MPI (II)

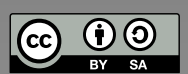
Distributed-memory programming

Juan Ángel Lorenzo del Castillo

CY Cergy Paris Université

ING2-GSI-MI Architecture et Programmation Parallèle

2023 - 2024



juan-angel.lorenzo-del-castillo@cyu.fr

Table of Contents

Table of Contents

Collective communication

- **Collective (or global) communication**: all processes from a communicator participate.
  - ▷ They could be implemented using point-to-point communications

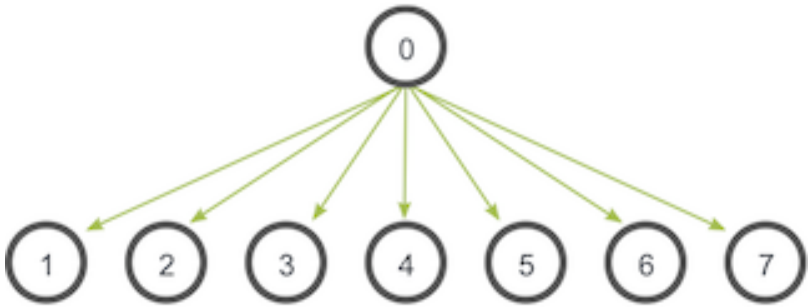
  E.g: A radiation (broadcast, data sent from one process to the rest) could be implemented with a `MPI_Send` on a loop and a `MPI_Recv` on each receiver.

Collective communication in MPI

- Collective communication calls:
  - Are blocking functions (with some exceptions in MPI-3)
  - All processes in the communicator must execute the function
  - The reception buffer must have the exact size
- Types:
  - Data movement
  - Group computation
  - Synchronisation

Radiation Data movement

- Data are sent from a root process to the rest



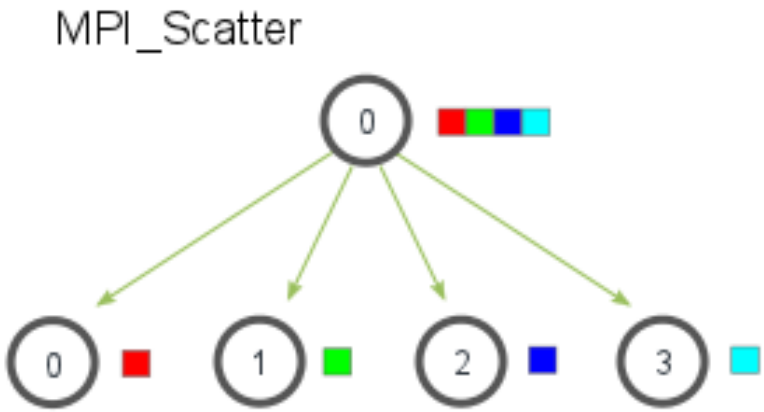
Source : mpitutorial.com

Radiation

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

Scatter
Data movement

- Data distribution from a process to the rest



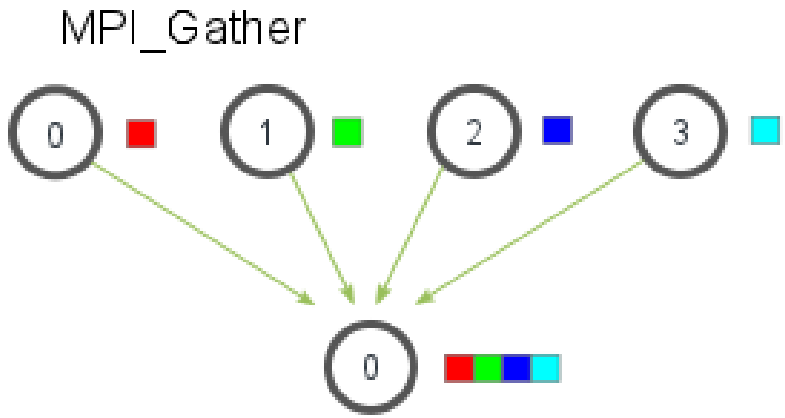
Source : mpitutorial.com

Scatter

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

Gather
Data movement

- Collect data from every process to a single one of them



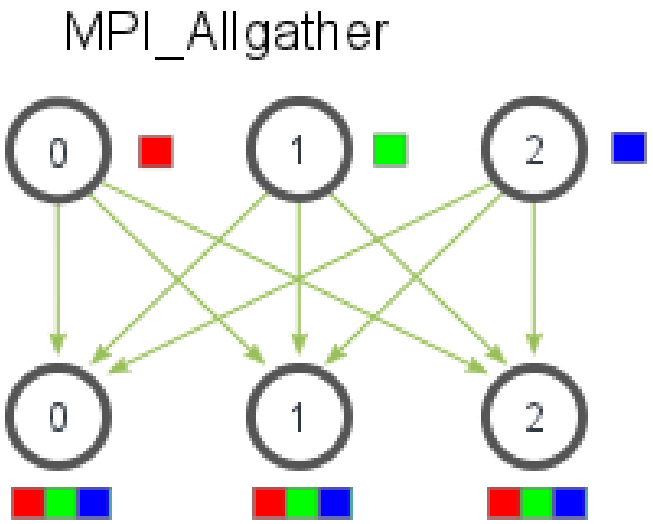
Source : mpitutorial.com

Gather

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

Allgather Data movement

- **Gather** from all to all
- In the end, all data will be available for every process



Source : mpitutorial.com

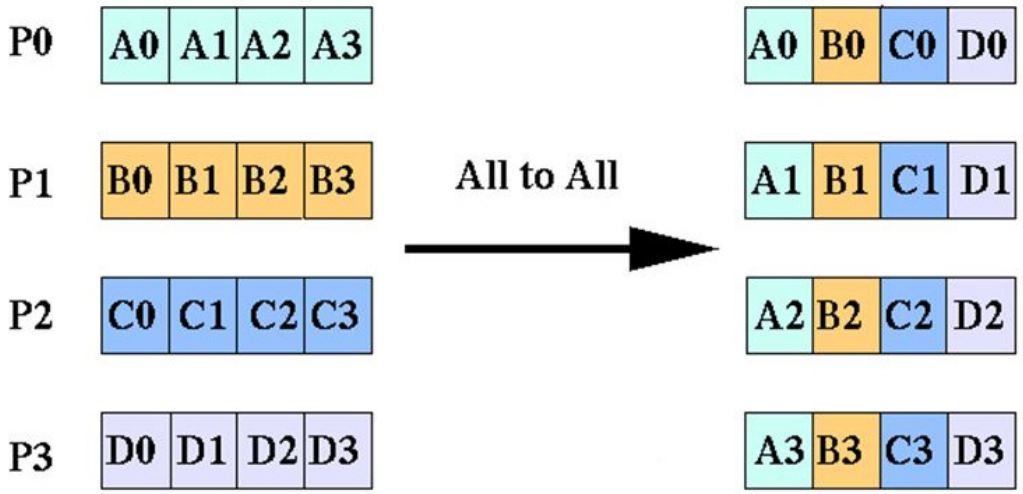
Allgather

```
MPI_Allgather(void *sendbuf, int sendcnt, MPI_Datatype sendtype,
              void *recvbuf, int recvcnt, MPI_Datatype recvtype,
              MPI_Comm comm)
```

Alltoall

Data movement

- Data are sent from all to all
- Combines multiple *scatters*
- Essentially, a matrix transposition



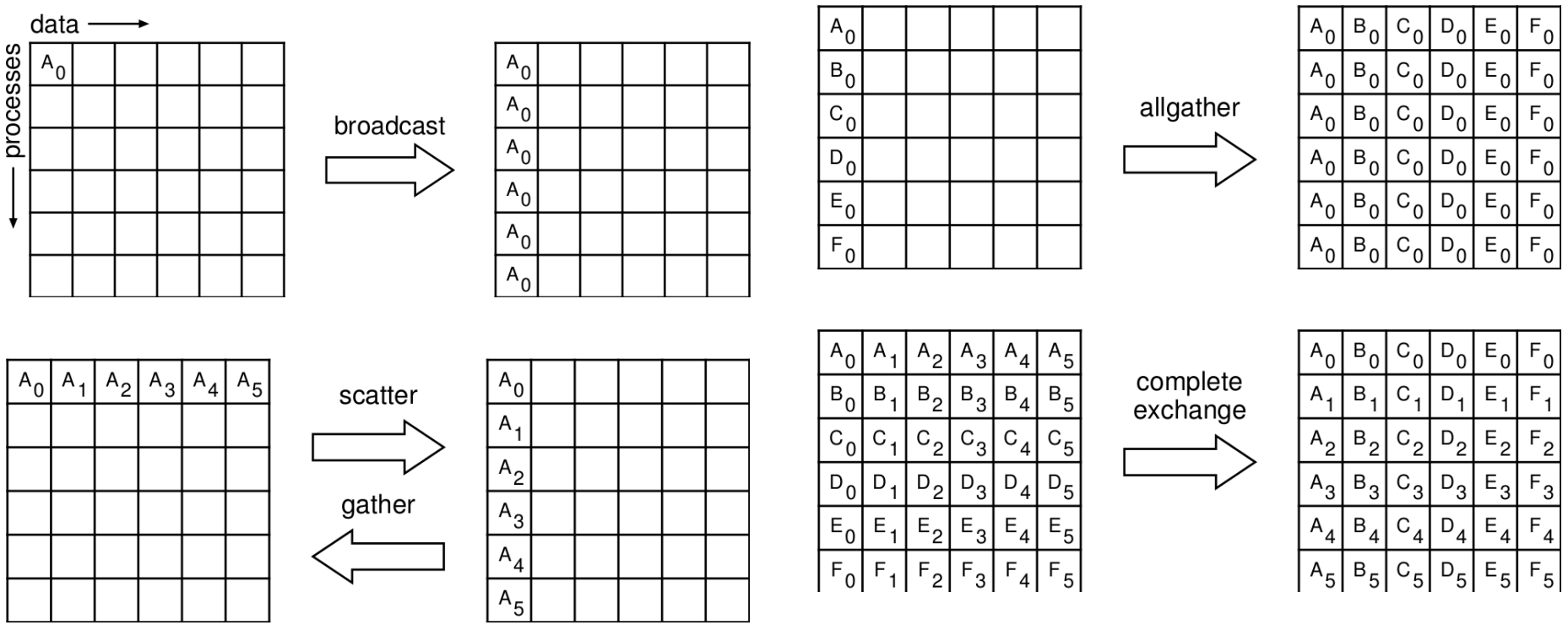
Source : UNC-Charlotte

Alltoall

```
int MPI_Alltoall(void *sbuf, int sendcnt, MPI_Datatype sendtype,
                 void *rbuf, int recvcnt, MPI_Datatype recvtype,
                 MPI_Comm comm);
```

Summary Data movement

Examples



Source : *MPI: A Message-Passing Interface Standard (v. 2.2)*
Message Passing Interface Forum
September 4, 2009

«v» variants Data movement

- A different number of elements is sent to each process

```
MPI_Scatterv
MPI_Gatherv
MPI_Allgatherv
MPI_Alltoallv
```

  - The number of elements sent to each process is defined in **vectors** of P elements
    - P: number of processes associated to the communicator
  - The number of element received from each process is defined in **vectors** of P elements
  - Example:

```
int MPI_Alltoallv(void *sendbuf,
                  int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf,
                  int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype,
                  MPI_Comm comm);
```

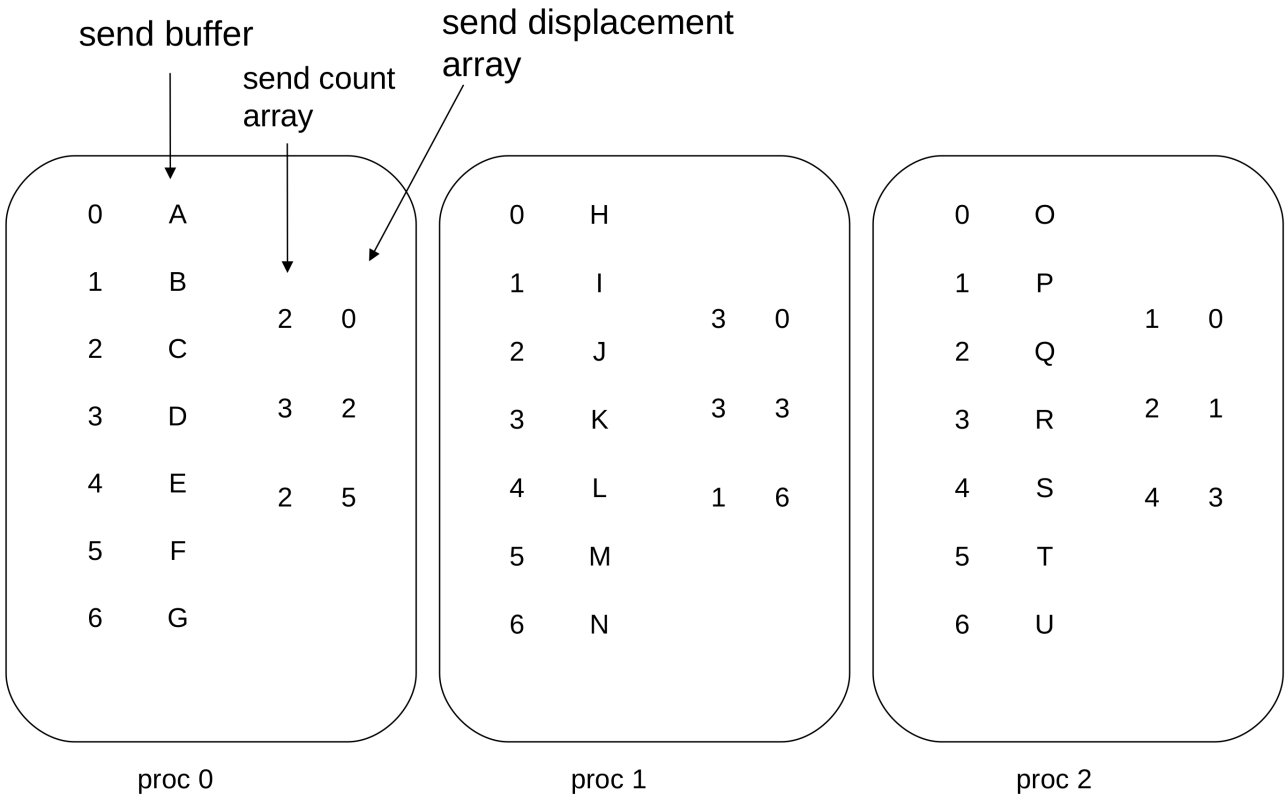
«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

- Input Parameters:
  - **sendbuf:** starting address of send buffer
  - **sendcounts:** integer array equal to the group size specifying the number of elements to send to each processor
  - **sdispls:** integer array (of length group size). Entry *j* specifies the displacement (relative to *sendbuf*) from which to take the outgoing data destined for process *j*
  - **recvcounts:** integer array equal to the group size specifying the maximum number of elements that can be received from each processor
  - **rdispls:** integer array (of length group size). Entry *i* specifies the displacement (relative to *recvbuf*) at which to place the incoming data from process *i*

«v» variations Data movement

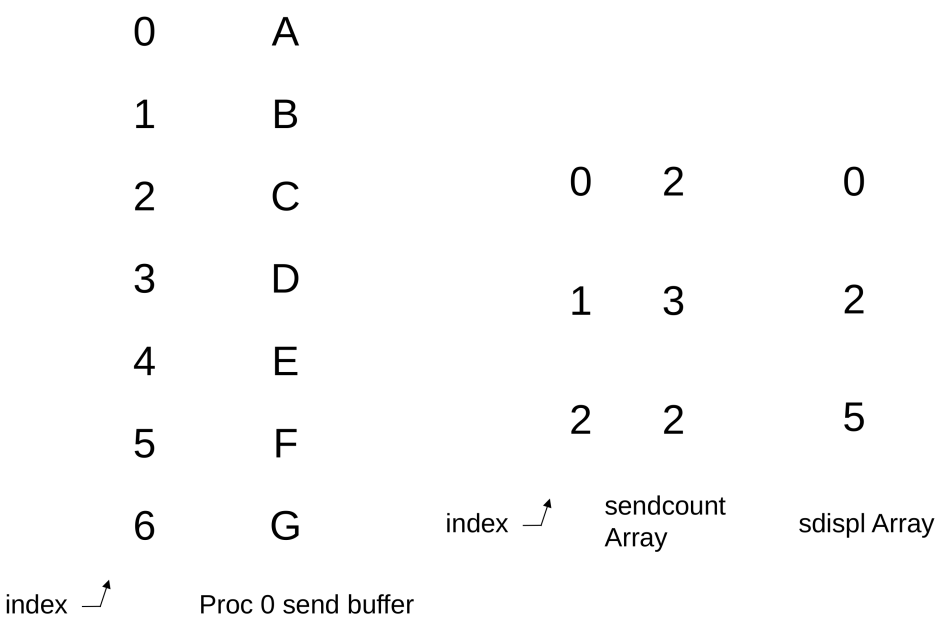
```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

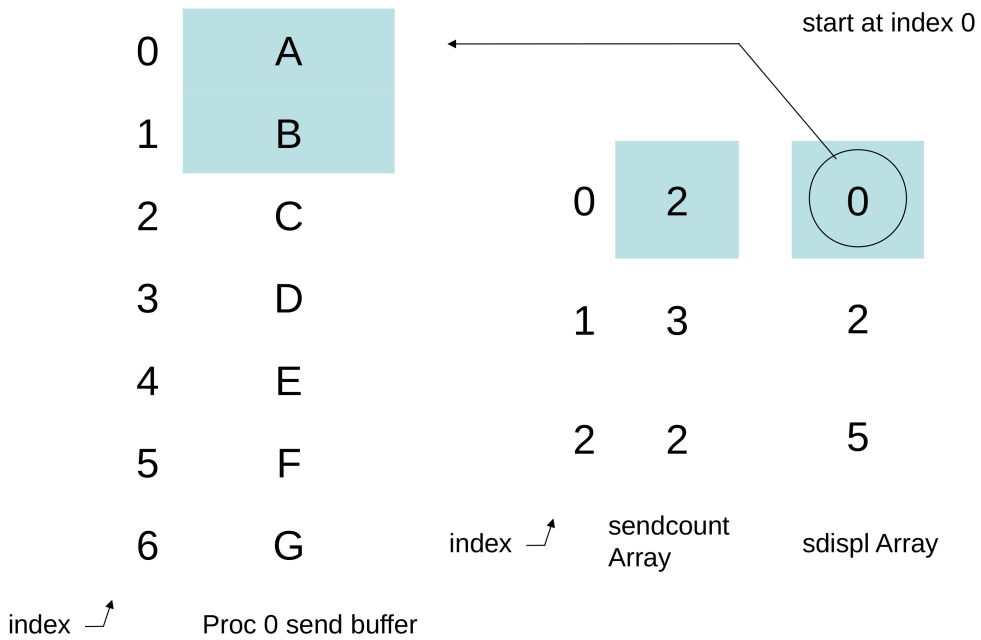
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

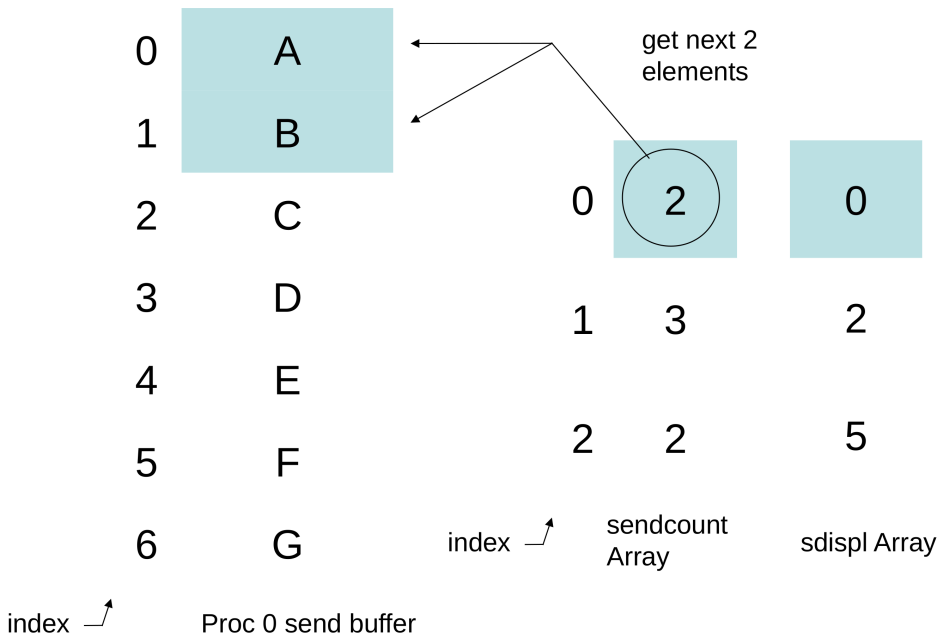
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

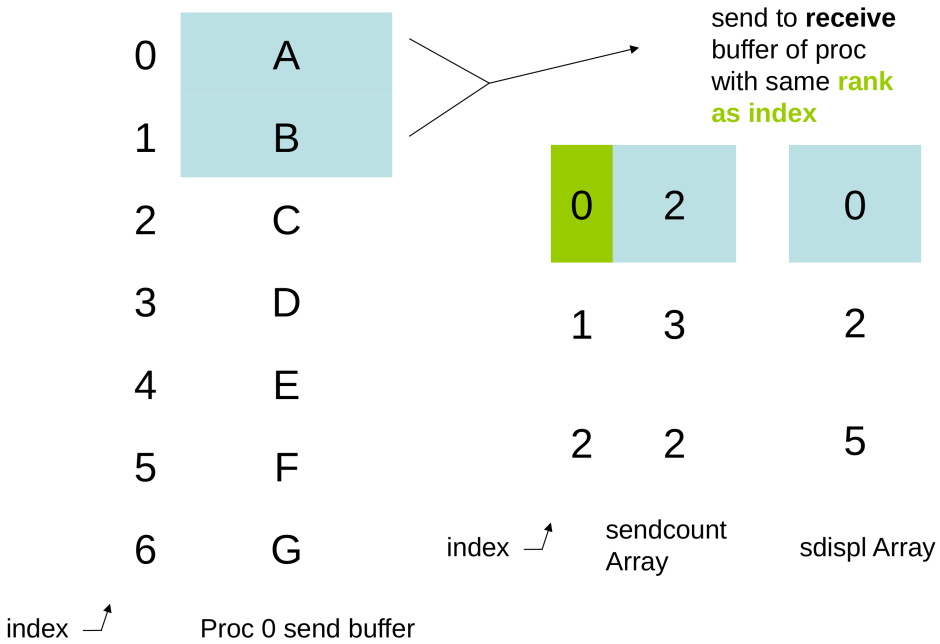
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

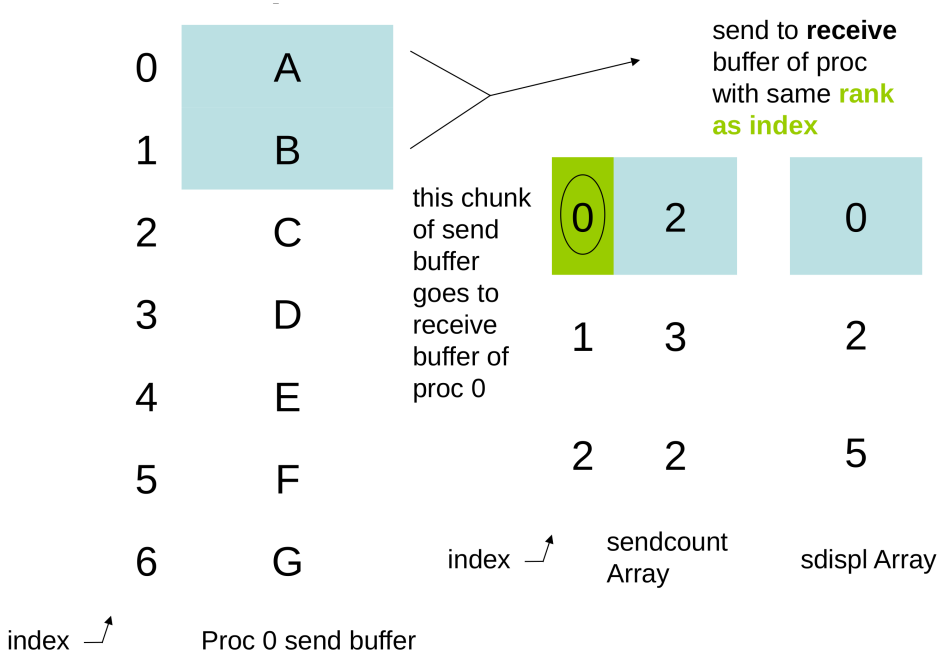
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

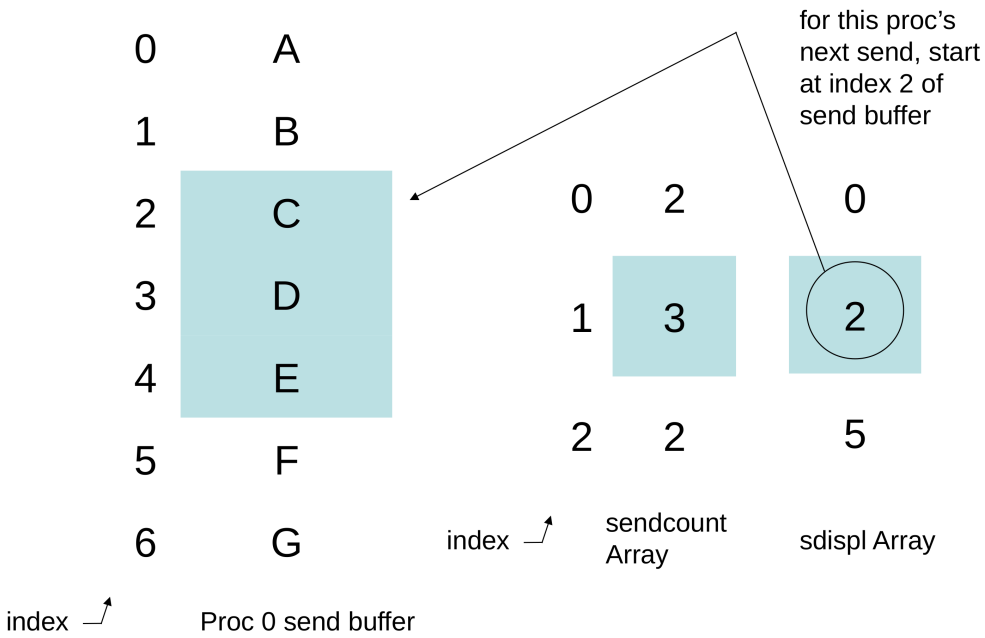
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

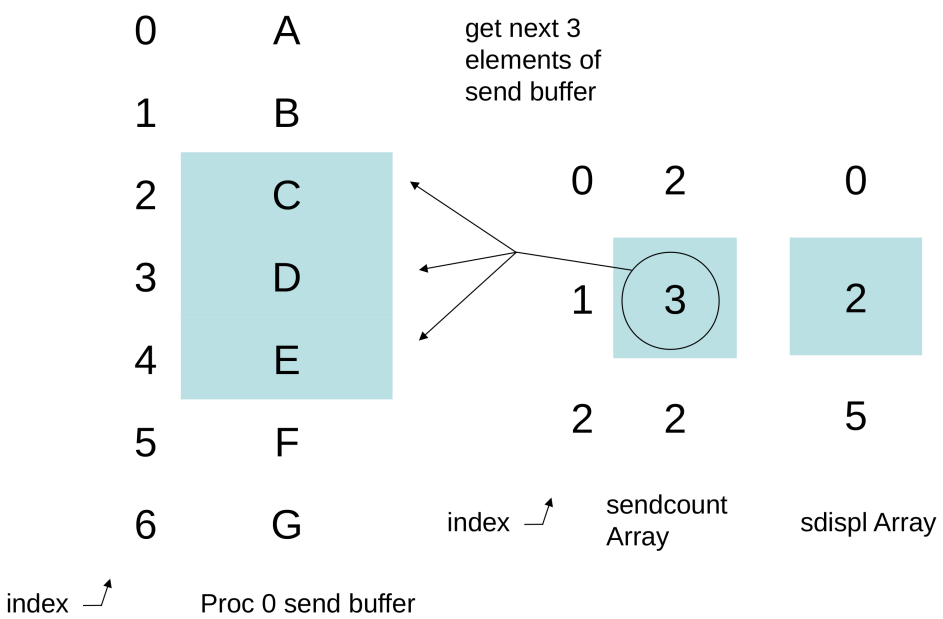
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

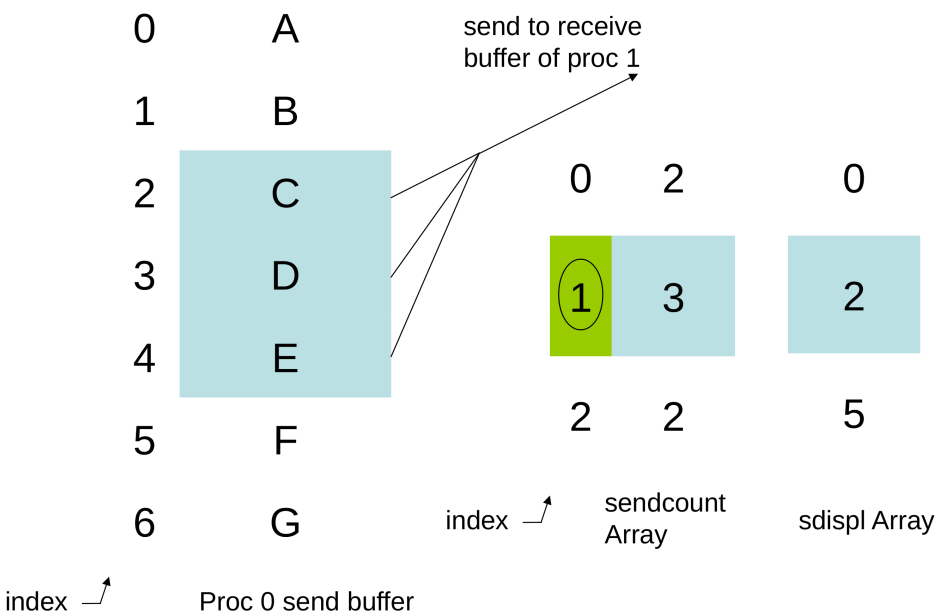
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

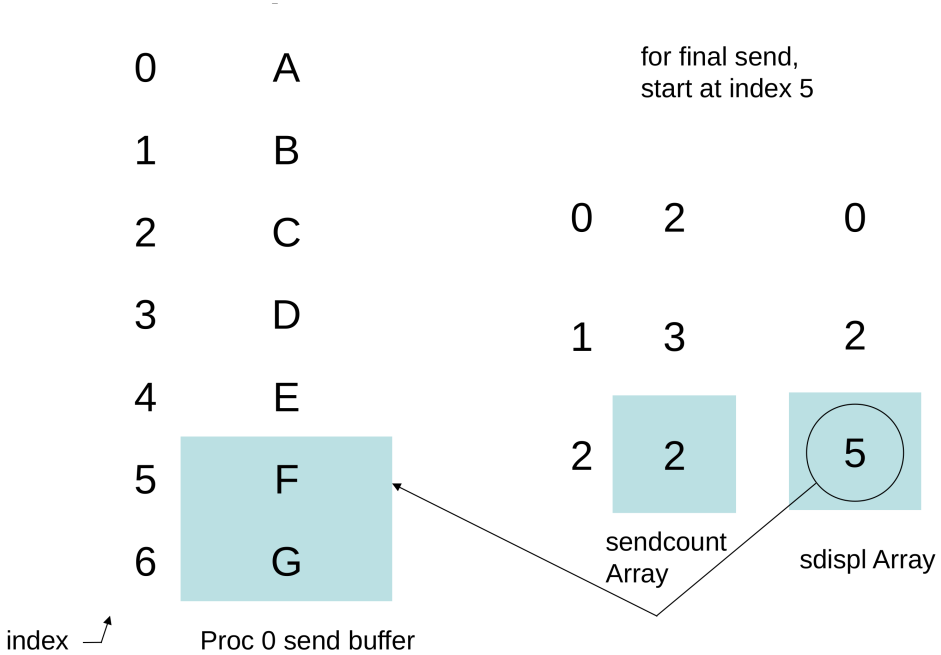
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

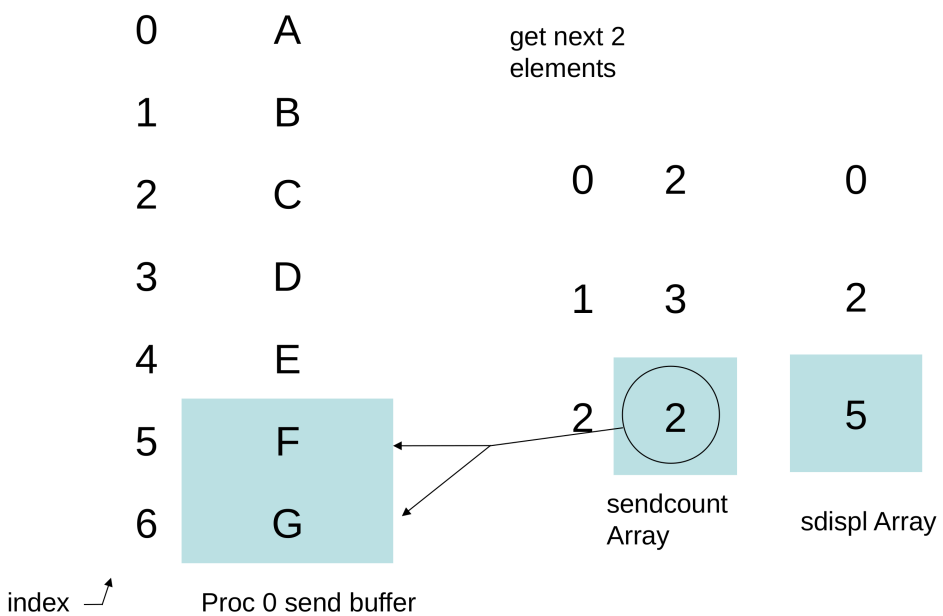
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

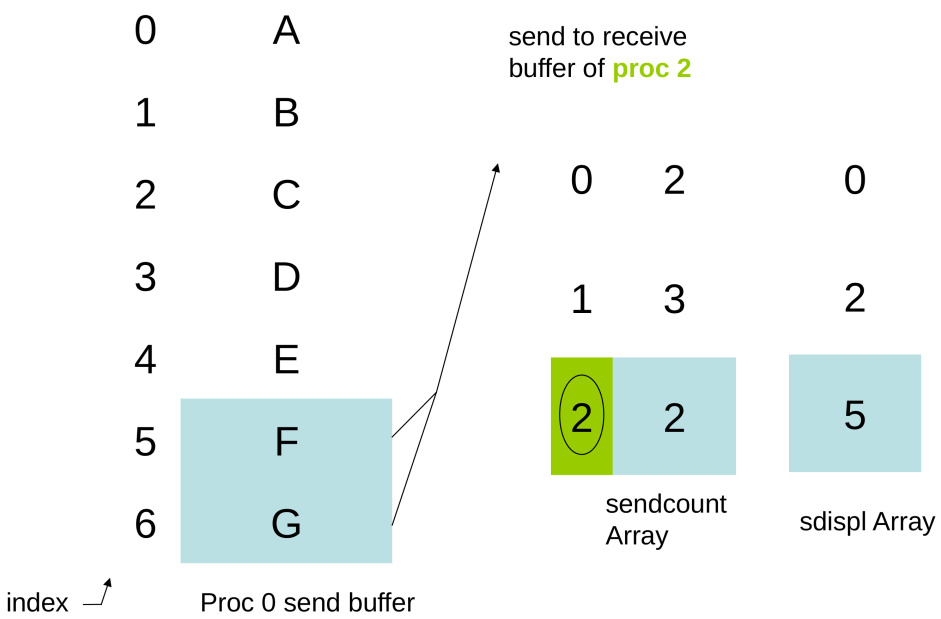
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

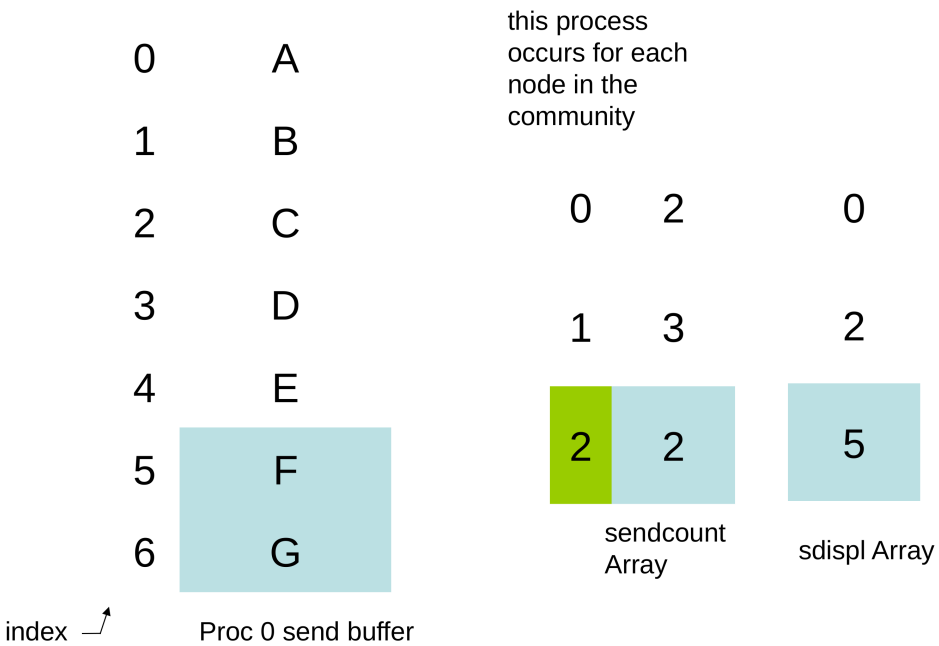
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

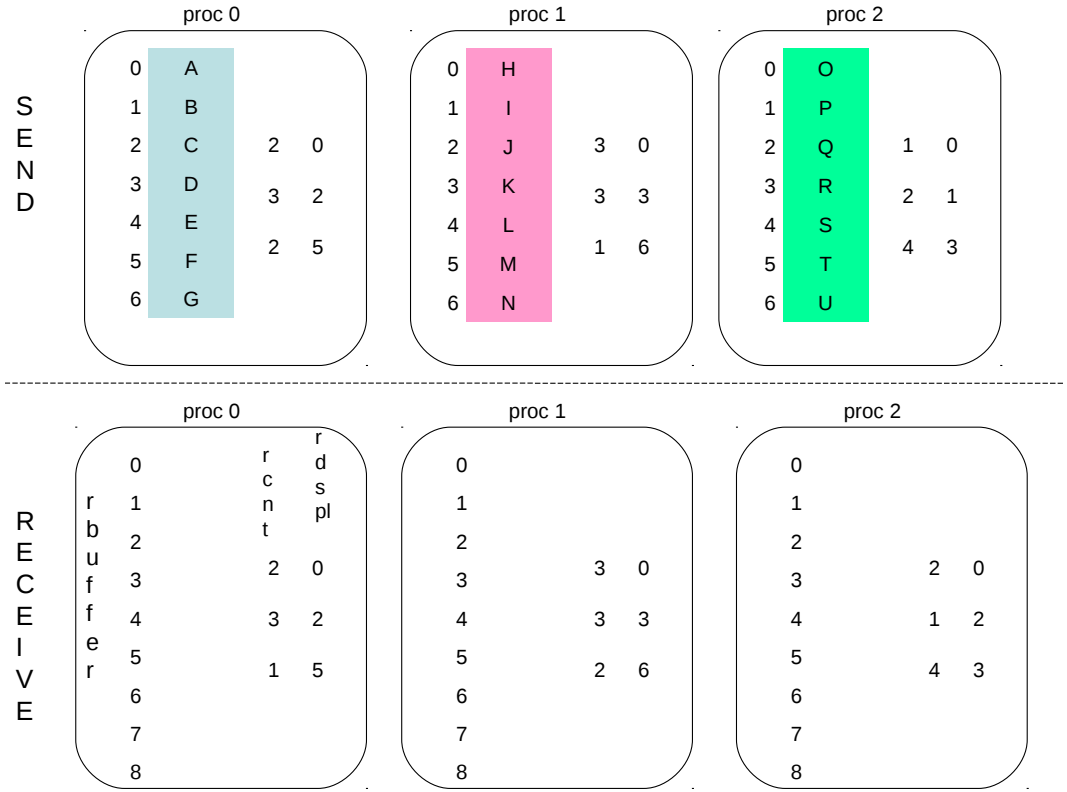
- Example of Send for Proc 0



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

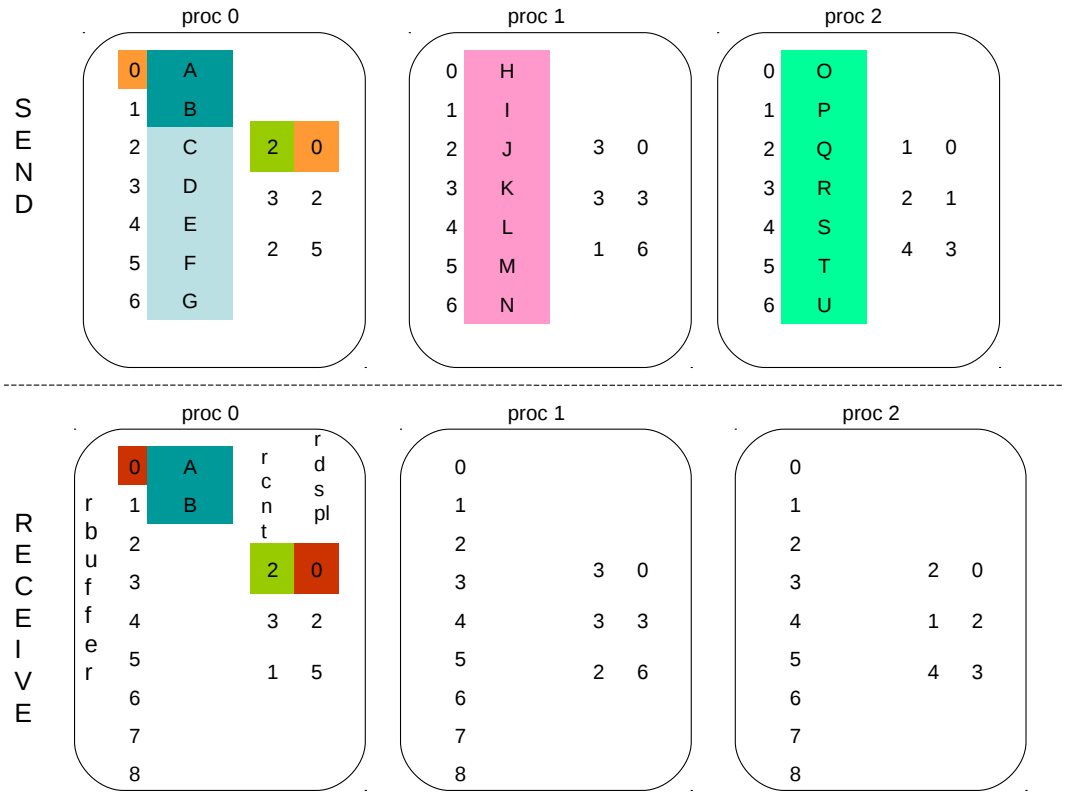
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

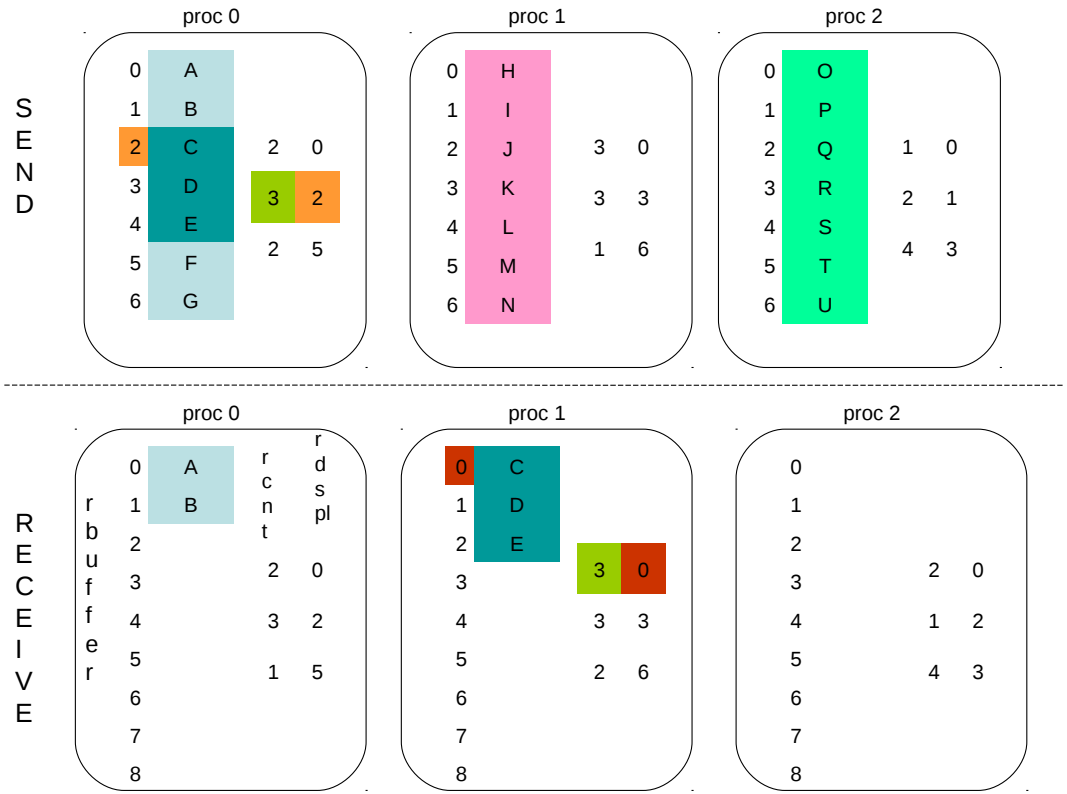
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

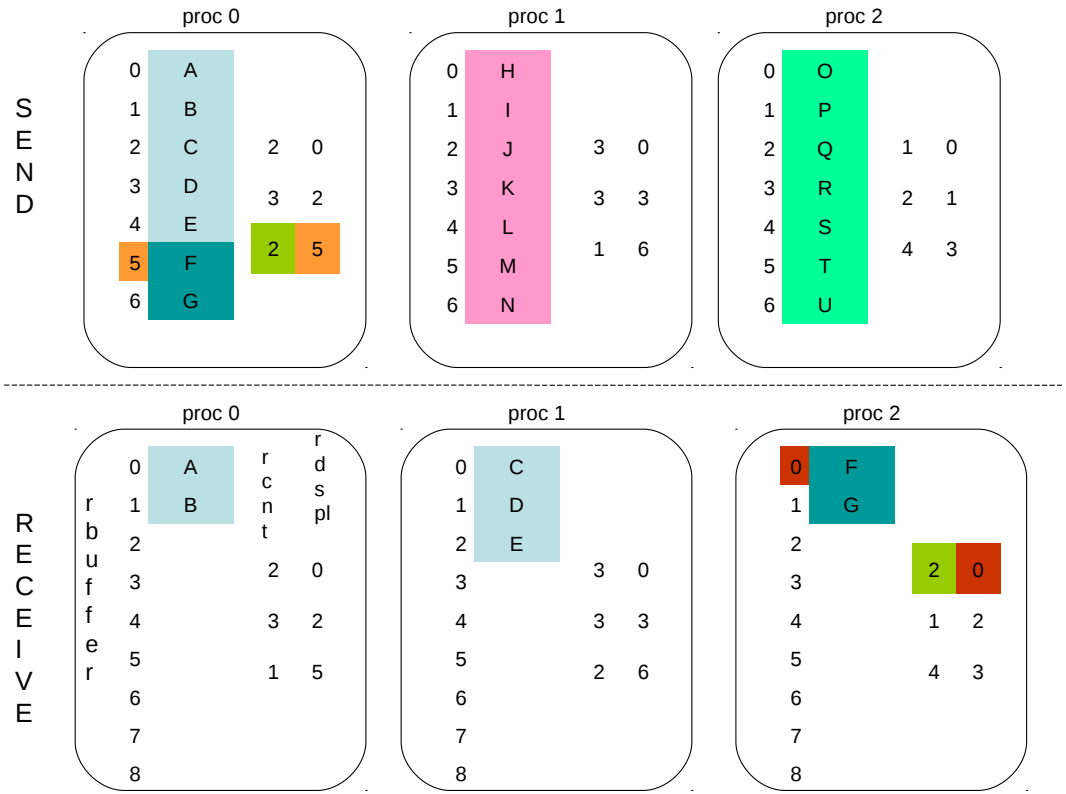
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

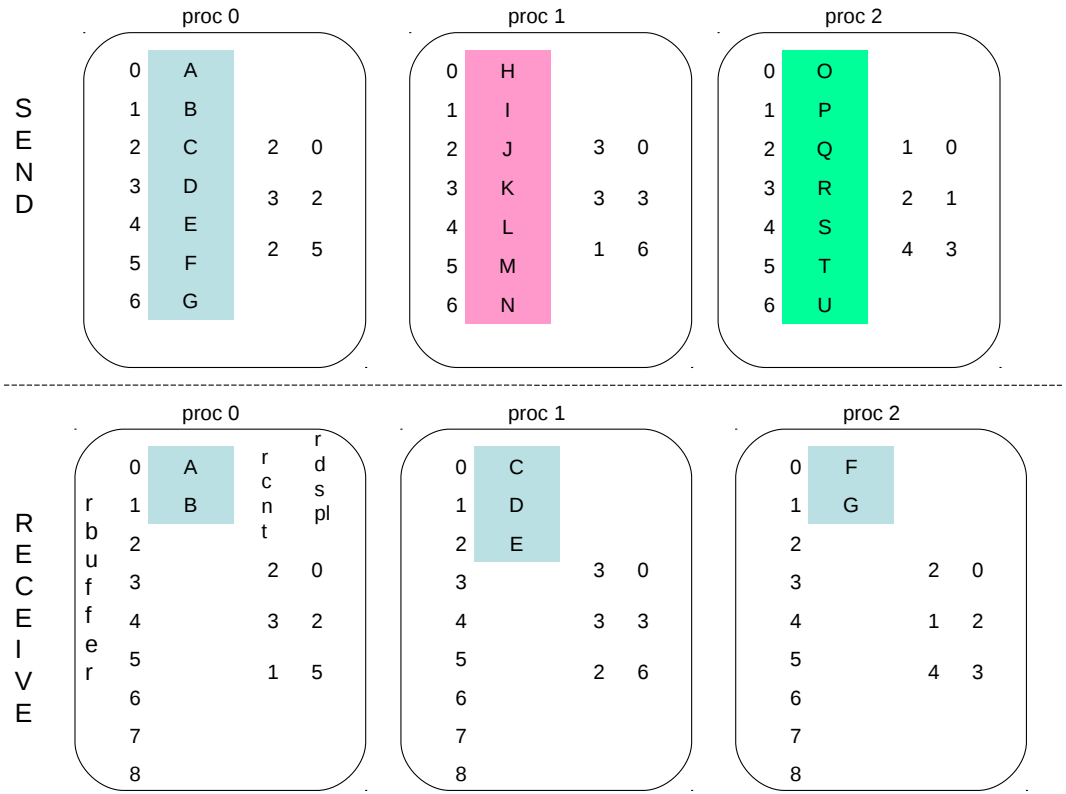
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

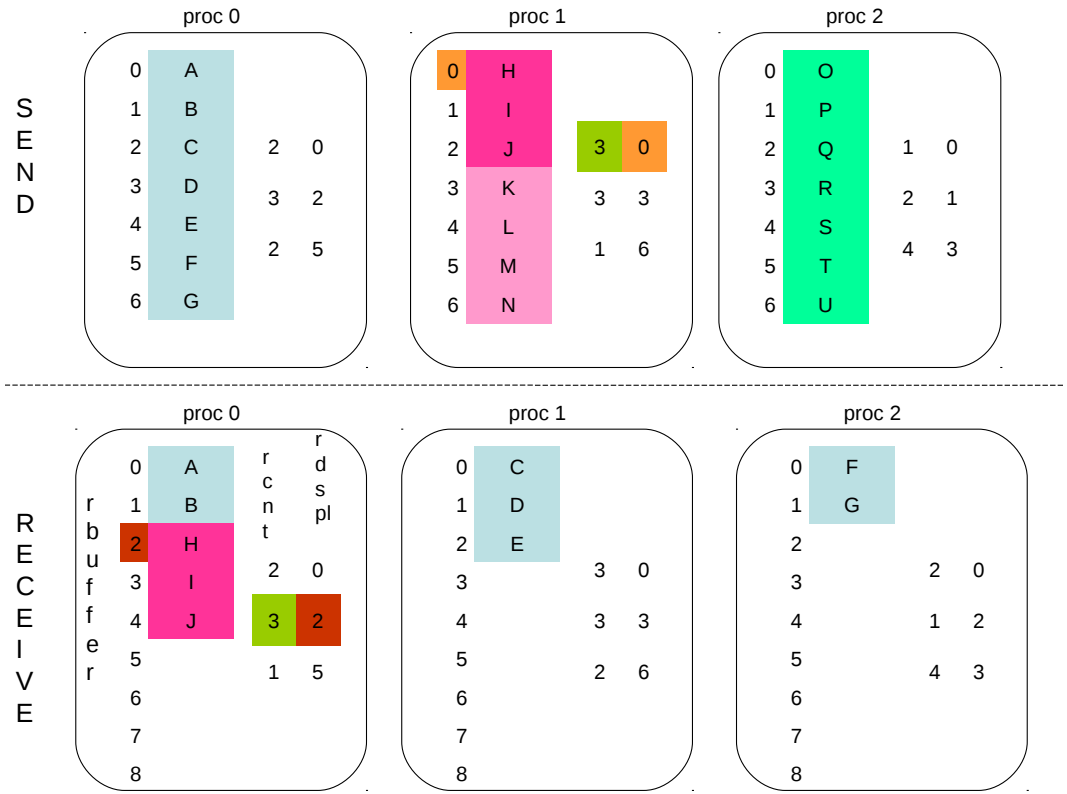
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

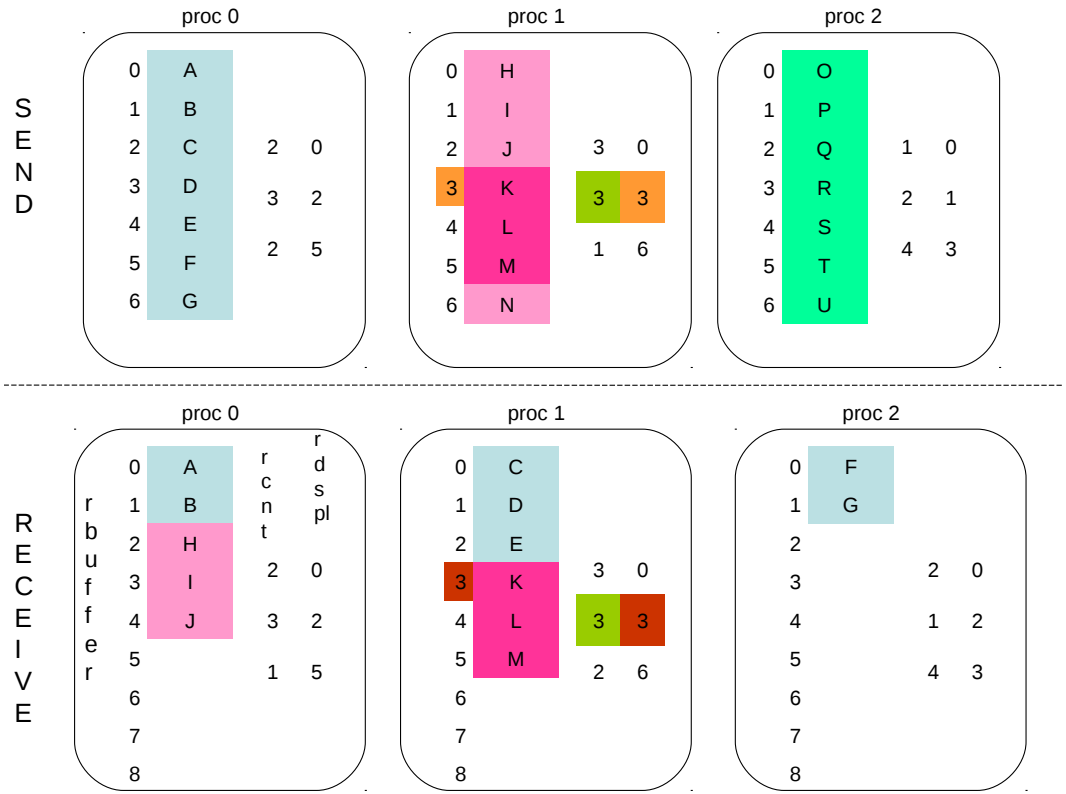
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

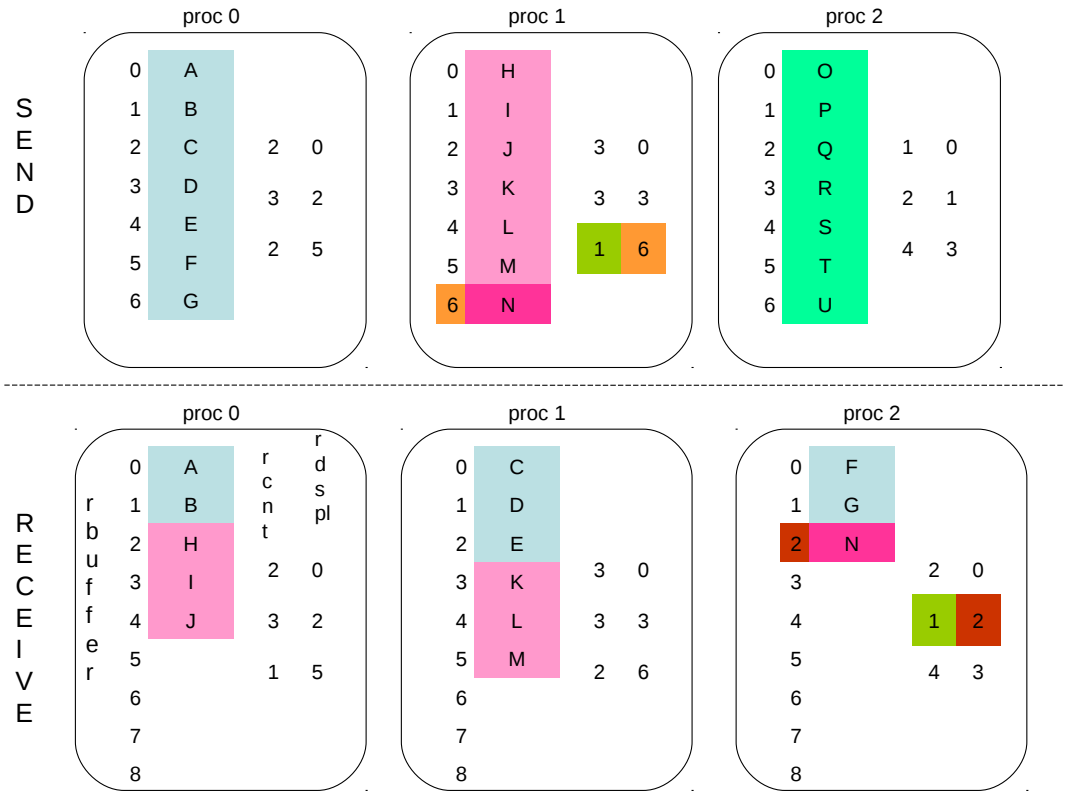
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

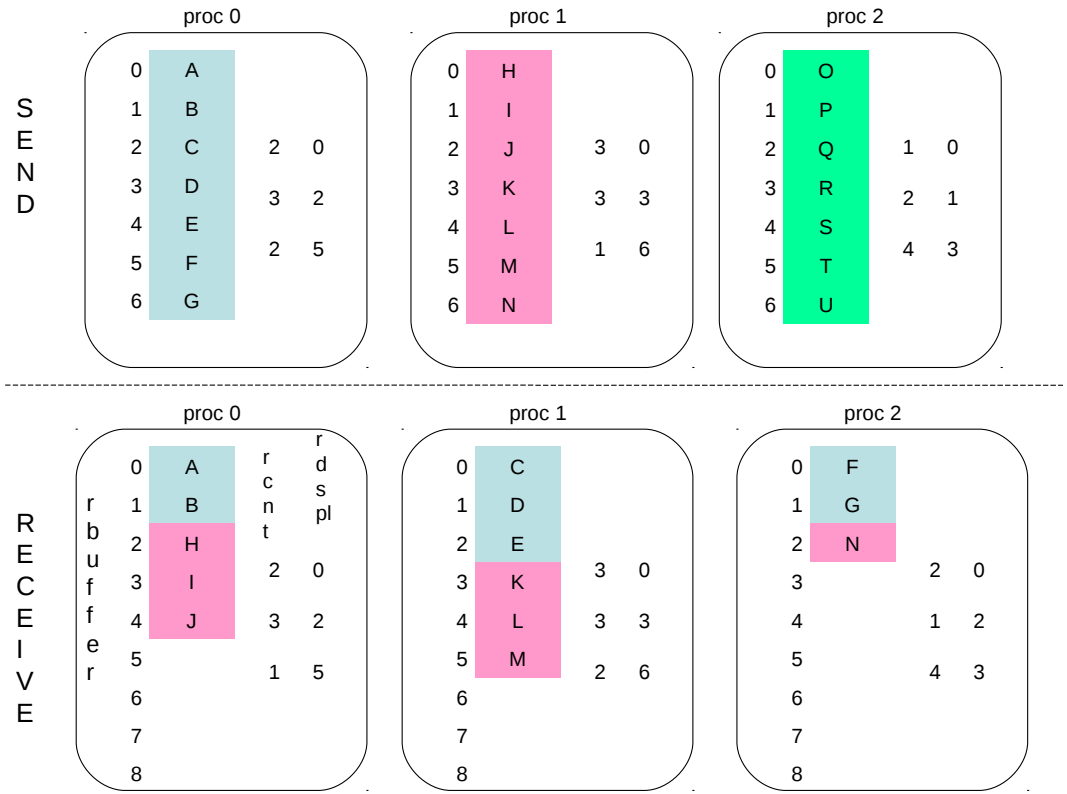
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

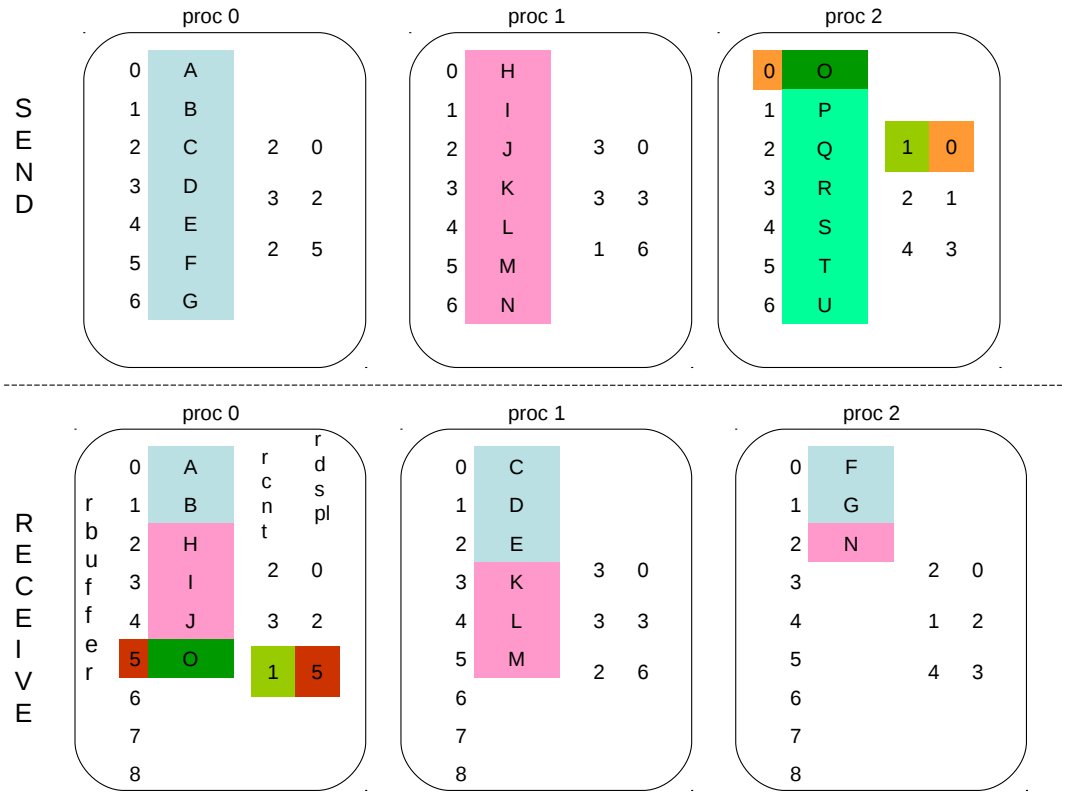
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

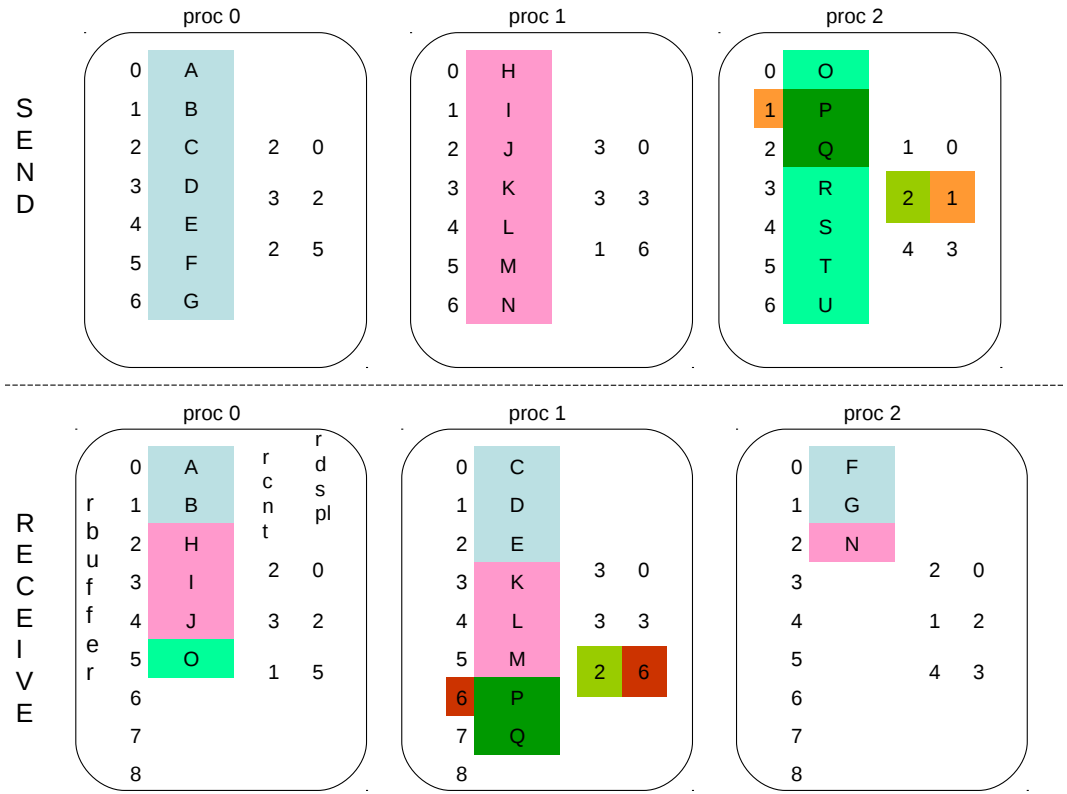
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

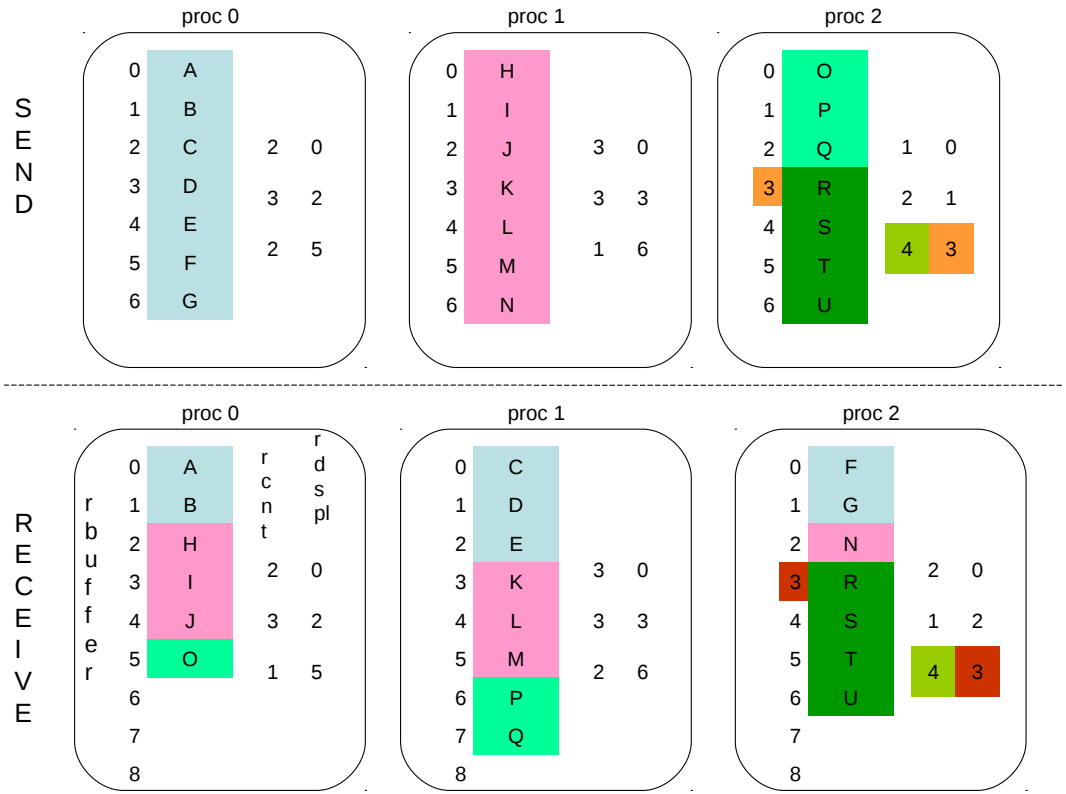
- All processes simultaneously



«v» variations Data movement

```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

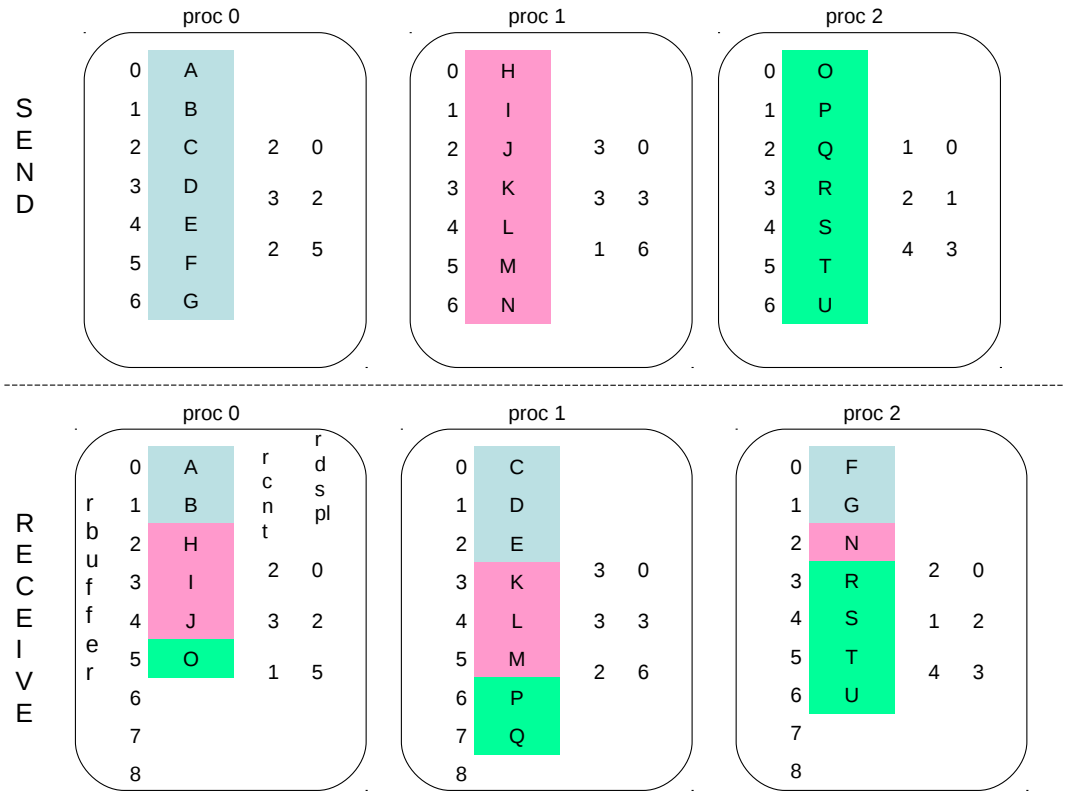
- All processes simultaneously



«v» variations Data movement

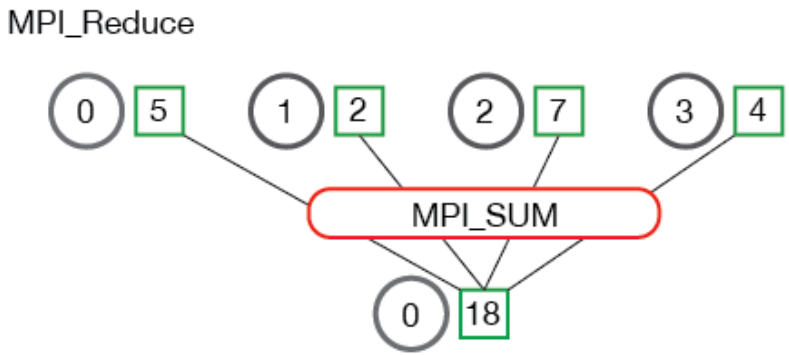
```
int MPI_Alltoallv(void *sendbuf, int* sendcounts, int* sdispls,
                  MPI_Datatype sendtype,
                  void *recvbuf, int* recvcounts, int* rdispls,
                  MPI_Datatype recvtype, MPI_Comm comm);
```

- All processes simultaneously



Reduction Group computations

- An operation with the data on each processor, sending the final result to the root process



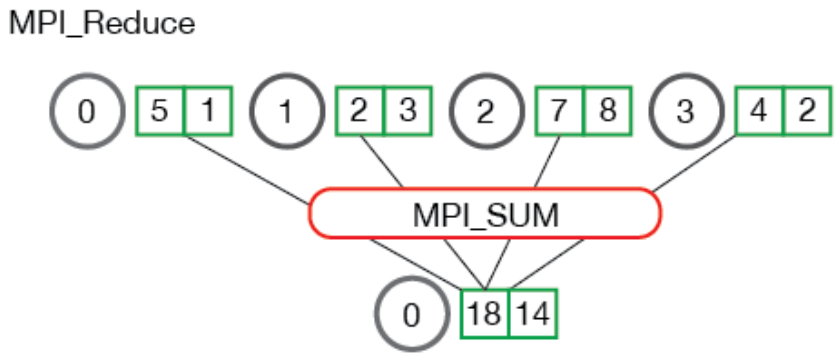
Source : mpitutorial.com

Reduction

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

Reduction
Group computations

- An operation with the data on each processor, sending the final result to the root process



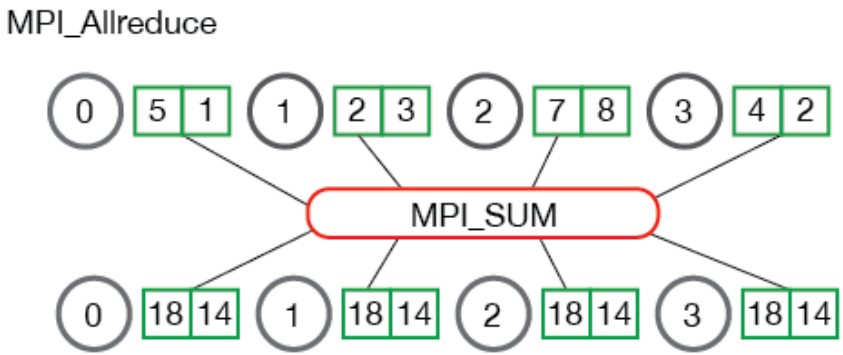
Source : mpitutorial.com

Reduction

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

Reduction Group computations

- `MPI_Allreduce` variant
  - leaves the final result on all processes



Source : mpitutorial.com

Reduction operations

Group computations

- Predefined reduction operations:

MPI_MAX	MPI_MIN	MPI_SUM	MPI_PROD
MPI_LAND	MPI_BAND	MPI_LOR	MPI_BOR
MPI_LXOR	MPI_BXOR	MPI_MAXLOC	MPI_MINLOC

- User-defined reduction operations
  - The user-defined function must follow the structure:

```
int User_function(void *invec, void *inoutvec, int *len,
                  MPI_Datatype datatype)
```

  - The function must be registered with:

```
int MPI_Op_create(MPI_User_function *function,
                  int conmute, MPI_Op *op)
```

  - The operation can be removed with:

```
int MPI_Op_free(MPI_Op *op)
```

Barriers Synchronisation

- Global synchronisation between the communicator processes
- The execution stops until all processes arrive to the barrier

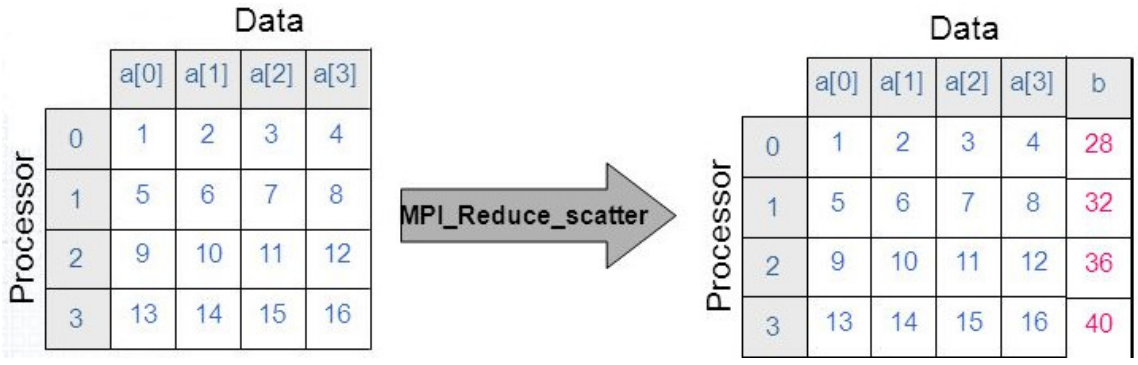
Barrier

```
int MPI_Barrier(MPI_Comm comm)
```

- Useful to measure execution times from a given part of a code

Other collective communications

- Reduction and scatter: combines `MPI_Reduce` y `MPI_Scatterv`
- Combines values according to `rcounts` and scatters the results



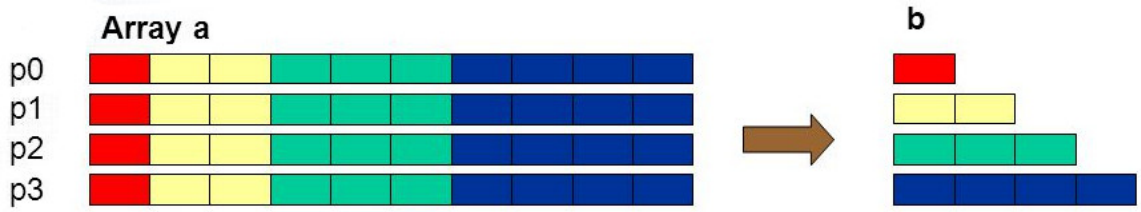
`rcounts = {1,1,1,1}`

Reduction and scatter

```
MPI_Reduce_scatter(void *sbuf, void *rbuf, int *rcounts,
                   MPI_Datatype datatype, MPI_Op op,
                   MPI_Comm comm)
```

Other collective communications

- Reduction and scatter: combines `MPI_Reduce` y `MPI_Scatterv`
- Combines values according to `rcounts` and scatters the results



`rcounts = {1,2,3,4}`

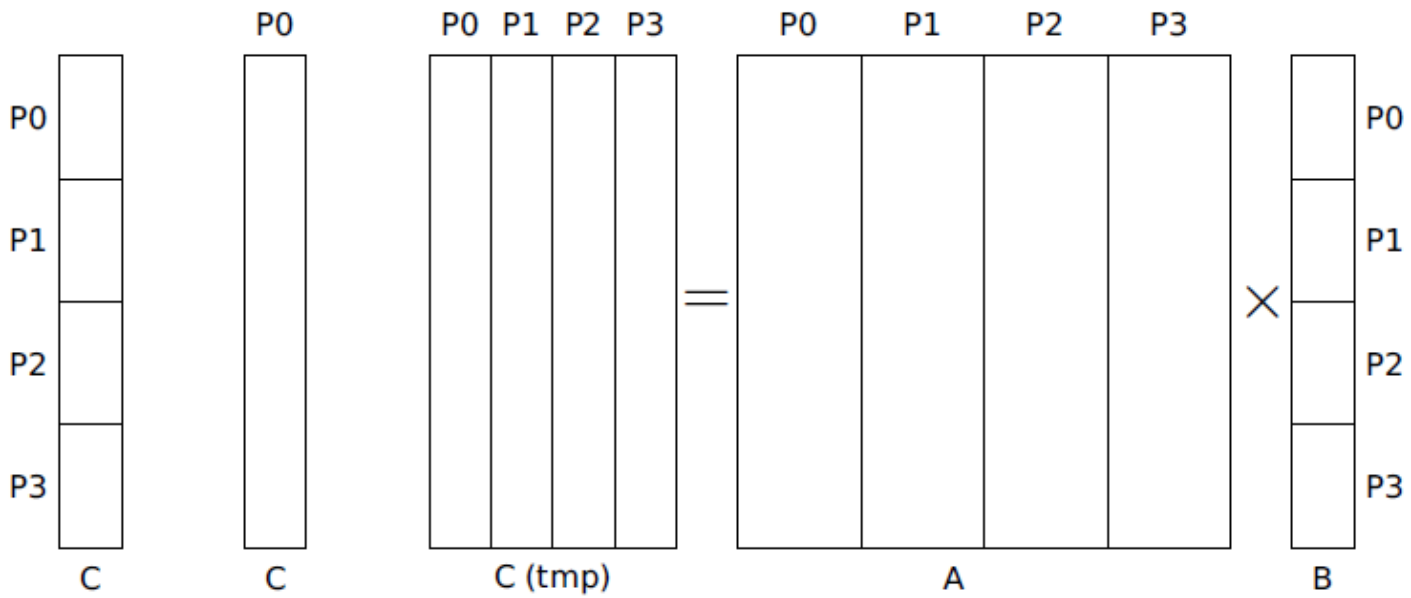
Reduction and scatter

```
MPI_Reduce_scatter(void *sbuf, void *rbuf, int *rcounts,
                   MPI_Datatype datatype, MPI_Op op,
                   MPI_Comm comm)
```

Other collective communications

```
MPI_Reduce_scatter(void *sbuf, void *rbuf, int *rcounts,
                   MPI_Datatype datatype, MPI_Op op,
                   MPI_Comm comm)
```

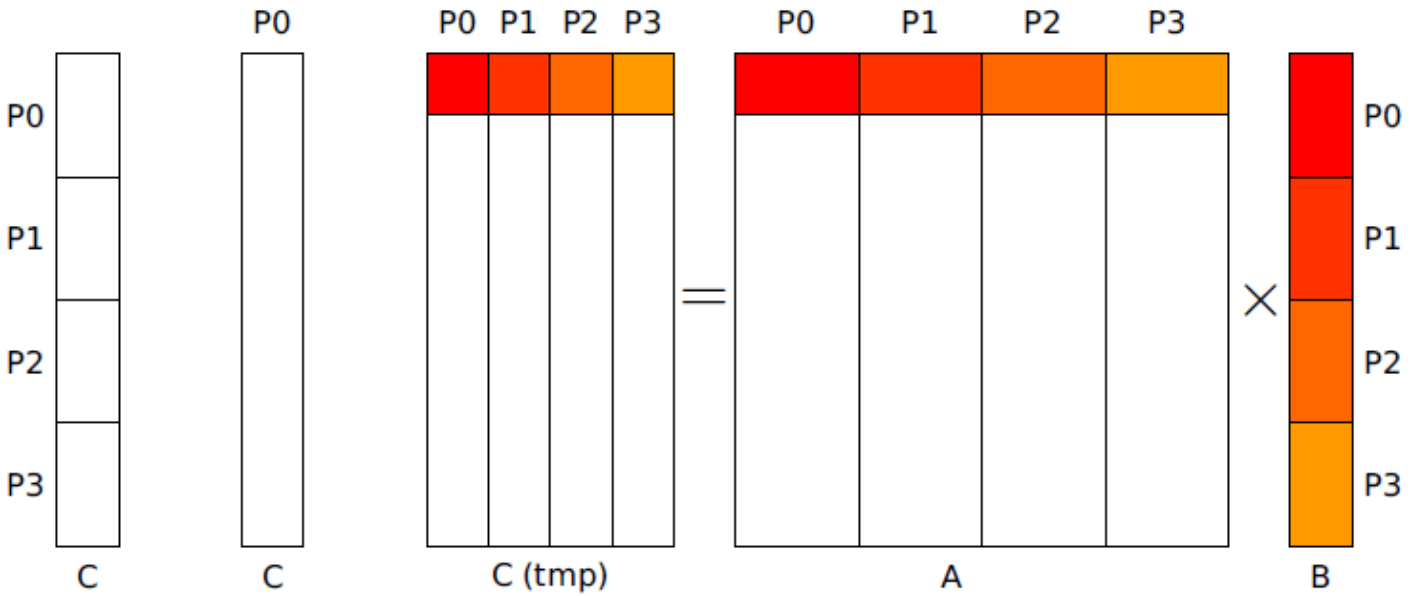
Example: Matrix-vector product



Other collective communications

```
MPI_Reduce_scatter(void *sbuf, void *rbuf, int *rcounts,
                   MPI_Datatype datatype, MPI_Op op,
                   MPI_Comm comm)
```

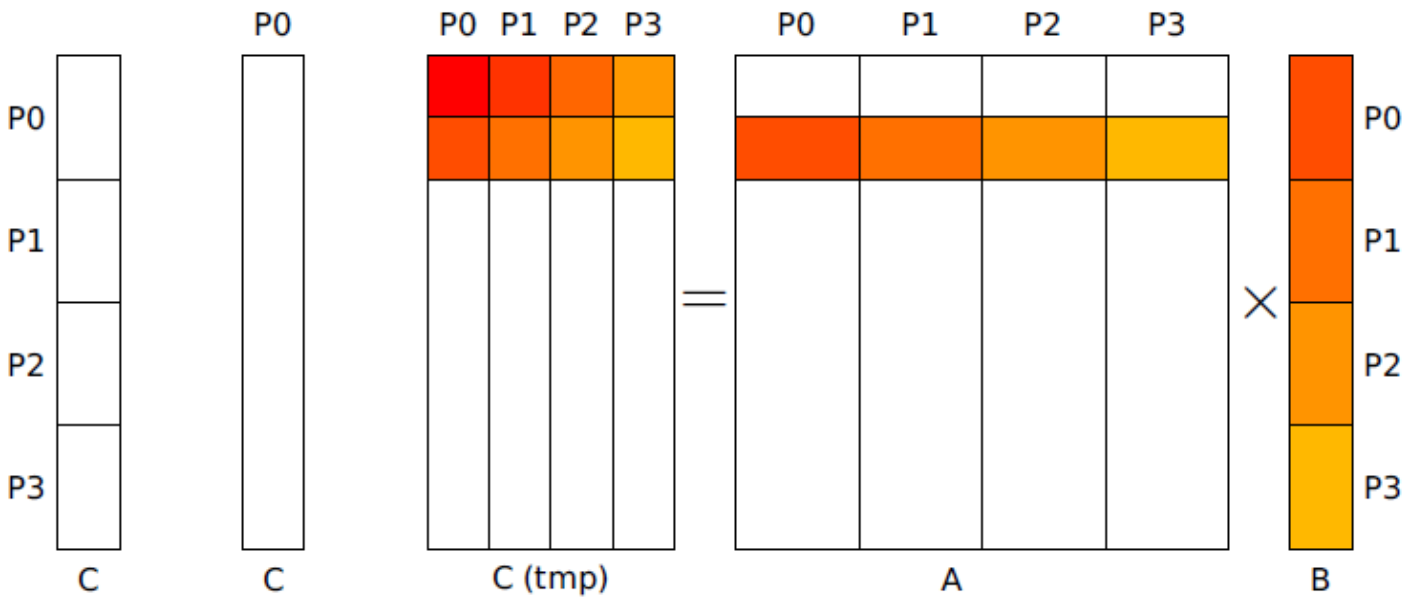
Example: Matrix-vector product



Other collective communications

```
MPI_Reduce_scatter(void *sbuf, void *rbuf, int *rcounts,
                   MPI_Datatype datatype, MPI_Op op,
                   MPI_Comm comm)
```

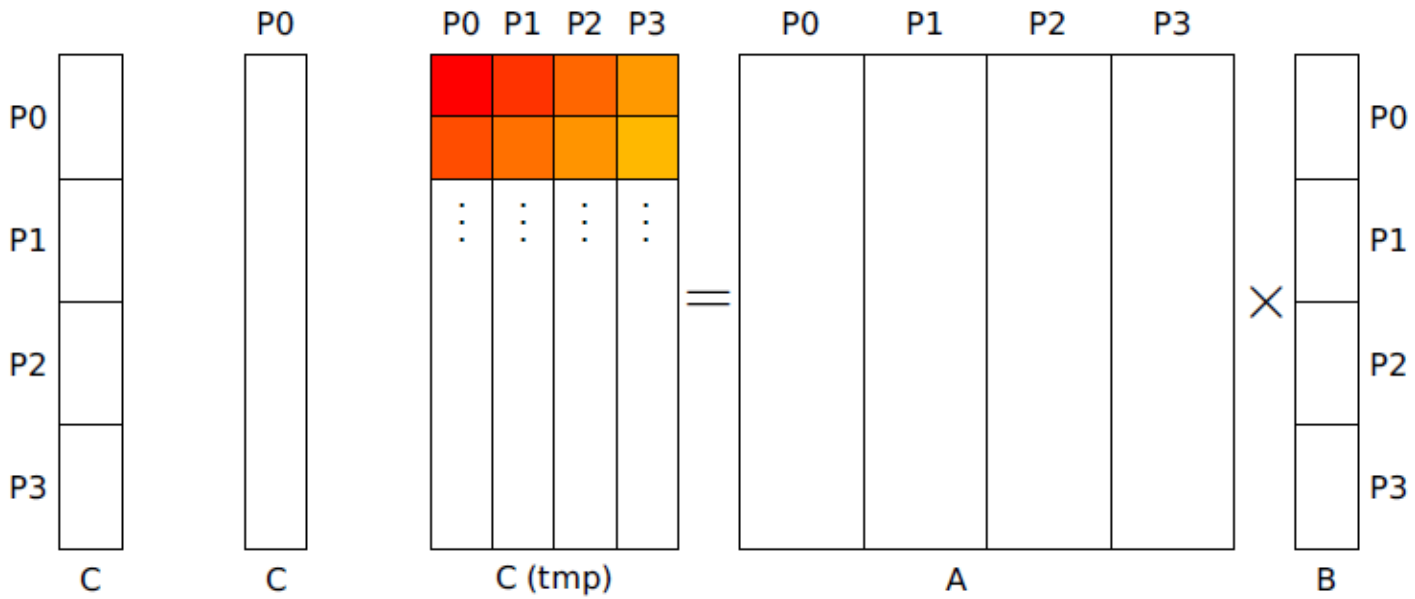
Example: Matrix-vector product



Other collective communications

```
MPI_Reduce_scatter(void *sbuf, void *rbuf, int *rcounts,
                   MPI_Datatype datatype, MPI_Op op,
                   MPI_Comm comm)
```

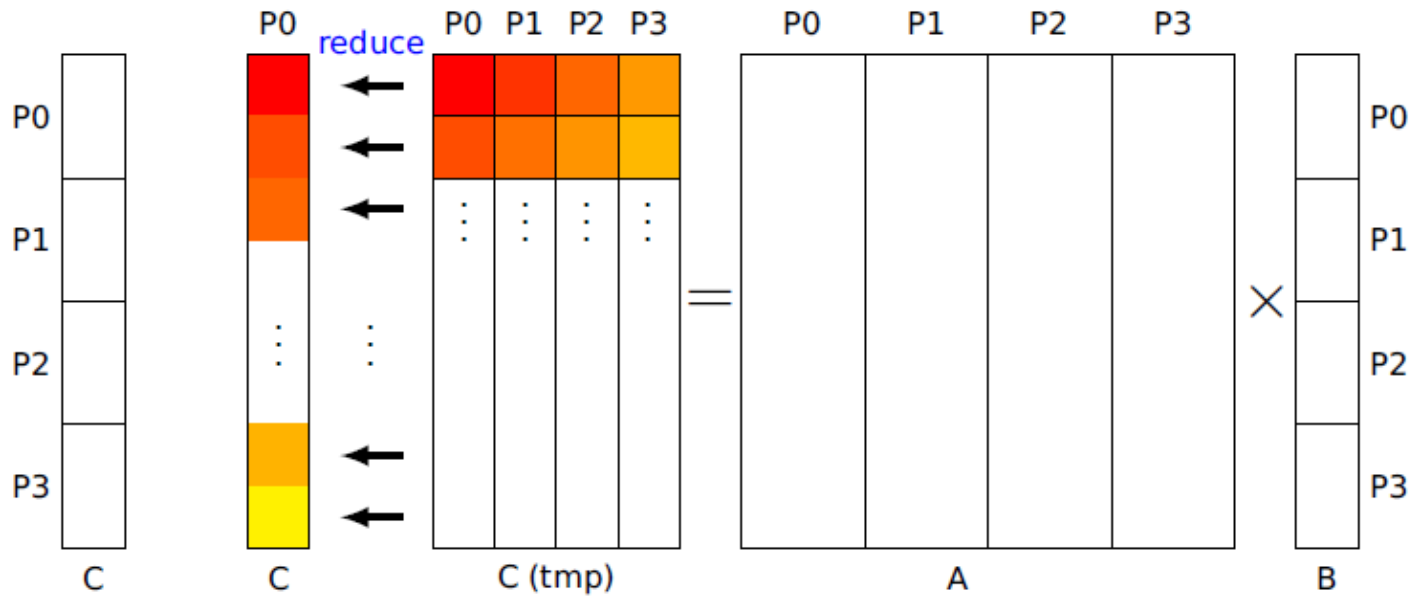
Example: Matrix-vector product



Other collective communications

```
MPI_Reduce_scatter(void *sbuf, void *rbuf, int *rcounts,
                   MPI_Datatype datatype, MPI_Op op,
                   MPI_Comm comm)
```

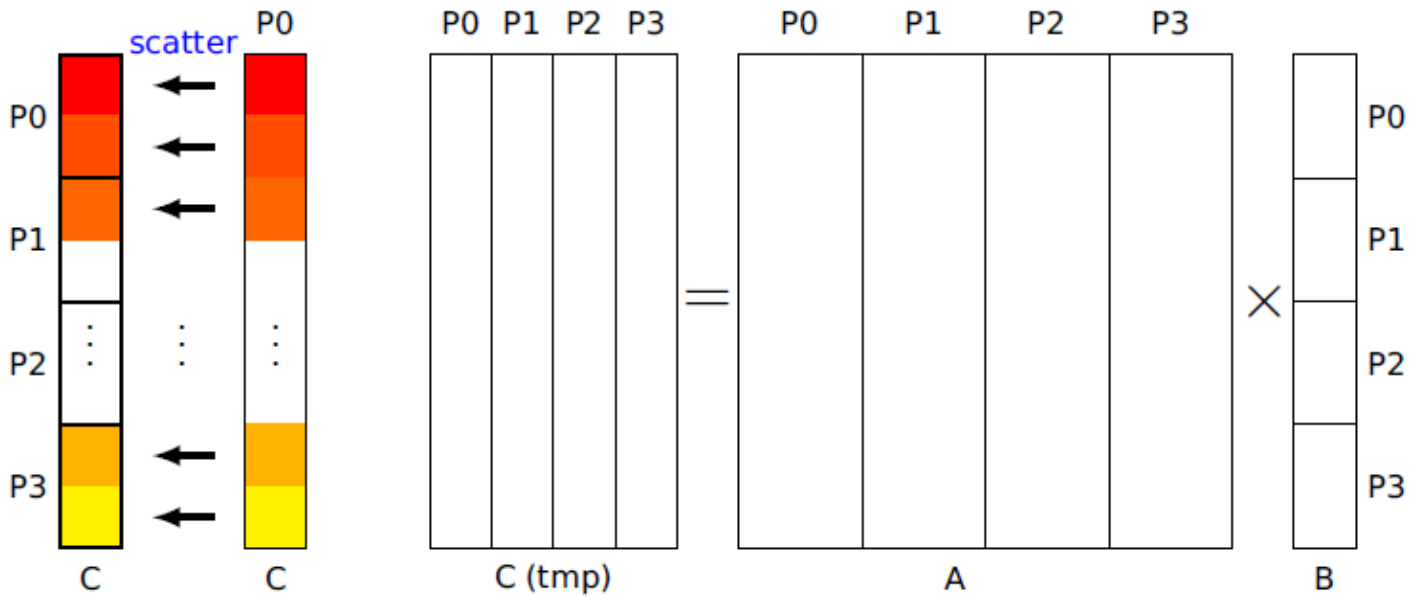
Example: Matrix-vector product



Other collective communications

```
MPI_Reduce_scatter(void *sbuf, void *rbuf, int *rcounts,
                   MPI_Datatype datatype, MPI_Op op,
                   MPI_Comm comm)
```

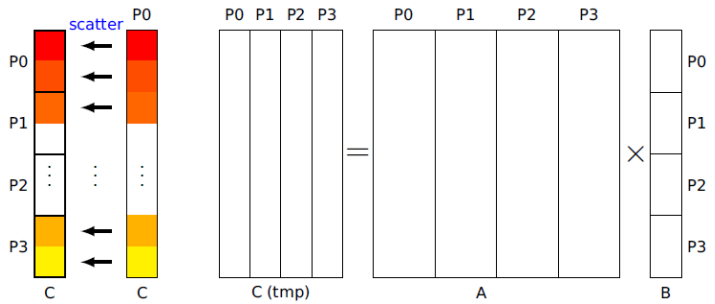
Example: Matrix-vector product



Other collective communications

```
MPI_Reduce_scatter(void *sbuf, void *rbuf, int *rcounts,
                   MPI_Datatype datatype, MPI_Op op,
                   MPI_Comm comm)
```

Example: Matrix-vector product



- Each process receives its part of the product on its *C* sub-array
  - For example, to start over a new iteration on an iterative matrix-vector product