Large-Scale Parallel Computing

Aamer Shah

shah@cs.tu-darmstadt.de

EXERCISE 4

Hands-on session

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

- Objective:
  - Everyone should be able to program in MPI by the end of the session
  - MPI very important for exam

Exercise task

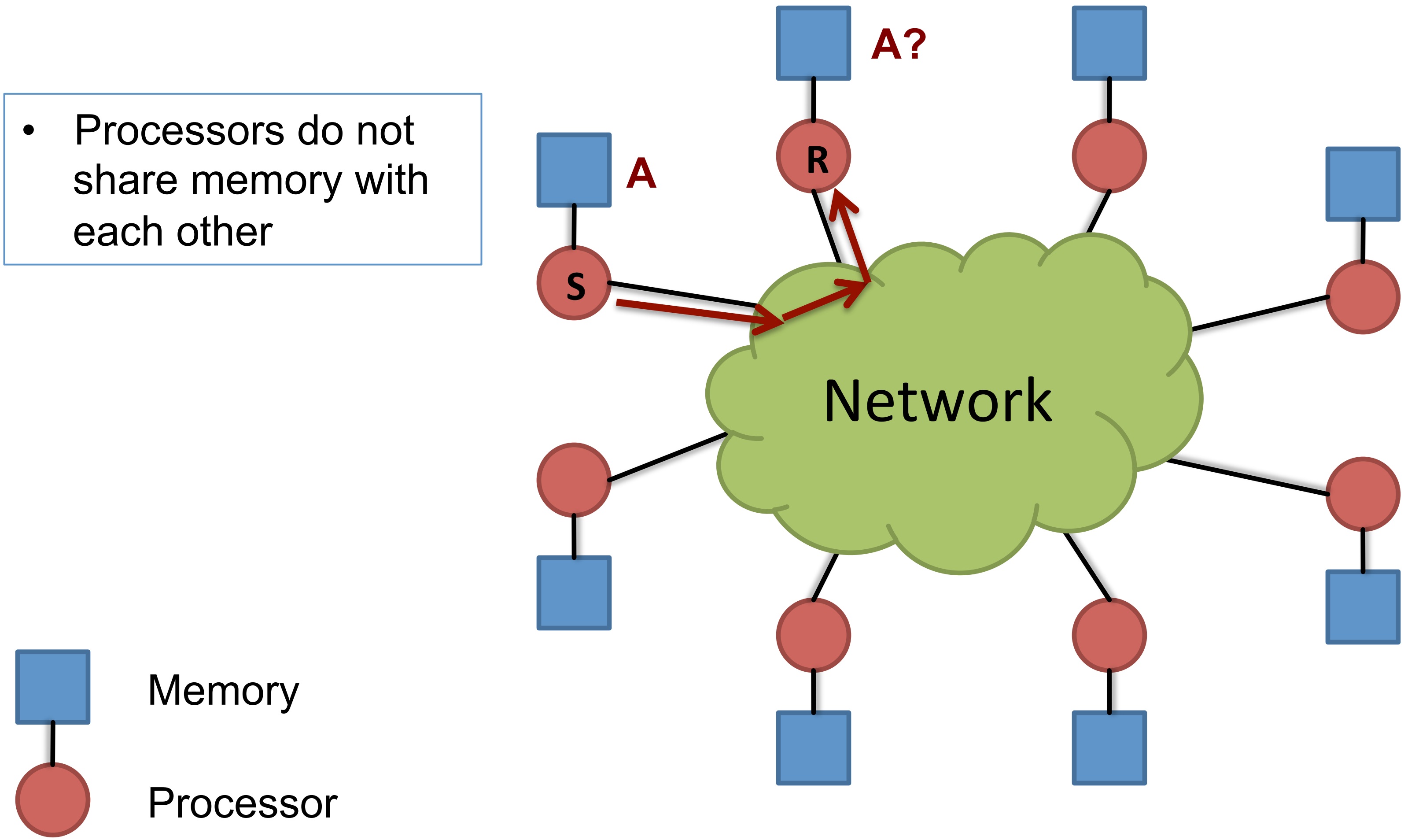
TECHNISCHE UNIVERSITÄT DARMSTADT

- A serial version of the NEWS median filter was provided
- Exercise task
  - Develop a parallel MPI version of the program
  - A small MPI program was also given for learning purpose
- Pre-requisite of the session:
  - ✓ Access to a computer
  - ✓ Login into the Lichtenberg cluster
  - ✓ Compile an MPI program on the cluster
  - ✓ Use a text editor on the cluster
  - ✓ Write and submit batch jobs on the cluster
  - ✓ Know basics of MPI

MPI – basic concepts

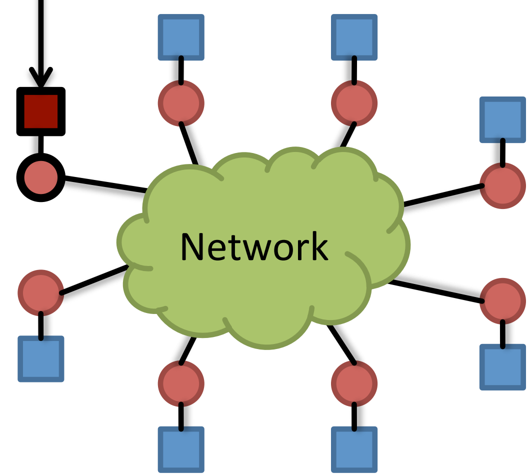
TECHNISCHE UNIVERSITÄT DARMSTADT

- Processors do not share memory with each other



MPI – basic concepts

ssh to cluster



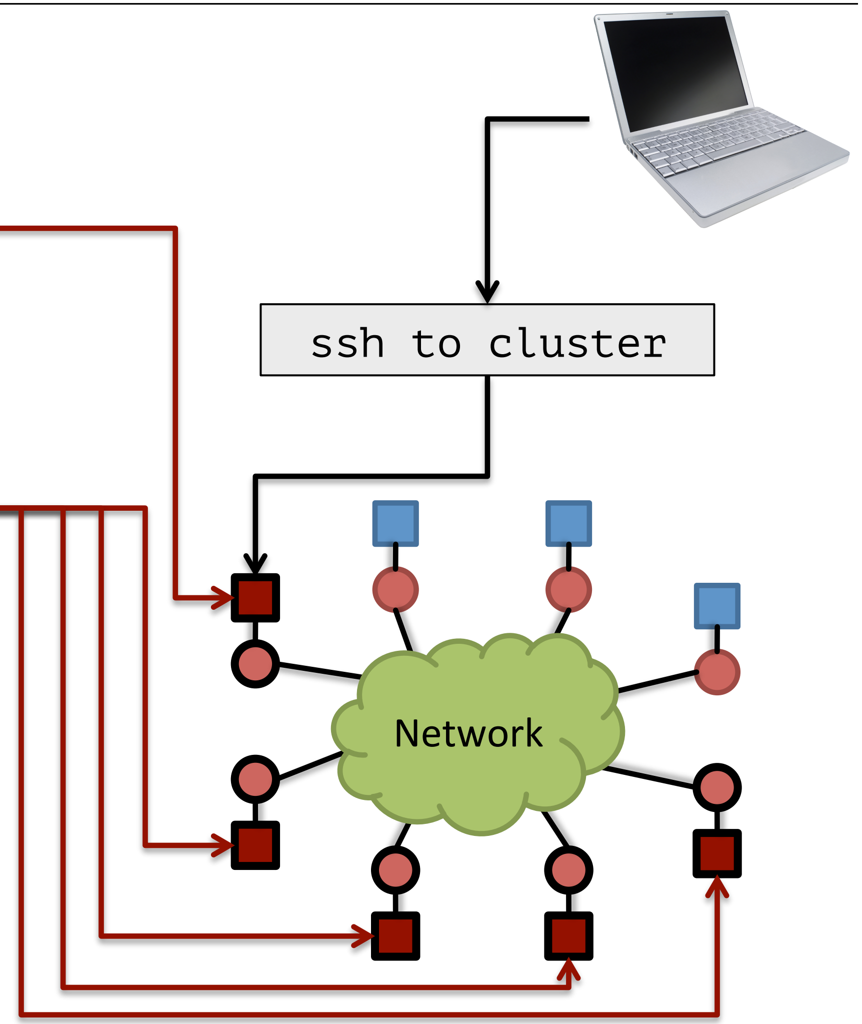
- When user connects to a cluster, they get access to a single processor
- Anything the user executes, gets executed on that processor
- How to execute the program on multiple processors?

MPI – basic concepts

```
printf("hello world!\n");
```

```
> ./hello_world
> hello world!
```

```
> mpiexec -np 4 ./hello_world
> hello world!
> hello world!
> hello world!
> hello world!
```



- Use **mpiexec** to execute programs on multiple processors
- ➢ However, the **same program** execute on the processors

MPI – basics

- Does it make sense to execute the same program on multiple processors?
  - Doing the same task multiple times
- Different types of parallel processing needs

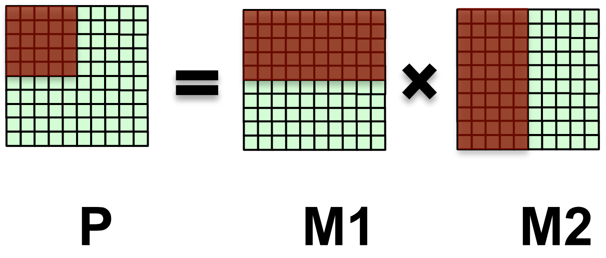
Word processor:

- Thread 1 takes user input
- Thread 2 updates the GUI
- Thread 3 does spell check
- etc
- MIMD:
  - Multiple Instruction Multiple Data

➢ For typical scientific code, running the same program on multiple processors makes sense

Typical scientific code

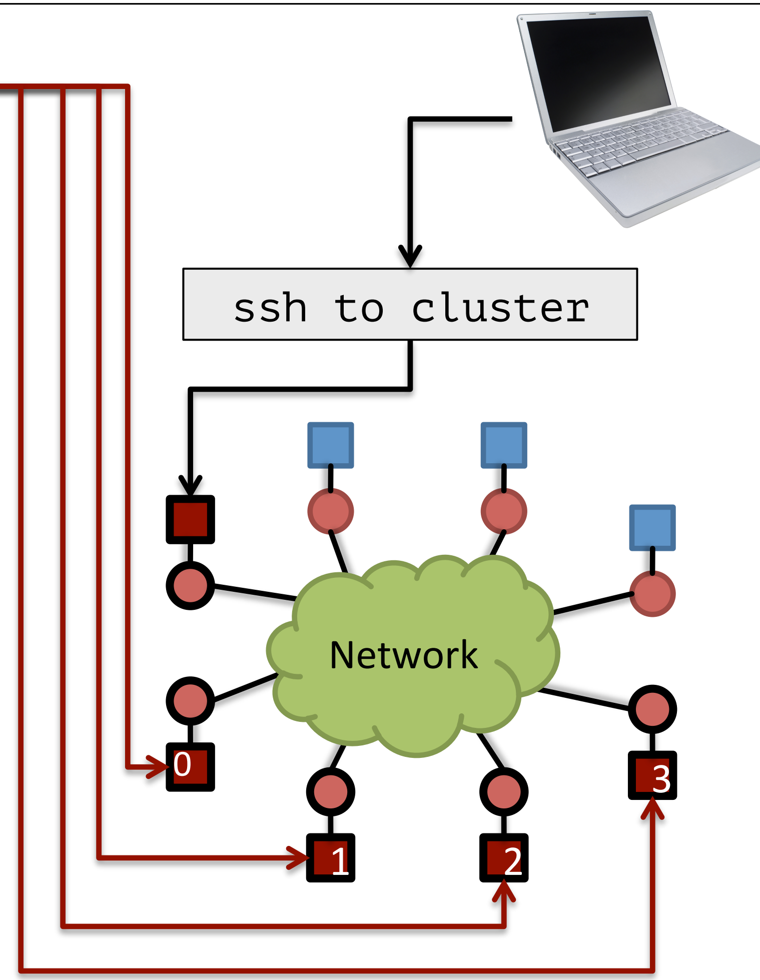
- Matrix multiplication
- Same algorithm on all processors
- Data is divided among the processes
- SIMD
  - Single Instruction Multiple Data



MPI – basic concepts

```
> mpiexec -np 4 ./matmul
```

- Makes sense to execute the same program on multiple processors, but
- Processes still need to make distinction among themselves
  - Some one has to read the input data
  - Some one has to write the output data
- Processes also need to communicate with each other
  - The data has to be distributed properly among the processes
- MPI provides
  - Rank ids to identify different processes
  - Functions to communicate between processes



MPI – basic concepts

- Processes still need to make distinction among themselves
  - MPI automatically assigns IDs (called **rank**) to each process
  - MPI also automatically groups all the processes in a communicator, called MPI_COMM_WORLD

```
MPI_Comm_rank(MPI_COMM comm, int *rank);
MPI_Comm_size(MPI_COMM comm, int *size);
```

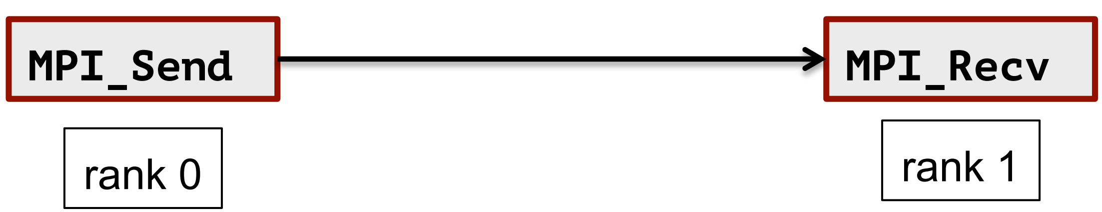
```
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
MPI_Comm_size(MPI_COMM_WORLD, &comm_size);
```

MPI – basic concepts

- Processes also need to communicate with each other
  - MPI provides two sets of functions
    - Point-to-point: between pair of processes

```
MPI_Send(const void *buf, int count, MPI_Datatype
datatype, int dest, int tag, MPI_COMM comm);

MPI_Recv(void *buf, int count, MPI_Datatype datatype, int
source, int tag, MPI_Comm comm, MPI_Status *status)
```



MPI – basic concepts

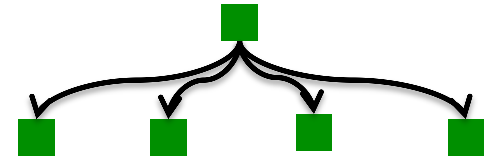
```
if(my_rank == 0)
  MPI_Send(sndbuf, 2, MPI_INT, 1, 0, MPI_COMM_WORLD);
if(my_rank == 1)
  MPI_Recv(rcvbuf, 2, MPI_INT, 0, 0, MPI_COMM_WORLD, &stat);
```

Every send should have a corresponding receive operation

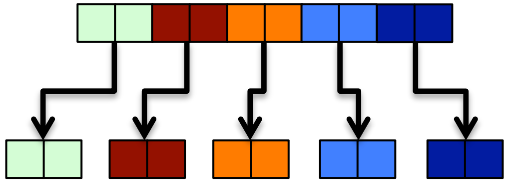
MPI – basic concepts

- Collectives: involves a complete communicator group

```
MPI_Bcast(void *buffer, int count, MPI_Datatype
datatype, int root, MPI_Comm comm )
```



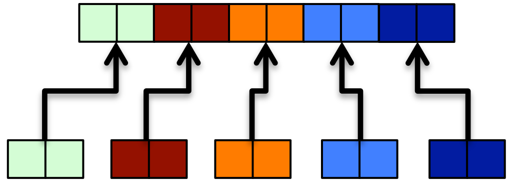
```
MPI_Scatter(const void *sendbuf, int sendcount,
MPI_Datatype sendtype, void *recvbuf, int
recvcount, MPI_Datatype recvtype, int root,
MPI_Comm comm)
```



sendbuf, sendcount, sendtype only valid on root

recvbuf, recvcount, recvtype only valid on root

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



MPI – basic concepts

TECHNISCHE UNIVERSITÄT DARMSTADT

```
> mpiexec -np 4./matmul
```

Rank 0 reads input file

```
if (my_rank == 0) read_input();
```

Rank 0 distributes the data to all processes

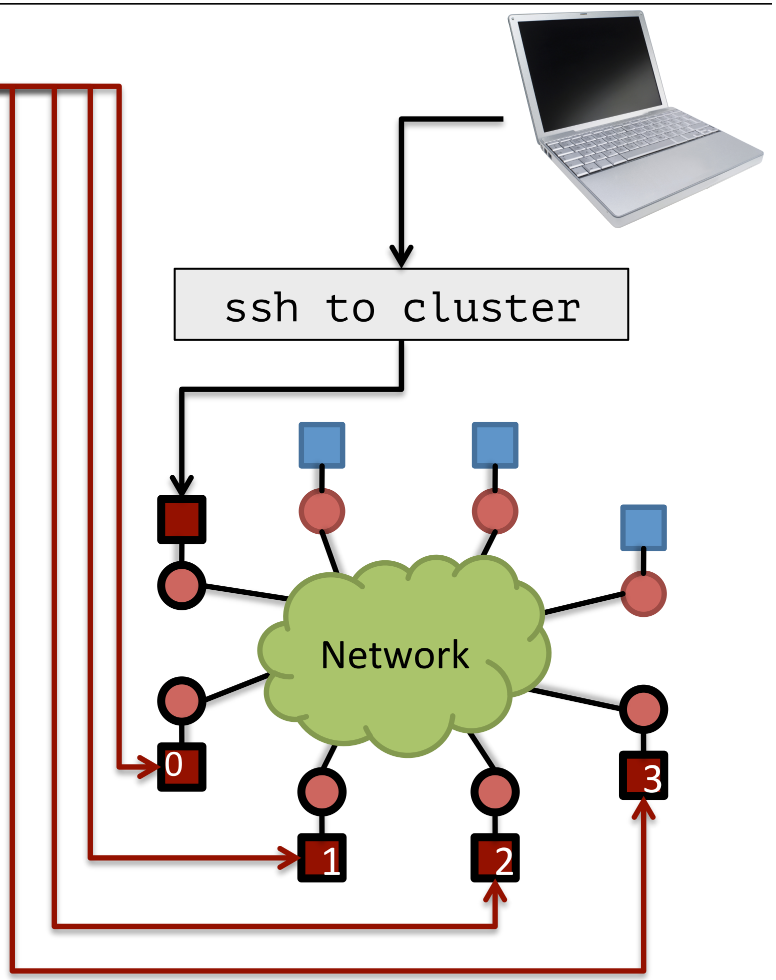
```
MPI_Scatter(sendbuf, cnt, MPI_INT, rcvbuf,
cnt, MPI_INT, 0, MPI_COMM_WORLD);
```

Each process performs matrix multiplication

```
mat_mul(mat_1, mat_2, prod);
```

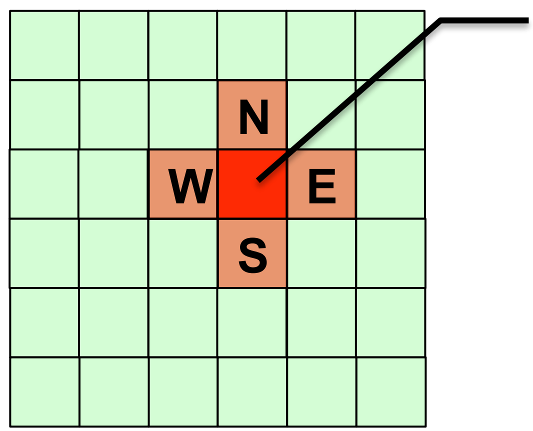
All processes send their product to rank 0

```
MPI_Gather(sendbuf, cnt, MPI_INT, rcvbuf,
cnt, MPI_INT, 0, MPI_COMM_WORLD);
```



NEWS filter

- NEWS median filter is used to remove salt-and-pepper noise from images

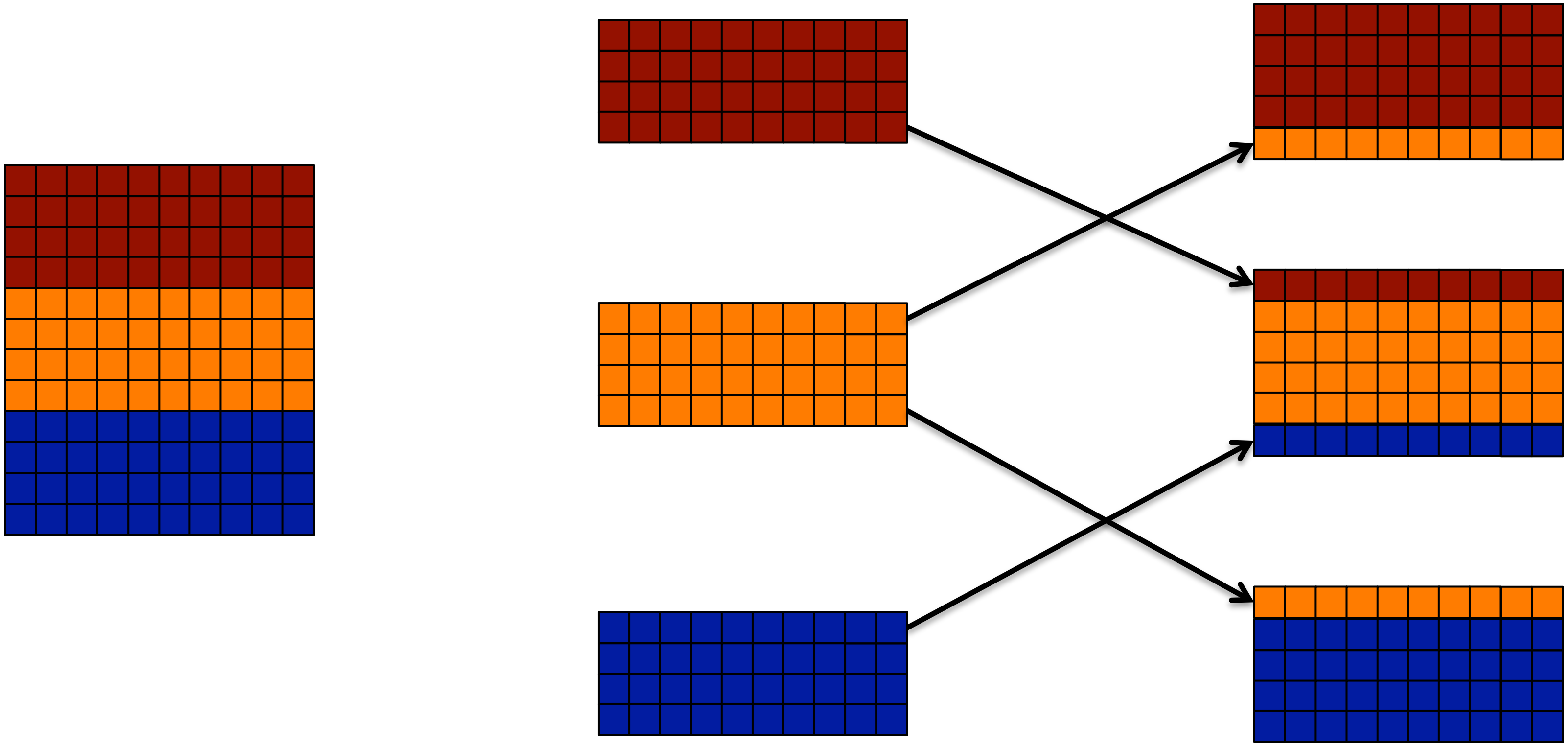


To apply the NEWS filter on a pixel (X,Y), the following pixels values are also needed

- North: (X,Y-1)
- East: (X+1,Y)
- West: (X-1,Y)
- South: (X,Y+1)

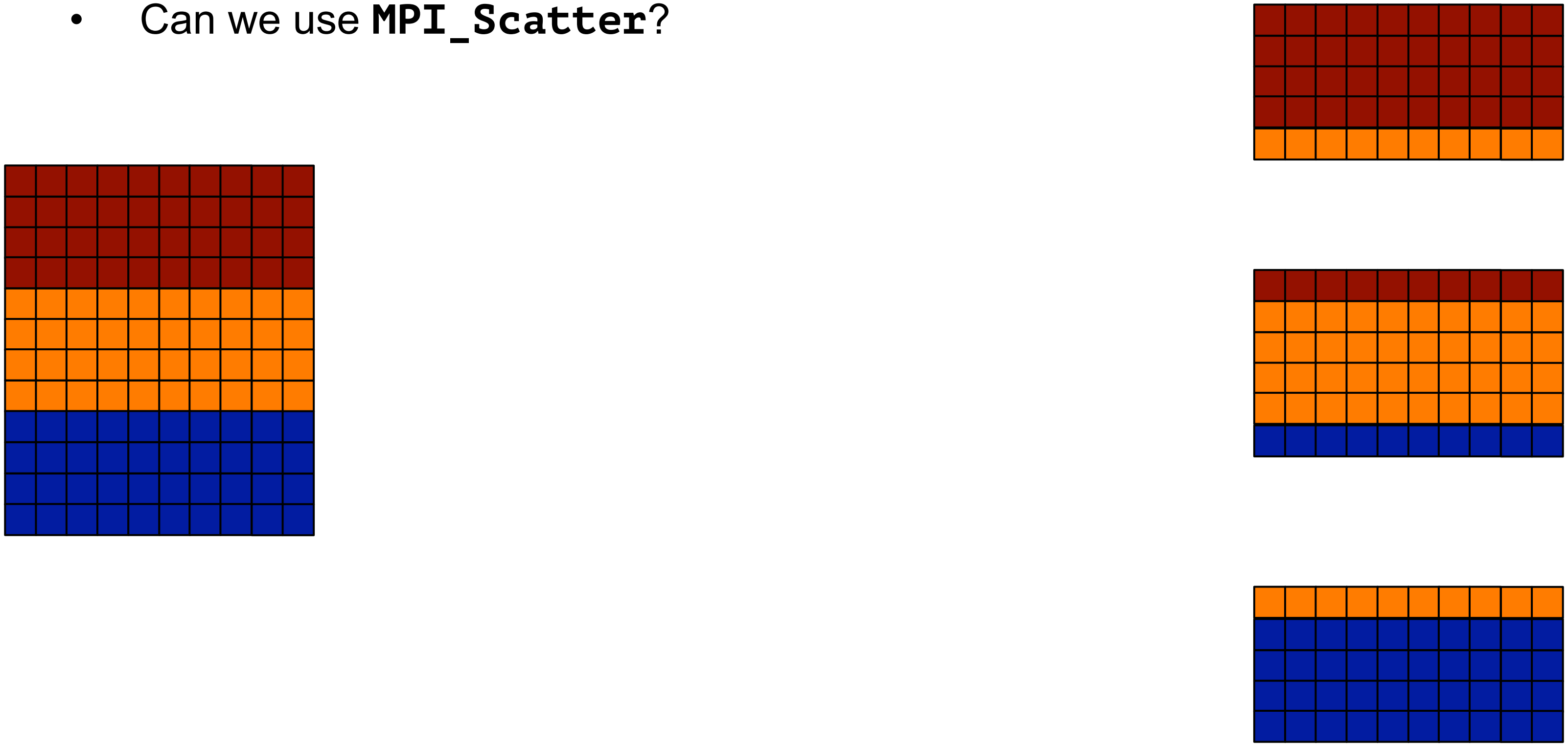
NEWS filter – data distribution

- Implement row wise data distribution among processes
  - Processes will need one row above and below their image share to apply the NEWS filter



NEWS filter – data distribution

- Implement row wise data distribution among processes
  - How to distribute the data among processes?
    - Can we use `MPI_Scatter`?



MPI – basic concepts

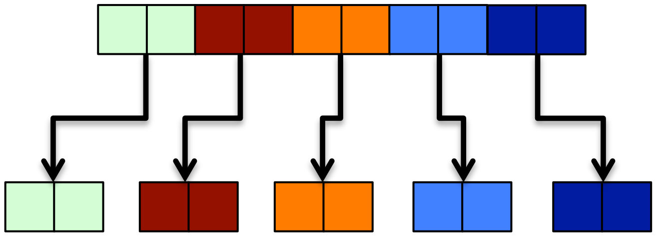
- Collectives: involves a complete communicator group

sendbuf, sendcount, sendtype only valid on root

The same amount of data is sent to all processes

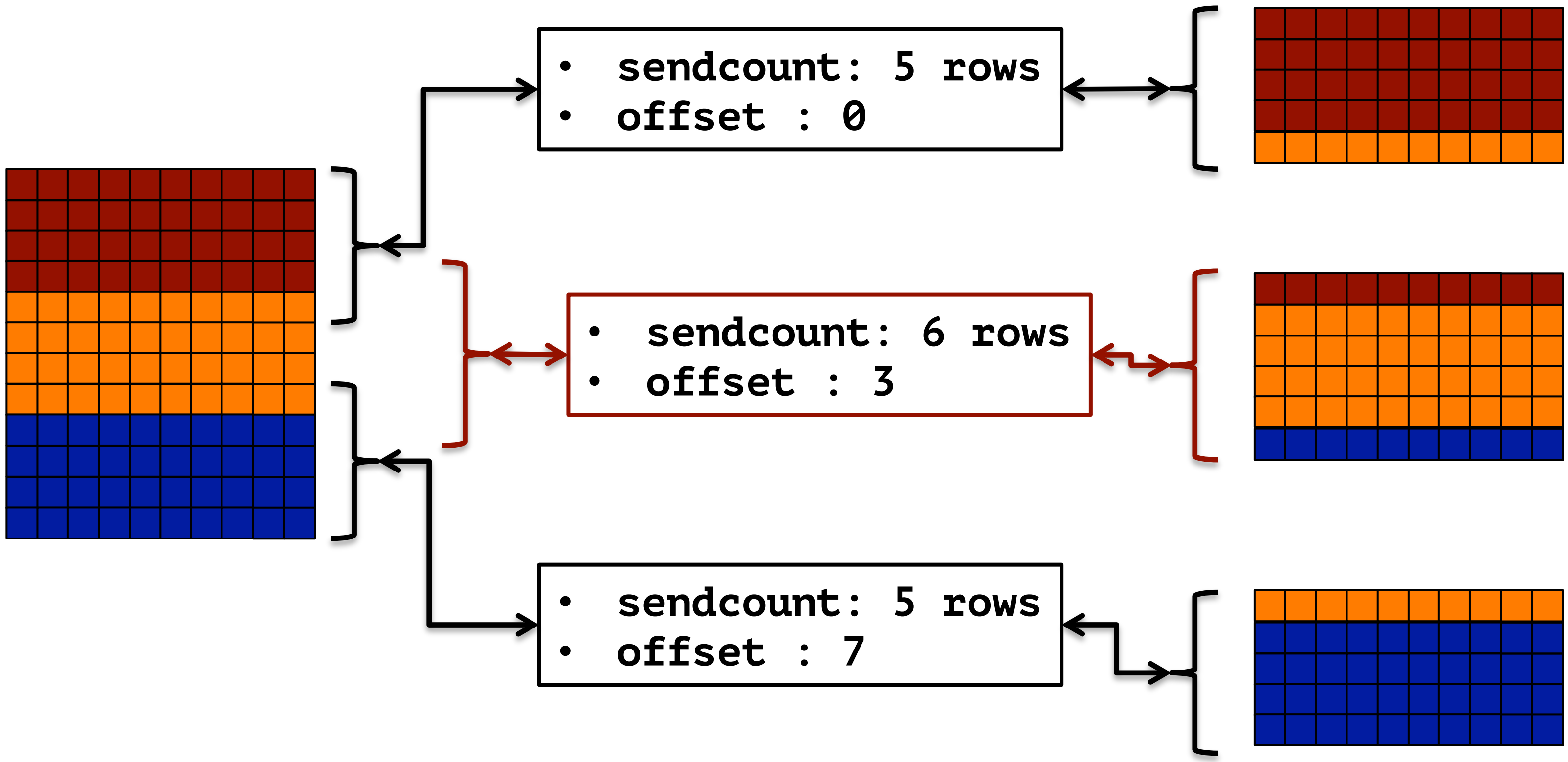
```
MPI_Scatter(const void *sendbuf, int sendcount, MPI_Datatype
sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype,
int root, MPI_Comm comm)
```

```
• Offset for rank 0 : 0
• Offset for rank 1 : sendcount
• Offset for rank 2 : 2 * sendcount
```



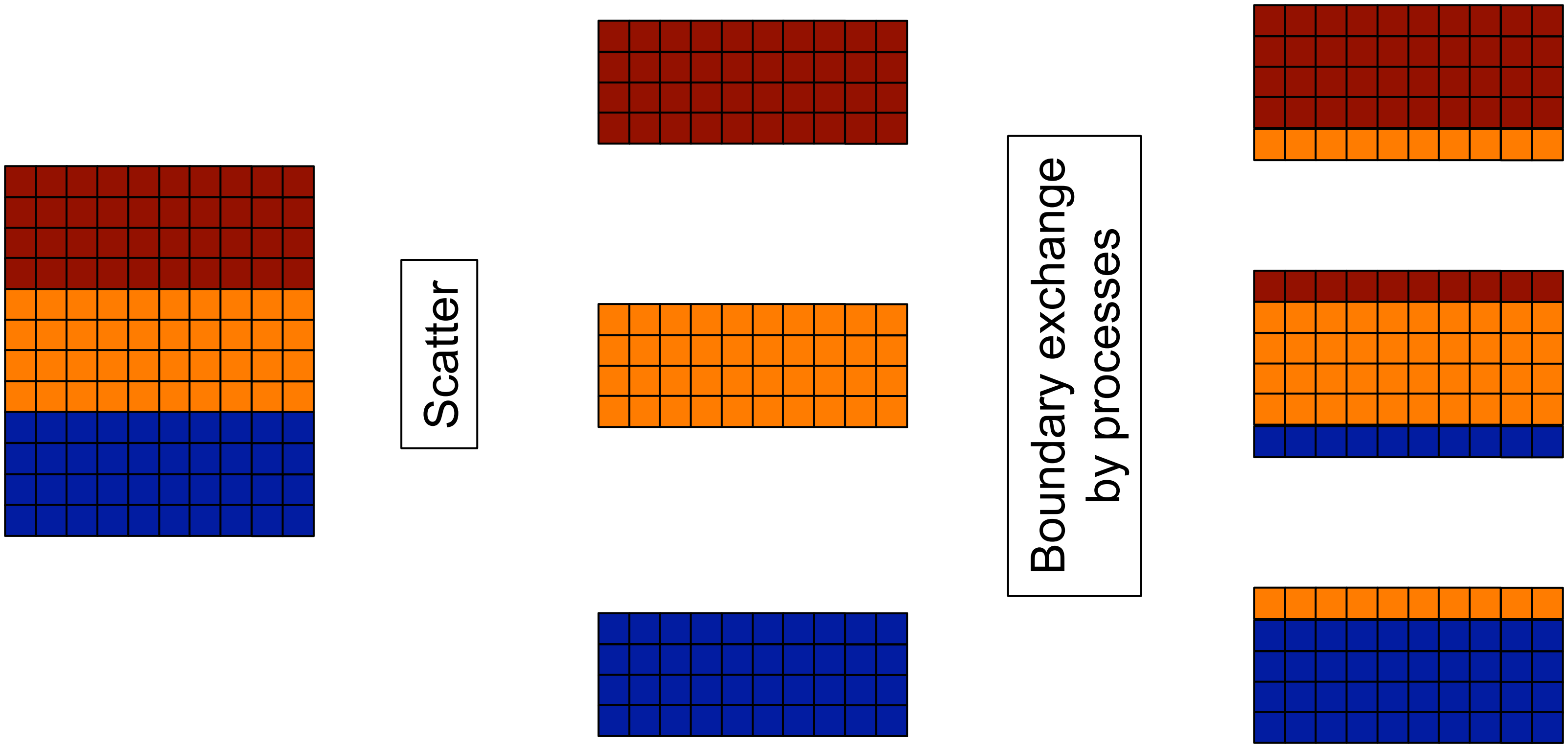
NEWS filter – data distribution

- Implement row wise data distribution among processes
  - **`MPI_Scatterv`**?



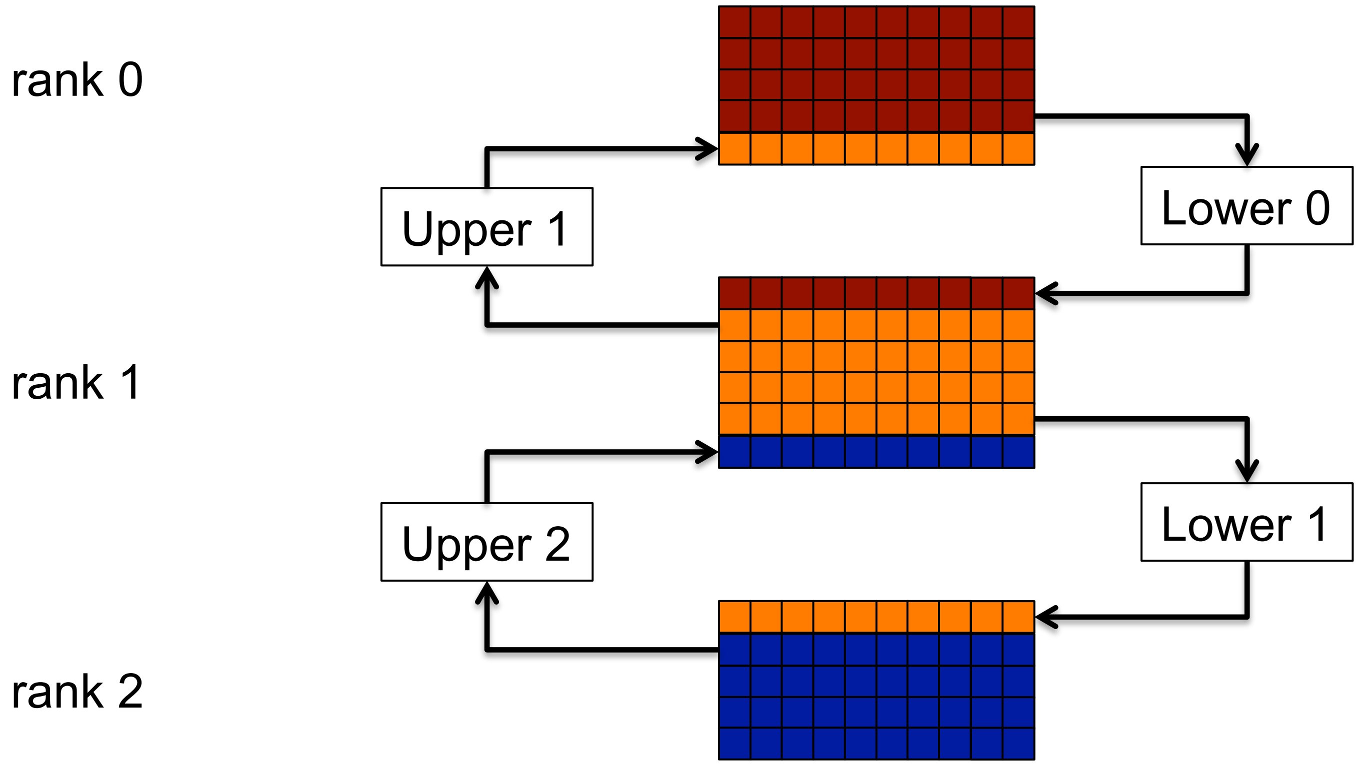
NEWS filter – data distribution

- First use scatter to distribute main image parts
- Then exchange boundary rows among processes
  - Gives flexibility to apply the filter twice



NEWS filter – data distribution

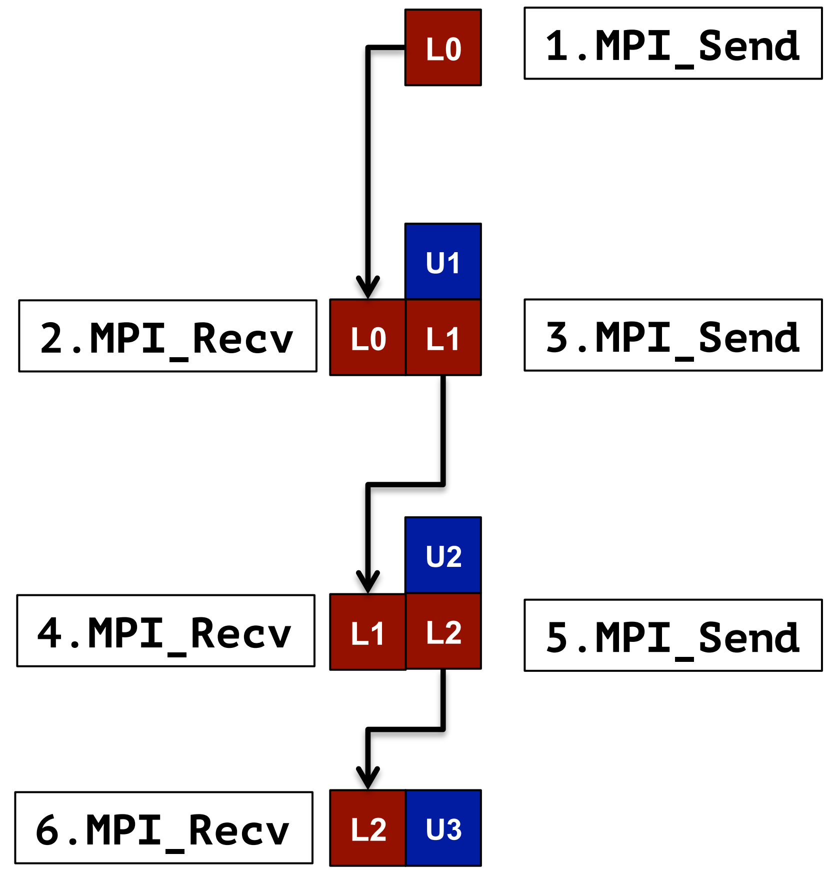
TECHNISCHE UNIVERSITÄT DARMSTADT



Boundary exchange by processes


```
if (my_rank > 0)
    MPI_Recv(L
from my_rank - 1)

if (my_rank <
comm_size - 1)
    MPI_Send(L to
my_rank + 1)
```



Exercise task – hands on session

- Everyone should first
  1. Login to Lichtenberg cluster
     - Make sure not to overload a single login node
  2. Copy the ex04.tgz archive and extract it
  3. Compile the helloworld.c program
  4. Submit the batch job
  5. Check the output
- Pre-requisite
  - Are you able to login with –Y option?
    - Quick test, run **`display`** command and see if a GUI opens

> Load the following modules first:
> - **`gcc`**
> - **`intel`**
> - **`openmpi/intel`**

Exercise task

- A serial version of the NEWS filter is provided, along with a sample PGM image file
- Implement a parallel MPI version of the program
- Use the **`display`** command to view the PGM image
- **Attention:** PGM images have large size. First convert the image to JPEG before displaying it
- Use **`convert <image_file.pgm> <image_file.jpg>`** to convert to JPEG
- Session will end at 2:50 pm

NEWS filter – implementation

1. Add MPI header file and MPI_Init/MPI_Finalize
2. Get process rank and comm size
3. Rank 0 reads the PGM file
4. Rank 0 broadcasts the image height, width and max color
5. Each process calculates the amount of data it will get
6. Rank 0 scatters main image data to all processes
7. Processes send/recv boundary rows to/from each other
8. Processes apply NEWS filter on their data
9. Rank 0 gathers data from all processes
10. Rank 0 writes data back to file

Bonus task

TECHNISCHE UNIVERSITÄT DARMSTADT

- Extend the MPI program so that the NEWS filter is applied twice by each process
- After applying the NEWS filter once, processes exchange boundary rows before applying the filter twice
- After applying the filter twice, rank 0 gathers the image data
- Compare the output of applying the filter once with the output of applying the filter twice