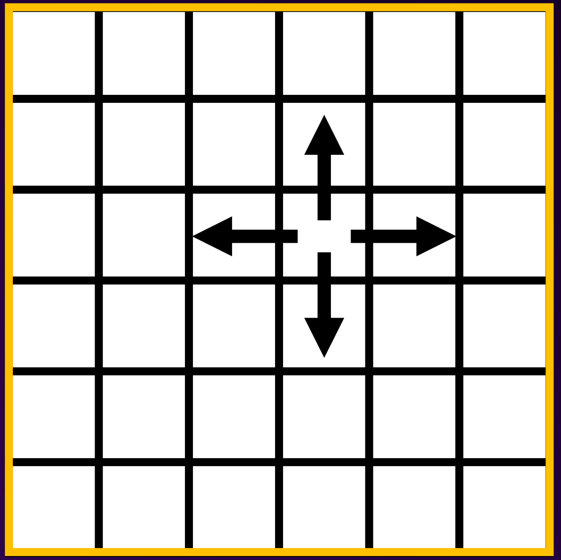
SOLUTIONS: `cd exercises/Exercise.1-DerivedDataTypes/.soln`

EXERCISE.2 CARTESIAN VIRTUAL TOPOLOGIES

Step 1: cd `exercises/Exercise.2-CartesianTopology/{c or f90}`

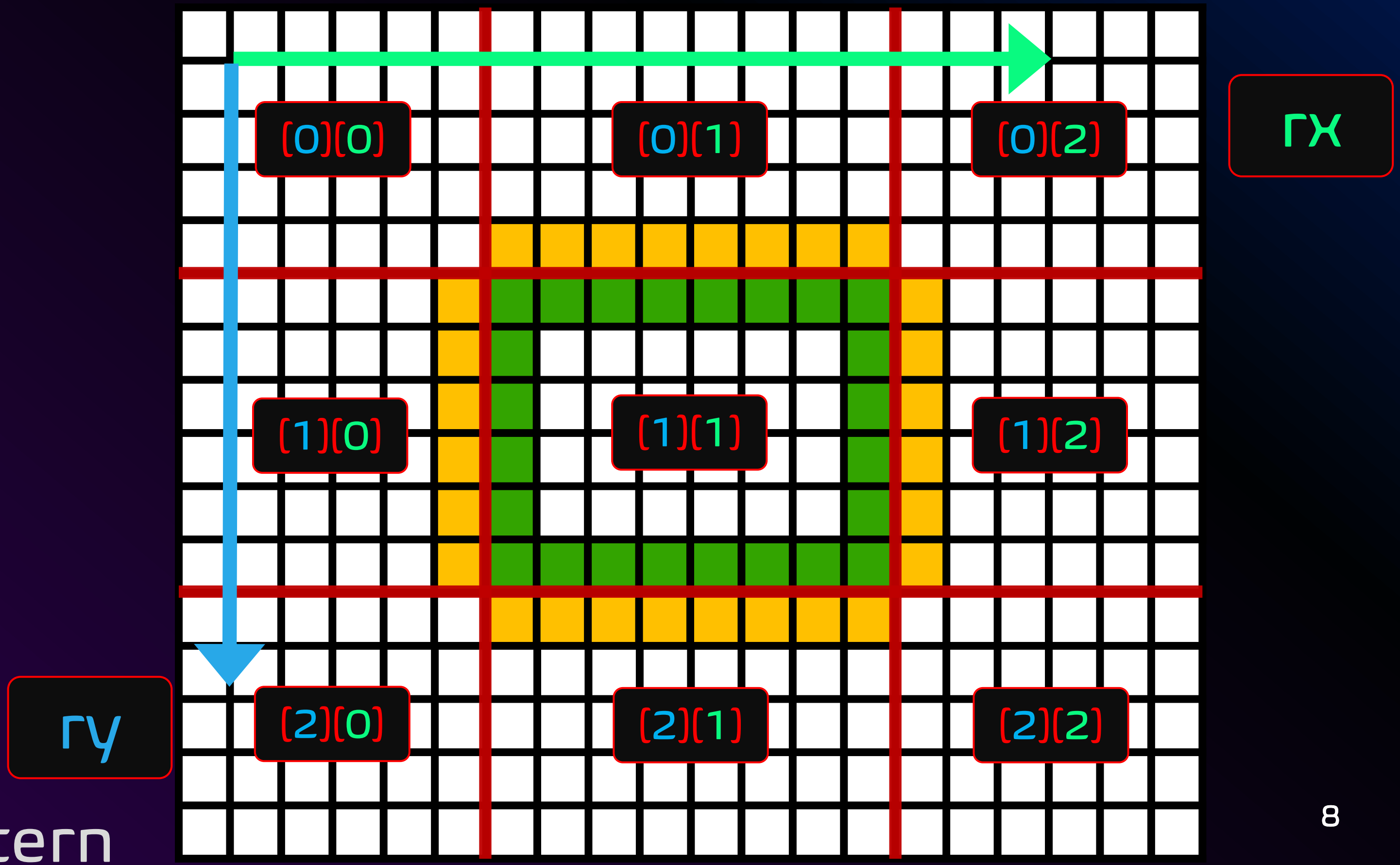
Step 2: Complete the "TODO" tasks in stencil_cart_shift.`{c or f90}`

$$-\Delta u(x,y) = f$$



4 Neighbors

Typical Communication Pattern

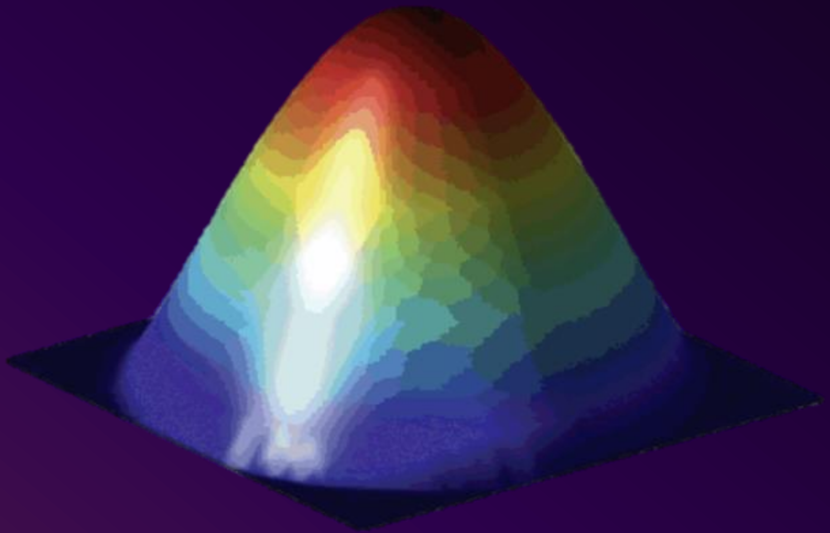


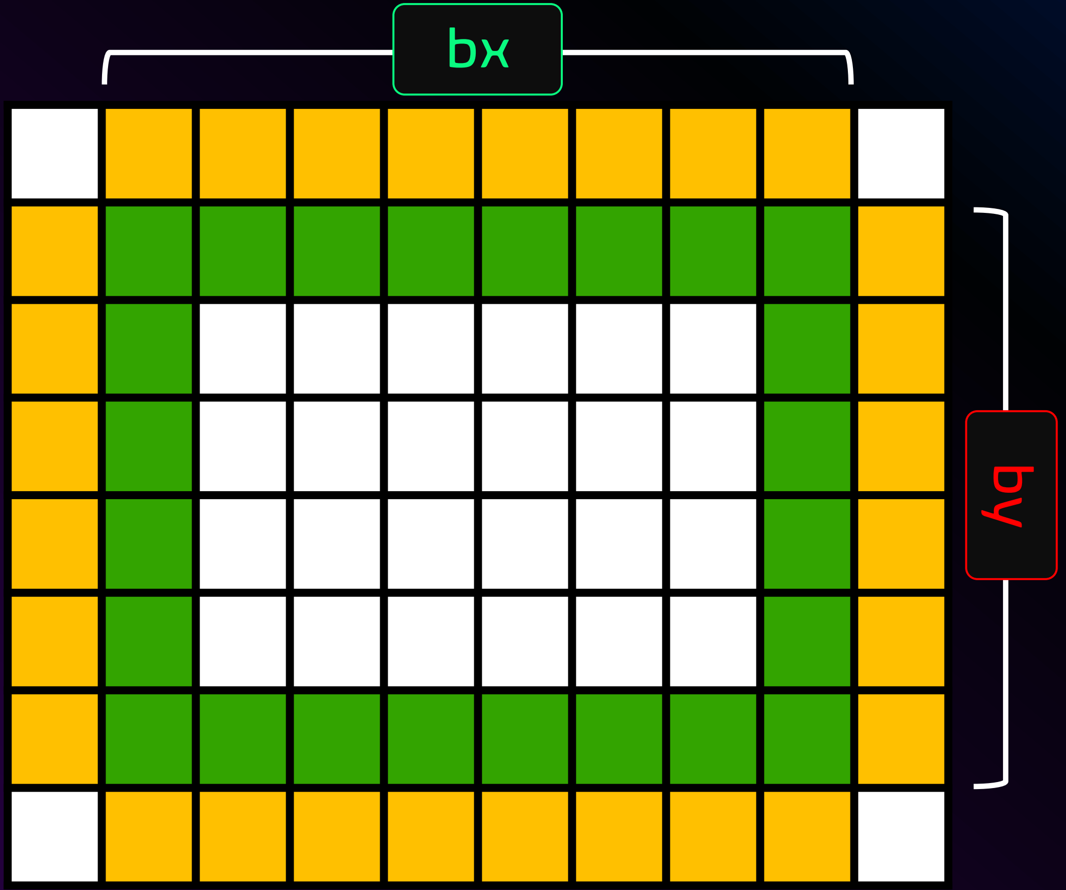
EXERCISE.2 CARTESIAN VIRTUAL TOPOLOGIES

Step 1: cd `exercises/Exercise.2-CartesianTopology/{c or f90}`

Step 2: Complete the "TODO" tasks in stencil_cart_shift.`{c or f90}`

$$-\Delta u(x,y) = f$$

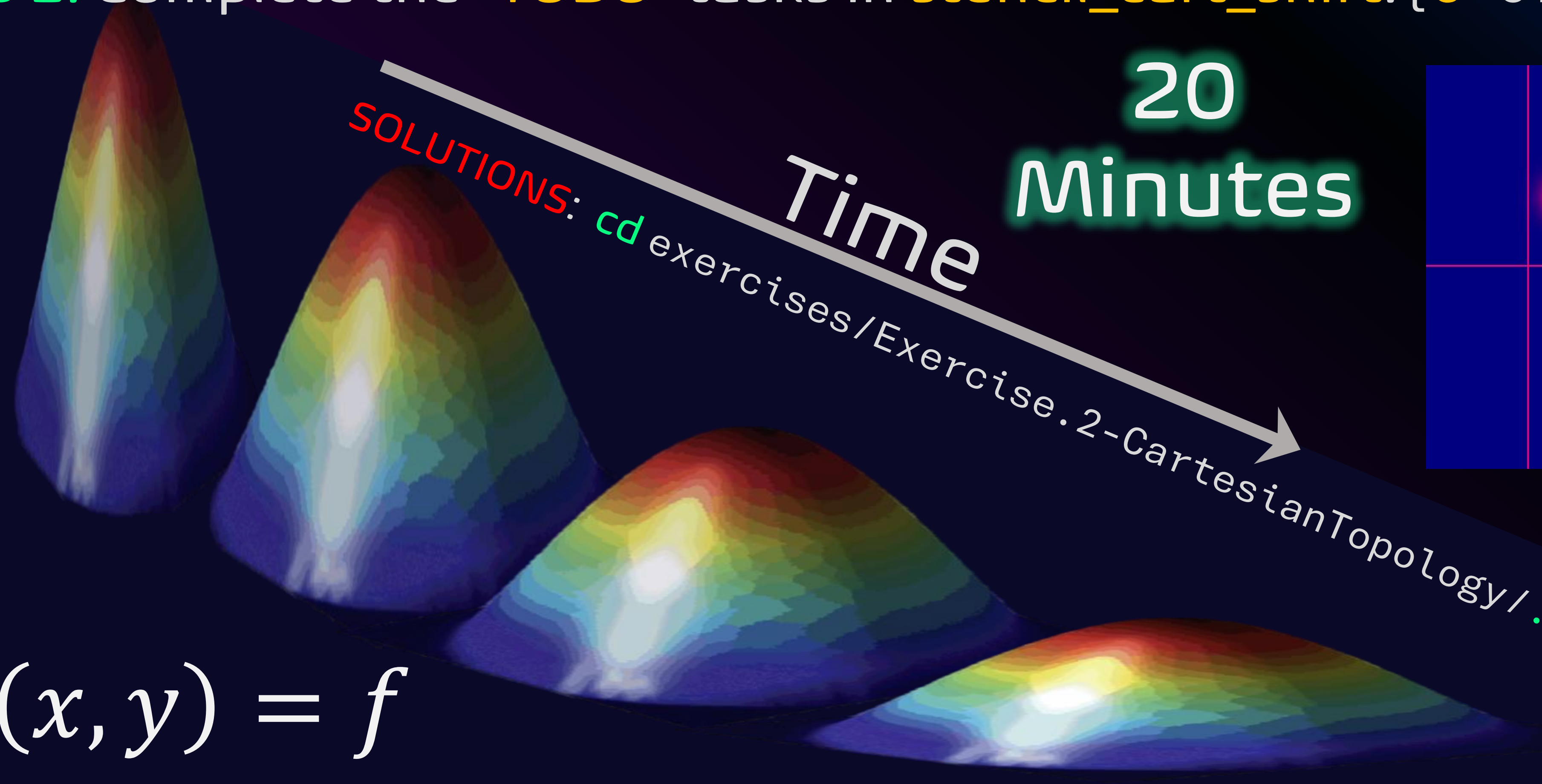




EXERCISE.2 CARTESIAN VIRTUAL TOPOLOGIES

Step 1: cd `exercises/Exercise.2-CartesianTopology/{c or f90}`

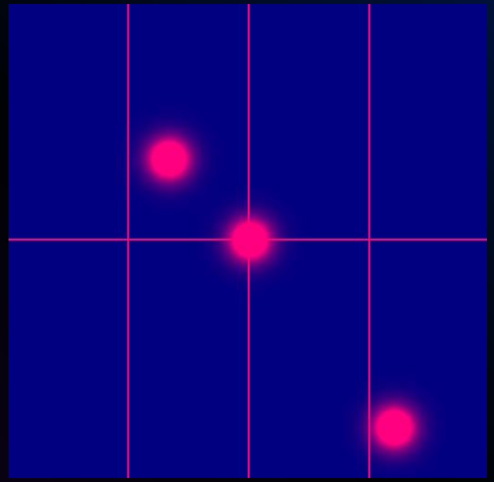
Step 2: Complete the "TODO" tasks in stencil_cart_shift.`{c or f90}`



Time

SOLUTIONS: cd `exercises/Exercise.2-CartesianTopology/.soln`

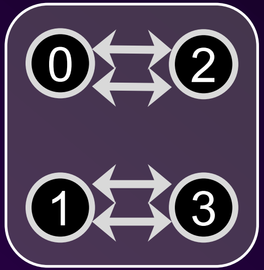
20 Minutes



5
10
15
20

$$-\Delta u(x, y) = f$$

EXERCISE.3 DISTRIBUTED GRAPH TOPOLOGIES



Makes a new communicator to which topology information has been attached.

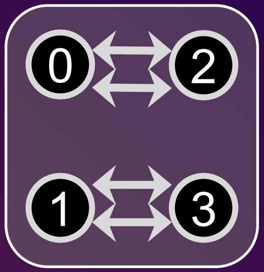
Every MPI process may specify 0, 1 or more edges.

The edges specified do not have to contain the MPI process that passes them.

EXERCISE.3 DISTRIBUTED GRAPH TOPOLOGIES

Step 1: cd `exercises/Exercise.3-GraphTopology/{c or f90}`

Step 2: Complete the "TODO" tasks in mpi_dist_graph_create.`{c or f90}`



Look at the new rank reordering.
Does it make sense? Why?*

20 Minutes

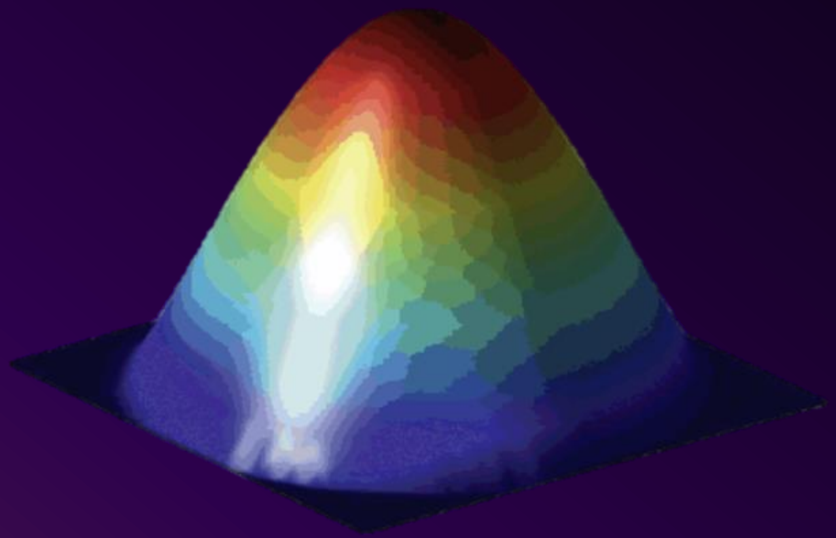
* May not reorder due to MPI Vendor implementation.

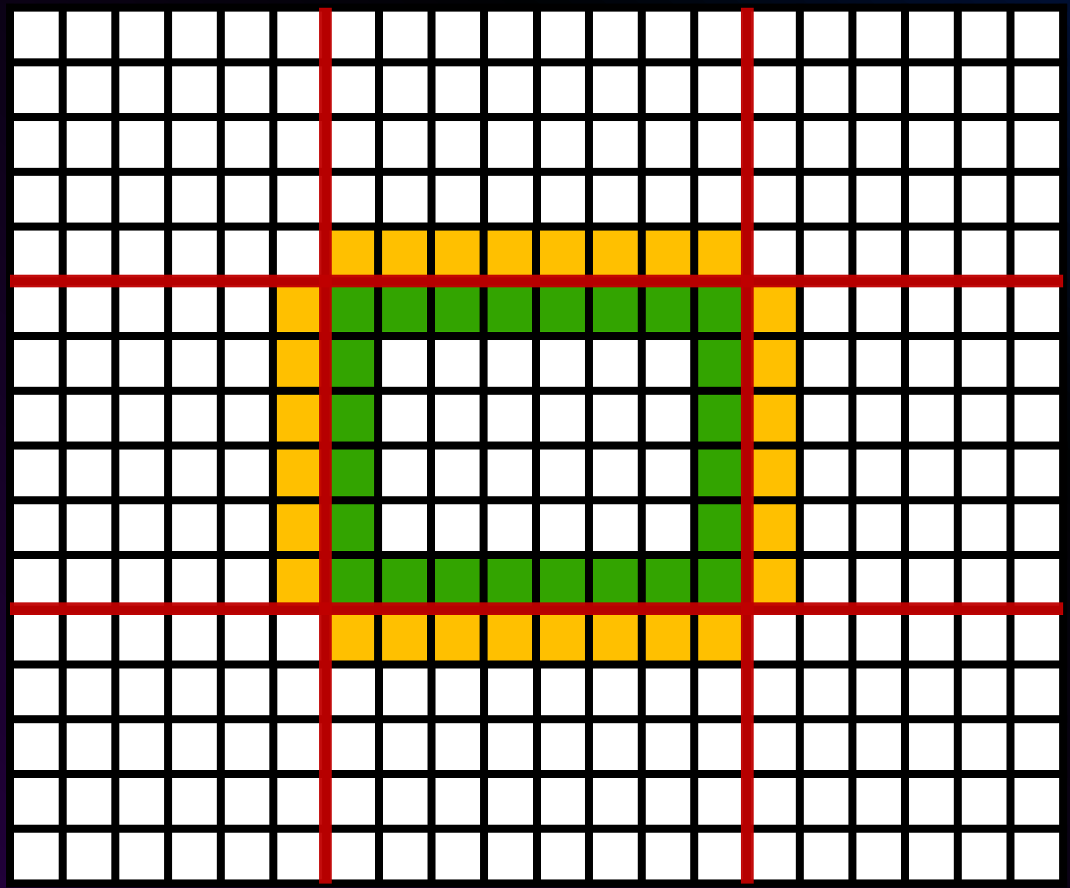
EXERCISE.4 NEIGHBORHOOD COLLECTIVES

Step 1: cd `exercises/Exercise.4-NeighborhoodCollectives/{c or f90}`

Step 2: Complete the "TODO" tasks in stencil_mpi_carttopo_neighcolls.`{c or f90}`

$$-\Delta u(x,y) = f$$





4 Neighbors
Typical Communication Pattern

MPI_Ineighbor_alltoallv

EXERCISE.4 NEIGHBORHOOD COLLECTIVES

Step 1: cd `exercises/Exercise.4-NeighborhoodCollectives/{c or f90}`

Step 2: Complete the "TODO" tasks in stencil_mpi_carttopo_neighcolls.`{c or f90}`

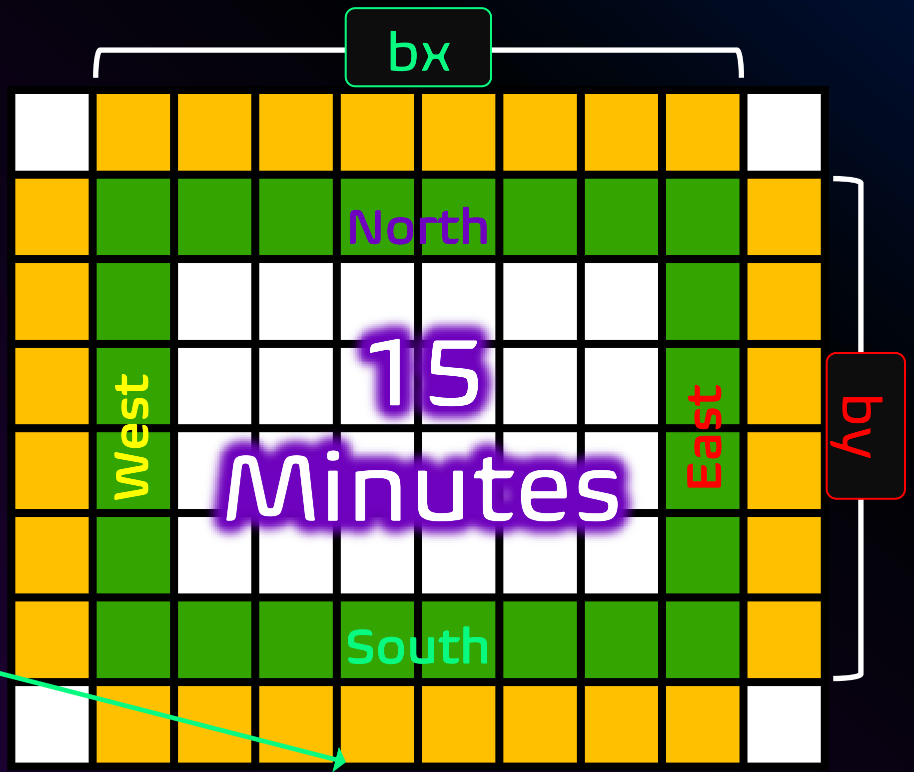
Pack Send Buffer (green cells)

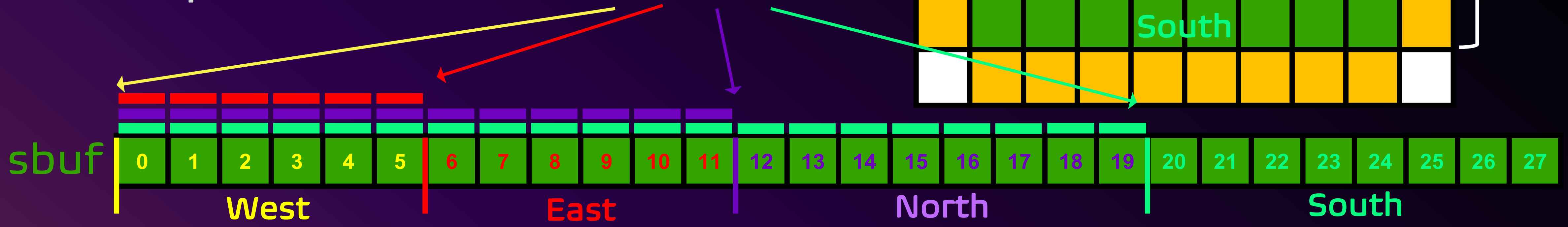
- West, East, North, South

  Recall Neighbor Ordering (X-, X+, Y-, Y+)

  Counts(4) = {?, ?, ?, ?}

  Displacements(4) = {?, ?, ?, ?}





EXERCISE.5 GRAPH PARTITIONING WITH METIS

Step 1: cd exercises/Exercise.5-METIS/c

Step 2: Execute the Demos Programs in

- ✓ Build METIS (./build-metis.sh)
- ✓ cd demo.1.box

  >> make; ./MetisDemo
- ✓ cd demo.2.mesh

  >> make; ./MetisDemo <MeshID> <nparts>

Step 3: Visualize the partitioned meshes (*.vtu) in VisIt/Paraview.

Step 4: Examine the source codes for METIS API calls.