MESSAGE PASSING INTERFACE - 3

Dr. Emmanuel Pilli

Contents

- Collective Communications
  - Barrier, Broadcast, Scatter, Gather
- Global Reduction Operations
  - Reduce, Allreduce, Reduce_scatter, Scan
- Predefined Reduction Operations
- Revision

Collective Communication

- Communications involving a group of processes
- Called by *all* processes in a communicator
- Examples:
  - Broadcast, scatter, gather, etc (Data Distribution)
  - Global sum, global maximum, etc. (Collective Operations)
  - Barrier synchronization

Characteristics of Collective Communication

- Collective communication will not interfere with point-to-point communication and vice-versa
- All processes must call the collective routine
- Synchronization not guaranteed (except for barrier)
- No non-blocking collective communication
- No tags
- Receive buffers must be exactly the right size

Barrier Synchronization

- Red light for each processor: turns green when all processors have arrived
- Slower than hardware barriers

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

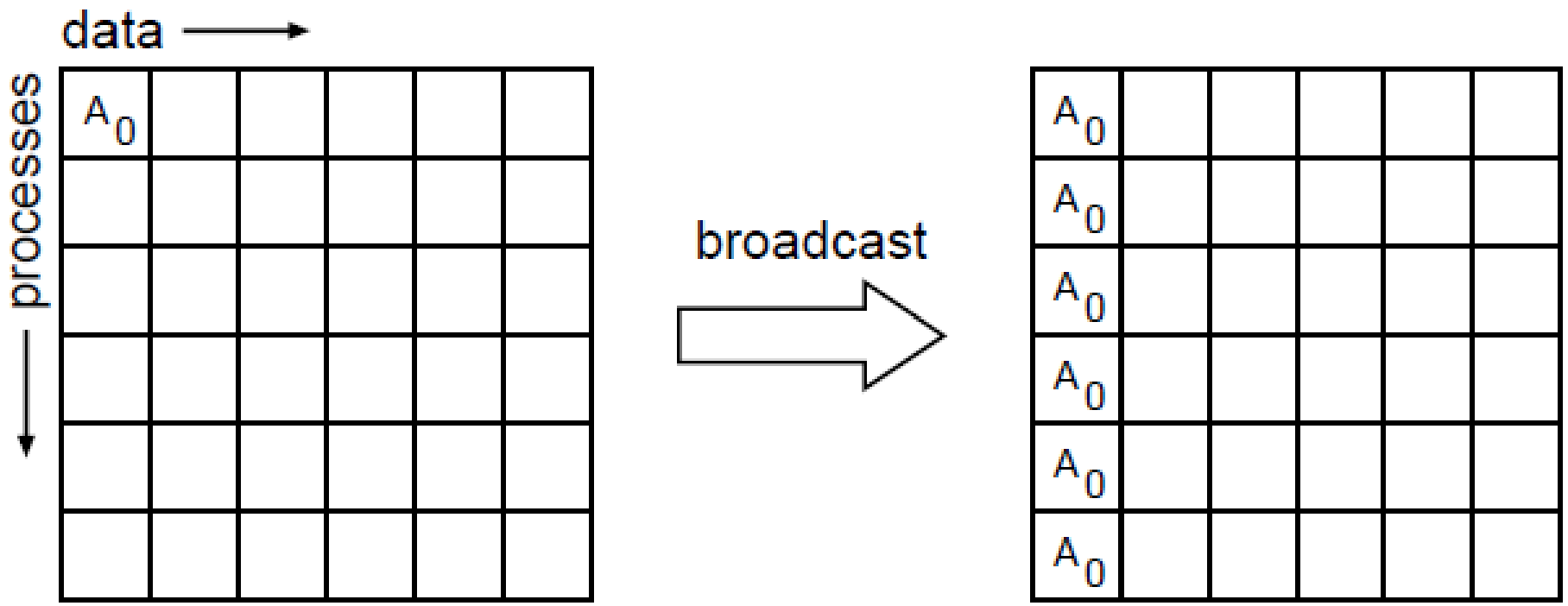
Broadcast

- One-to-all communication: same data sent from root process to all the others in the communicator

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

- All processes must specify same root rank and communicator

Broadcast



Sample Program

```
#include<mpi.h>
void main (int argc, char *argv[])
{
  int rank;
  double param;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  if(rank==5) param=23.0;
    MPI_Bcast(&param,1,MPI_DOUBLE,5,
                  MPI_COMM_WORLD);
  printf("P:%d after broadcast parameter
                is %f\n",rank,param);
  MPI_Finalize();
}
```

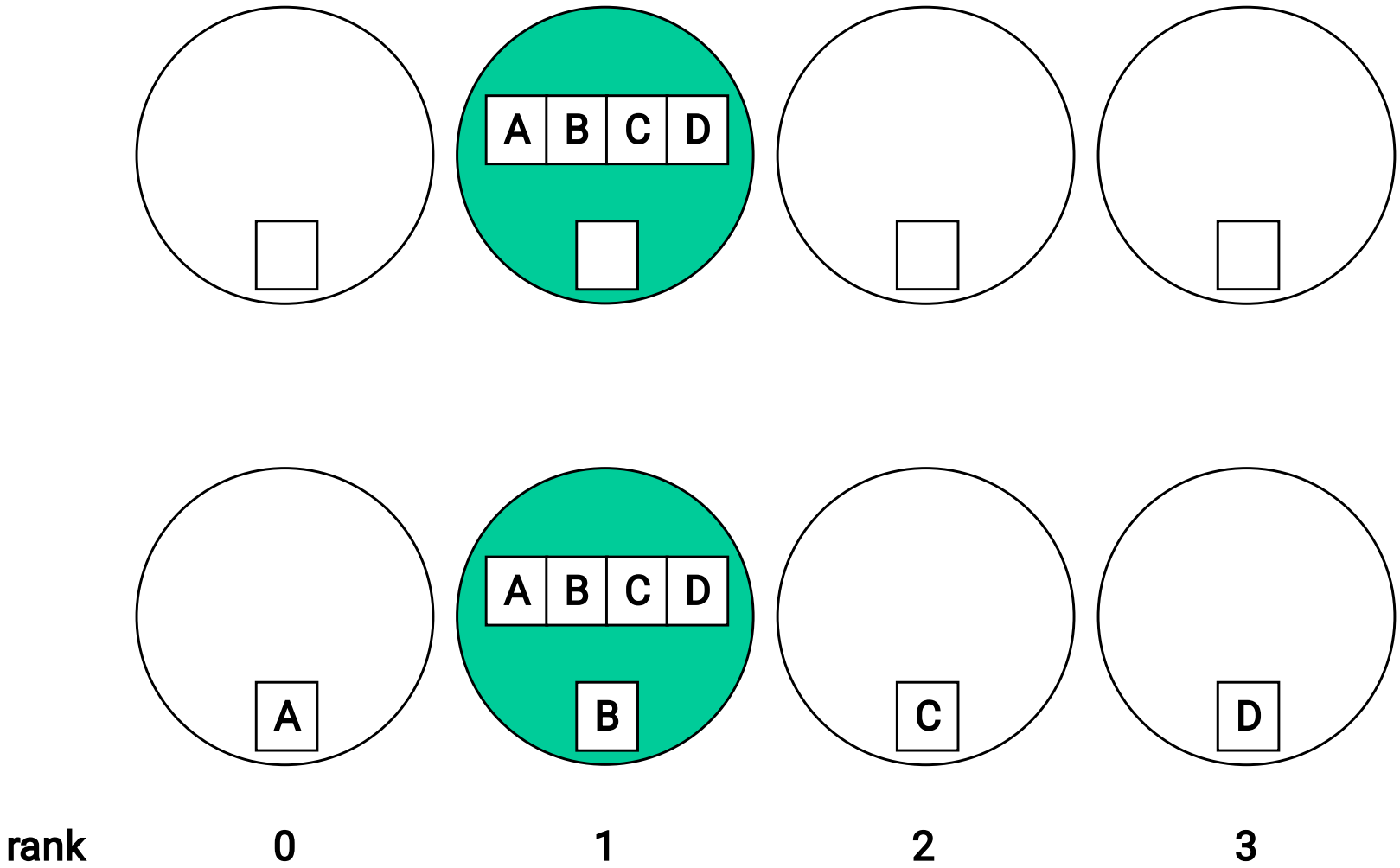
Scatter

- One-to-all communication: different data sent to each process in the communicator (in rank order)

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

- **sendcount** is the number of elements sent to each process, not the “total” number sent
  - send arguments are significant only at the root process

Scatter Example



Sample Program

```
#include <mpi.h>
 void main (int argc, char *argv[]) {
    int rank,size,i,j;
    double param[4],mine;
    int sndcnt,revcnt;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);
    revcnt=1;
    if(rank==3){
       for(i=0;i<4;i++) param[i]=23.0+i;
       sndcnt=1;
    }
    MPI_Scatter(param, sndcnt, MPI_DOUBLE, &mine, revcnt,
                        MPI_DOUBLE, 3, MPI_COMM_WORLD);
    printf("P:%d mine is %f\n",rank,mine);
    MPI_Finalize();
 }
```

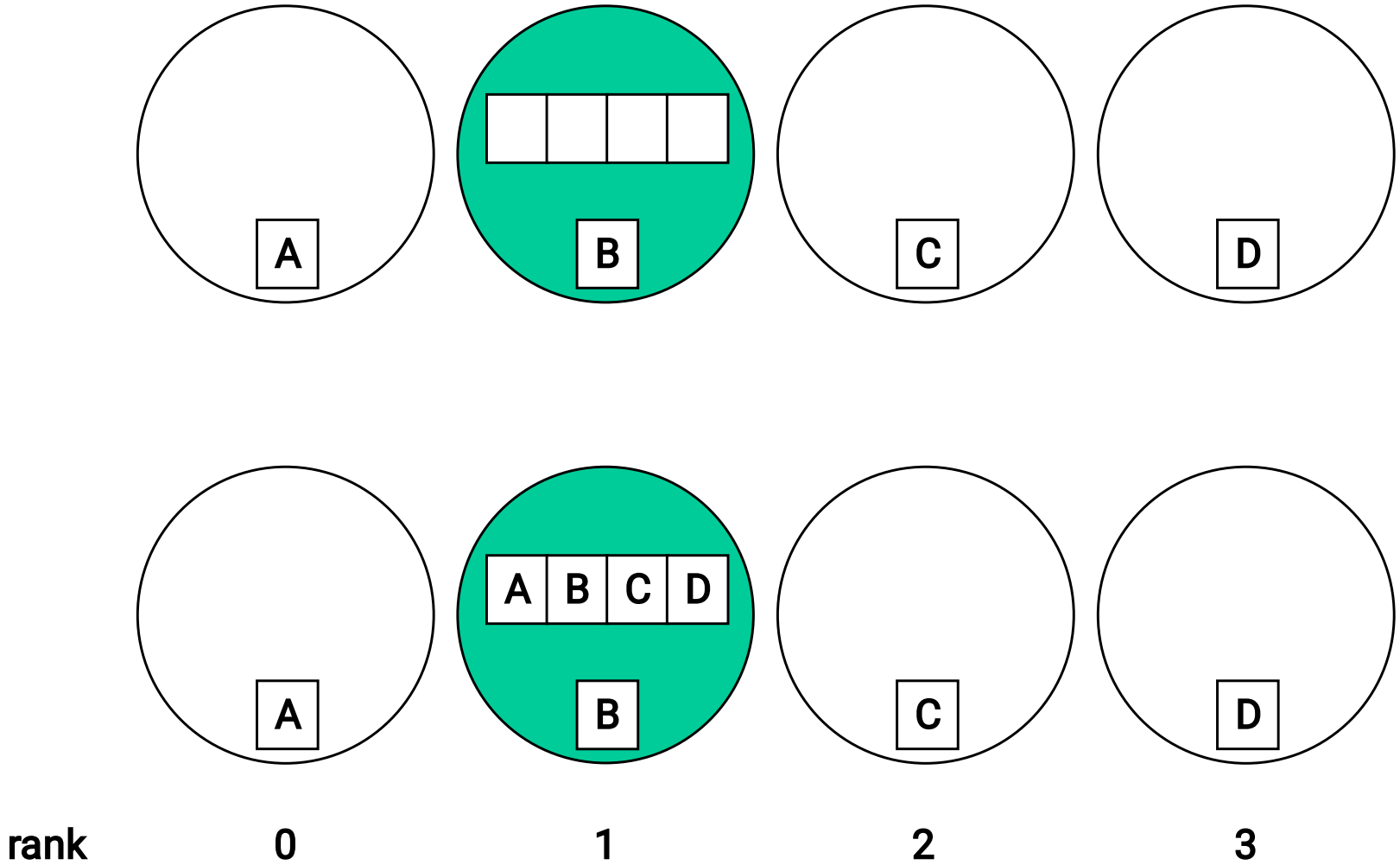
Gather

- All-to-one communication: different data collected by root process
  - Collection done in rank order

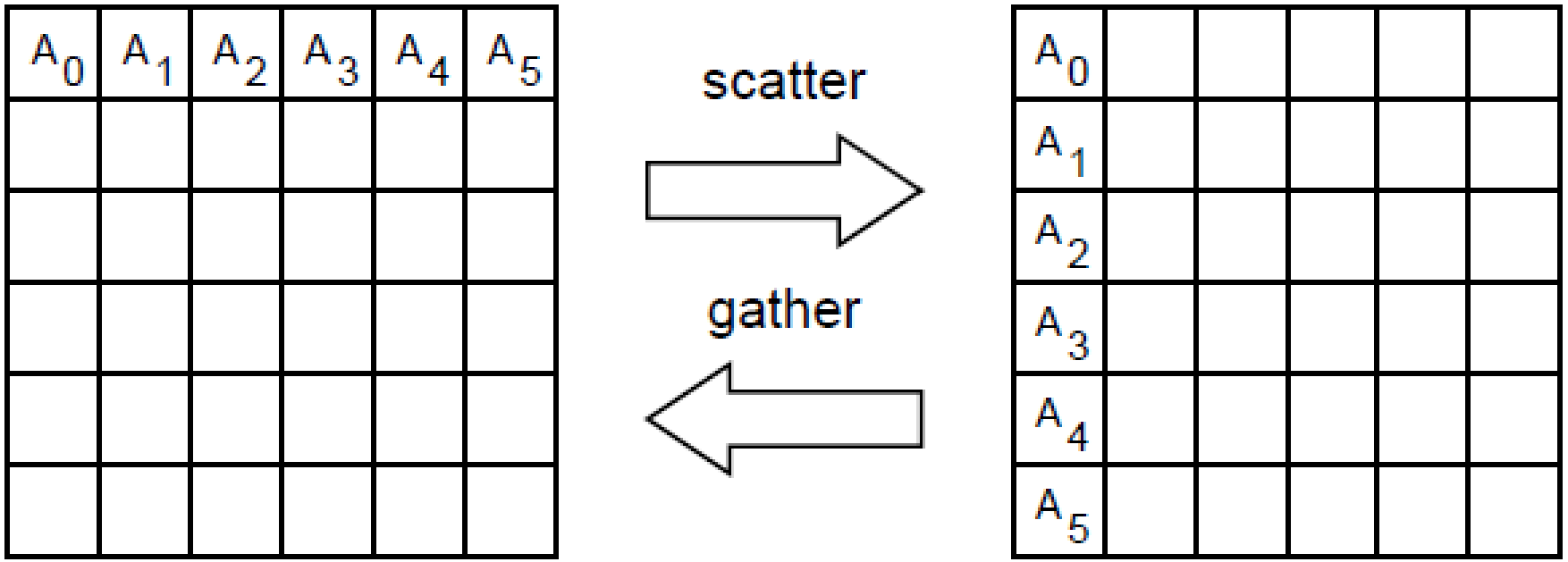
```
int MPI_Gather (void* sendbuf,          int sendcount,
 MPI_Datatype sendtype,
 void* recvbuf, int recvcount, MPI_Datatype recvtype,
int root,
  MPI_Comm comm)
```

- Receive arguments only meaningful at the root process

Gather Example



Scatter / Gather



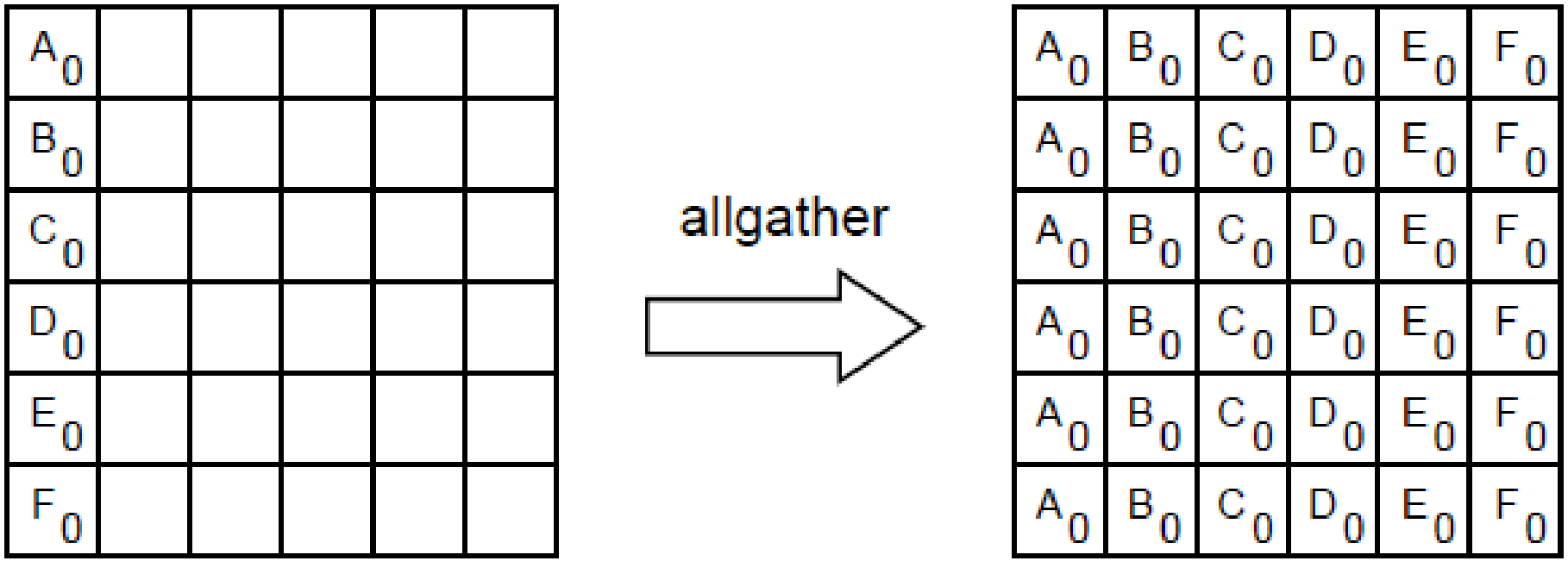
Scatter / Gather Variations

- MPI_Allgather
- MPI_Alltoall
- No root process specified: all processes get gathered or scattered data
- Send and receive arguments significant for all processes

Scatter / Gather Variations

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

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

Scatter / Gather Variations



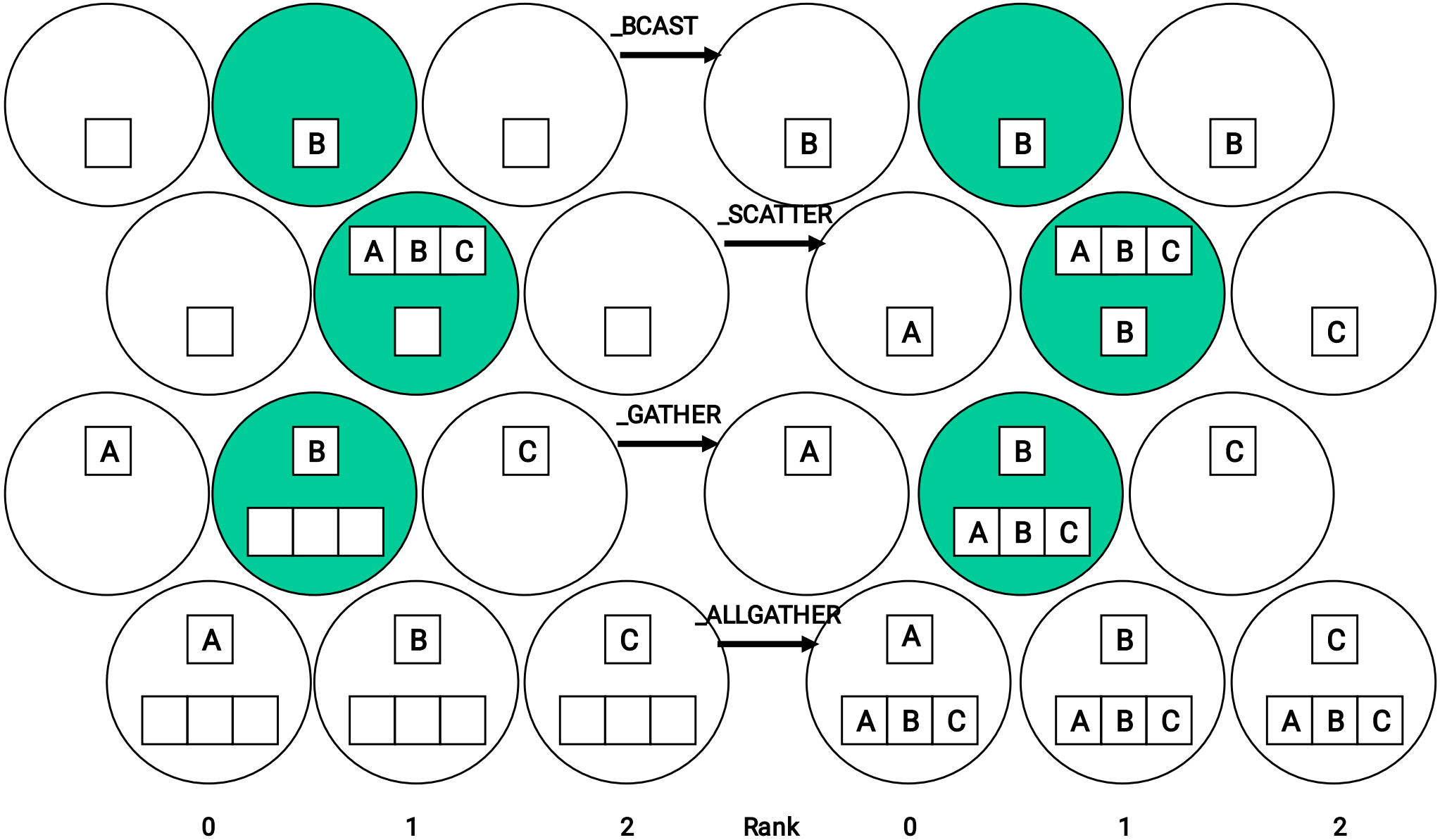
Scatter / Gather Variations

A_0	A_1	A_2	A_3	A_4	A_5
B_0	B_1	B_2	B_3	B_4	B_5
C_0	C_1	C_2	C_3	C_4	C_5
D_0	D_1	D_2	D_3	D_4	D_5
E_0	E_1	E_2	E_3	E_4	E_5
F_0	F_1	F_2	F_3	F_4	F_5

alltoall

A_0	B_0	C_0	D_0	E_0	F_0
A_1	B_1	C_1	D_1	E_1	F_1
A_2	B_2	C_2	D_2	E_2	F_2
A_3	B_3	C_3	D_3	E_3	F_3
A_4	B_4	C_4	D_4	E_4	F_4
A_5	B_5	C_5	D_5	E_5	F_5

Summary


_BCAST
B
B
B
B
_SCATTER
A B C
A B C
A
B
C
_GATHER
A
B
C
A
B
C
A B C
_ALLGATHER
A
B
C
A
B
C
A B C
A B C
A B C
0
1
2
Rank
0
1
2

Summary

- Root sends data to all processes (itself included): **Broadcast** and **Scatter**
- Root receives data from all processes (itself included): **Gather**
- Each process will communicate with each process (itself included): **Allgather** and **Alltoall**

Global Reduction Operations

- Used to compute a result involving data distributed over a group of processes
- Perform a global reduce operation such as sum, max, logical AND, etc across all the members of a group
- The reduction operation can be either one of a predefined list of operations or a user-defined operation

Global Reduction Operations

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

- **count** is the number of "*ops*" done on consecutive elements of **sendbuf** (it is also size of **recvbuf**)
- **op** is an associative operator that takes two operands of type **datatype** and returns a result of the same type

Global Reduction Operations

- The global reduction functions come in several flavors
  - a reduce that returns the result of the reduction at one node
  - an allreduce that returns this result at all nodes
  - a scan parallel prefix operation
- A reduce-scatter operation combines the functionality of a reduce and of a scatter operation

Example – Global Sum

- Sum of all the **x** values is placed in **result** only on processor 0

```
MPI_Reduce(&x,&result,1, MPI_INTEGER,
   MPI_SUM, 0, MPI_COMM_WORLD)
```

Predefined Reduction Operations

MPI Name	Function
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Sum
MPI_PROD	Product
MPI_LAND	Logical AND
MPI_BAND	Bitwise AND
MPI_LOR	Logical OR
MPI_BOR	Bitwise OR
MPI_LXOR	Logical exclusive OR
MPI_BXOR	Bitwise exclusive OR
MPI_MAXLOC	Maximum and location
MPI_MINLOC	Minimum and location

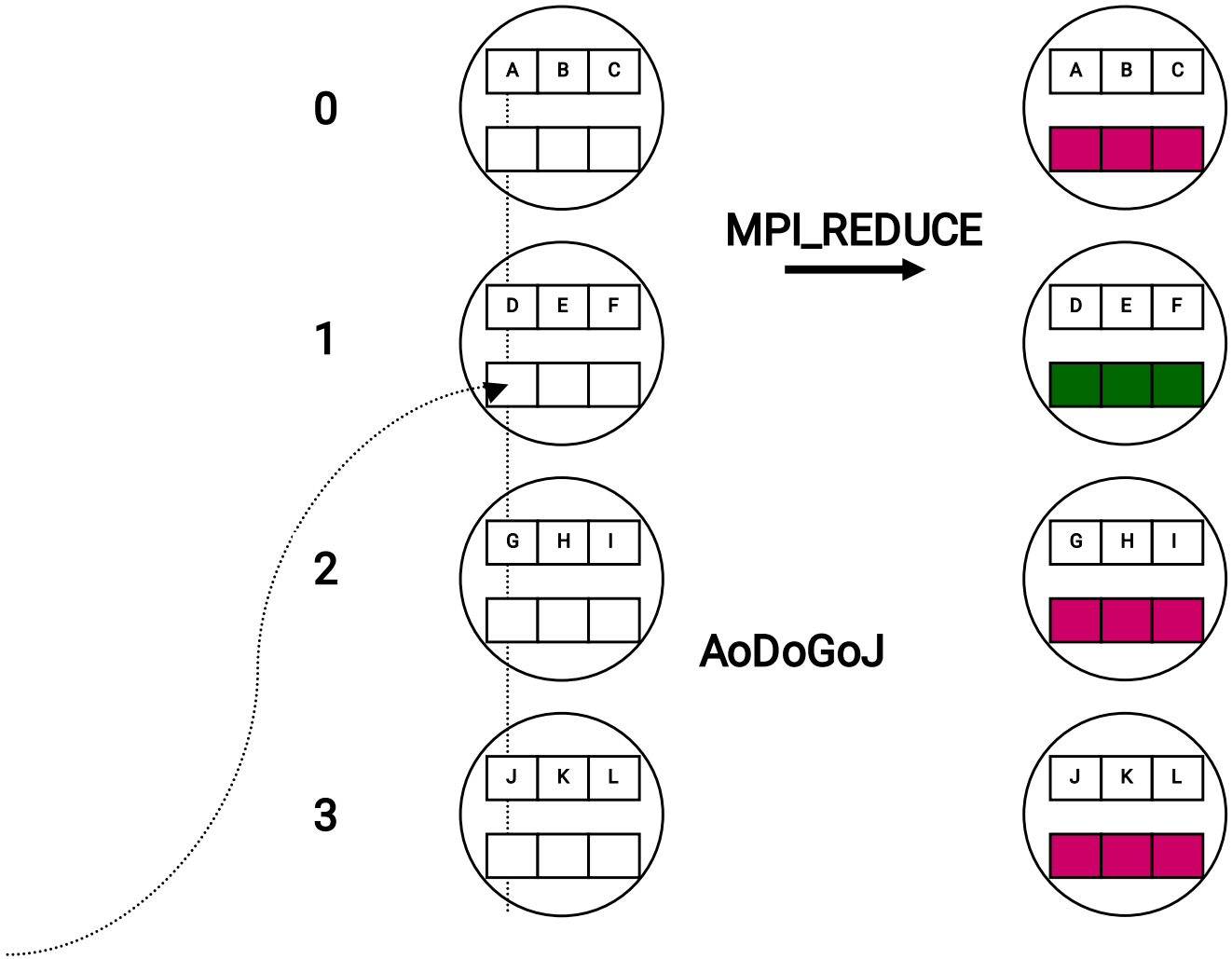
Sample Program

```
#include <mpi.h>
/* Run with 16 processes */
void main (int argc, char *argv[])
{
   int rank;
   struct {
     double value;
     int rank;
   } in, out;
   int root;
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD,&rank);
   in.value=rank+1;
   in.rank=rank;
   root=7;
   MPI_Reduce(&in, &out, 1, MPI_DOUBLE_INT, MPI_MAXLOC, root,
                                    MPI_COMM_WORLD);
   if(rank==root) printf("PE:%d max=%lf at rank %d\n", rank,
                                 out.value, out.rank);
   MPI_Reduce(&in, &out, 1, MPI_DOUBLE_INT, MPI_MINLOC, root,
                                    MPI_COMM_WORLD);
   if(rank==root) printf("PE:%d min=%lf at rank %d\n", rank,
                                 out.value, out.rank);
   MPI_Finalize();
}
```

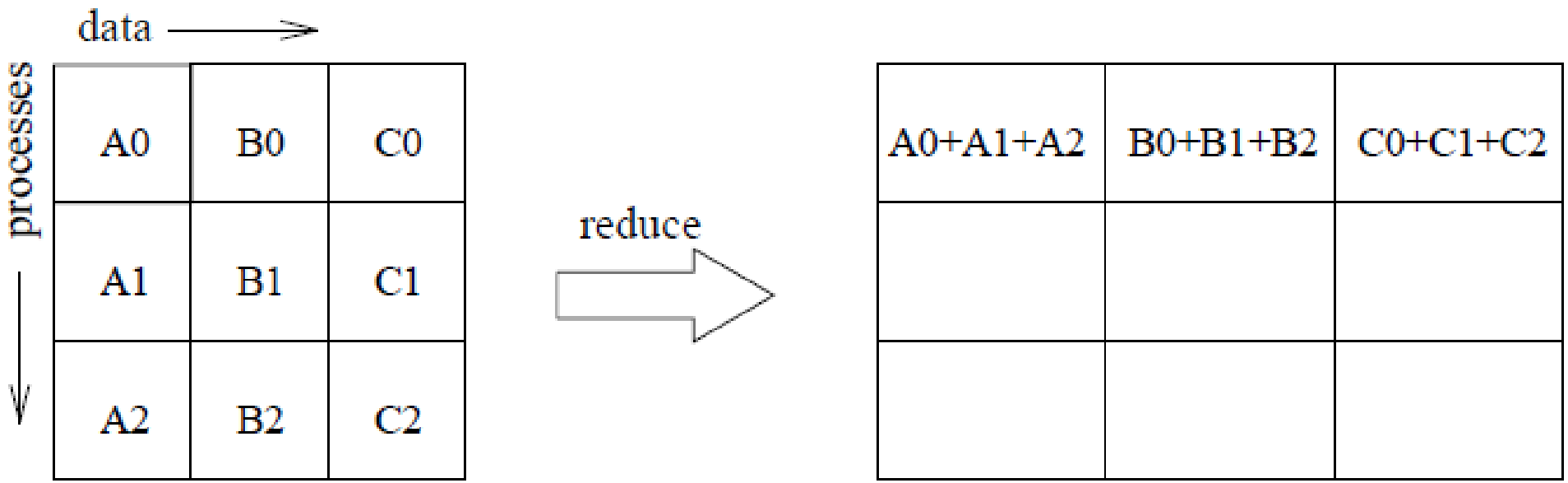
Variations of Reduce

- MPI_Allreduce -- no root process (all get results)
- MPI_Reduce_scatter -- multiple results are scattered
- MPI_Scan -- “parallel prefix”

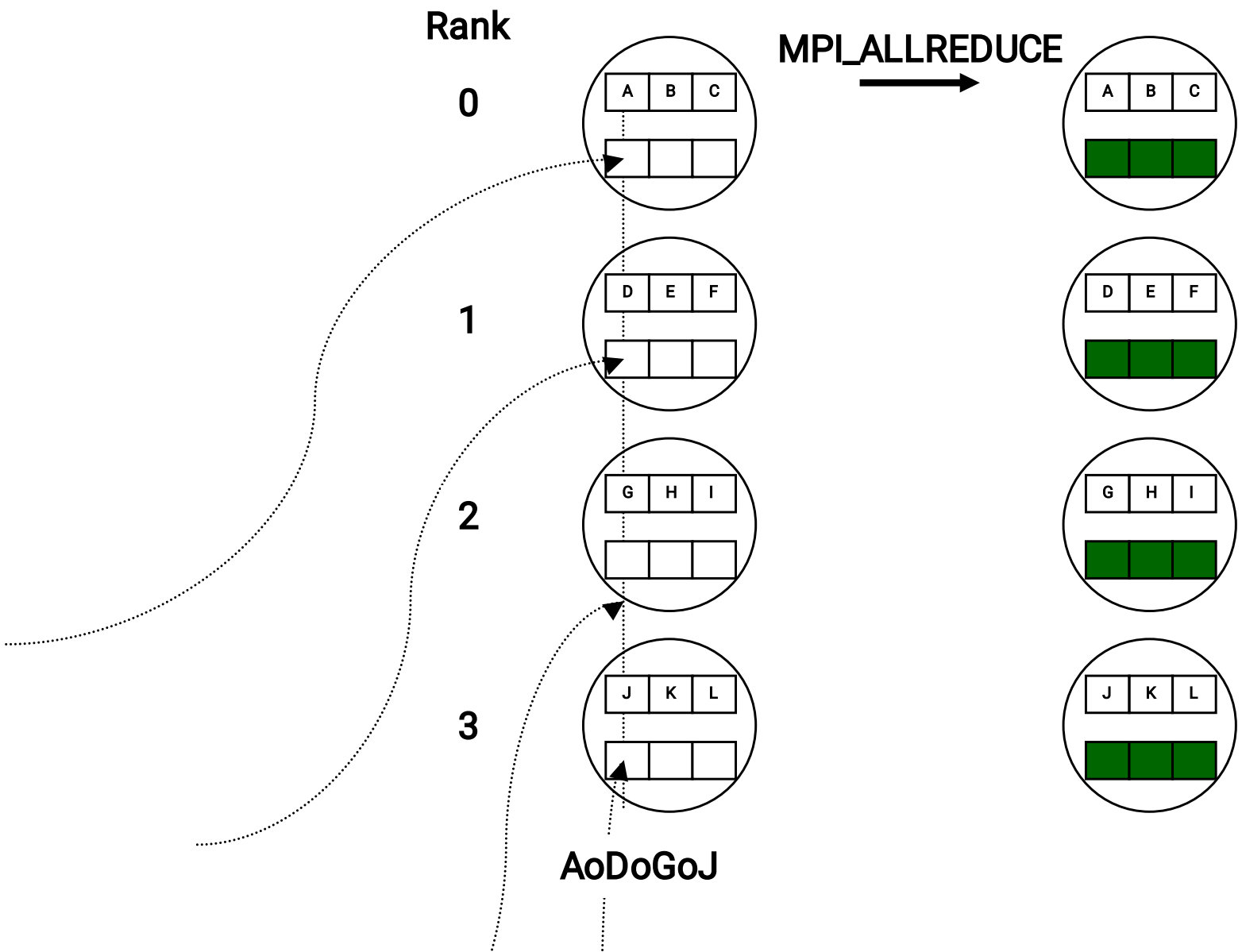
MPI_Reduce



MPI_Reduce


data
processes
A0
B0
C0
A1
B1
C1
A2
B2
C2
reduce
A0+A1+A2
B0+B1+B2
C0+C1+C2

MPI_Allreduce



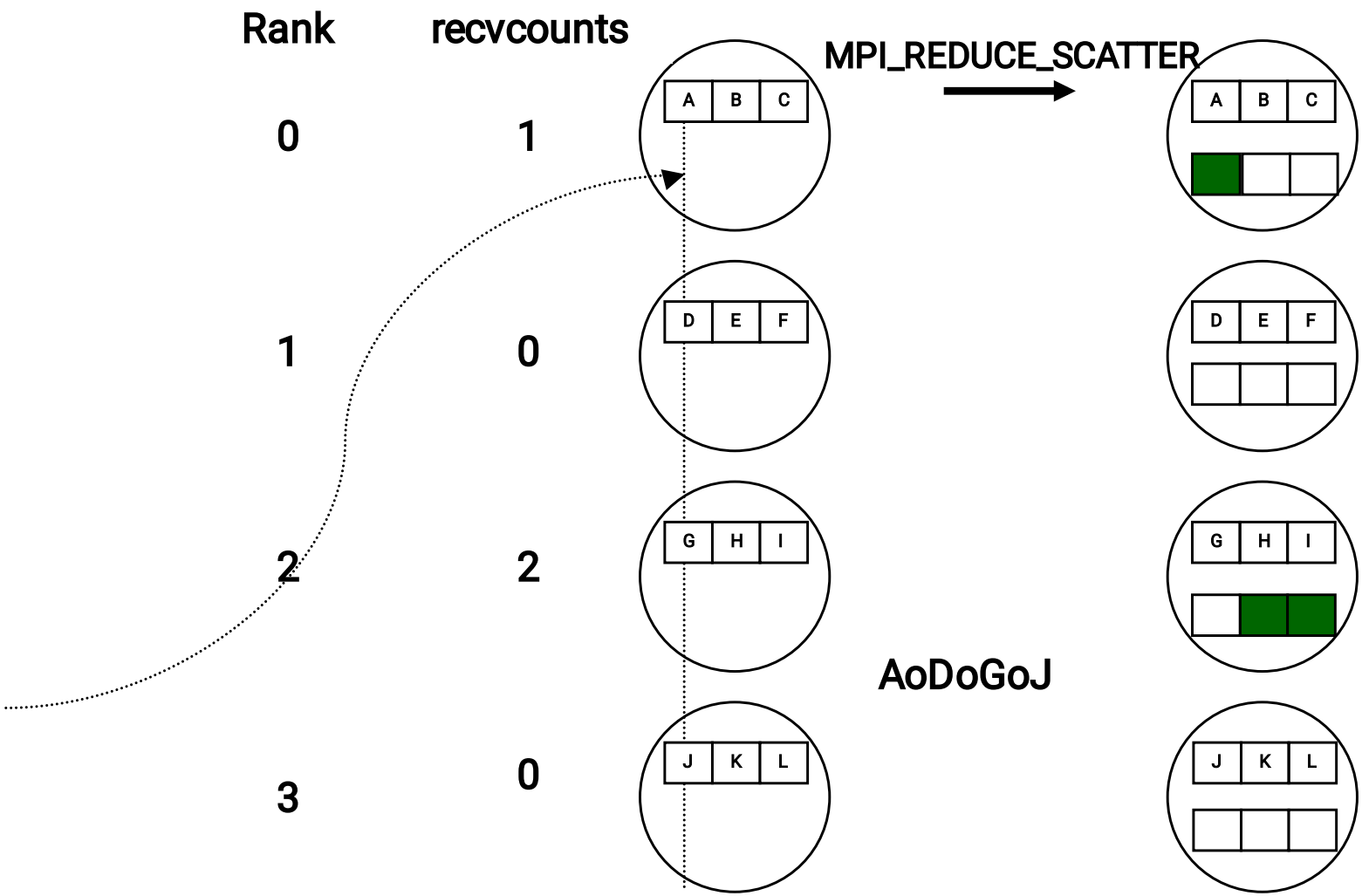
MPI_Allreduce

A0	B0	C0
A1	B1	C1
A2	B2	C2

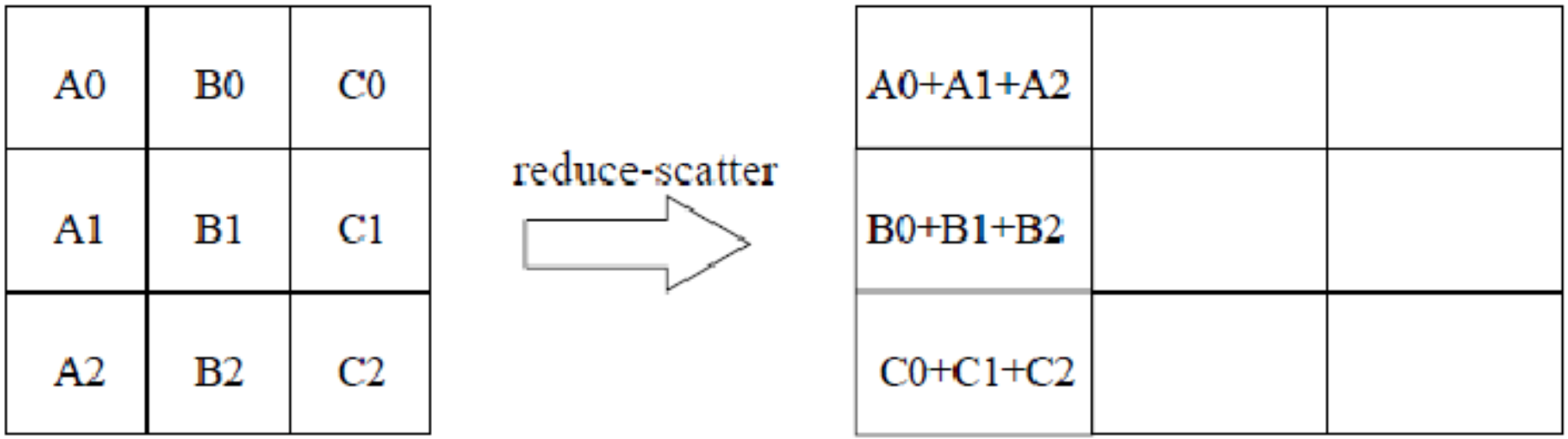
allreduce

A0+A1+A2	B0+B1+B2	C0+C1+C2
A0+A1+A2	B0+B1+B2	C0+C1+C2
A0+A1+A2	B0+B1+B2	C0+C1+C2

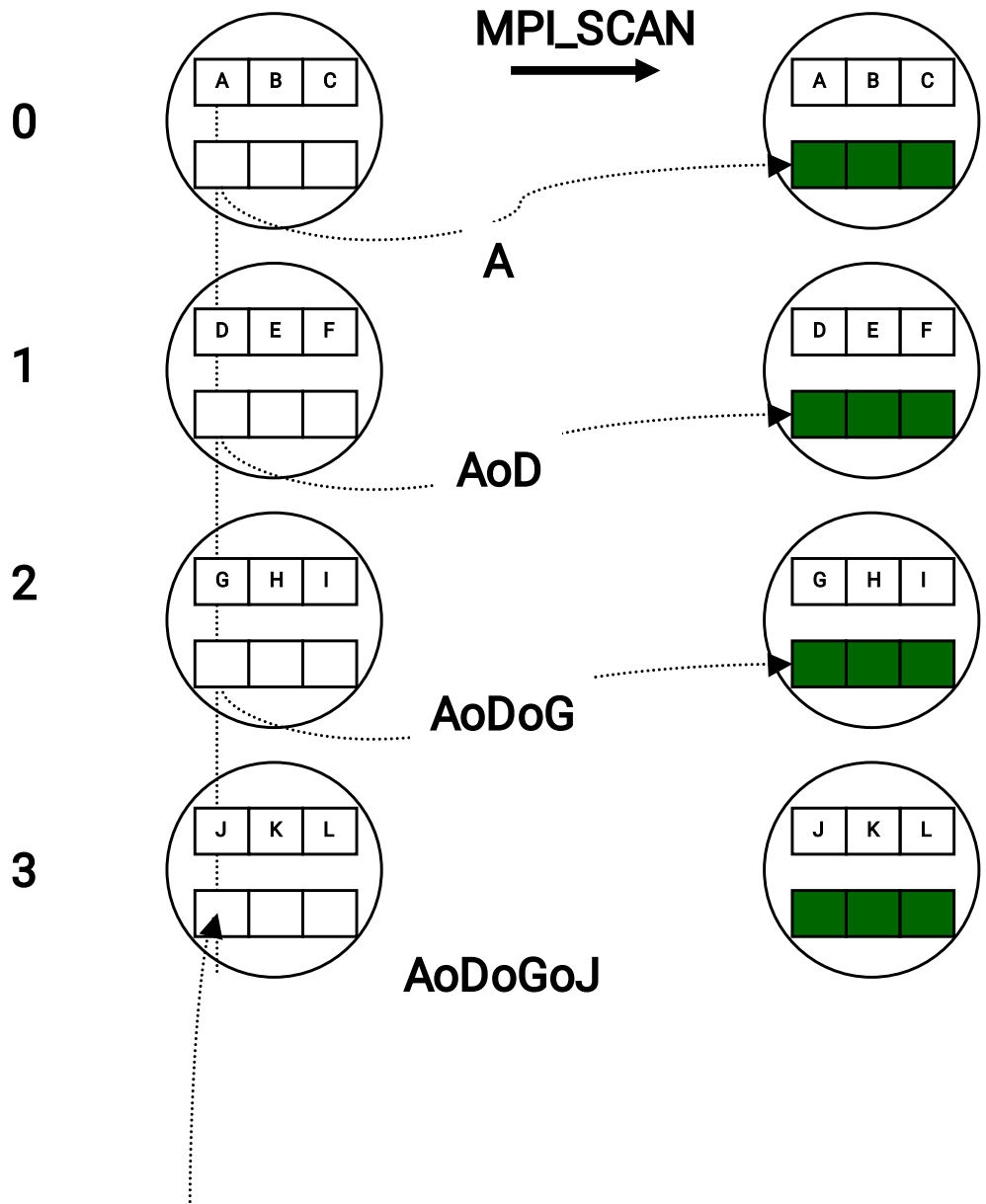
MPI_Reduce_scatter



MPI_Reduce_scatter



MPI_Scan



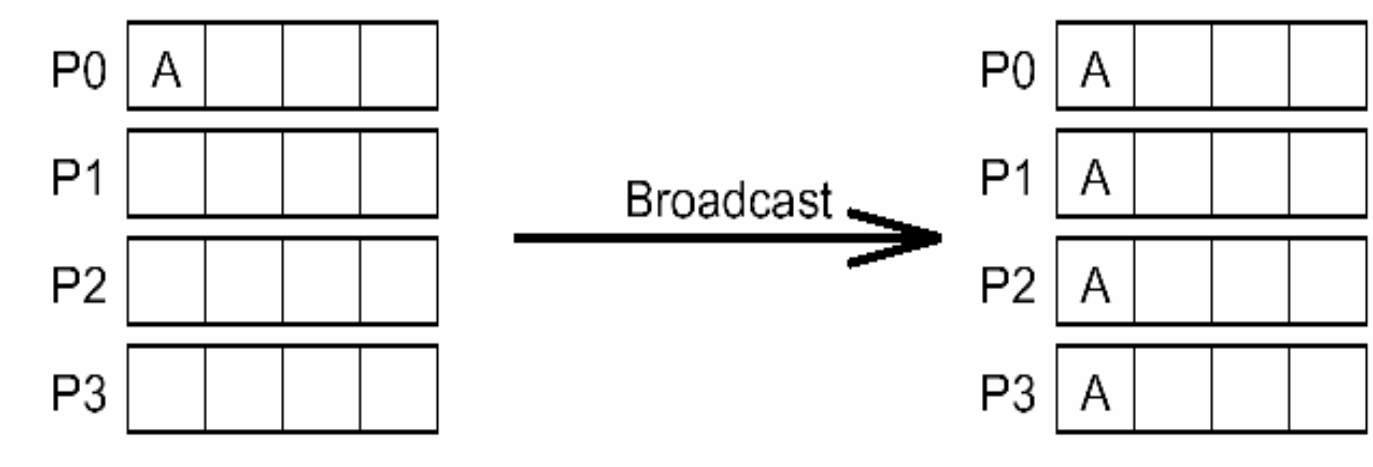
MPI_Scan

A0	B0	C0
A1	B1	C1
A2	B2	C2

scan →

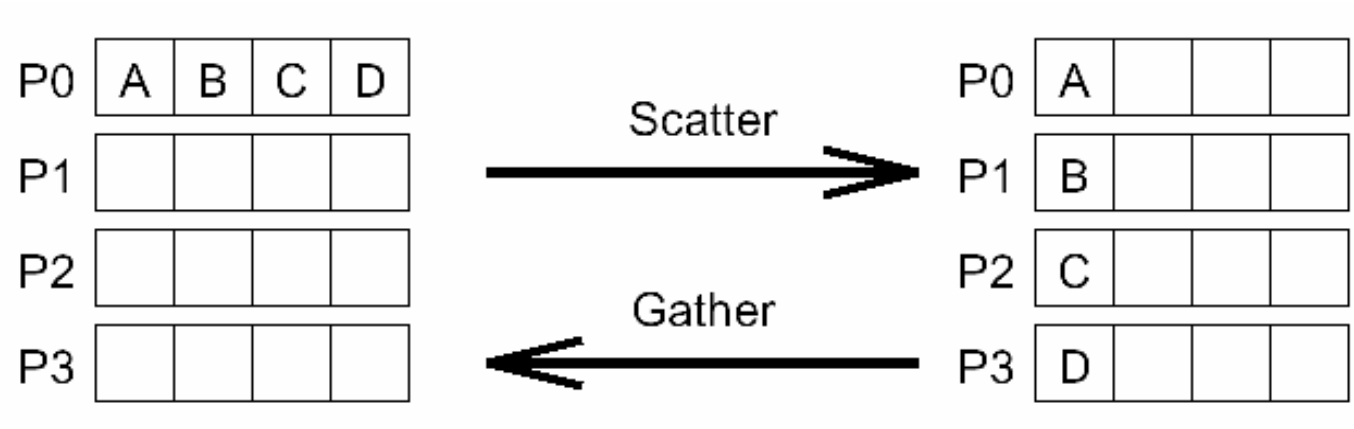
A0	B0	C0
A0+A1	B0+B1	C0+C1
A0+A1+A2	B0+B1+B2	C0+C1+C2

Revision



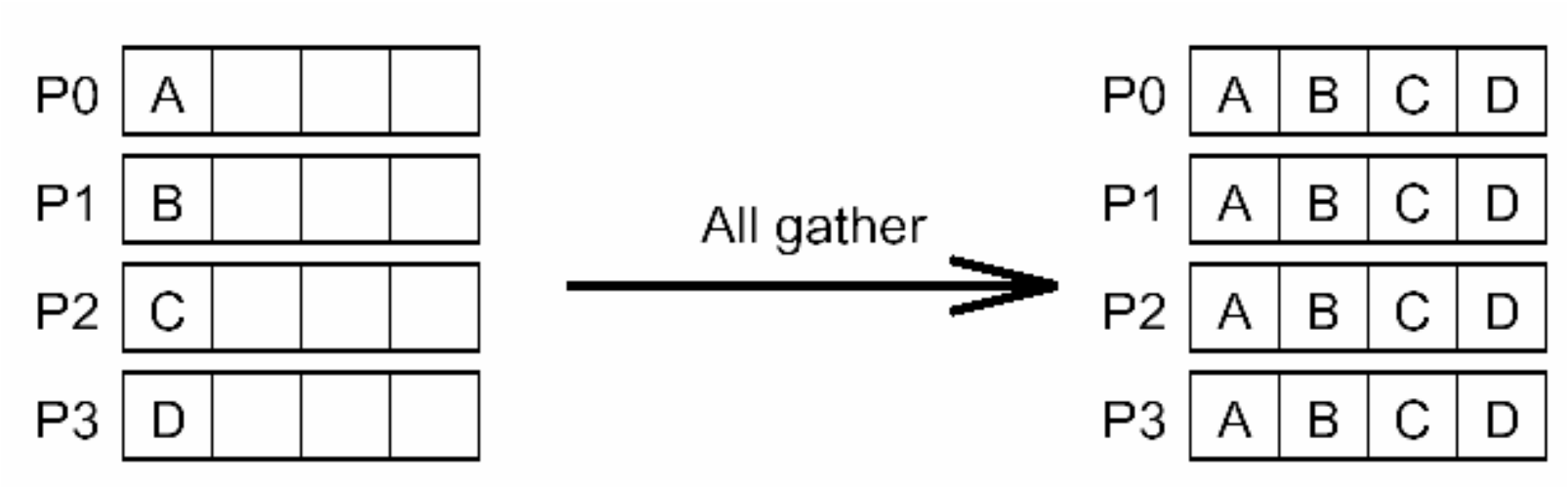

```
int MPI_Bcast(void *buf, int count, MPI_Datatype datatype,
        int source, MPI_Comm comm)
```

Revision



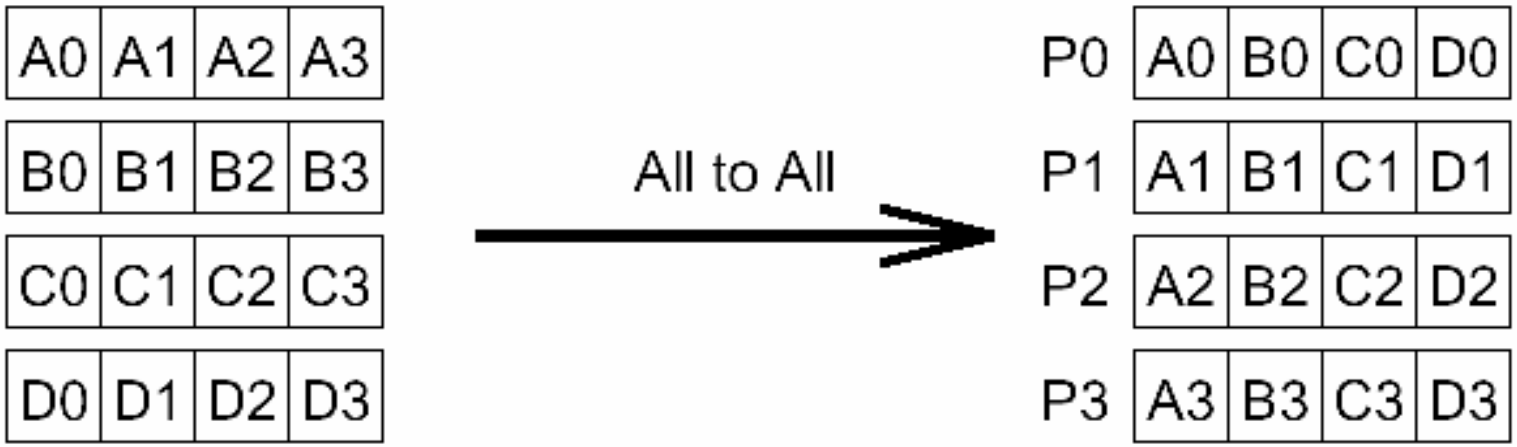

```
int MPI_Scatter(void *sendbuf, int sendcount,
        MPI_Datatype senddatatype, void *recvbuf, int recvcount,
        MPI_Datatype recvdatatype, int source, MPI_Comm comm)

 int MPI_Gather(void *sendbuf, int sendcount,
         MPI_Datatype senddatatype, void *recvbuf, int recvcount,
         MPI_Datatype recvdatatype, int target, MPI_Comm comm)
```

Revision



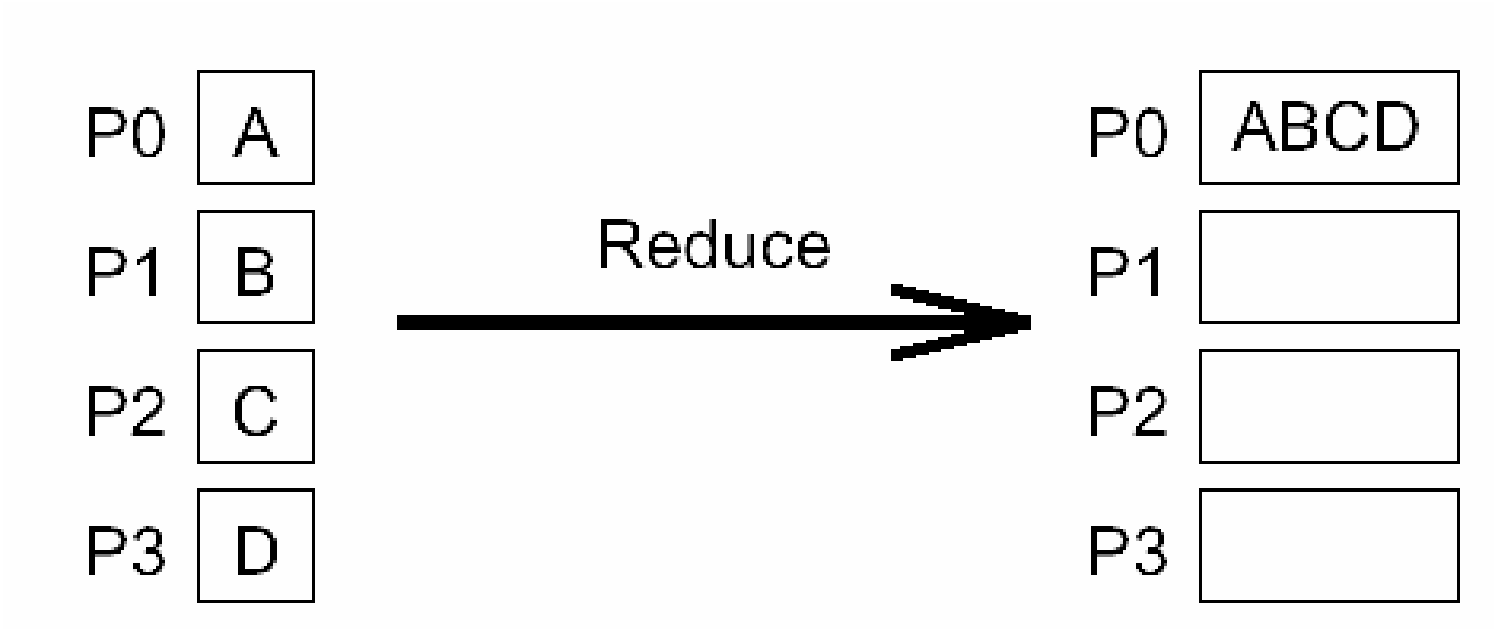

```
int MPI_Allgather(void *sendbuf, int sendcount,
        MPI_Datatype senddatatype, void *recvbuf, int recvcount,
        MPI_Datatype recvdatatype, MPI_Comm comm)
```

Revision



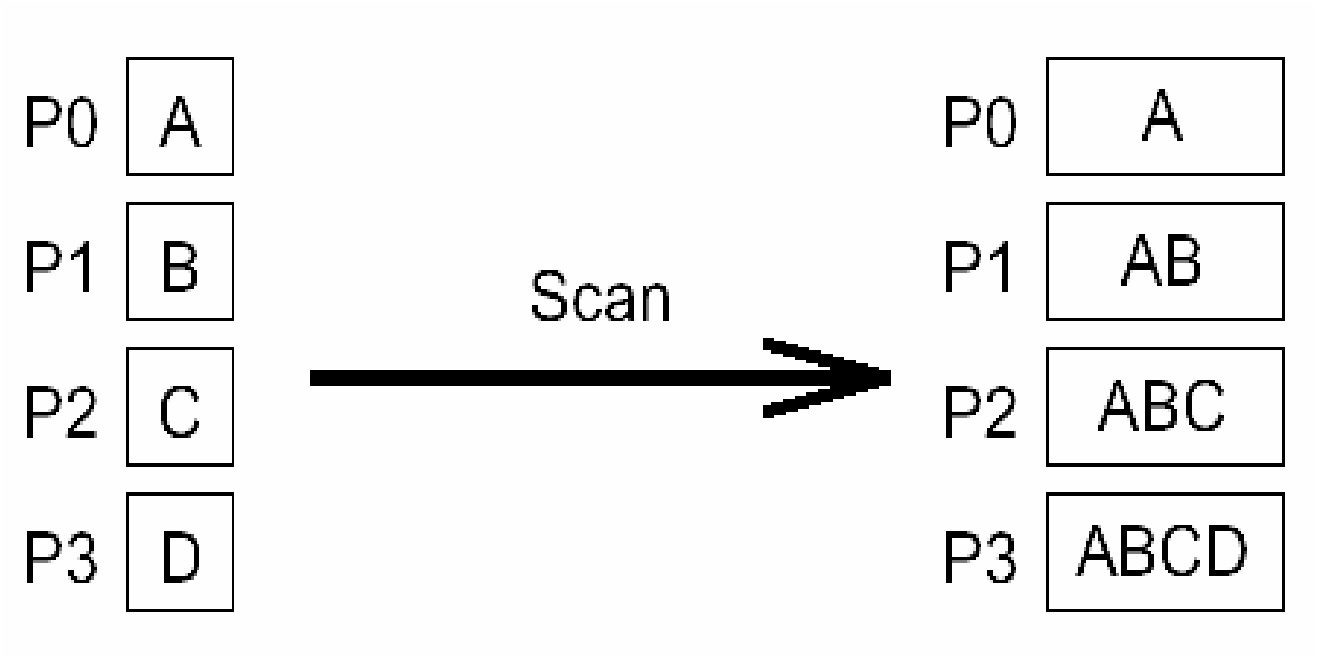

```
int MPI_Alltoall(void *sendbuf, int sendcount,
        MPI_Datatype senddatatype, void *recvbuf, int recvcount,
        MPI_Datatype recvdatatype, MPI_Comm comm)
```

Revision




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

Revision




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

MPI Functions

Operation	MPI Name
One-to-all broadcast	MPI_Bcast
All-to-one reduction	MPI_Reduce
All-to-all broadcast	MPI_Allgather
All-to-all reduction	MPI_Reduce_scatter
All-reduce	MPI_Allreduce
Gather	MPI_Gather
Scatter	MPI_Scatter
All-to-all personalized	MPI_Alltoall