

# MPI Refresher

Sebastian Kuckuk

Zentrum für Nationales Hochleistungsrechnen Erlangen (NHR@FAU)



# An aside on parallelization

- **Shared** memory parallelism
  - **CUDA** (single GPU), **OpenMP**
- All threads run on the **same compute node** (the same GPU)
- All threads can access a **common memory**
- Data exchange via **shared memory** spaces
- **Distributed** memory parallelism
  - **MPI**, **PGAS**
- Processes can run on **different compute nodes**
- Each process works on **its own memory**
- Data exchange via **explicit communication** between ranks

# An aside on parallelization

- MPI and OpenMP are standardized
  - MPI 4.1 (11/2023) – 1166 pages
  - OpenMP 5.2 (11/2021) – 669 pages
- Implementation is separated
  - Open MPI, MPICH, MVAPICH
  - Intel MPI, CRAY MPICH, IBM MPI, ...
  - OpenMP support in various compilers<sup>[1]</sup>
  - Implemented specification version may vary
- Roots in CPU-only applications, extensions for GPUs

[1] <https://www.openmp.org/resources/openmp-compilers-tools/>

# From CUDA to MPI

- CUDA
  - Each CUDA thread executes the kernel body
  - Threads can be identified by their thread and block IDs
  - Without synchronization, all ranks execute asynchronously
- MPI
  - Each MPI rank executes the whole application
  - Ranks can be identified by their rank IDs
  - Without synchronization, all ranks execute asynchronously

# MPI Hello World

- Hello world example

```
#include <mpi.h>
```

```
int main(int argc, char *argv[]) {  
    MPI_Init(&argc,&argv);
```

```
    MPI_Finalize();  
}
```

# MPI Hello World

- Hello world example

```
#include <mpi.h>

int main(int argc, char *argv[]) {
    MPI_Init(&argc,&argv);

    int rank, numRanks;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &numRanks);

    MPI_Finalize();
}
```

# MPI Hello World

- Hello world example

```
#include <mpi.h>

int main(int argc, char *argv[]) {
    MPI_Init(&argc,&argv);

    int rank, numRanks;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &numRanks);

    std::cout << "Hello from rank " << rank << " of " << numRanks << std::endl;

    MPI_Finalize();
}
```

# MPI Hello World

- Compile with

```
mpicxx -O3 mpi-hello-world.cpp \  
-o mpi-hello-world
```

- Run with

```
mpirun -n 4 ./mpi-hello-world
```

- Could print

```
Hello from rank 0 of 4  
Hello from rank 1 of 4  
Hello from rank 2 of 4  
Hello from rank 3 of 4
```

- But could also print

```
Hello from rank 2 of 4  
Hello from rank 3 of 4  
Hello from rank 1 of 4  
Hello from rank 0 of 4
```



# Communication

- Communication between ranks via point-to-point and collective operations
- Point-to-point: send and receive messages between two ranks, e.g.

MPI\_Send

MPI\_Recv

- Collective: all ranks participate in the same operation, e.g.

MPI\_Barrier

MPI\_Reduce

MPI\_Bcast

# Example

---

Rank 1

Rank 3

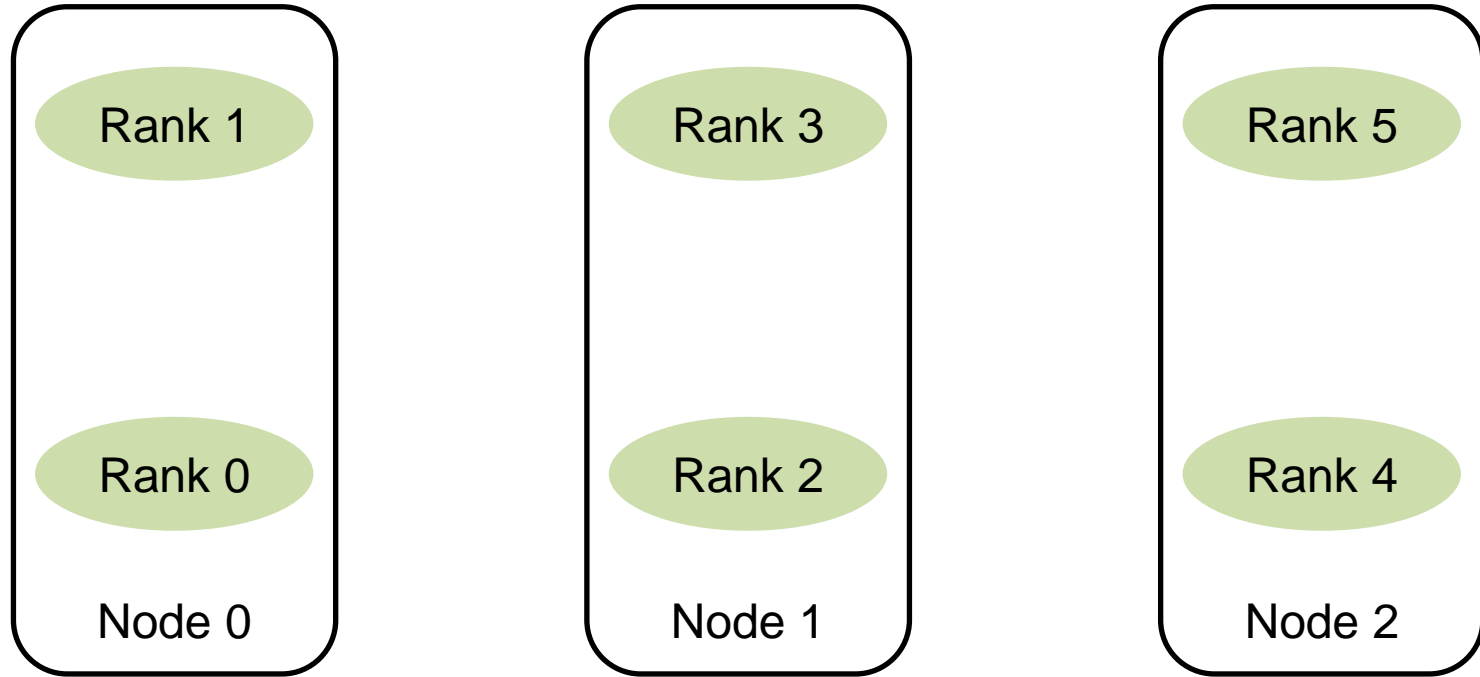
Rank 5

Rank 0

Rank 2

Rank 4

# Example Node Mapping



# Example with Values

Rank 1  
5

Rank 3  
0

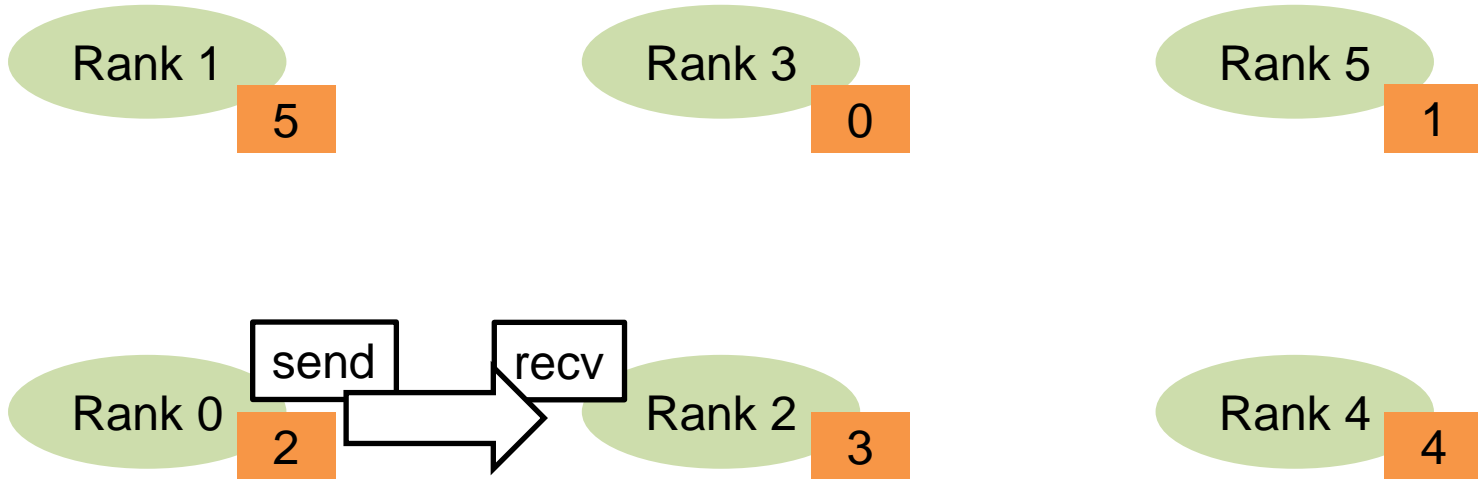
Rank 5  
1

Rank 0  
2

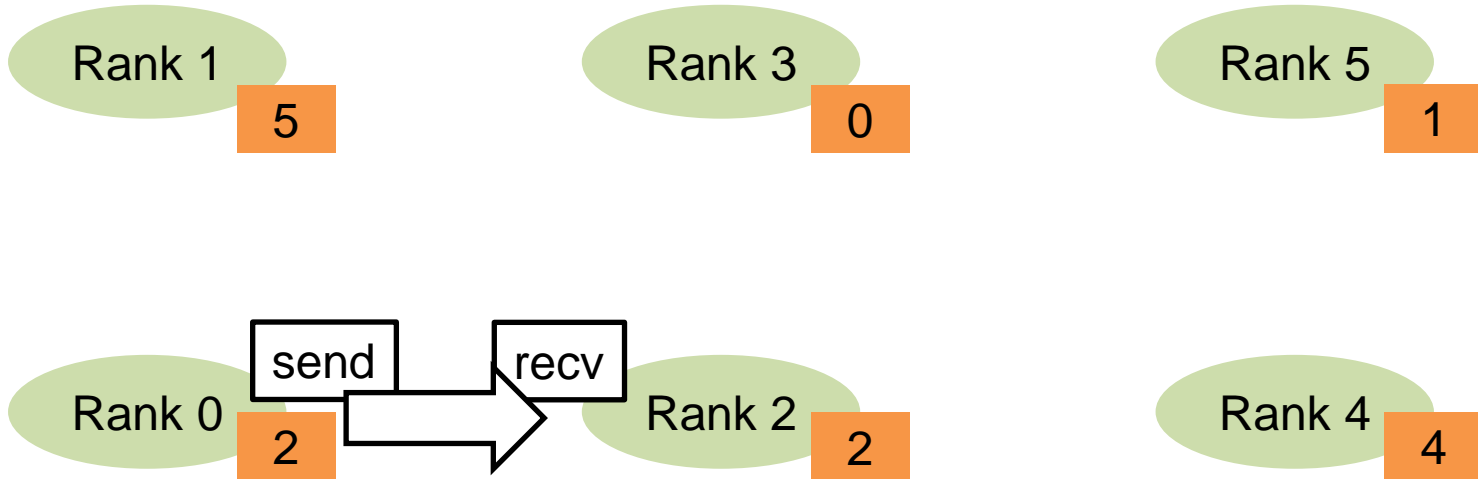
Rank 2  
3

Rank 4  
4

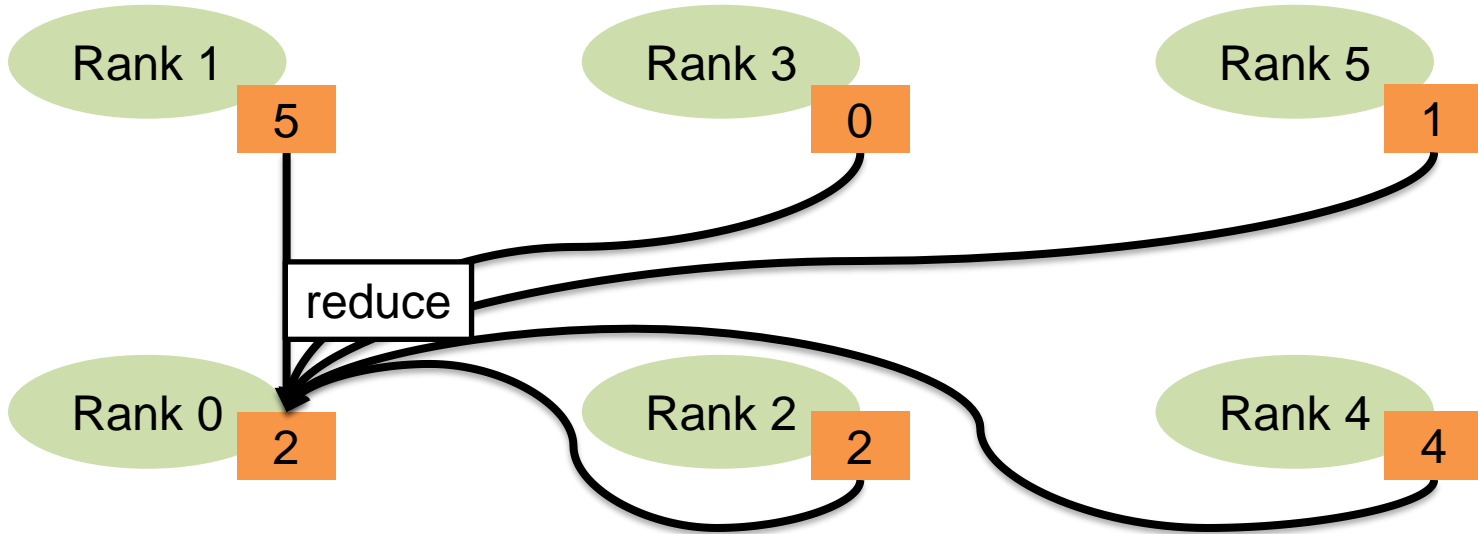
# MPI Send/Recv



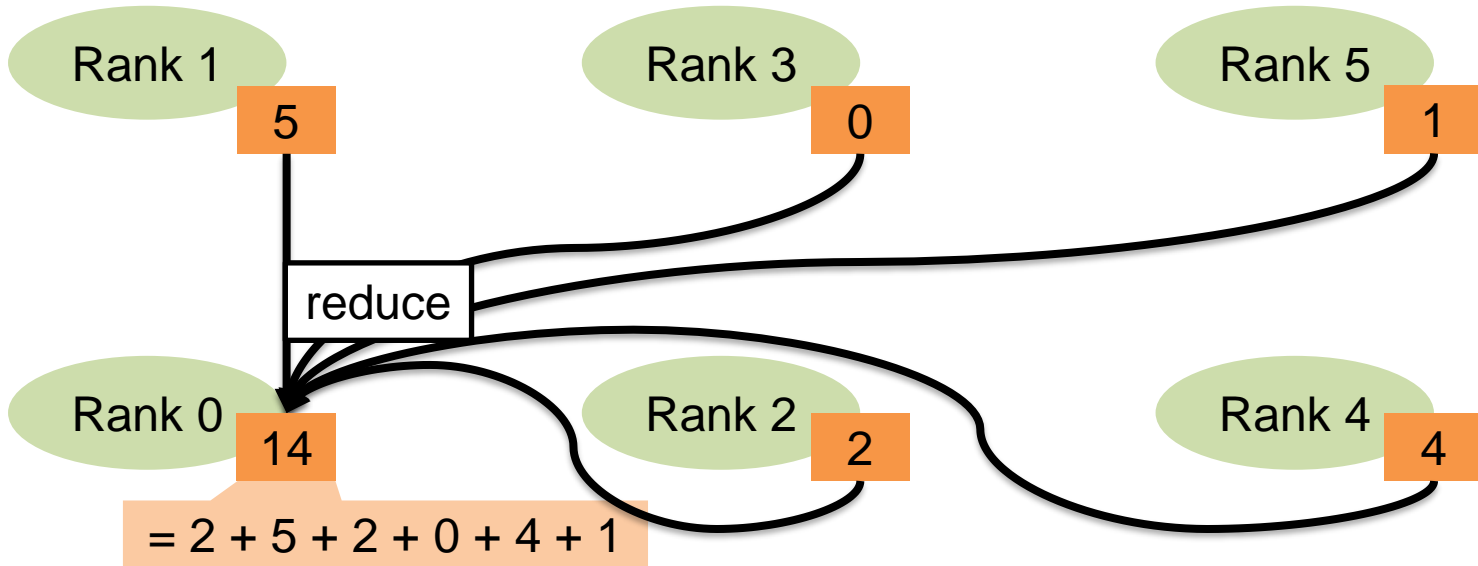
# MPI Send/Recv



# MPI Reduce

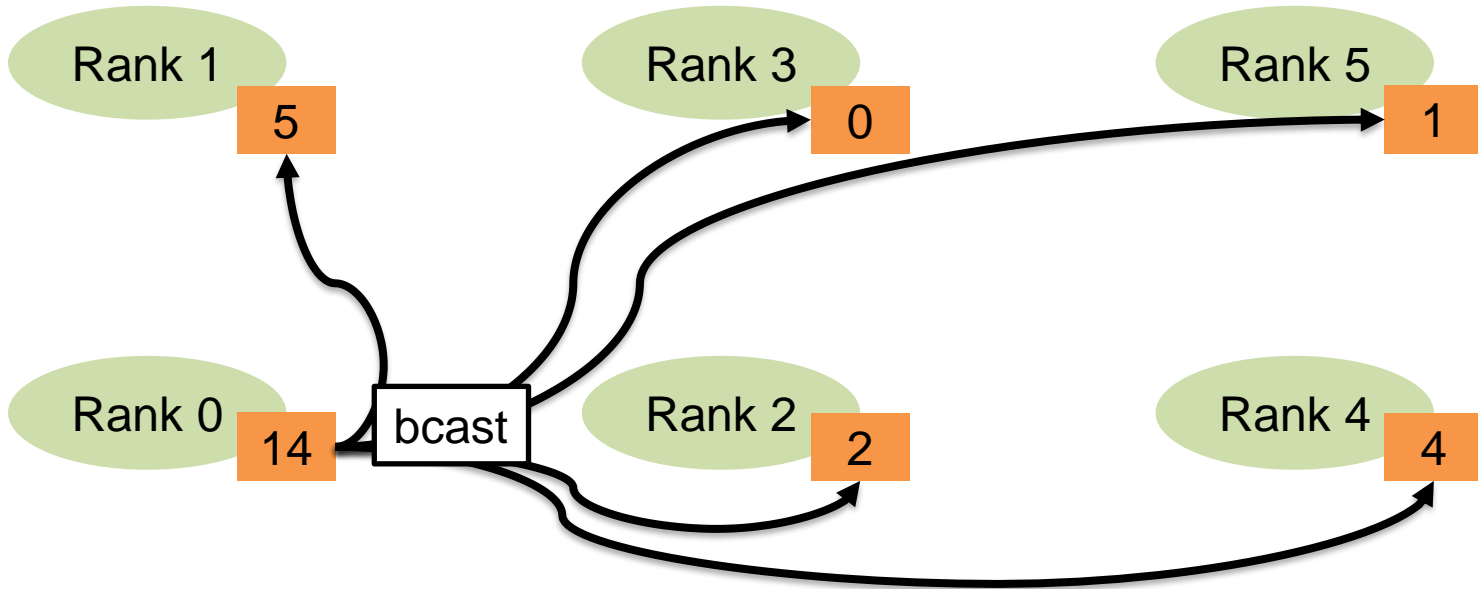


# MPI Reduce

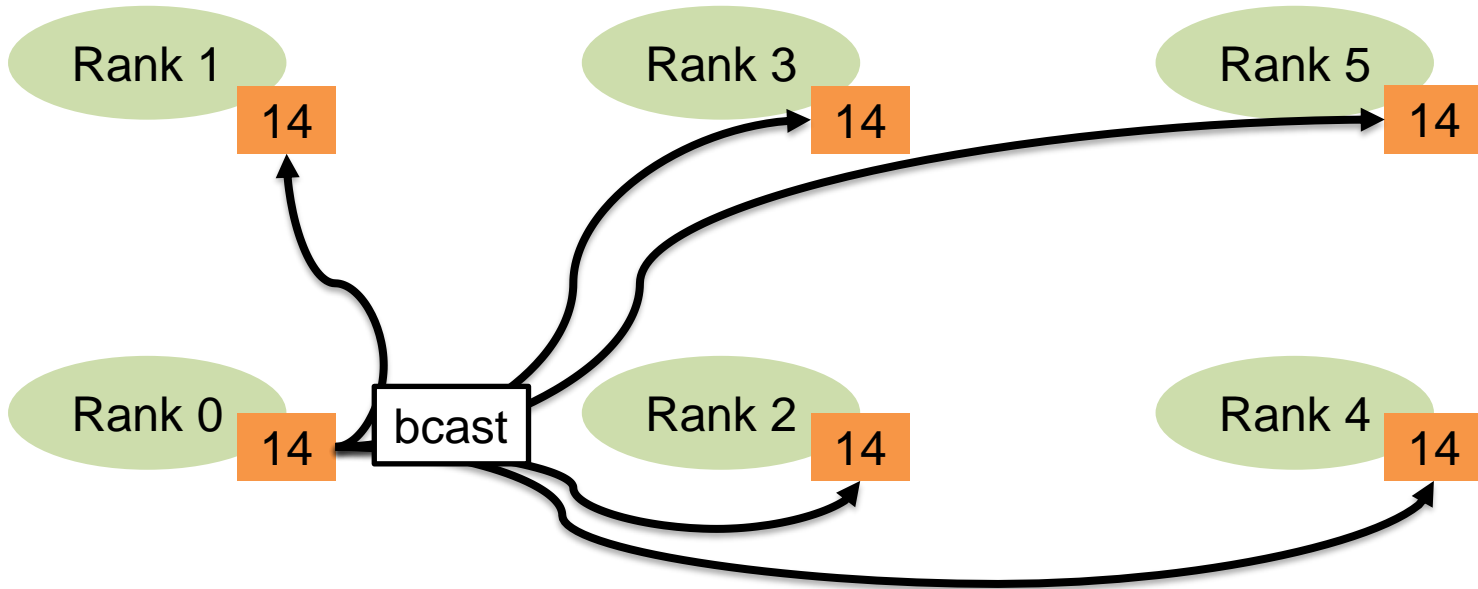




# MPI Broadcast



# MPI Broadcast



# Communication

- Communication between ranks via point-to-point and collective operations
- Point-to-point: send and receive messages between two ranks, e.g.

```
MPI_Send(&sendData, count, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);  
MPI_Recv(&recvData, count, MPI_DOUBLE, src, tag, MPI_COMM_WORLD, &mpiStatus);
```

- Collective: all ranks participate in the same operation, e.g.

```
MPI_Barrier(MPI_COMM_WORLD);  
MPI_Reduce(&sendData, &recvData, count, MPI_DOUBLE, MPI_SUM, root, MPI_COMM_WORLD);  
MPI_Bcast(&data, count, MPI_DOUBLE, root, MPI_COMM_WORLD);
```

# Beyond First Steps

- Other collective operations
  - MPI\_Gather, MPI\_Scatter, MPI\_Scan, ...
  - MPI\_Allreduce, MPI\_Allgather, ...
- Non-blocking communication
  - MPI\_Isend, MPI\_Irecv, MPI\_Ireduce, ...
  - MPI\_Test, MPI\_Wait, ...
- One-sided communication
  - MPI\_Get, MPI\_Put, ...
- Custom (sub-) communicators
  - MPI\_Cart\_create, MPI\_Dims\_create, ...
- MPI I/O, Error handling, timing/profiling, ...

CUDA-Aware MPI