# Collective communications

## Global Reduction Communications

What are these communications ?

Communications that involve computation operation on the data which is transmitted over the processes in the communicator

**Advantage:**

- collect all the date across the communicator
- reduce multiple data to a single one
- perform desired operation with the data
- store on root or share date over the processes

### MPI_Reduce

In C:

```
MPI_Reduce(void *sendbuffer, void *recvbuffer, int count,
    MPI_Datatype data_type, MPI_Op operation, int root, MPI_Comm comm);
```

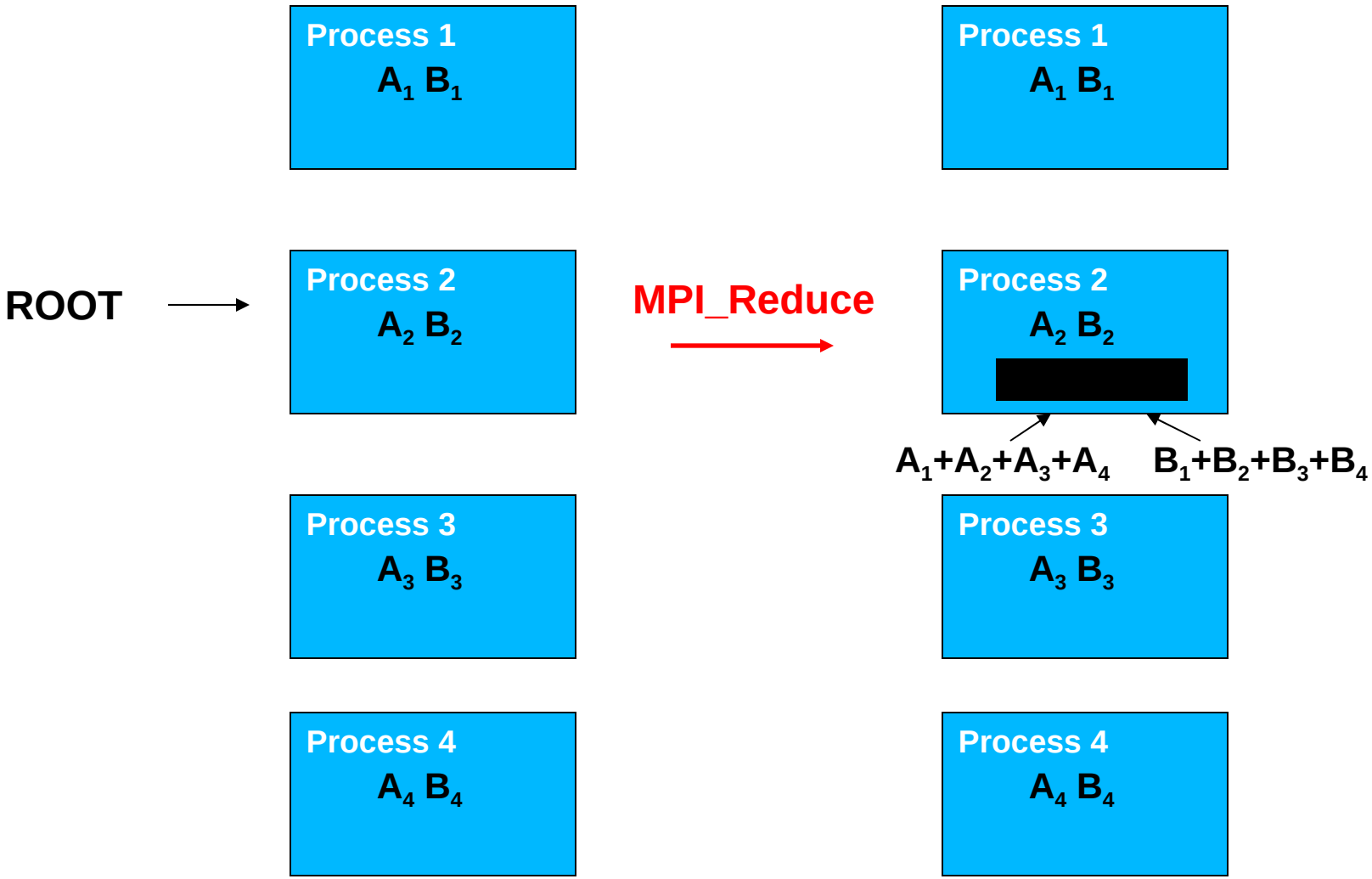
In Fortran:

```
call MPI_Reduce(sendbuffer, recvbuffer, count,
```

# Collective communications

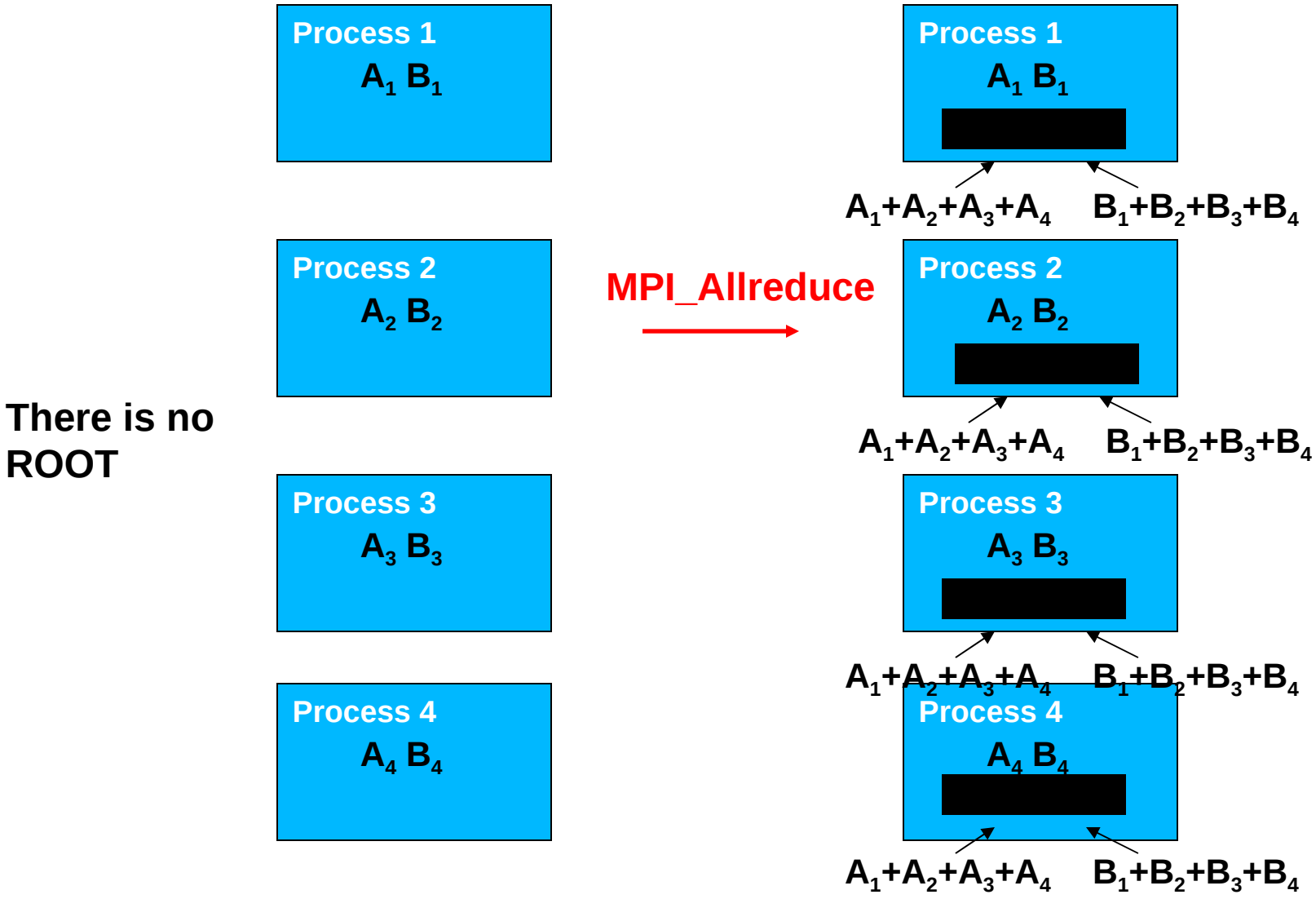
**MPI_Reduce :**

**OPERATION = SUMMATION**



# Collective communications

**MPI_Allreduce :** **OPERATION = SUMMATION**



# Collective communications

## Reduction operations

| MPI Name | Operation |
|---|---|
| MPI_SUM | summation |
| MPI_PROD | product |
| MPI_MIN | minimum |
| MPI_MAX | maximum |
| MPI_LAND | logical AND |
| MPI_LOR | logical OR |

**Other useful reduction subroutines:**

MPI_REDUCE_SCATTER
MPI_SCAN

**Other useful collective subroutines:**

MPI_GATHER, MPI_ALLGATHER
MPI_ALLTOALL

# Collective communications

## Example demonstrating MPI_Reduce operation: Calculation of PI

Look for "**calculate_pi.c**"

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
int main( int argc, char *argv[] )
{
    int n, myid, numprocs, i;
    double PI25DT = 3.14159265358979323846264 3;
    double mypi, pi, h, sum, x;
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    while (1) {
        if (myid == 0) {
            printf("Enter the number of intervals: (0 quits) "); fflush(stdout);
            scanf("%d",&n);
        }
        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
        if (n == 0)
            break;
```

# Collective communications

## Example demonstrating MPI_Reduce operation: Calculation of PI

Look for "**calculate_pi.c**"

```
else {
        h   = 1.0 / (double) n;
        sum = 0.0;
        for (i = myid + 1; i <= n; i += numprocs) {
            x = h * ((double)i - 0.5);
            sum += (4.0 / (1.0 + x*x));
        }
        mypi = h * sum;
        MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
                MPI_COMM_WORLD);
        if (myid == 0)
            printf("pi is approximately %.16f, Error is %.16f\n",
                pi, fabs(pi - PI25DT));
    }
  }
  MPI_Finalize();
  return 0;
}
```

# Collective communications

## Example demonstrating MPI_Reduce operation: Calculation of PI

Running "**calculate_pi.c**"

**mpirun -np 4 ./calculate_pi.exe**

Output:

```
Enter the number of intervals: (0 quits) 100
pi is approximately 3.1416009869231249, Error is 0.0000083333333318
Enter the number of intervals: (0 quits) 500
pi is approximately 3.1415929869231269, Error is 0.0000003333333338
Enter the number of intervals: (0 quits) 1000
pi is approximately 3.1415927369231262, Error is 0.0000000833333331
```