15-440
Distributed Systems
Collective Routines in MPI

Zeinab Khalifa

جامعة كارنيجي ميلون في قطر
Carnegie Mellon University Qatar

Agenda

- **Last recitation:** Point-to-point communication in MPI
- **Today's recitation:** collective communication in MPI

Point-to-point communication

P2P – Exercise (1)

1. Download Parallel_Sum.c from course website
2. Run the code:
    - Login to hadoop@<andrewid>-n01.qatar.cmu.edu
    - password: hadoop
    - mpicc Parallel_Sum.c –o Parallel_Sum
    - scp ./Parallel_Sum <andrewid>-n02.qatar.cmu.edu:<path>(similarly for n03 and n04)
    - mpiexec –f machinefile –np 4 ./Parallel_Sum (note: -np is the number of processes)
3. What have we done in parallel sum using point-to-point communication?

Parallel Sum – P2P

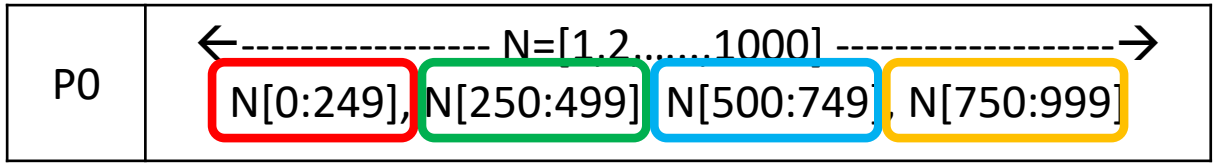
- Root initializes the array

P0	←---------- N=[1,2,…..,1000] ----------→

P1				
P2				
P3				

Parallel Sum – P2P

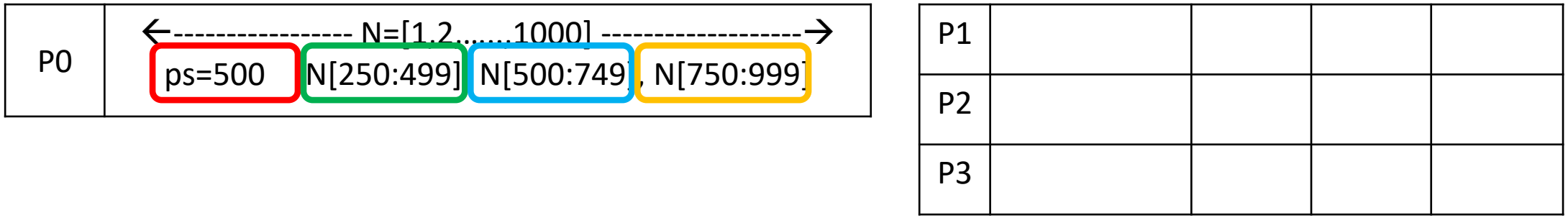
- Root initializes the array
- Root breaks down the array with a loop (process #, num_elements_per_process, etc.)



P1				
P2				
P3				

Parallel Sum – P2P

- Root initializes the array
- Root breaks down the array with a loop (process #, num_elements_per_process, etc.)
- Root calculates it's own partial sum



Parallel Sum – P2P

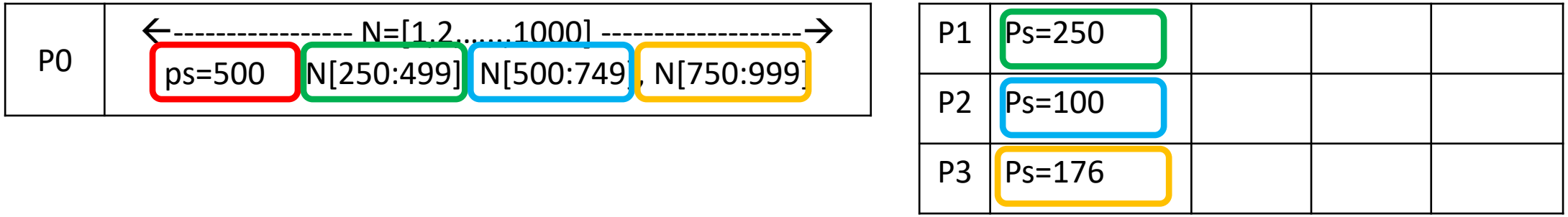
- Root initializes the array
- Root breaks down the array with a loop (process #, num_elements_per_process, etc.)
- Root calculates it's own partial sum
- Root sends each process elements to calculate

P0	←---------- N=[1,2,.....,1000] ----------→ ps=500 N[250:499] N[500:749] N[750:999]

P1	N[250:499]			
P2	N[500:749]			
P3	N[750:999]			

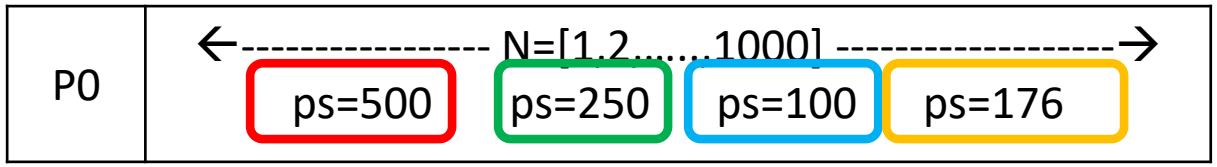
Parallel Sum – P2P

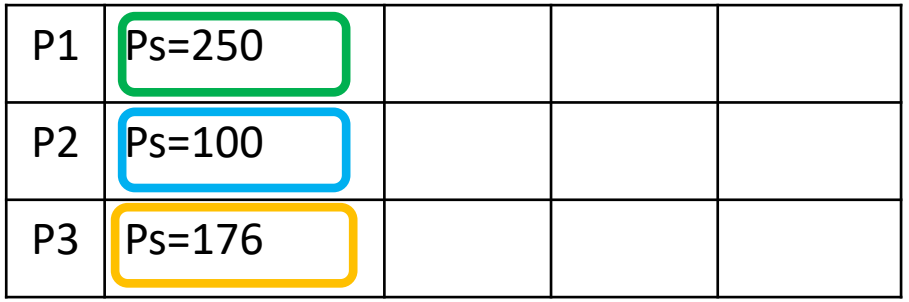
- Root initializes the array
- Root breaks down the array with a loop (process #, num_elements_per_process, etc.)
- Root calculates it's own partial sum
- Root sends each process elements to calculate
- Each process calculates the sum



Parallel Sum – P2P

- Root initializes the array
- Root breaks down the array with a loop (process #, num_elements_per_process, etc.)
- Root calculates it's own partial sum
- Root sends each process elements to calculate
- Each process calculates the sum
- Each process sends back the result





Parallel Sum – P2P

- Root initializes the array
- Root breaks down the array with a loop (process #, num_elements_per_process, etc.)
- Root calculates it's own partial sum
- Root sends each process elements to calculate
- Each process calculates the sum
- Each process sends back the result
- Root adds all partial sums and has the result

P0	←---------- N=[1,2,.....,1000] ----------→ Total sum = 1026

P1	Ps=250			
P2	Ps=100			
P3	Ps=176			

Collective communication

Collective routines

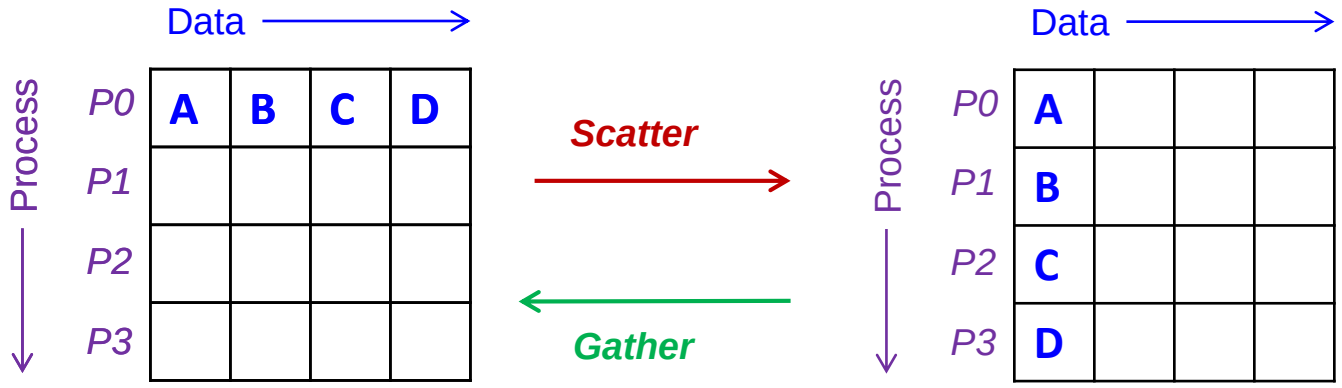
- Broadcast
- Gather
- AllGather
- Scatter
- Reduce
- AllReduce
- ReduceScatter
- Scan
- Alltoall

Collective routines

- Broadcast
- **Gather**
- AllGather
- **Scatter**
- **Reduce**
- **AllReduce**
- ReduceScatter
- Scan
- Alltoall

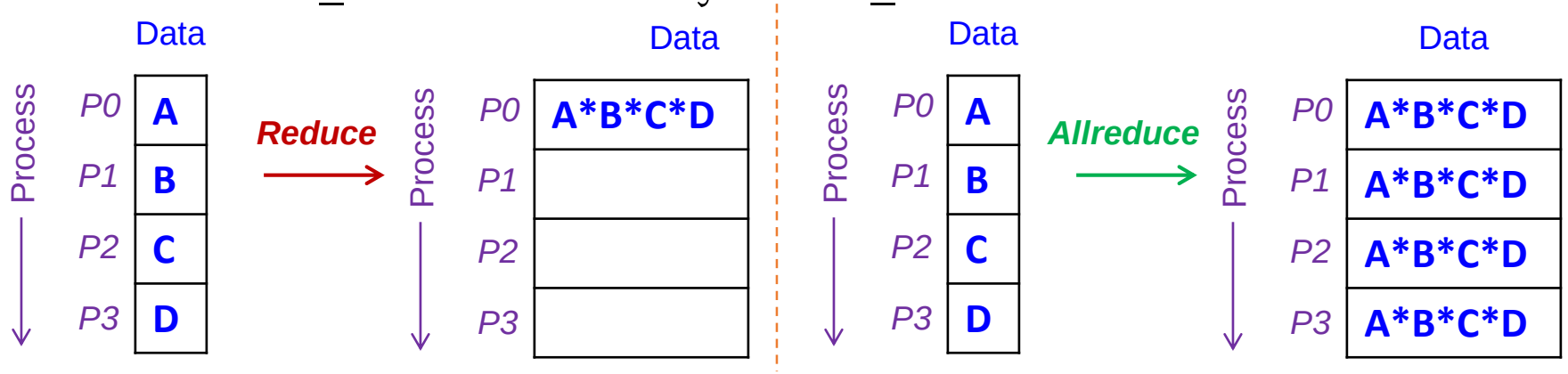
Scatter and Gather

- **Scatter** distributes distinct messages from a single source task to each task in the group
- **Gather** gathers distinct messages from each task in the group to a single destination task



Reduce and All Reduce

- **Reduce** applies a reduction operation on all tasks in the group and places the result in one task
- **Allreduce** applies a reduction operation and places the result in all tasks in the group. This is equivalent to an MPI_Reduce followed by an MPI_Bcast



Parallel Sum – CR

Exercise: How can we compute the parallel sum using CR?

Hints (1): which CR is similar to this step in P2P communication?

P0	←---------- N=[1,2,……,1000] ----------→ ps=500 N[250:499] N[500:749] N[750:999]

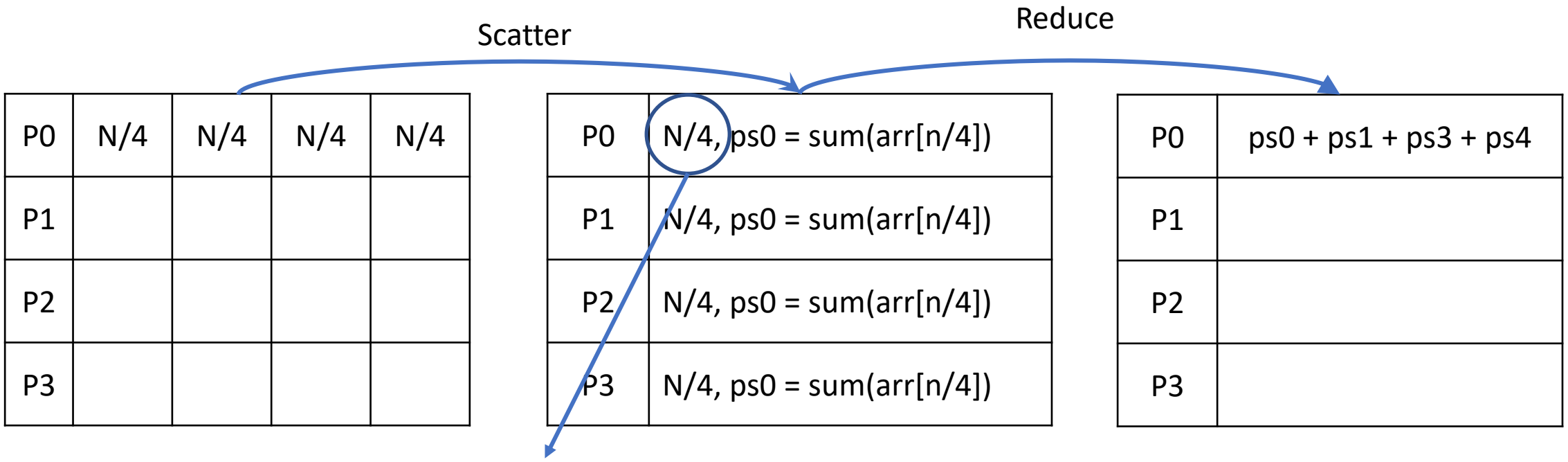
P1	N[250:499]			
P2	N[500:749]			
P3	N[750:999]			

Hint (2): which CR is similar to the following steps we did in P2P communication?

1. Root sends each process elements to calculate
2. Each process calculates the sum
3. Each process sends back the result
4. Root adds all partial sums and has the result

Parallel Sum – CR

Exercise: How can we compute the parallel sum using CR?



Parallel Sum – CR

Exercise: Download the starter code and implement parallel sum using the collective routines.

MPI_Scatter & MPI_Reduce

MPI_SCATTER(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, root, comm)

sendbuf: address of send buffer
sendcount: number of elements sent to each process
sendtype: data type of send buffer elements
recvbuf: address of receive buffer
recvcount: number of elements in receive buffer
recvtype: data type of receive buffer elements
root: rank of sending process
comm: communicator

MPI_REDUCE(sendbuf, recvbuf, count, datatype, op, root, comm)

sendbuf: address of send buffer
recvbuf: address of receive buffer
count: number of elements in send buffer
datatype: data type of elements of send buffer
op: reduce operation
root: rank of root process
comm: communicator

P2P vs. CR

What is the difference?

- Point-to-point communication

One process sends a message to another one

- Collective communication
- Collective communication operations are composed of several point-to-point operations & Optimized internal implementations

- Can broadcast be implemented using MPI_Send & MPI_Recv?
- Yes
- However, it is less efficient