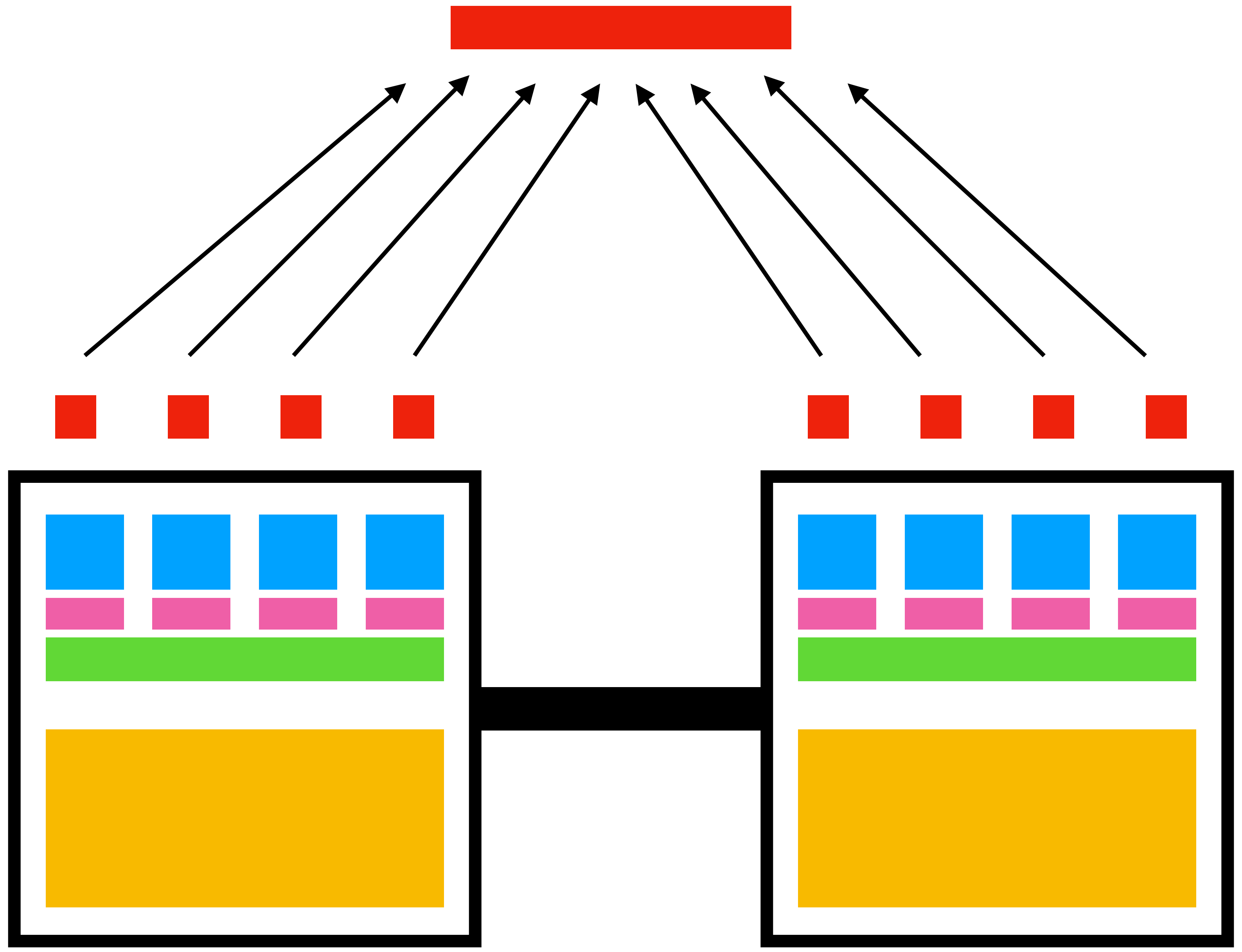
Introduction to Parallel Processing

Lecture 4 : Collective Communication Introduction

Professor Amanda Bienz

Collective Operations

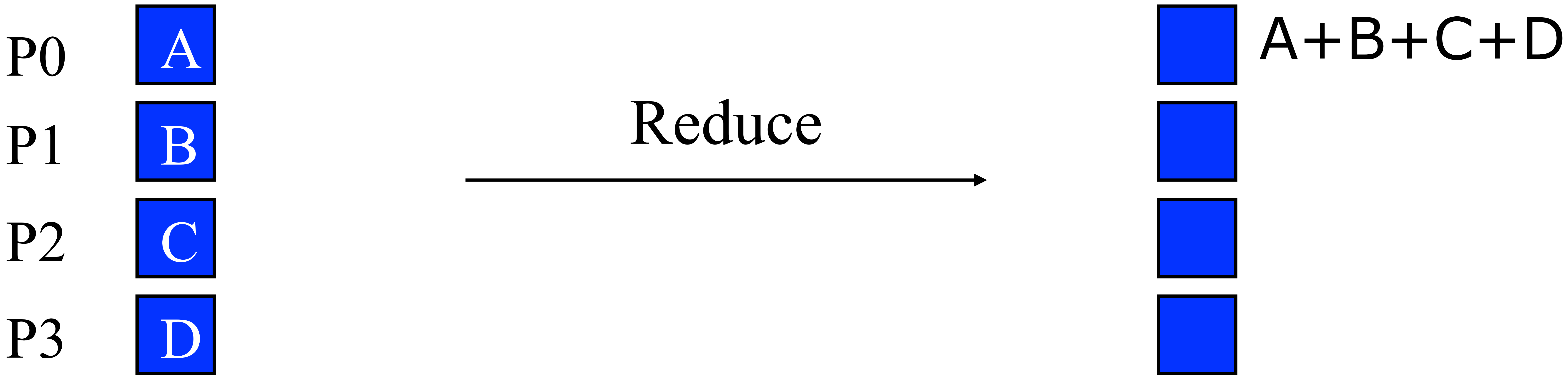


- Consider the case where each process has a portion of data, and we want to combine all of this data in some manner

- E.G. Each process holds a single number, and we want to find the sum of all of these numbers

- How do we handle this?

Reduction



MPI_Reduce

- Reduction to a single processor (only one process will hold final result)

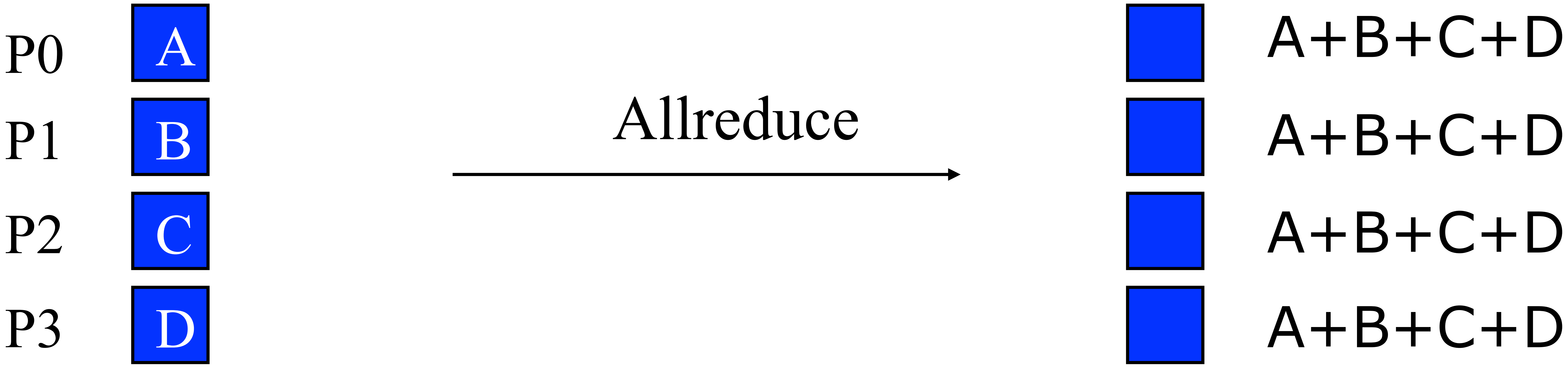
-
  ```
  MPI_Reduce(const void* sendbuf,   // Buffer containing original data
          void* recvbuf,            // Buffer to hold reduced data
          int count,                // Buffer size
          MPI_Datatype datatype,    // Type of data (e.g. MPI_INT)
          MPI_Op op,                // Operation (MPI_SUM, MPI_MAX,…)
          int root,                 // Process to hold reduced value
          MPI_Comm comm)            // MPI_COMM_WORLD
  ```

- **MPI_Reduce(&sum, &global_sum, 1, MPI_DOUBLE, MPI_SUM,**
  0, MPI_COMM_WORLD);

When is MPI_Reduce used?

- When do we want to reduce a variable only to a master ‘root’ process?
- Often, if something should be printed out
- For example, if each process times a method, we only want one process to print out the maximum time
- Similarly, an MPI program that calculates and prints a sum

Reduction



P0 A

P1 B

P2 C

P3 D

Allreduce

A+B+C+D

A+B+C+D

A+B+C+D

A+B+C+D

MPI_Allreduce

- Reduction to every process (all will hold final result)

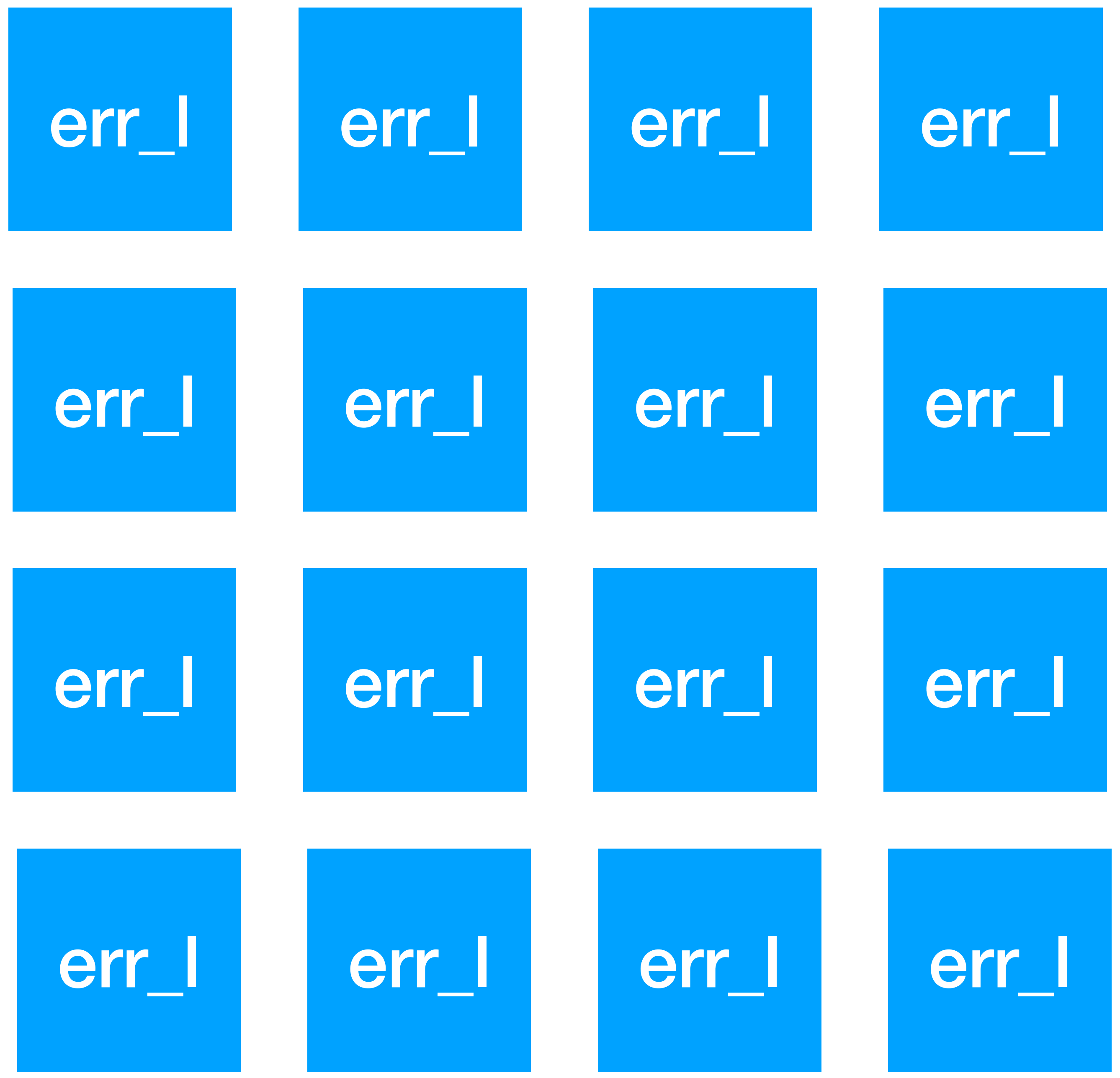
- MPI_Allreduce(const void* sendbuf, // Buffer containing original data
  void* recvbuf, // Buffer to hold reduced data
  int count, // Buffer size
  MPI_Datatype datatype, // Type of data (e.g. MPI_INT)
  MPI_Op op, // Operation (MPI_SUM, MPI_MAX,…)

  MPI_Comm comm) // MPI_COMM_WORLD

- **MPI_Allreduce(&sum, &global_sum, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);**

When is MPI_Allreduce used?

- Iterative Methods : Converging to solution and want to stop when close enough



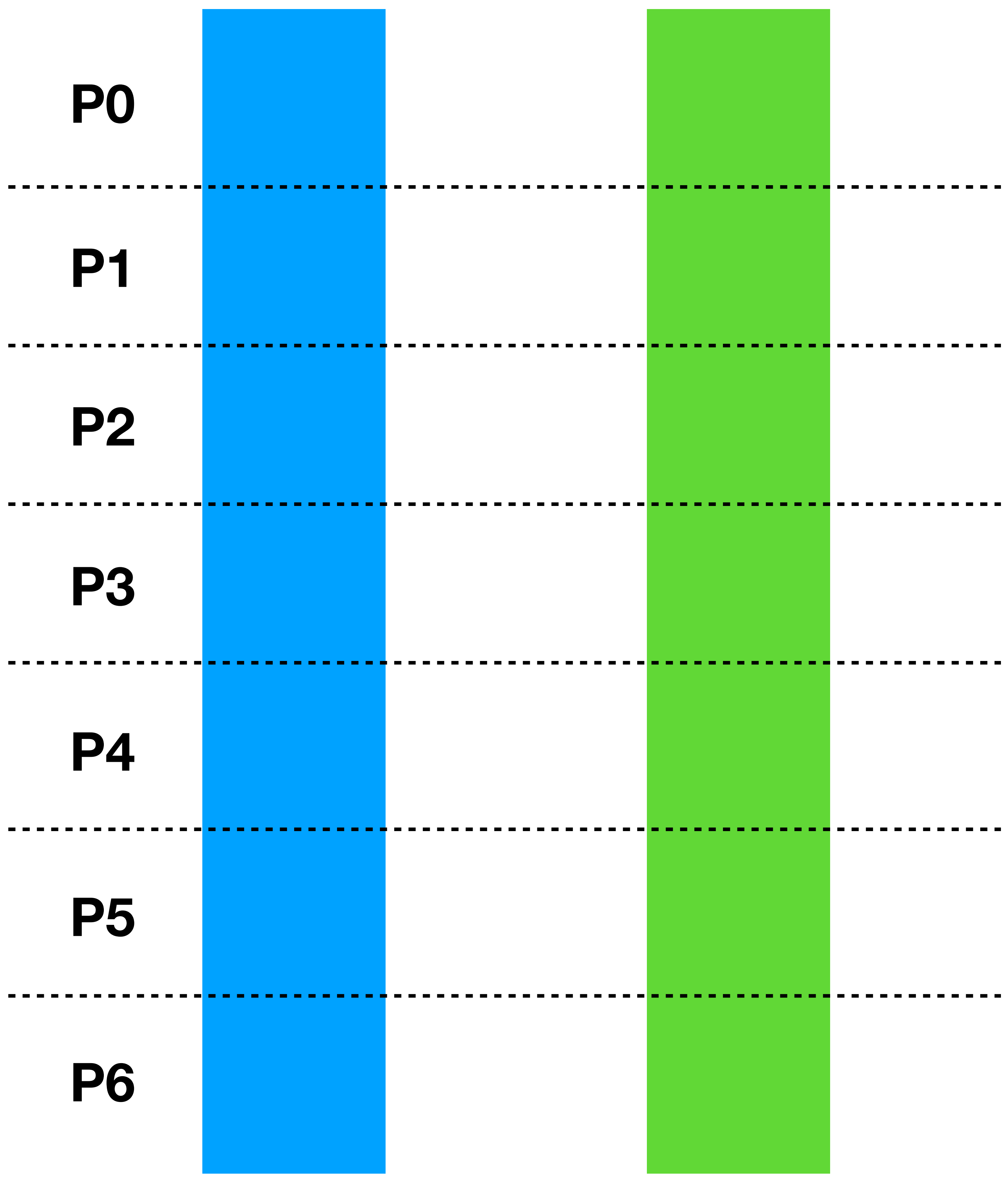
err_l	err_l	err_l	err_l
err_l	err_l	err_l	err_l
err_l	err_l	err_l	err_l
err_l	err_l	err_l	err_l

16 Process, each hold a local error

When is MPI_Allreduce used?

- Dot product of two vectors:

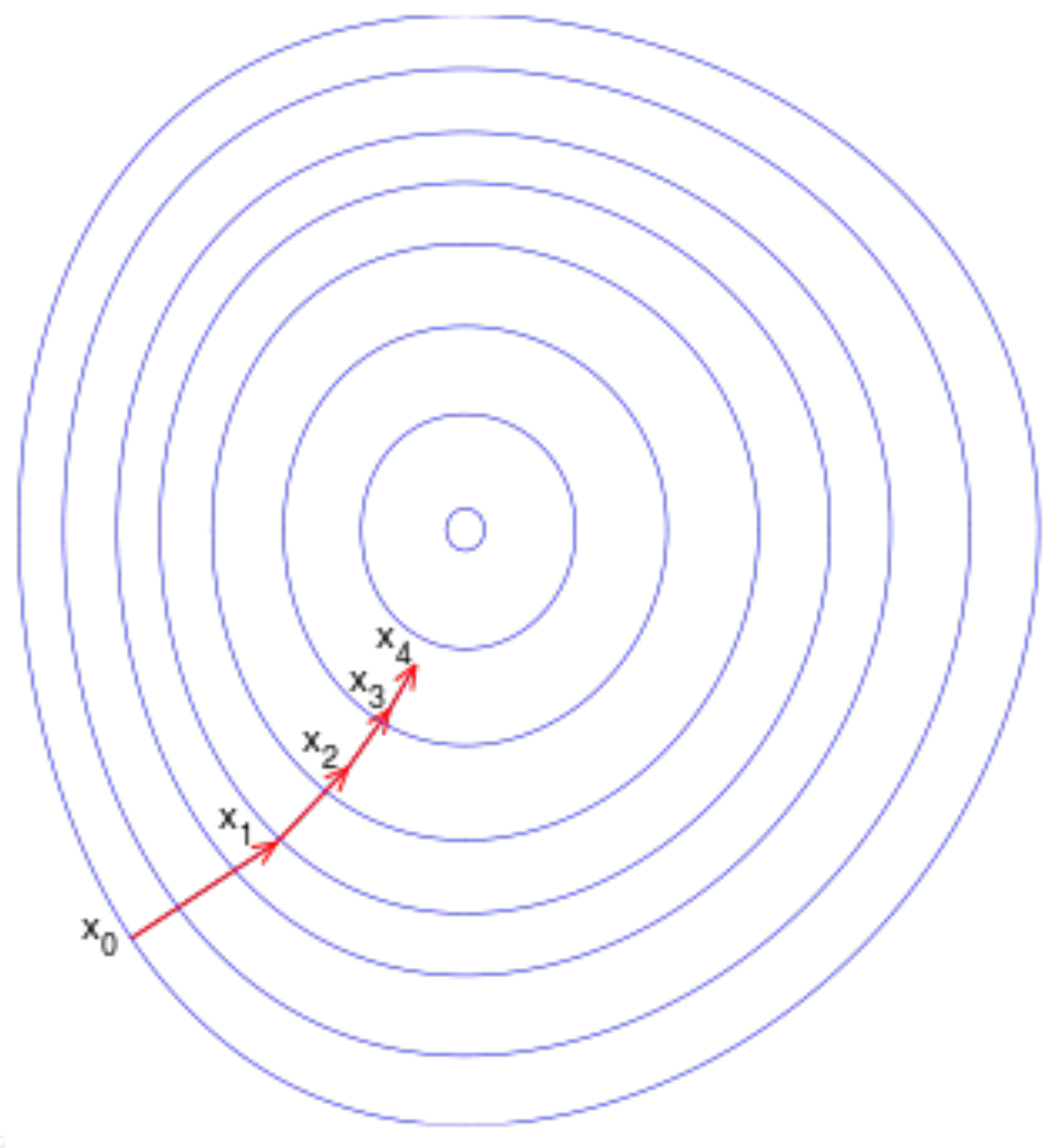
```
for (i = 0 to n)
    sum += x[i] * y[i]
```



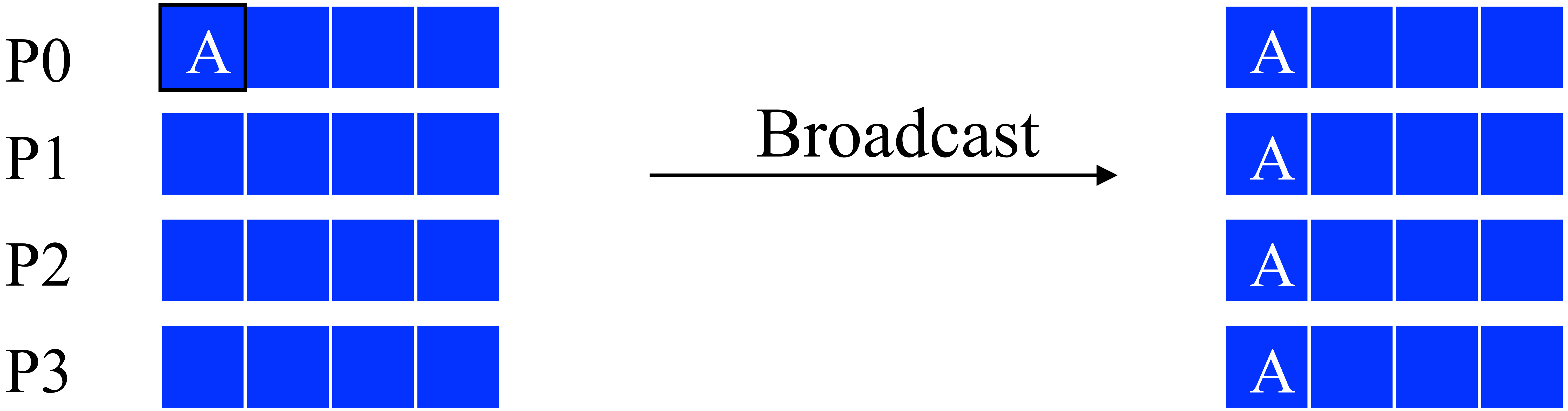
Two vectors, split across 7 processors

When is MPI_Allreduce used?

- Training neural networks
- Stochastic Gradient Descent : finds gradients, representing direction to go towards the minimum
- Tons of data spread across processors, each resulting in gradients
- Often have hundreds of millions of gradients, or directions, that need reduced



Broadcast



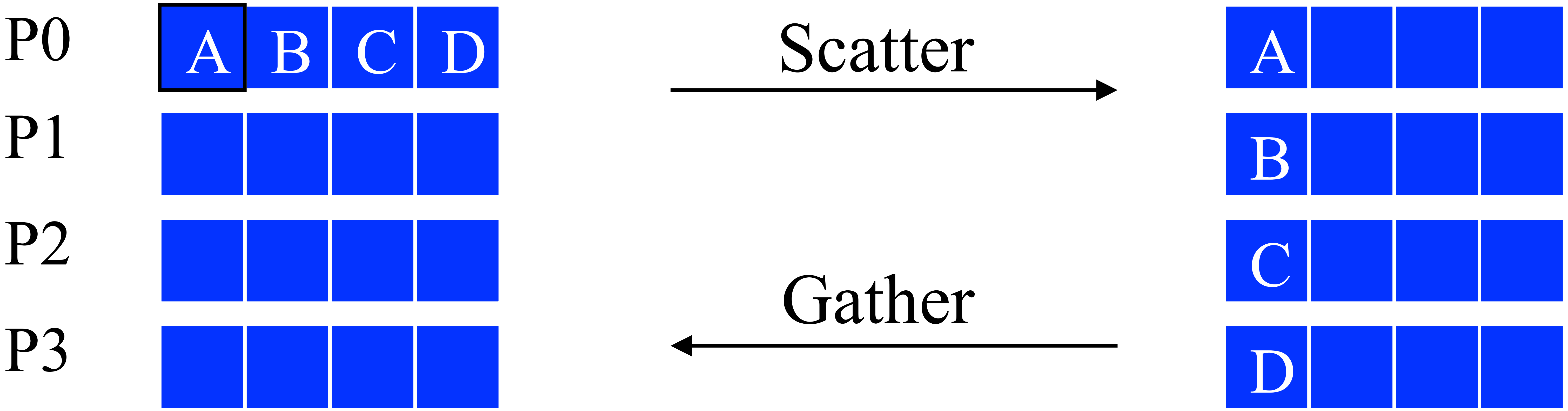
Broadcast

- Send data from one process to all other processes

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

-
  ```
  int size = rand();
  MPI_Bcast(&size, 1, MPI_INT, 0, MPI_COMM_WORLD);
  ```

Scatter and Gather



Scatter

- Sends data from one process to all other processes. Each process gets separate portion of data.

-
  ```
  MPI_Scatter(const void* sendbuf,
          int sendcount,                  // Number vals to send to each proc
          MPI_Datatype sendtype,
          void* recvbuf,
          int recvcount,                  // Number vals to recv
          MPI_Datatype recvtype,
          int root,
          MPI_Comm comm)
  ```

-
  ```
  int* vals = new vals[num_procs];
  int recv_val;
  MPI_Scatter(vals, 1, MPI_INT, &recv_val, 1, MPI_INT, 0, MPI_COMM_WORLD);
  ```

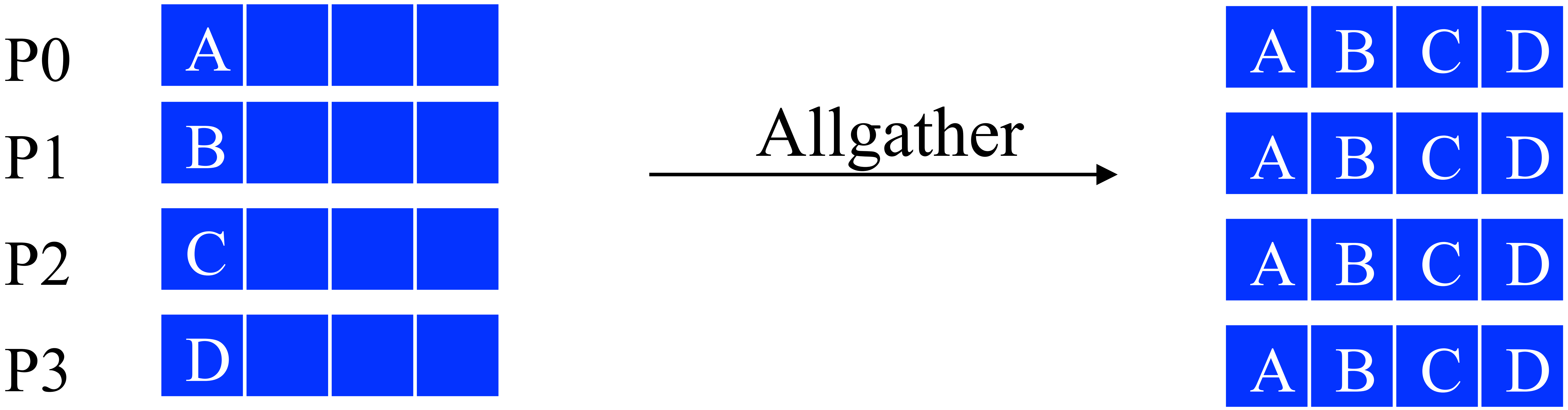
Gather

- Opposite of scatter, each process starts with some values and want to gather all of these values onto a single process

-
  ```
  MPI_Gather(const void* sendbuf,
          int sendcount,                // Number vals, send all to every proc
          MPI_Datatype sendtype,
          void* recvbuf,
          int recvcount,                // Number vals to recv from each proc
          MPI_Datatype recvtype,
          int root,
          MPI_Comm comm)
  ```

-
  ```
  int* vals = new vals[num_procs];
  int send_val = …;
  MPI_Gather(&send_val, 1, MPI_INT, vals, 1, MPI_INT, 0, MPI_COMM_WORLD);
  ```

Allgather



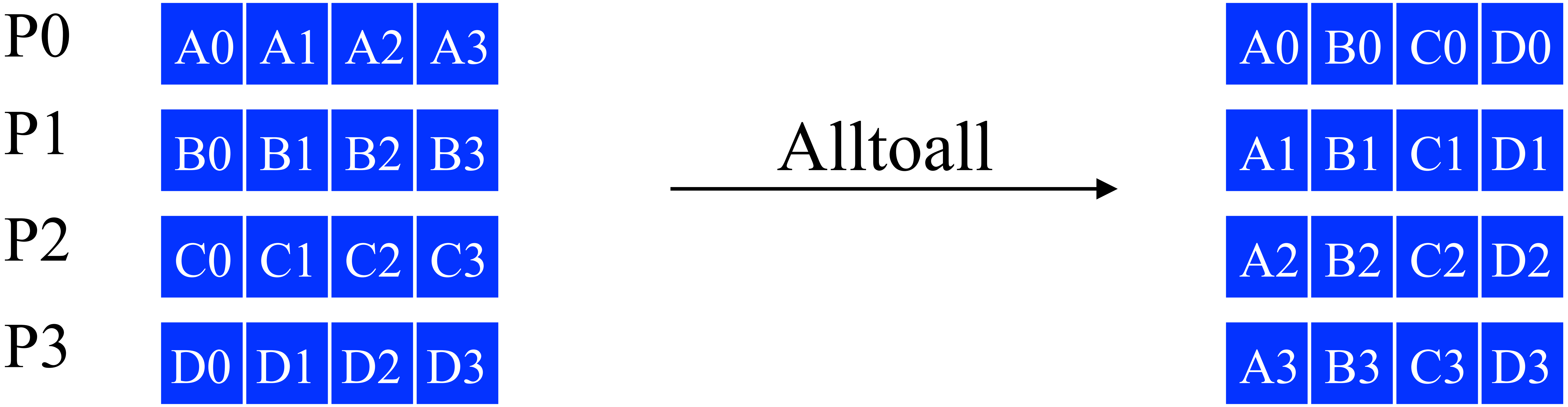
Allgather

- Same as gather, but have all data end up on every process

-
```
MPI_Allgather(const void* sendbuf,
        int sendcount,                  // Number vals, send all to every proc
        MPI_Datatype sendtype,
        void* recvbuf,
        int recvcount,                  // Number vals to recv from each proc
        MPI_Datatype recvtype,
        MPI_Comm comm)
```

-
```
int* vals = new vals[num_procs];
int send_val = …;
MPI_Allgather(&send_val, 1, MPI_INT, vals, 1, MPI_INT, MPI_COMM_WORLD);
```

All-to-all



P0	A0	A1	A2	A3
P1	B0	B1	B2	B3
P2	C0	C1	C2	C3
P3	D0	D1	D2	D3

Alltoall

A0	B0	C0	D0
A1	B1	C1	D1
A2	B2	C2	D2
A3	B3	C3	D3

All To All

- Similar to allgather, but instead of getting entire array, each process gets different part of data

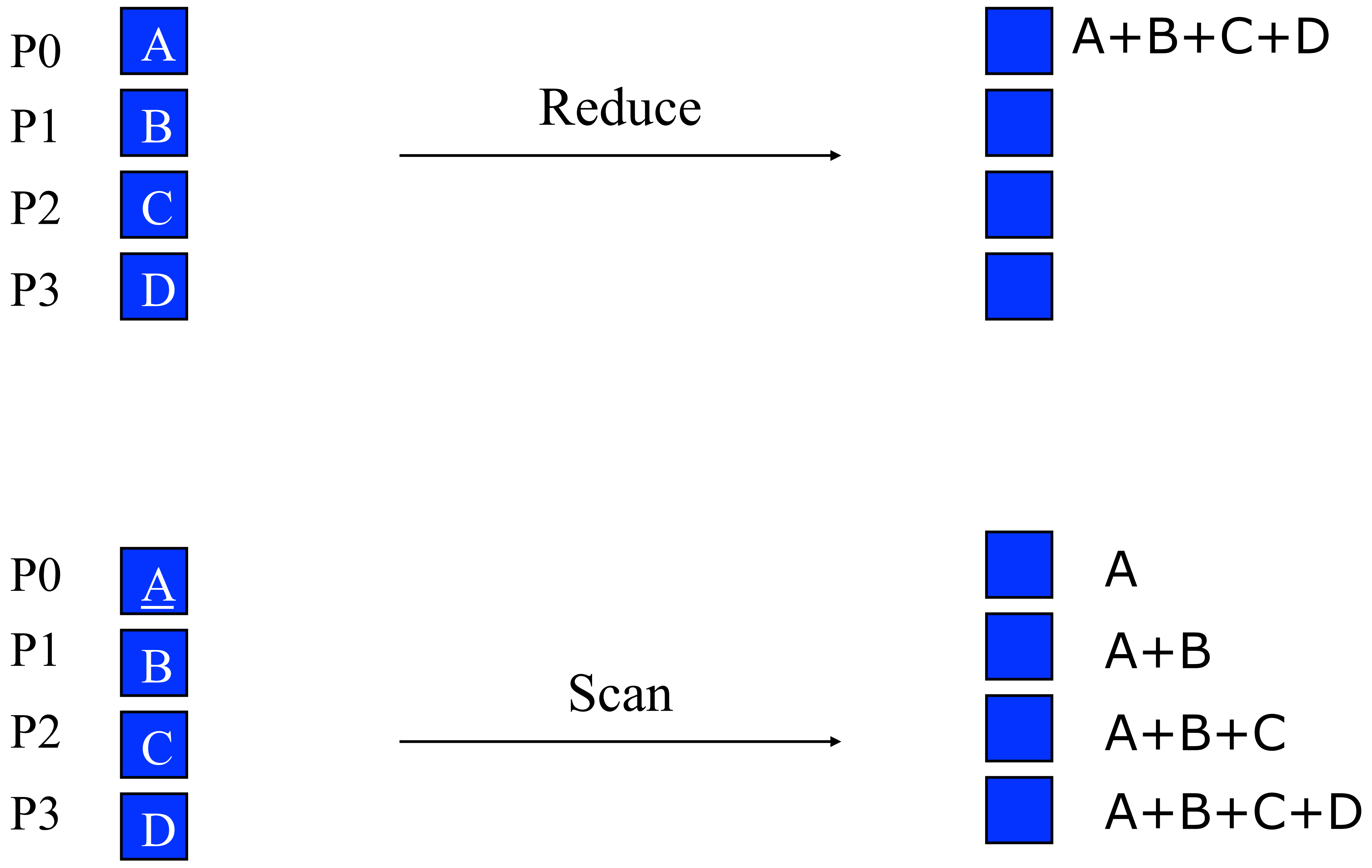
-
  ```
  MPI_Alltoall(const void* sendbuf,
          int sendcount,                      // Number vals to send to each proc
          MPI_Datatype sendtype,
          void* recvbuf,
          int recvcount,                      // Number vals to recv from each proc
          MPI_Datatype recvtype,
          MPI_Comm comm)
  ```

-
  ```
  int* send_vals = new int[num_procs] …
  int* recv_vals = new int[num_procs] …
  MPI_Alltoall(send_vals, 1, MPI_INT, recv_vals, 1, MPI_INT, MPI_COMM_WORLD)
  ```

Collective Operations With Computation

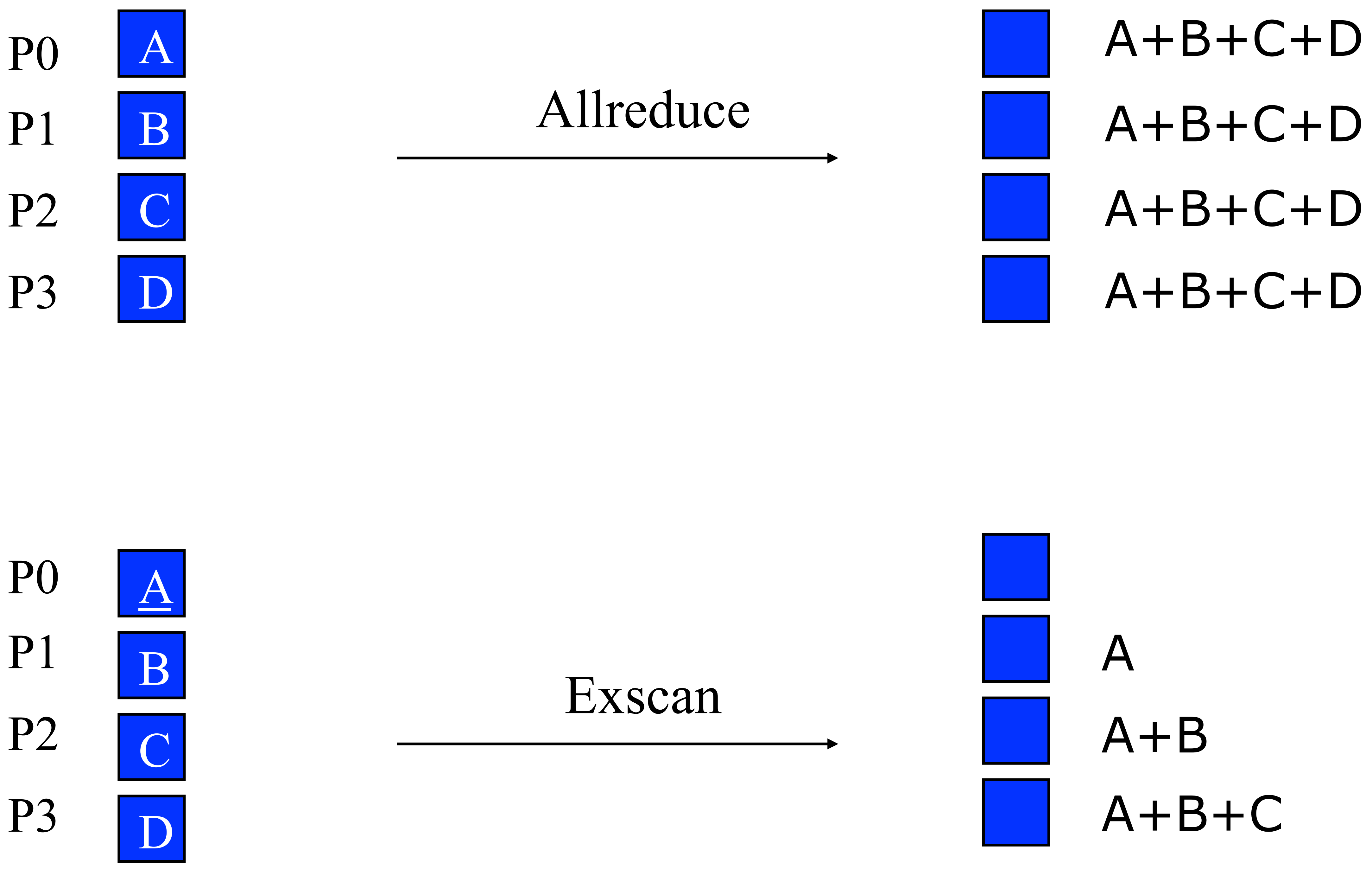
- Reduce, Allreduce (previously discussed) have computation such as summing values together
- Scan, Exscan : Combination of data from all prior ranks
- Reduce_scatter : All to all, but combines results

Collective Computation



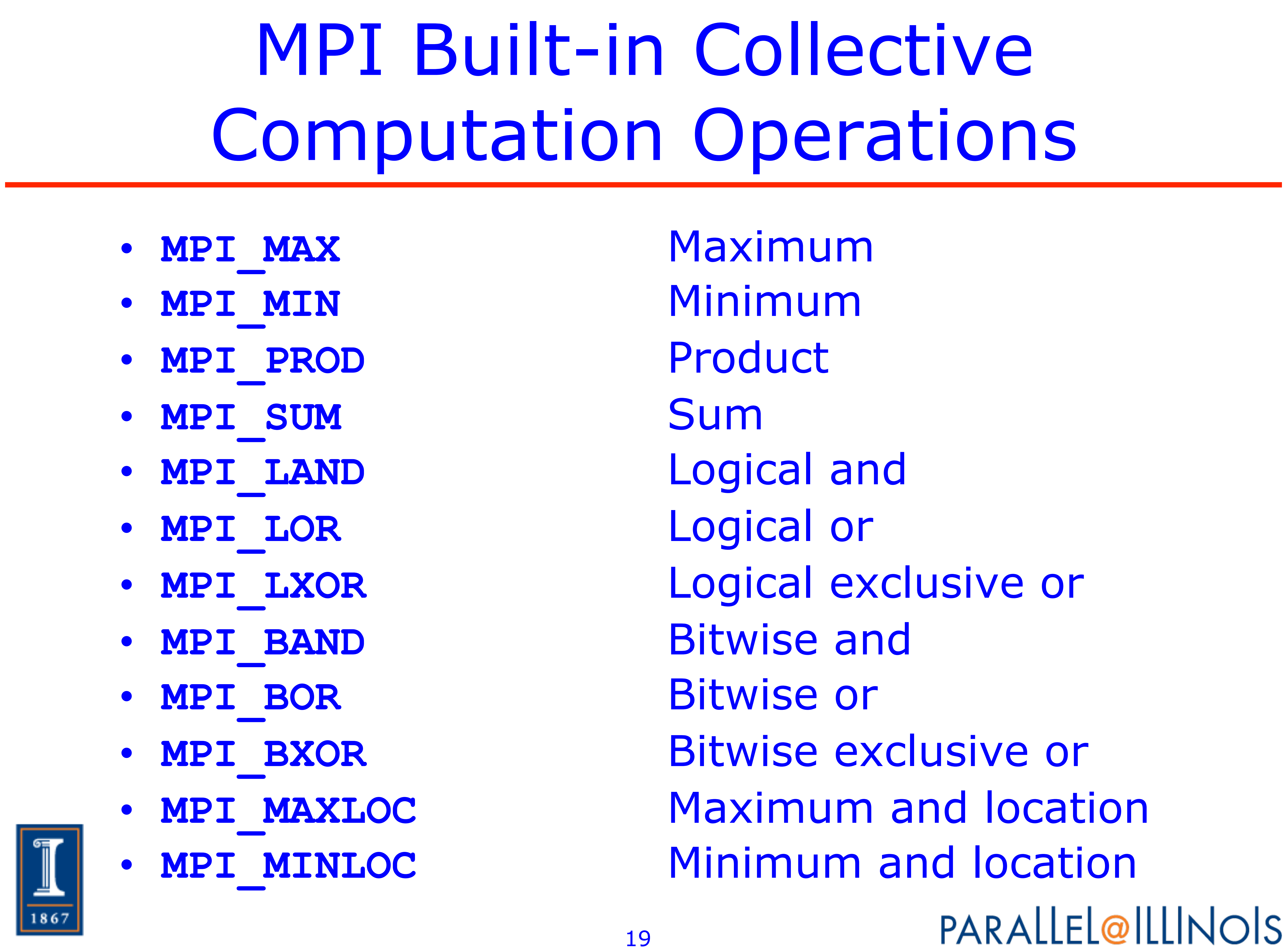
https://wgropp.cs.illinois.edu/courses/cs598-s15/lectures/lecture29.pdf

Collective Computation



https://wgropp.cs.illinois.edu/courses/cs598-s15/lectures/lecture29.pdf

Built In MPI_Op Values



MPI Built-in Collective Computation Operations

MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_PROD	Product
MPI_SUM	Sum
MPI_LAND	Logical and
MPI_LOR	Logical or
MPI_LXOR	Logical exclusive or
MPI_BAND	Bitwise and
MPI_BOR	Bitwise or
MPI_BXOR	Bitwise exclusive or
MPI_MAXLOC	Maximum and location
MPI_MINLOC	Minimum and location

PARALLEL@ILLINOIS