TECHNISCHE UNIVERSITÄT DARMSTADT

Large-Scale Parallel Computing

Aamer Shah

shah@cs.tu-darmstadt.de

EXERCISE 5

Hands-on session

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hands-on session
- Students will develop the solution during the exercise session

1. Login to Lichtenberg cluster (with –Y option)
2. Copy **`ex05.tgz`** from **`/home/as65huly/public`**
3. Run the **`find_prime_serial`** program
   - Usage: **`./find_prime_serial <starting_val> <ending_val>`**
4. Submit the batch job **`job_find_prime.sh`** and check the output
   - Attention: Change the email address in the batch job before submitting!

Task 1

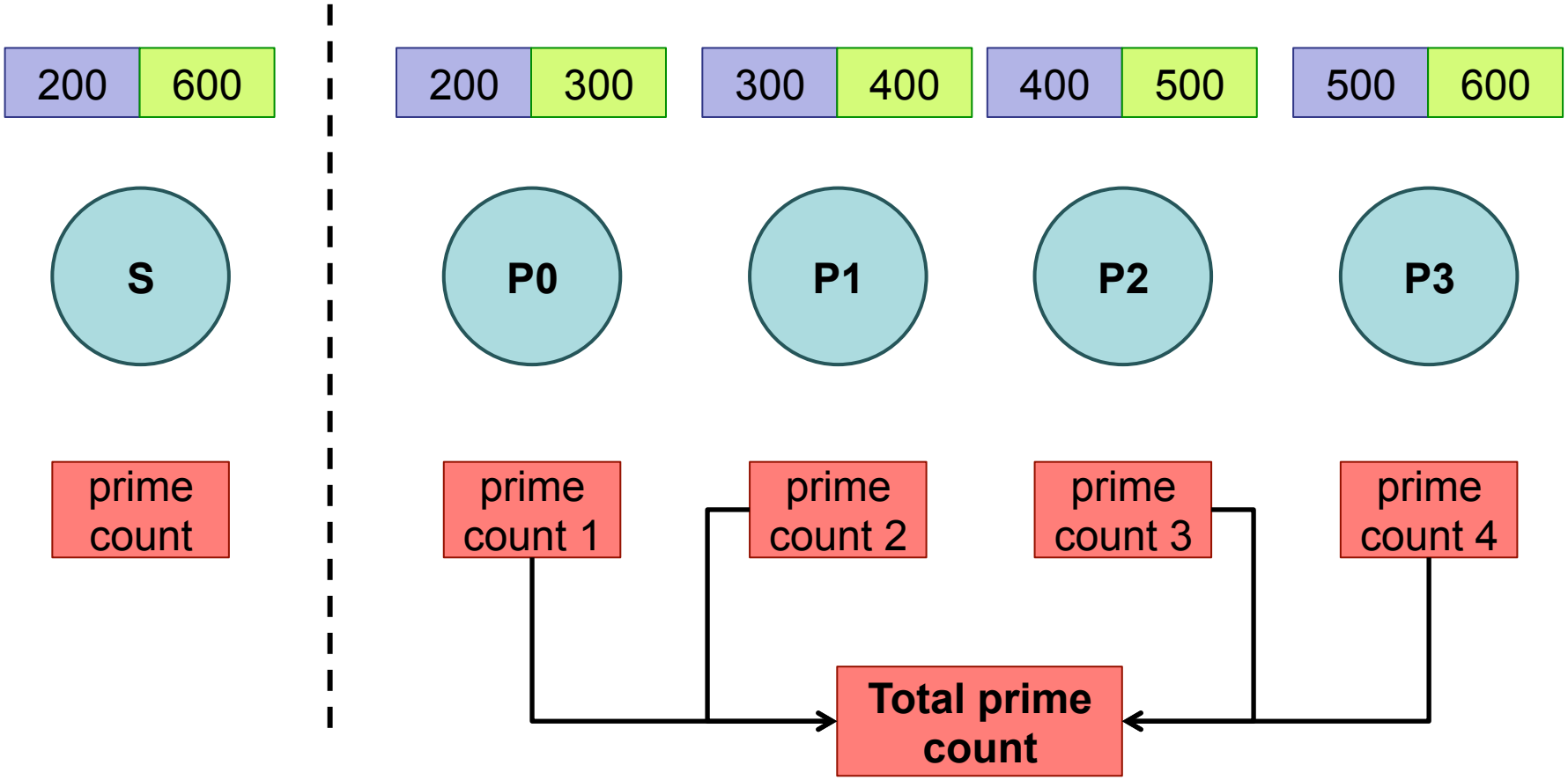
TECHNISCHE UNIVERSITÄT DARMSTADT

- Given is a serial program that counts the total number of prime numbers in a given range

- Task
  - Implement an MPI version of the program
    - Distribute the number range equally among the processes
  - Count the total number of prime numbers in the range
  - Calculate the total time taken by each process
  - Calculate time taken by each process
  - Find the percentage difference between maximum, minimum and average times

Task 1

TECHNISCHE UNIVERSITÄT DARMSTADT



Task 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- Percentage difference between maximum, average and minimum

$$perc\_diff\_max\_min = \frac{(maximum - minimum)}{(maximum)}$$

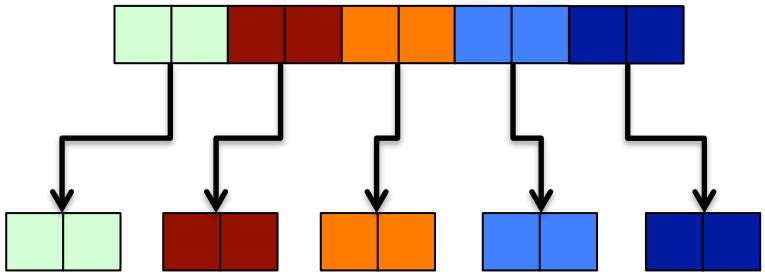
- Similar for
  - Percentage difference between maximum and average
  - Percentage difference between average and minimum

Task 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- How to distribute the number range?
  - Make an array with **`starting_val`** and **`ending_val`** for each process
  - Scatter the array among processes

```
MPI_Scatter(const void *sendbuf, int sendcount,
MPI_Datatype sendtype, void *recvbuf, int recvcount,
MPI_Datatype recvtype, int root, MPI_Comm comm)
```



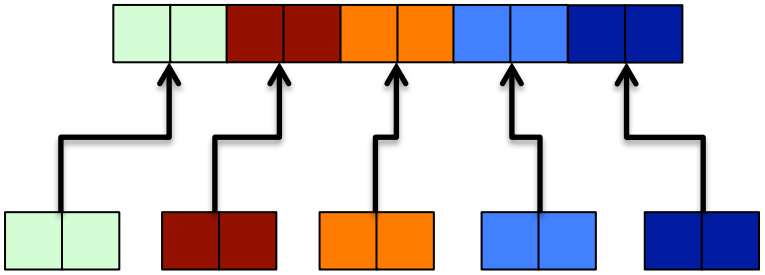
Task 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- How to find total count? Maximum, minimum and average times?
  - Approach 1:
    - Gather all the data at process 0
    - Serially find sum, maximum, minimum and average

Inefficient serial execution does not take advantage of multiple processes

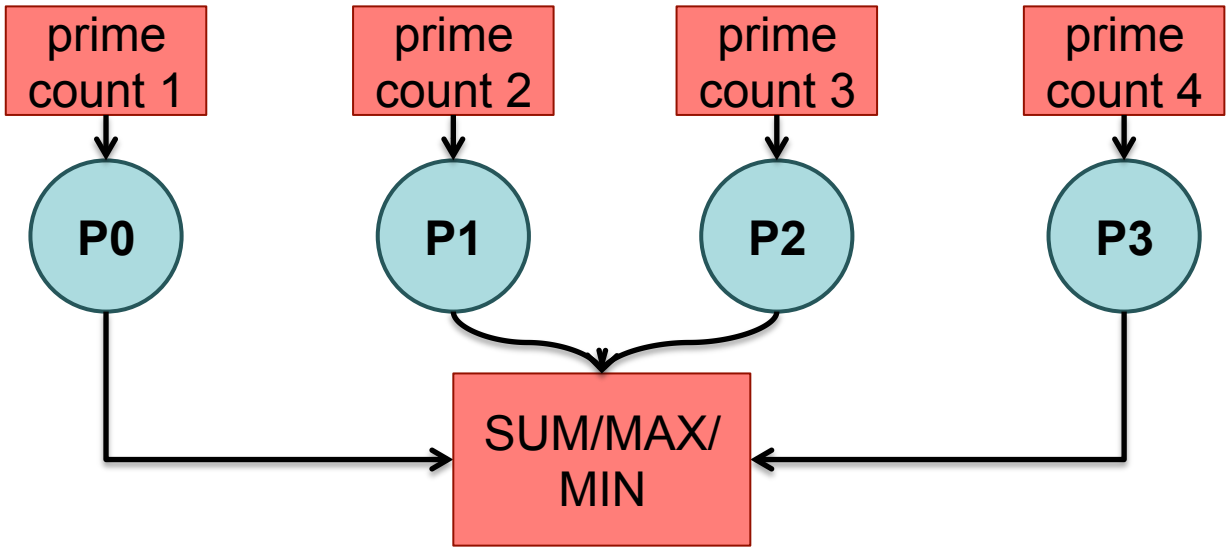
```
MPI_Gather(const void *sendbuf, int sendcount,
MPI_Datatype sendtype, void *recvbuf, int recvcount,
MPI_Datatype recvtype, int root, MPI_Comm comm)
```



Task 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- How to find total count? Maximum, minimum and average times?
  - An operation, similar to gather (mirror of broadcast), but that also applies a certain operator on the data, like sum, maximum, etc.

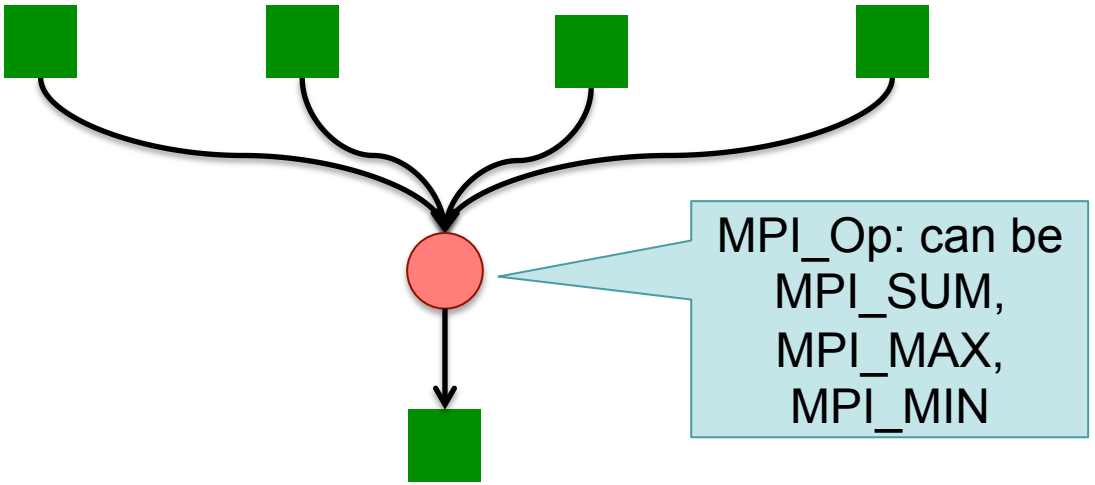


Task 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- Reduction operations

```
MPI_Reduce(void *sendbuf, void* recvbuf, int count,
MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm
comm )
```



Task 1

TECHNISCHE UNIVERSITÄT DARMSTADT

- Given is a serial program that counts the total number of prime numbers in a given range

- Task
  - Implement an MPI version of the program
    - Distribute the number range equally among the processes
  - Count the total number of prime numbers in the range
  - Calculate the total time taken by each process
  - Calculate time taken by each process
  - Find the percentage difference between maximum, minimum and average times

Task 1 steps

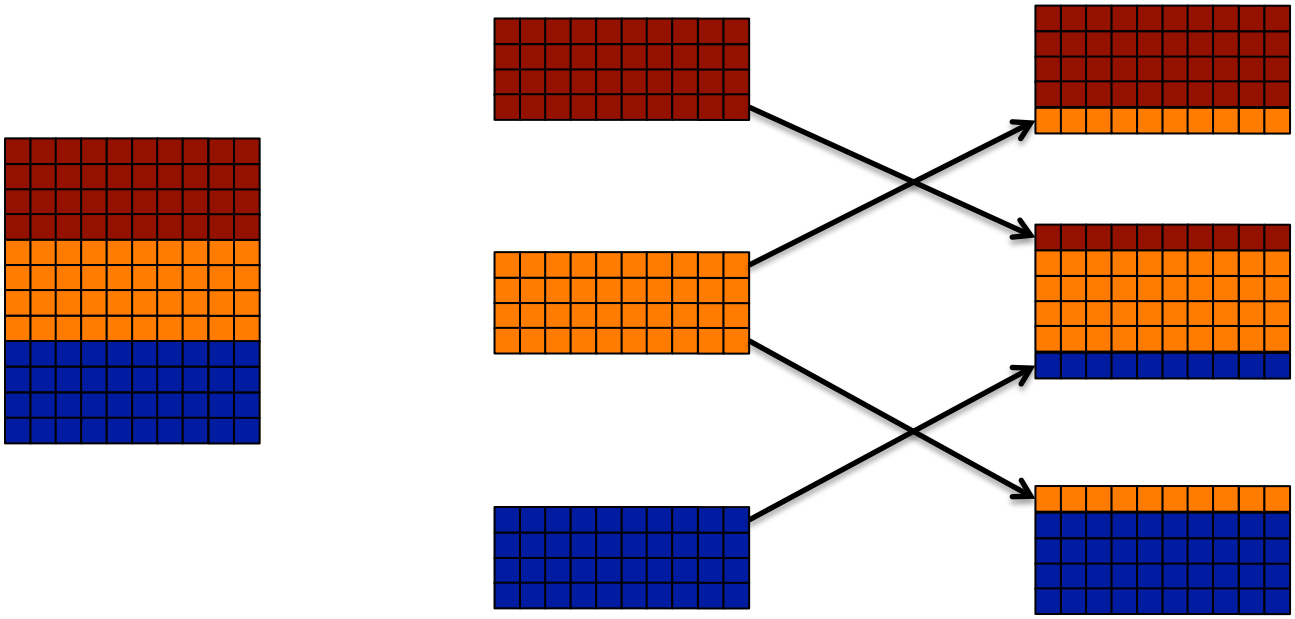
TECHNISCHE UNIVERSITÄT DARMSTADT

1. Initialize MPI
2. Distribute number range among processes
3. Each process prime in its range
4. Use MPI_Reduce to collect prime count
5. Use MPI_Reduce to collect maximum, minimum and average time
6. Rank 0 prints the output

NEWS filter – data distribution

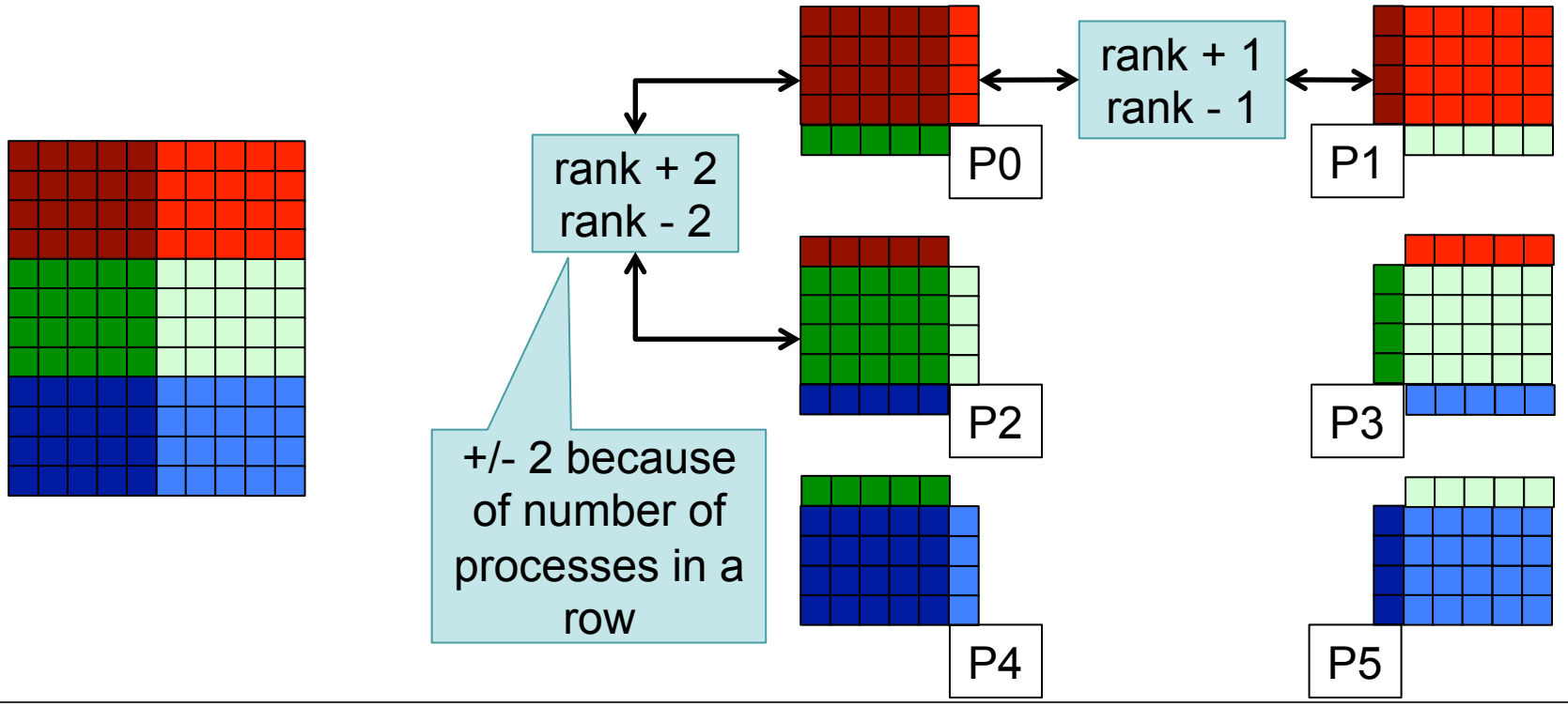
TECHNISCHE UNIVERSITÄT DARMSTADT

- Implement row wise data distribution among processes
  - Processes will need one row above and below their image share to apply the NEWS filter
  - How to identify a process above and below:
    - rank – 1 is above, rank + 1 is below



NEWS filter – data distribution

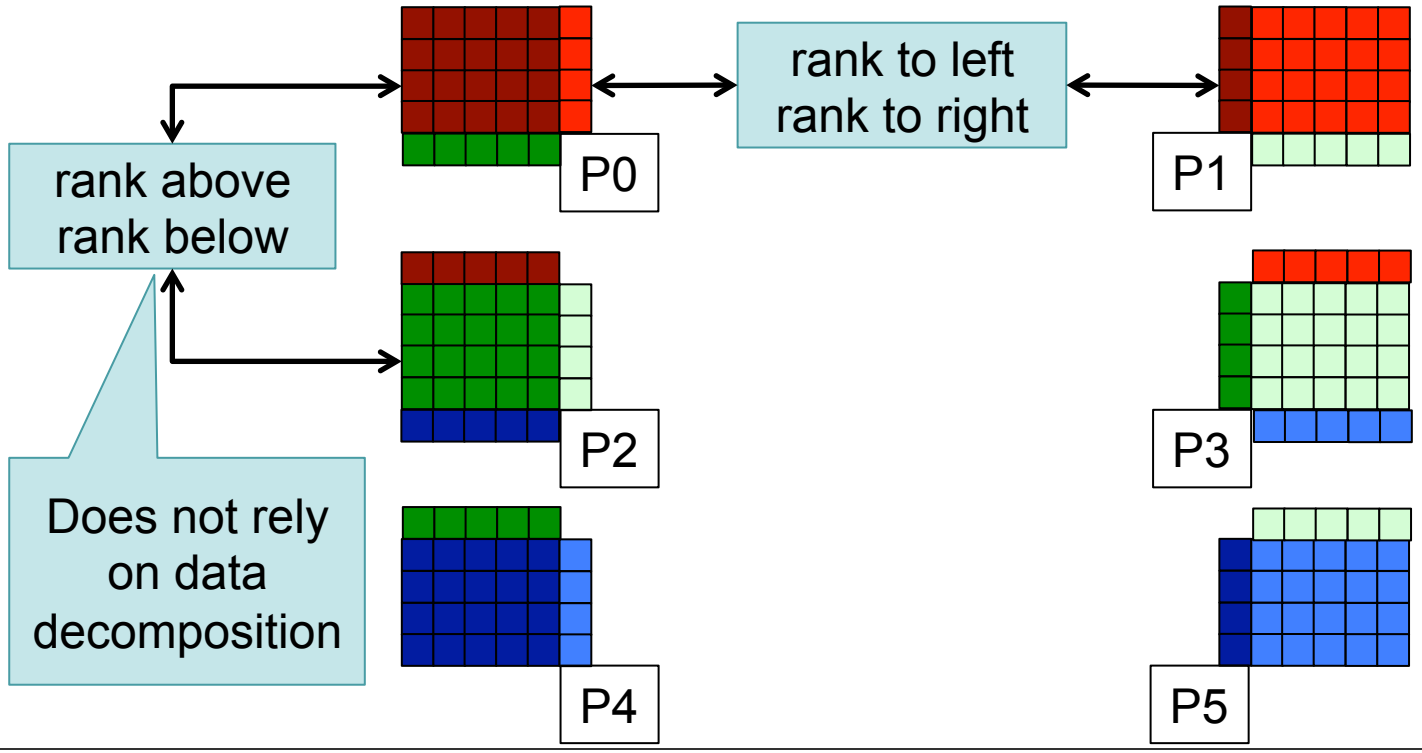
- What if we data distribution is complex, and we need data from all neighbors?
  - Finding neighbors can become complicated



Virtual topology

TECHNISCHE UNIVERSITÄT DARMSTADT

- Instead of using ranks, why not simply say left neighbor, right neighbor, above neighbor, below neighbor?



Virtual topology

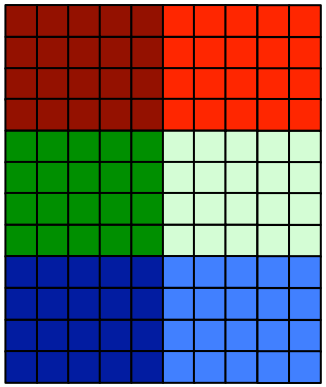
- How to achieve such configuration?
- What is a process rank?
  - Just a unique identifier assigned by MPI runtime
  - To find out a process rank, we use MPI_COMM_WORLD, which acts as an identifier for the default naming scheme
- Can we create another unique identifier scheme that allows left, right, above, below operations?
  - Virtual topologies
    - The virtual topology should have its own identifier, similar to MPI_COMM_WORLD, to find a process's unique id in the virtual topology

Virtual topology

TECHNISCHE UNIVERSITÄT DARMSTADT

- A way of assigning unique identifiers to processes following a Cartesian topology

```
int MPI_Cart_create(MPI_Comm oldcomm, int ndims,
                    int *dims, int *isperiodic,
                    int reorder, MPI_comm *newcomm);
```



```
int MPI_Cart_create(
MPI_Comm MPI_COMM_WORLD,
int 2, //ndims 2D
Int {2, 3},  //2 columns, 3 rows
int 0, //periodic - no
int 0, //reorder, ignore for now
MPI_comm *virtual_topo_comm);
```

Virtual topology

TECHNISCHE UNIVERSITÄT DARMSTADT

- How to find neighbors?
  - Above, below, etc. limited to 2 dimensions
- MPI provides generic function to find neighbors

Virtual topology communicator

Direction means the dimension in which to move:
Left, right is along the row dimension
Up, down is along the column dimension

```
int MPI_Cart_shift(MPI_Comm comm, int direction,
                   int displacement, int *src, int *dest);
```

Displacement is movement in the dimension:
1 step neighbors, 2 step neighbors, etc.

Process at my_position - displacement

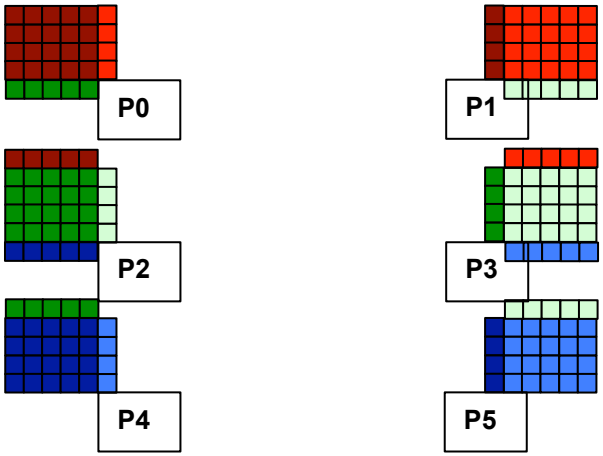
Process at my_position + displacement

Virtual topology

TECHNISCHE UNIVERSITÄT DARMSTADT

```
int MPI_Cart_shift(MPI_Comm virtual_topo_comm,
int 0 /*direction*/, int 1 /*displacement*/,
int *proc_left, int *proc_right);
```

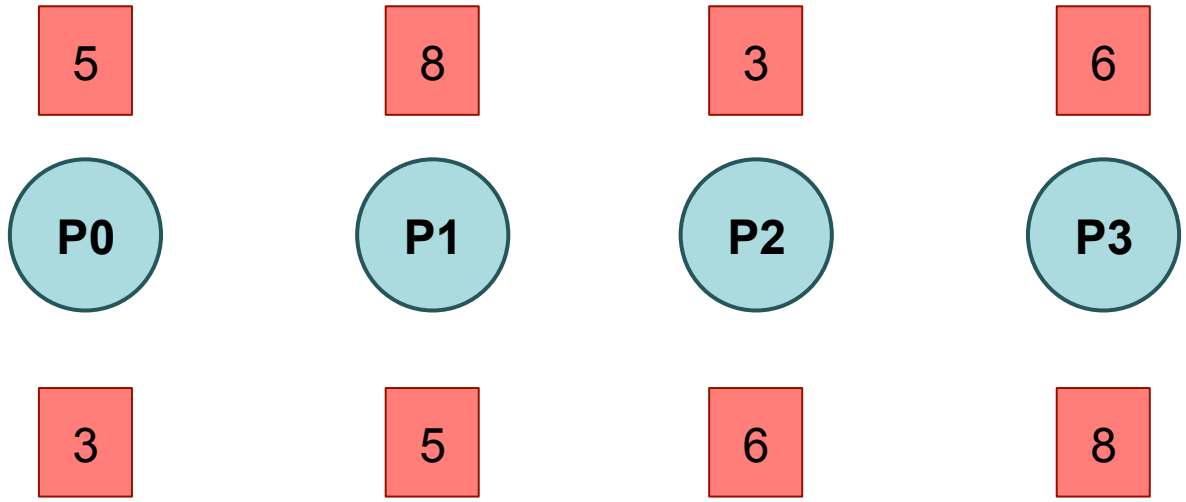
```
int MPI_Cart_shift(MPI_Comm virtual_topo_comm,
int 1 /*direction*/, int 1 /*displacement*/,
int *proc_below, int *proc_above);
```



Task 2

TECHNISCHE UNIVERSITÄT DARMSTADT

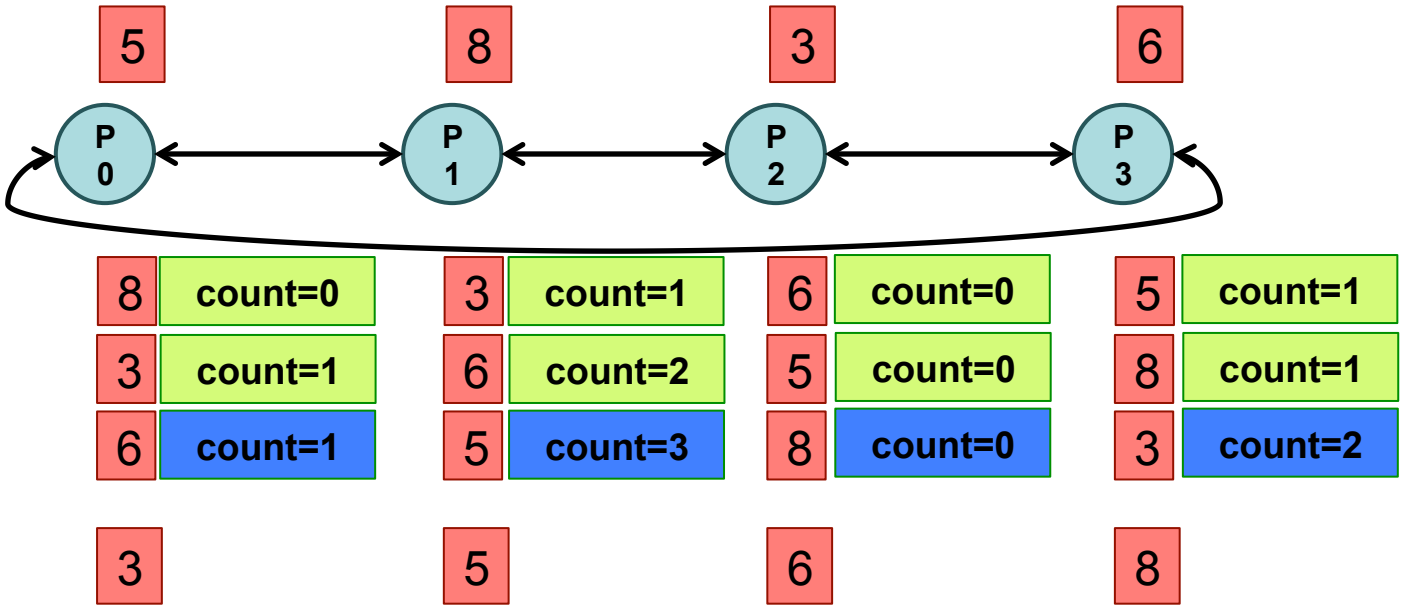
- Implement counting sort using ring topology



Task 2

TECHNISCHE UNIVERSITÄT DARMSTADT

- Implement counting sort using ring topology
  1. Create a ring topology (ndims = 1, periodic = true)
  2. Shift values along the ring and count how many are smaller than original value
  3. The count value at the end gives the destination rank



Task 2

TECHNISCHE UNIVERSITÄT DARMSTADT

- Use MPI_Sendrecv to exchange values between processes in ring
- For comparison and shifting, use MPI_Cart_shift for finding neighbors
- For sending values in final sorted order, each process knows where to send the data, but **does not know from which process to receive the data**
- Use MPI_ANY_SOURCE for that

```
int MPI_Sendrecv(void *sendbuf, int sendcount,
                 MPI_Datatype sendtype, int dest,
                 int sendtag,
                 void *recvbuf, int recvcount,
                 MPI_Datatype recvtype, int source,
                 int recvtag,
                 MPI_Comm comm, MPI_Status *status)
```