# Gather

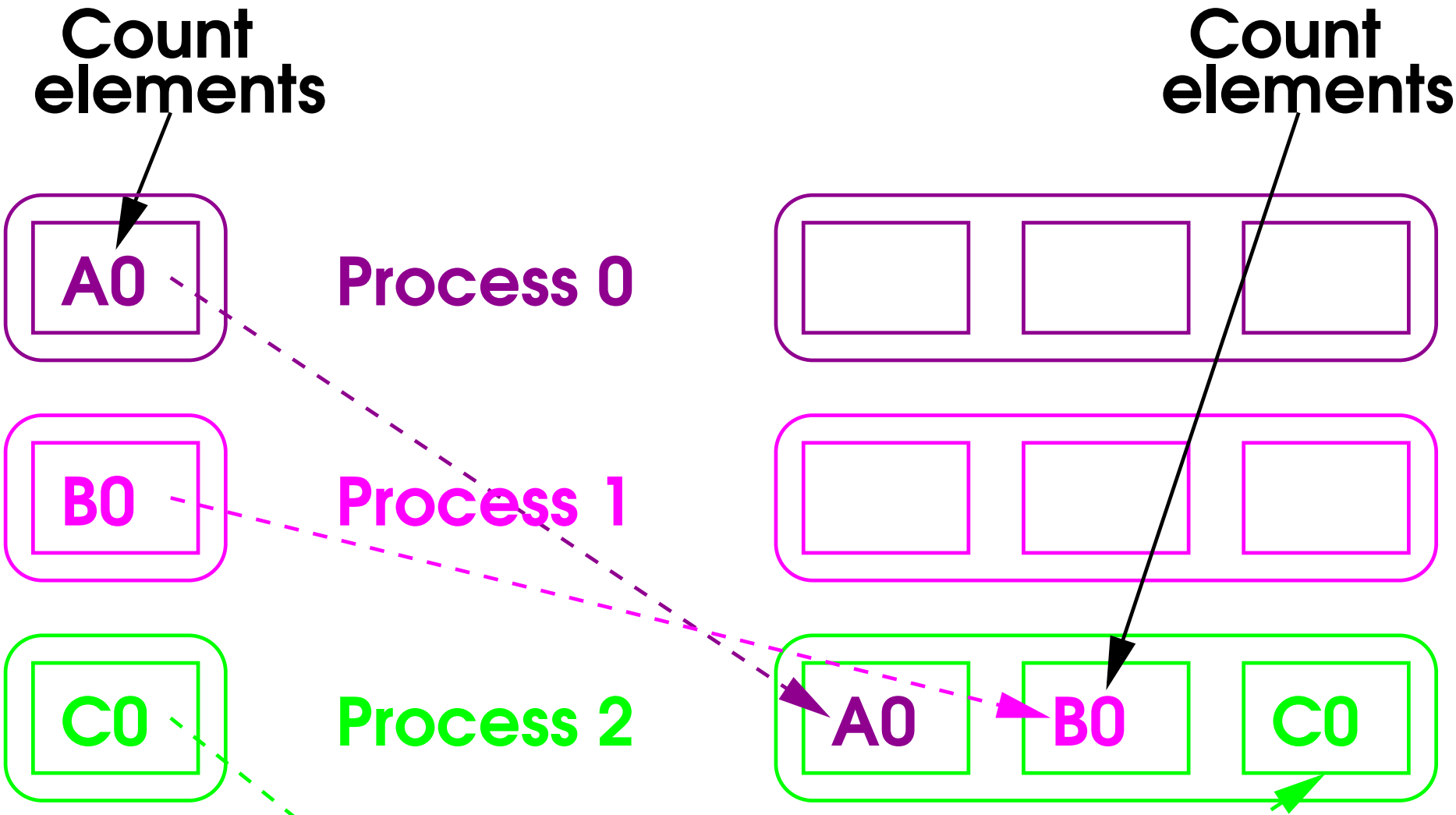
Gather is precisely the converse of scatter

- Just change MPI_Scatter to MPI_Gather
  And Scatter to Gather for C++, of course

Of course, the array sizes need changing
- It is the receive buffer that needs to be bigger

The send buffer is used on all processes
The receive buffer is used only on the root

# Gather


Count elements
Count elements
A0
Process 0
B0
Process 1
C0
Process 2
A0
B0
C0

# Allgather (1)

You can gather data and then broadcast it
The interface is very similar, with one difference

- This is now a symmetric operation

So has no argument specifying the root process

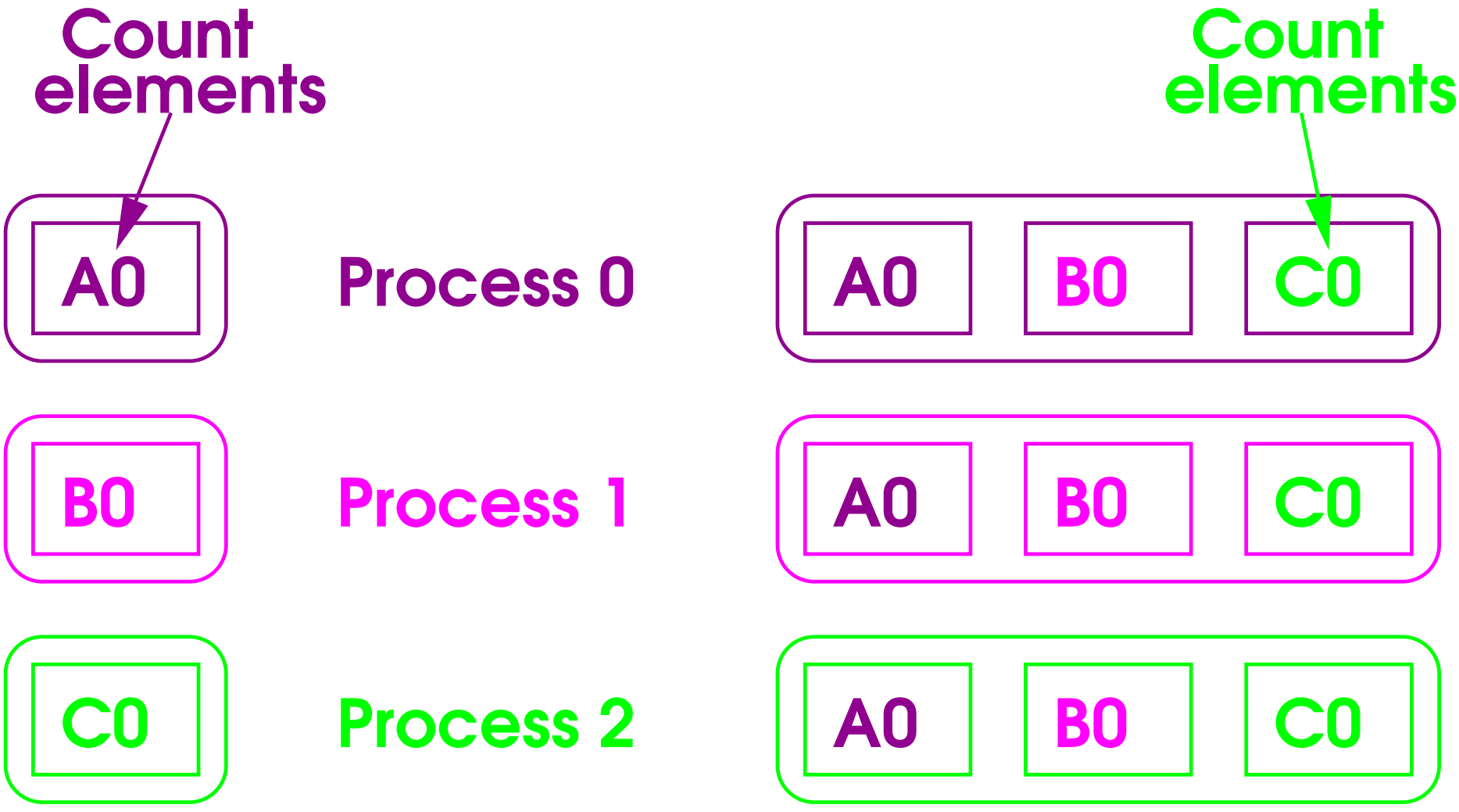
- Change MPI_Gather to MPI_Allgather

  And Gather to Allgather for C++

And remove the root process argument, of course

- The receive buffer is now used on all processes

# Allgather


Count elements
Count elements
A0
Process 0
A0
B0
C0
B0
Process 1
A0
B0
C0
C0
Process 2
A0
B0
C0

# Allgather (2)

Fortran example:

```
REAL(KIND=KIND(0.0D0)) ::       &
      sendbuf ( 100 ) , recvbuf ( 100 , 30 )
INTEGER :: error
CALL MPI_Allgather (       &
   sendbuf , 100 , MPI_DOUBLE_PRECISION ,     &
   recvbuf , 100 , MPI_DOUBLE_PRECISION ,     &
      MPI_COMM_WORLD , error )
```

# Allgather (3)

C example:

```
double sendbuf [ 100 ] , recvbuf [ 30 ] [ 100 ] ;
int error ;
error = MPI_Allgather (
        sendbuf , 100 , MPI_DOUBLE ,
        recvbuf , 100 , MPI_DOUBLE ,
        MPI_COMM_WORLD )
```

C++ example:

```
double sendbuf [ 100 ] , recvbuf [ 30 ] [ 100 ] ;
MPI::COMM_WORLD . Allgather (
        sendbuf , 100 , MPI::DOUBLE ,
        recvbuf , 100 , MPI::DOUBLE )
```

# Alltoall

You can do a composite gather/scatter operation
Essentially the same interface as MPI_Allgather

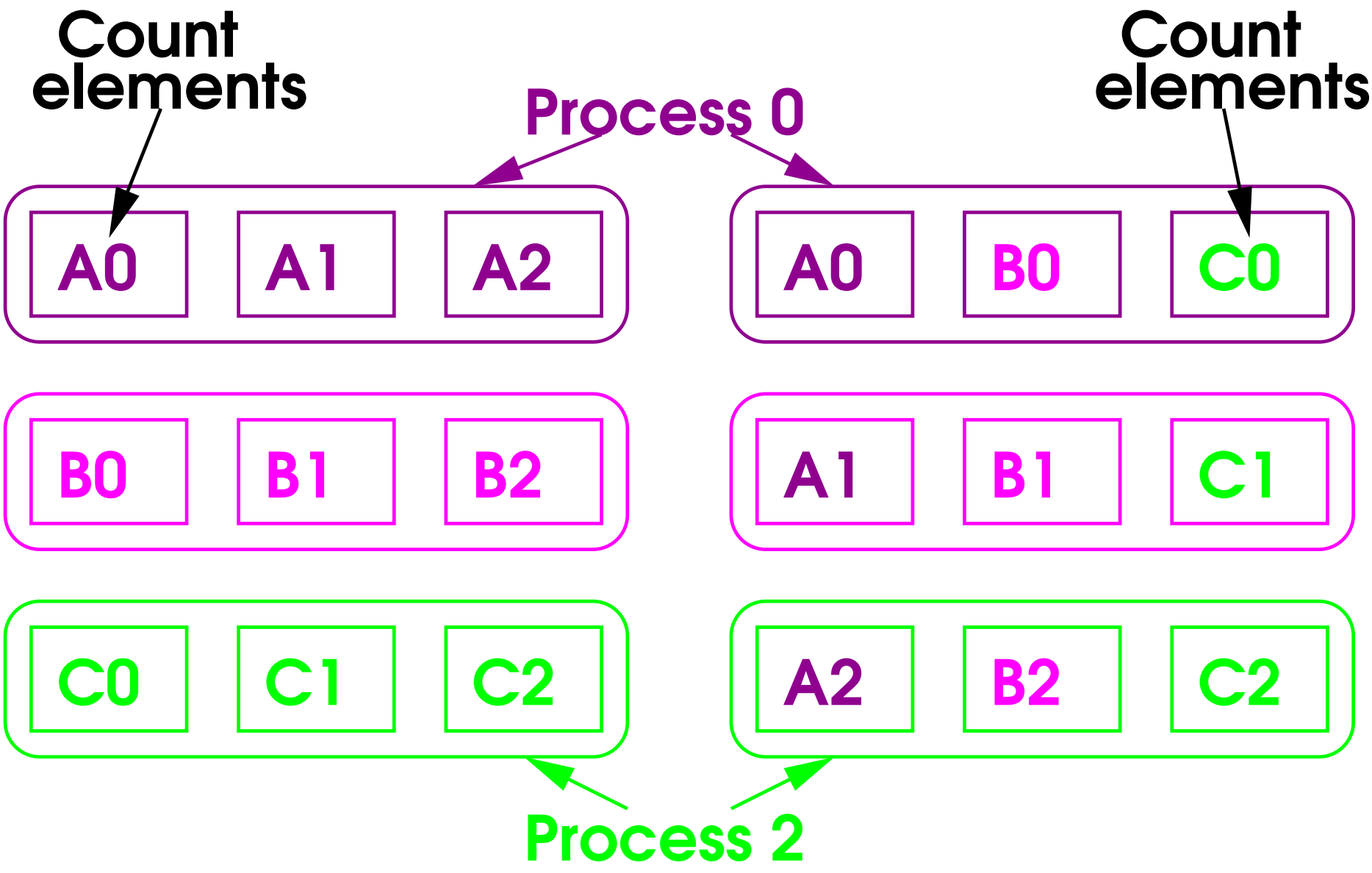
- Just change MPI_Allgather to MPI_Alltoall
  And Allgather to Alltoall for C++

- Now, both buffers need to be bigger

Think of this as a sort of parallel transpose
Used when implementing matrix transpose

- It's very powerful – a key for performance

# Alltoall


Count elements
Process 0
Count elements
A0
A1
A2
A0
B0
C0
B0
B1
B2
A1
B1
C1
C0
C1
C2
A2
B2
C2
Process 2