UNIVERSITY OF
LUXEMBOURG

Introduction to Message Passing Interfaces

2nd Lesson

Dr. Matteo Barborini

HPC Platform – University of Luxembourg

Maison du Nombre
6, avenue de la Fonte L-4364 Esch-sur-Alzette Luxembourg

matteo.barborini@uni.lu

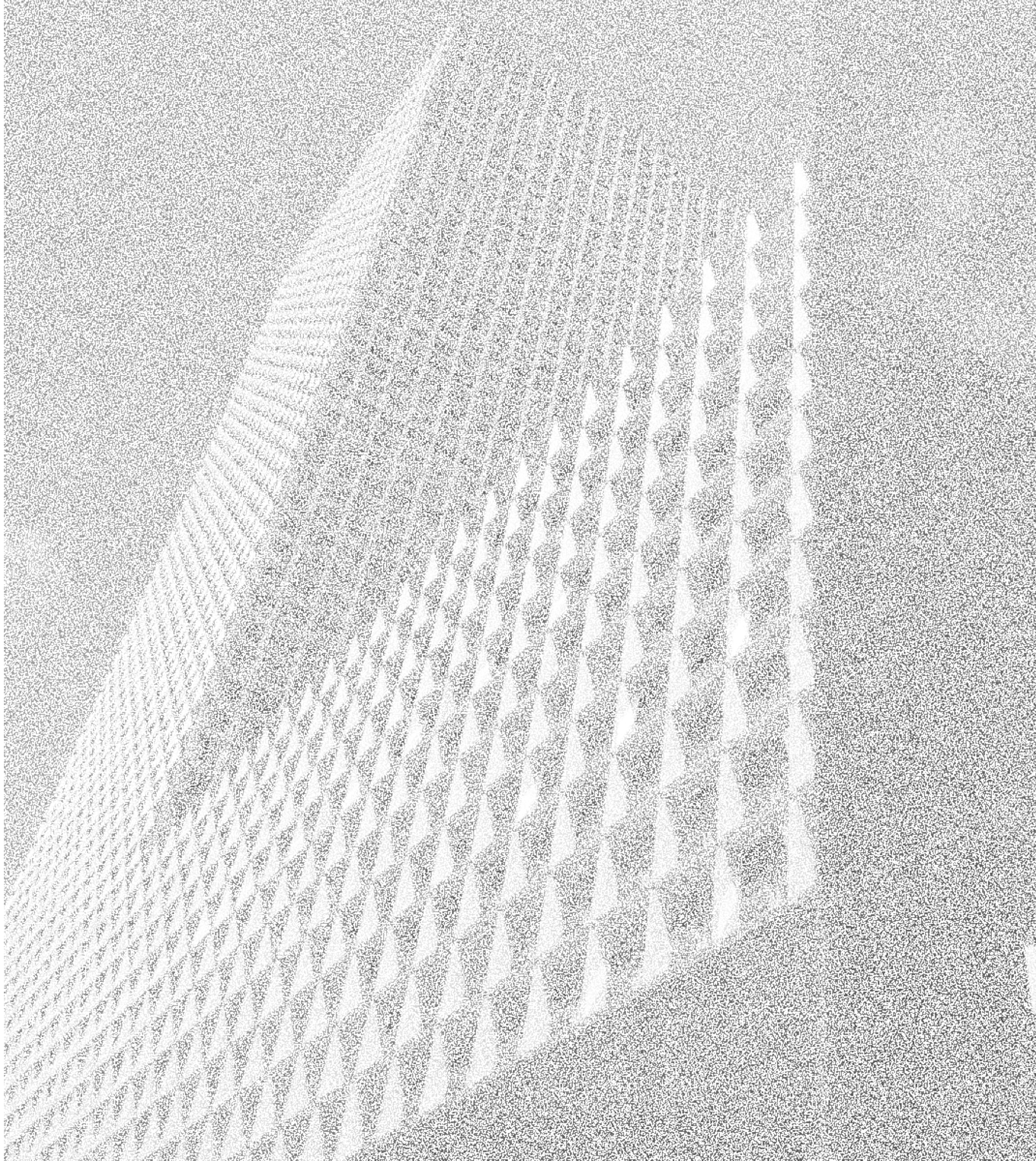


Table of Contents

1. MPI Functions (Second Part)

2. Scaling of codes

3. Matrix-Mastrix multiplication

4. First Exercise

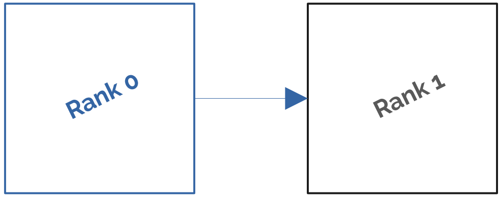
uni.lu
UNIVERSITY OF
LUXEMBOURG

MPI_Send/MPI_Recv

Introduction to MPI programming in C – 2025/2026
Lesson 2

Example: 3.1.MPI_send_recv_arrays.c

The commands MPI_Send/MPI_Recv are used for point-to-point communications between ranks



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

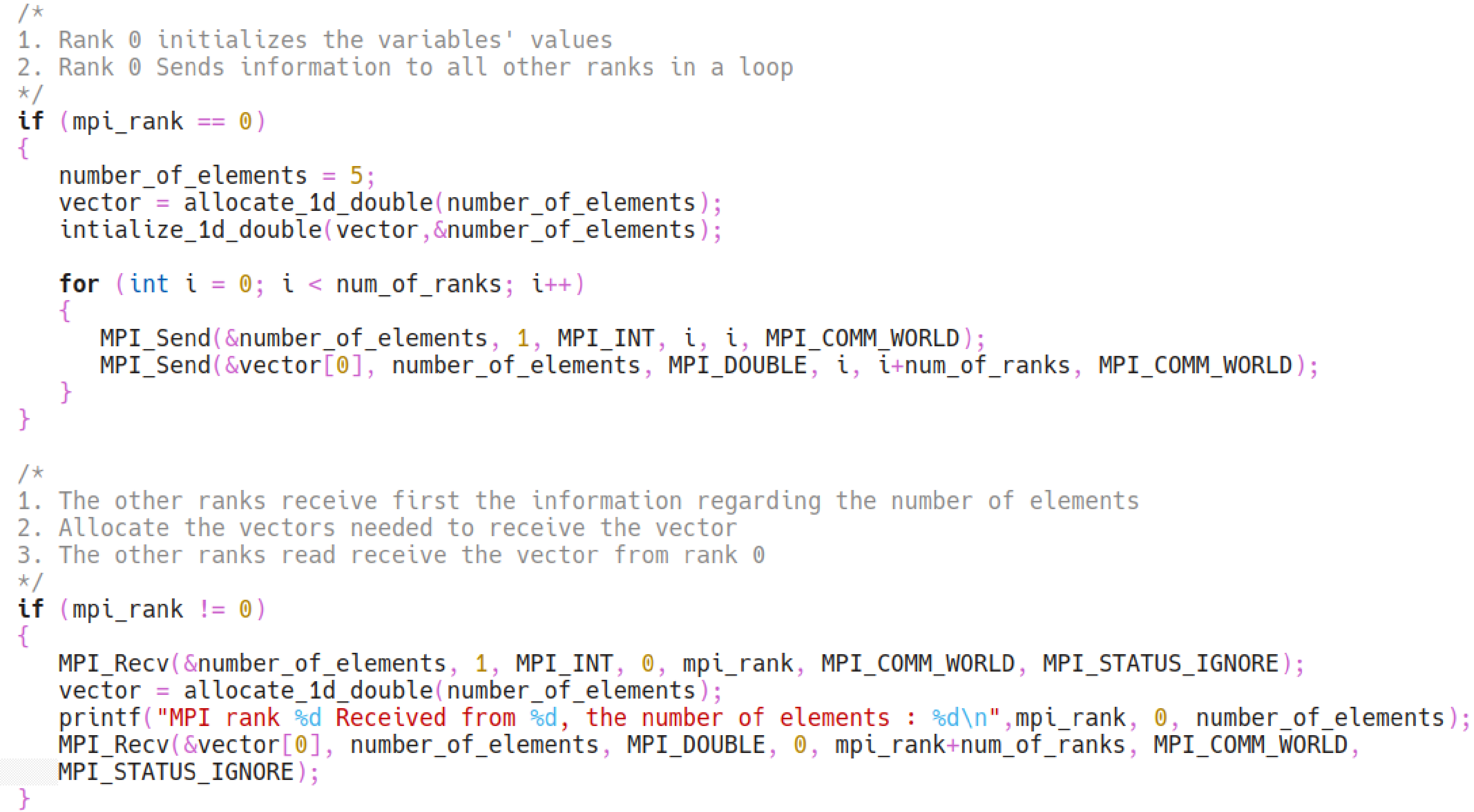
```
int MPI_Recv(void *buf,
             int count,
MPI_Datatype datatype,
           int source,
              int tag,
          MPI_Comm comm,
    MPI_Status *status)
```

The MPI task that calles the command MPI_send, sends the buffer to the dest, with a tag that is used to recognize the communication (Many communications can be executed between the same ranks).

The **task that executes the MPI_recv command awaits for the communcation** from the source with a given tag.

www.mpi-forum.org/docs/, www.open-mpi.org

Example: 3.1.MPI_send_recv_arrays.c



```
/*
1. Rank 0 initializes the variables' values
2. Rank 0 Sends information to all other ranks in a loop
*/
if (mpi_rank == 0)
{
    number_of_elements = 5;
    vector = allocate_1d_double(number_of_elements);
    intialize_1d_double(vector,&number_of_elements);

    for (int i = 0; i < num_of_ranks; i++)
    {
        MPI_Send(&number_of_elements, 1, MPI_INT, i, i, MPI_COMM_WORLD);
        MPI_Send(&vector[0], number_of_elements, MPI_DOUBLE, i, i+num_of_ranks, MPI_COMM_WORLD);
    }
}

/*
1. The other ranks receive first the information regarding the number of elements
2. Allocate the vectors needed to receive the vector
3. The other ranks read receive the vector from rank 0
*/
if (mpi_rank != 0)
{
    MPI_Recv(&number_of_elements, 1, MPI_INT, 0, mpi_rank, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    vector = allocate_1d_double(number_of_elements);
    printf("MPI rank %d Received from %d, the number of elements : %d\n",mpi_rank, 0, number_of_elements);
    MPI_Recv(&vector[0], number_of_elements, MPI_DOUBLE, 0, mpi_rank+num_of_ranks, MPI_COMM_WORLD,
    MPI_STATUS_IGNORE);
}
```

www.mpi-forum.org/docs/, www.open-mpi.org

UNIVERSITY OF
LUXEMBOURG

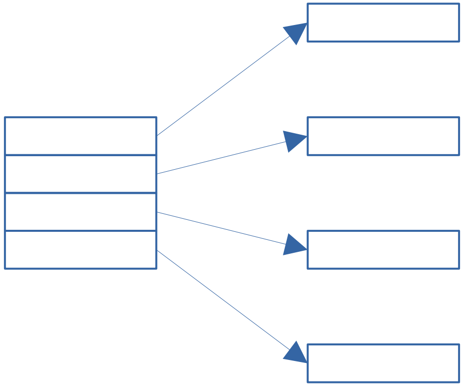
MPI_Scatter/MPI_Gather

Introduction to MPI programming in C – 2025/2026
Lesson 2

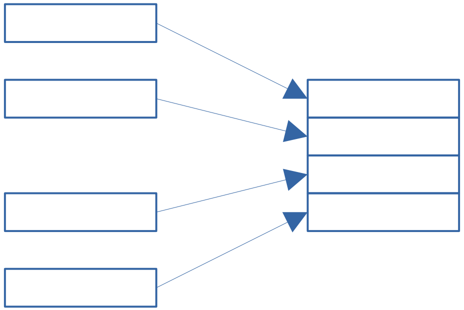
Example: 4.1.MPI_scatter_gather_arrays.c

The command MPI_Scatter is used to partition the sending buffer (usually array or matrix) on all ranks of the MPI_COMM_WORLD.

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



The command MPI_Gather is used to collect the sending buffers to a recivieving one.



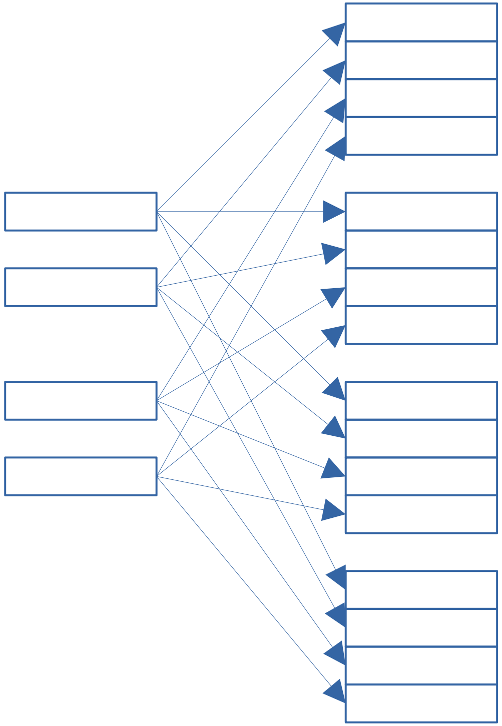
```
int MPI_Gather(const void *sendbuf,
                    int sendcount,
              MPI_Datatype sendtype,
                    void *recvbuf,
                    int recvcount,
              MPI_Datatype recvtype,
                         int root,
                    MPI_Comm comm)
```

www.mpi-forum.org/docs/, www.open-mpi.org

Example: 4.1.MPI_scatter_gather_arrays.c

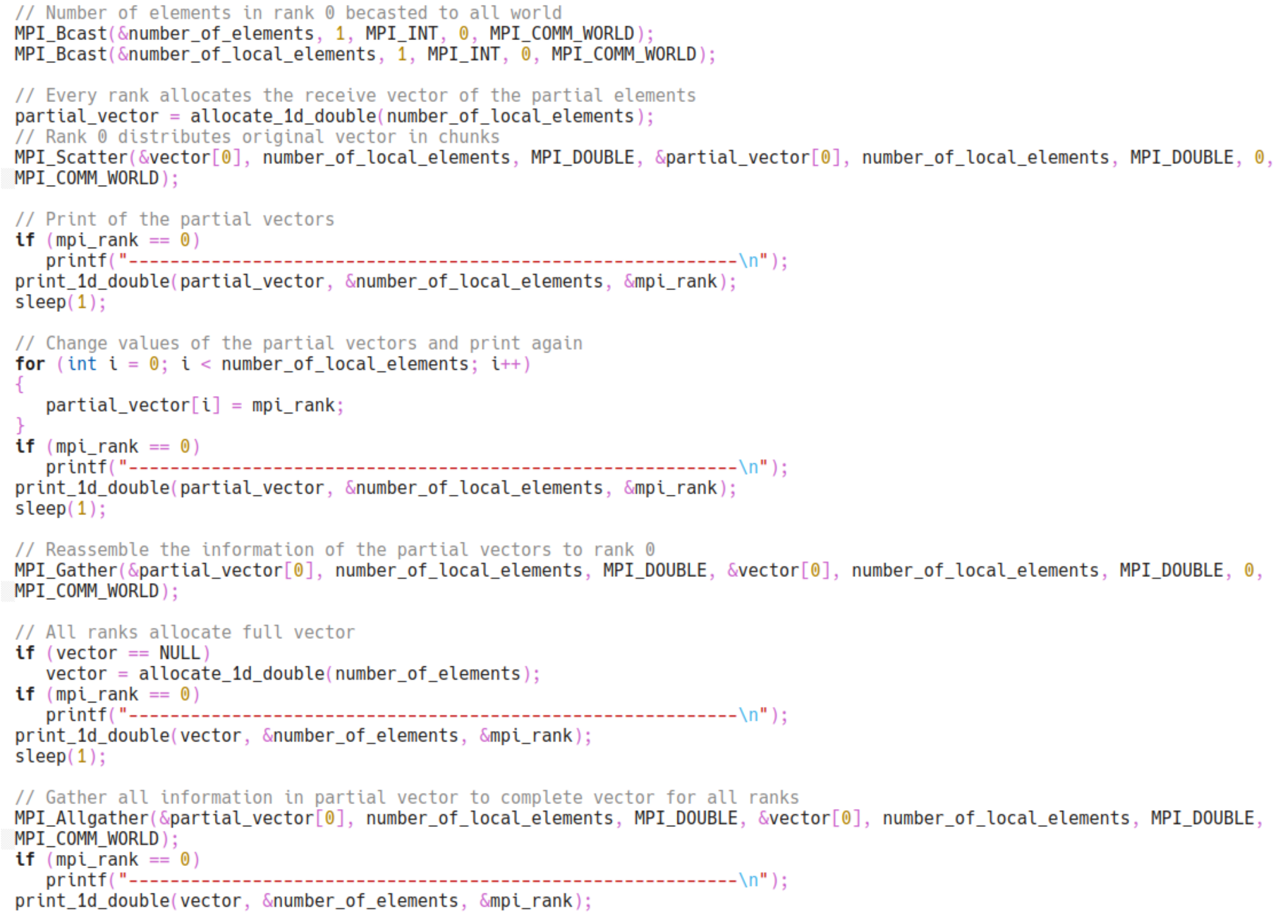
The command MPI_Allgather is used to collect the sending buffers for all ranks.

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



www.mpi-forum.org/docs/, www.open-mpi.org

Example: 4.1.MPI_scatter_gather_arrays.c



```c
// Number of elements in rank 0 becasted to all world
MPI_Bcast(&number_of_elements, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Bcast(&number_of_local_elements, 1, MPI_INT, 0, MPI_COMM_WORLD);

// Every rank allocates the receive vector of the partial elements
partial_vector = allocate_1d_double(number_of_local_elements);
// Rank 0 distributes original vector in chunks
MPI_Scatter(&vector[0], number_of_local_elements, MPI_DOUBLE, &partial_vector[0], number_of_local_elements, MPI_DOUBLE, 0,
MPI_COMM_WORLD);

// Print of the partial vectors
if (mpi_rank == 0)
   printf("-----------------------------------------------------------\n");
print_1d_double(partial_vector, &number_of_local_elements, &mpi_rank);
sleep(1);

// Change values of the partial vectors and print again
for (int i = 0; i < number_of_local_elements; i++)
{
   partial_vector[i] = mpi_rank;
}
if (mpi_rank == 0)
   printf("-----------------------------------------------------------\n");
print_1d_double(partial_vector, &number_of_local_elements, &mpi_rank);
sleep(1);

// Reassemble the information of the partial vectors to rank 0
MPI_Gather(&partial_vector[0], number_of_local_elements, MPI_DOUBLE, &vector[0], number_of_local_elements, MPI_DOUBLE, 0,
MPI_COMM_WORLD);

// All ranks allocate full vector
if (vector == NULL)
   vector = allocate_1d_double(number_of_elements);
if (mpi_rank == 0)
   printf("-----------------------------------------------------------\n");
print_1d_double(vector, &number_of_elements, &mpi_rank);
sleep(1);

// Gather all information in partial vector to complete vector for all ranks
MPI_Allgather(&partial_vector[0], number_of_local_elements, MPI_DOUBLE, &vector[0], number_of_local_elements, MPI_DOUBLE,
MPI_COMM_WORLD);
if (mpi_rank == 0)
   printf("-----------------------------------------------------------\n");
print_1d_double(vector, &number_of_elements, &mpi_rank);
```

www.mpi-forum.org/docs/, www.open-mpi.org

UNIVERSITY OF
LUXEMBOURG

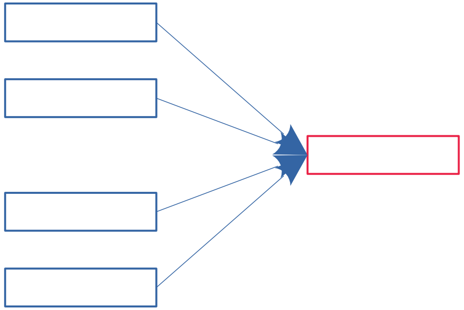
MPI_Reduce /MPI_Allreduce

Introduction to MPI programming in C – 2025/2026
Lesson 2

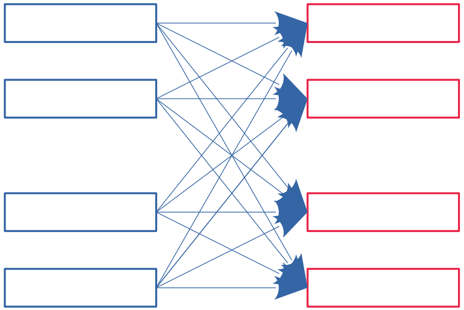
Example: 5.1.MPI_reduce_allreduce.c

The commands MPI_Reduce and MPI_Allreduce are used to perform the operation `MPI_Op op` executed over all MPI ranks, on the data from the sendbuf and collected on the recvbuf.

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



While in the case of MPI_Reduce the result is only made available to the rank 0, or root, in the case of `MPI_Allreduce` all MPI ranks receive the information:



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

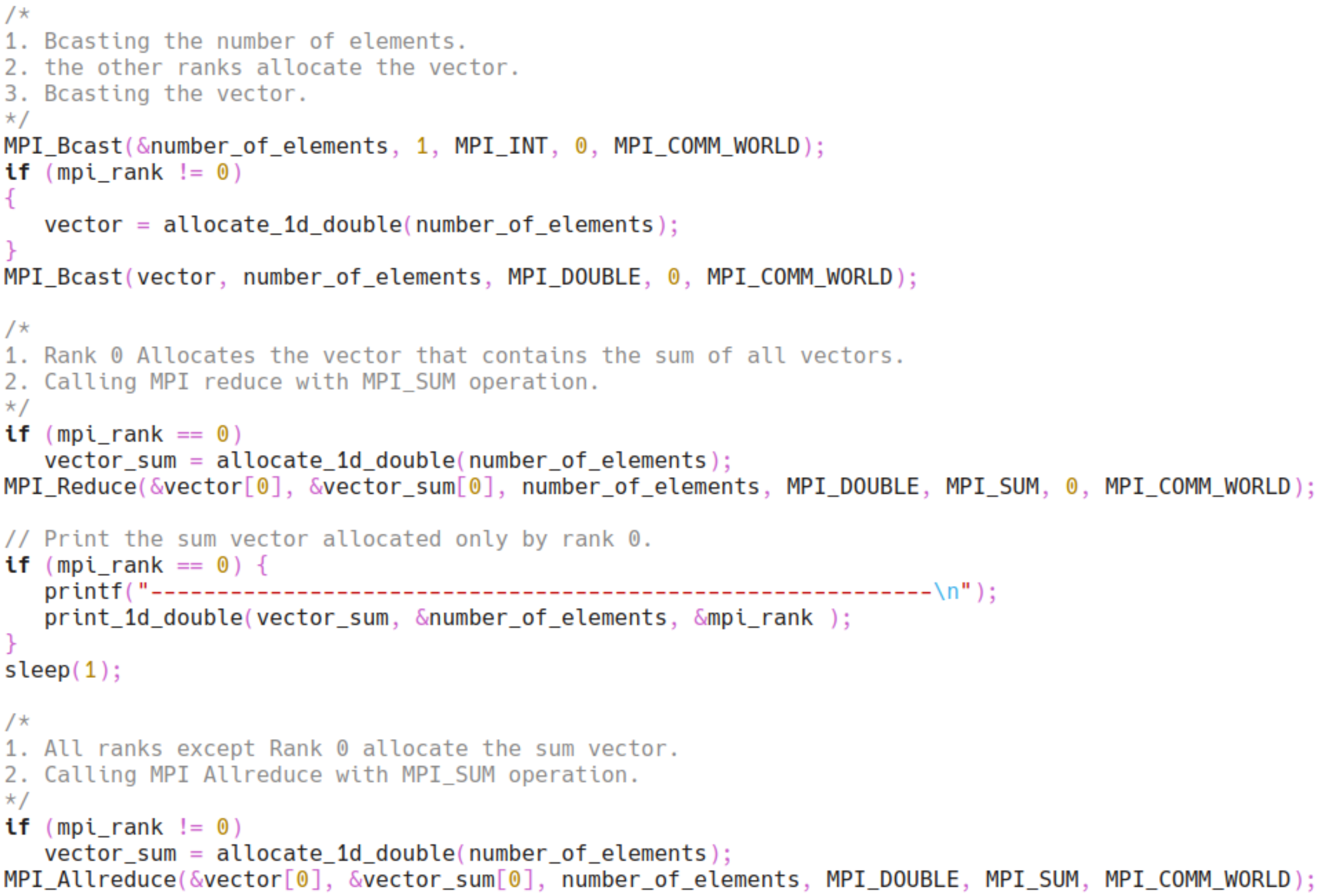
www.mpi-forum.org/docs/, www.open-mpi.org

MPI Operations

MPI Operations	Explanation
MPI_OP_NULL	dummy operation
MPI_MAX	Find the maximum value within ranks
MPI_MIN	Find the minimum value within ranks
MPI_SUM	Sum variables along all ranks
MPI_PROD	Mulitply variables along all ranks
MPI_LAND	Logical and
MPI_BAND	Bit-wise and
MPI_LOR	Logical or
MPI_BOR	Bit-wise or
MPI_LXOR	Logical xor (exclusive OR)
MPI_BXOR	Bit-wise xor
MPI_MINLOC	Computes min value and its location
MPI_MAXLOC	Computes max value and its location

www.mpi-forum.org/docs

Example: 5.1.MPI_reduce_allreduce.c



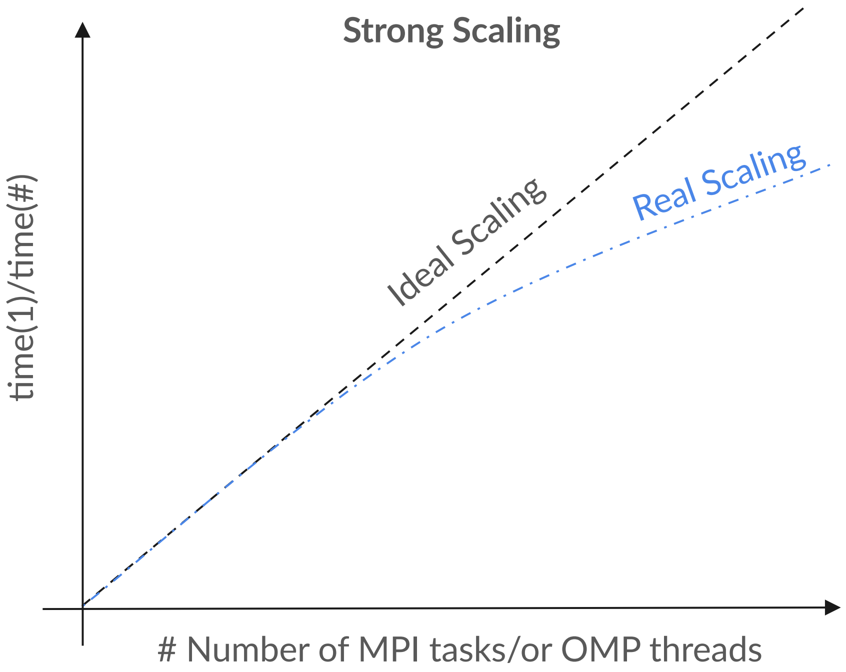
```c
/*
1. Bcasting the number of elements.
2. the other ranks allocate the vector.
3. Bcasting the vector.
*/
MPI_Bcast(&number_of_elements, 1, MPI_INT, 0, MPI_COMM_WORLD);
if (mpi_rank != 0)
{
   vector = allocate_1d_double(number_of_elements);
}
MPI_Bcast(vector, number_of_elements, MPI_DOUBLE, 0, MPI_COMM_WORLD);

/*
1. Rank 0 Allocates the vector that contains the sum of all vectors.
2. Calling MPI reduce with MPI_SUM operation.
*/
if (mpi_rank == 0)
   vector_sum = allocate_1d_double(number_of_elements);
MPI_Reduce(&vector[0], &vector_sum[0], number_of_elements, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

// Print the sum vector allocated only by rank 0.
if (mpi_rank == 0) {
   printf("-----------------------------------------------------------\n");
   print_1d_double(vector_sum, &number_of_elements, &mpi_rank );
}
sleep(1);

/*
1. All ranks except Rank 0 allocate the sum vector.
2. Calling MPI Allreduce with MPI_SUM operation.
*/
if (mpi_rank != 0)
   vector_sum = allocate_1d_double(number_of_elements);
MPI_Allreduce(&vector[0], &vector_sum[0], number_of_elements, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
```

www.mpi-forum.org/docs/, www.open-mpi.org

Performances

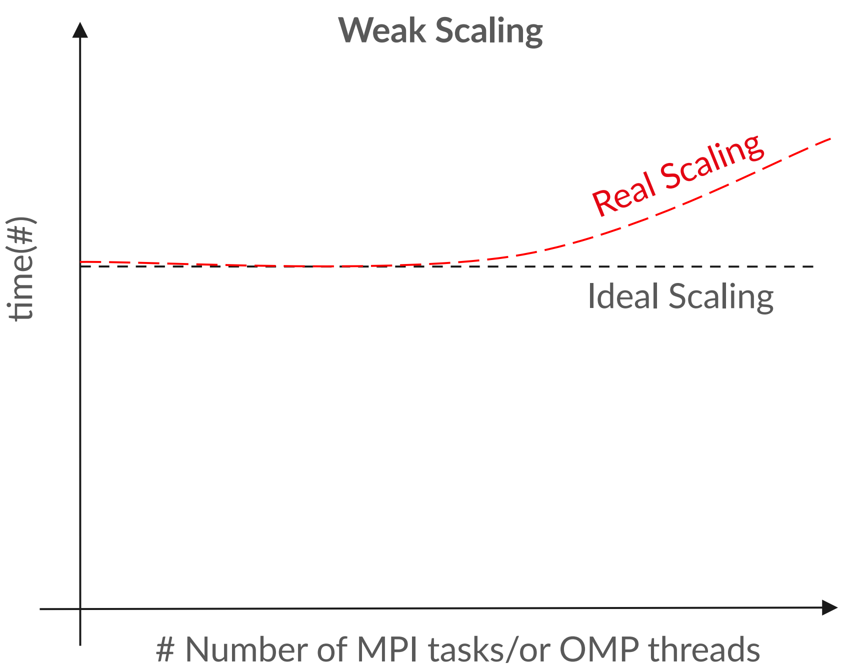
Introduction to MPI programming in C – 2025/2026
Lesson 2

Basic scaling laws



Strong Scaling

The total task is partitioned over the cores



Weak Scaling

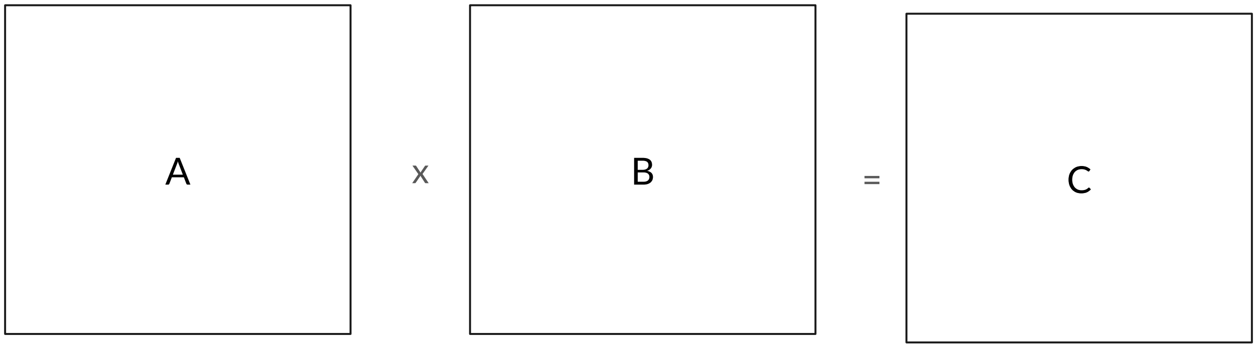
Each core does the same task

UNIVERSITY OF
LUXEMBOURG

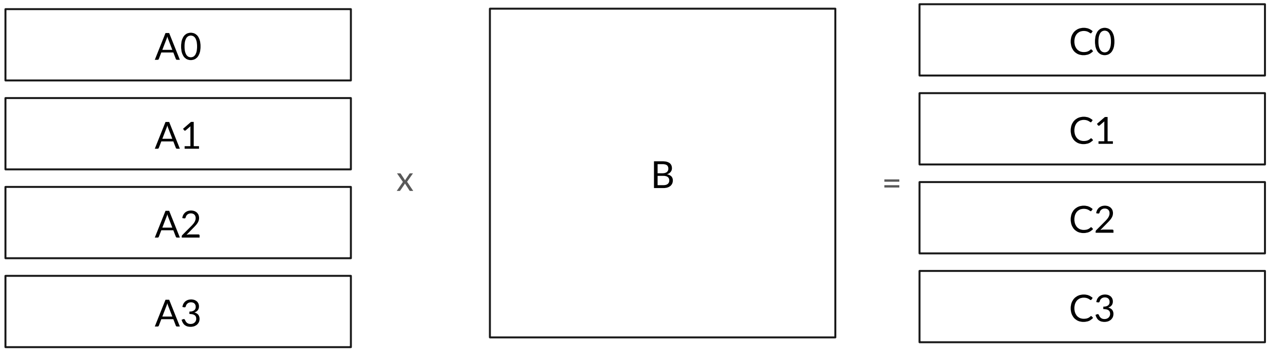
Matrix-Matrix multiplication

Introduction to MPI programming in C – 2025/2026
Lesson 2

Matrix-Matrix multiplication

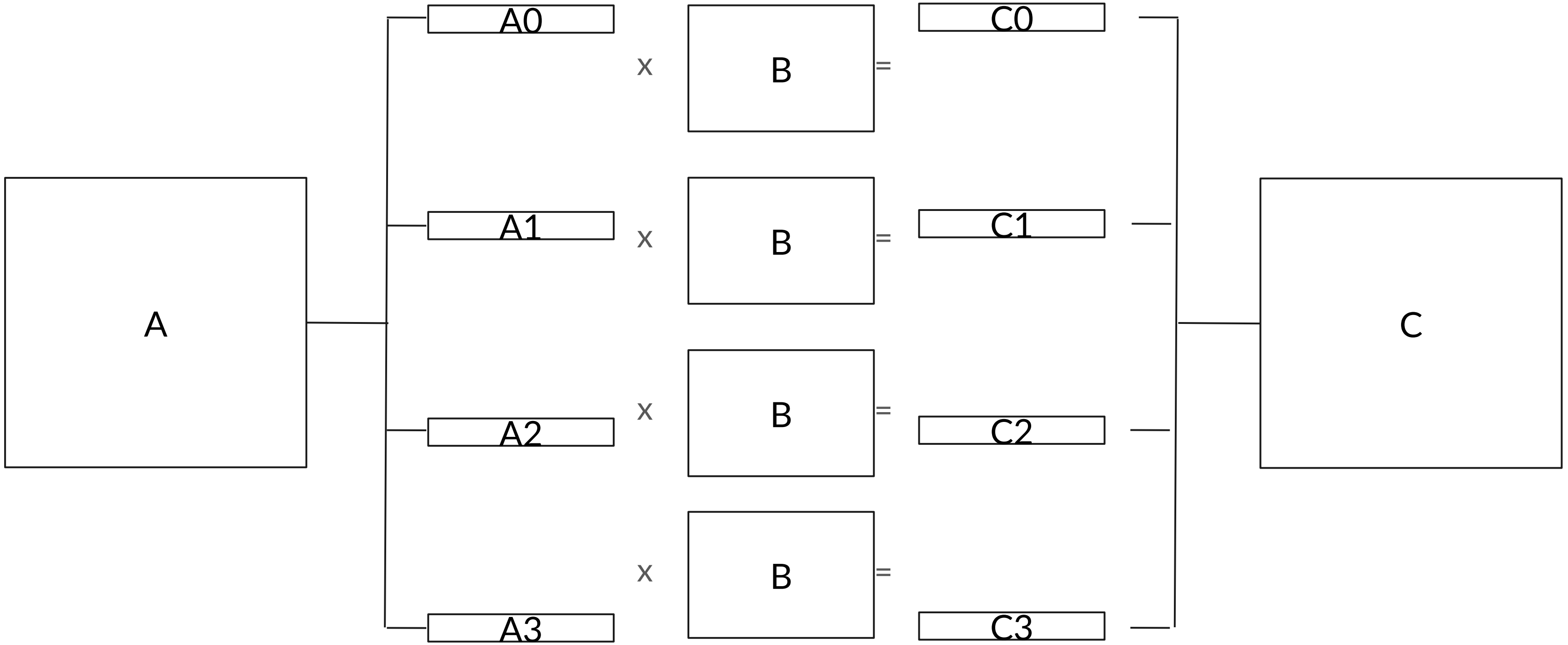


We want to split the matrix multiplication on 4 ranks:



Matrix-Matrix multiplication

A is split on 4 ranks, B is common to all ranks. The various MPI rands at the end communicate their result to rank 0.



Clearly you can split also the matrix B!!!.