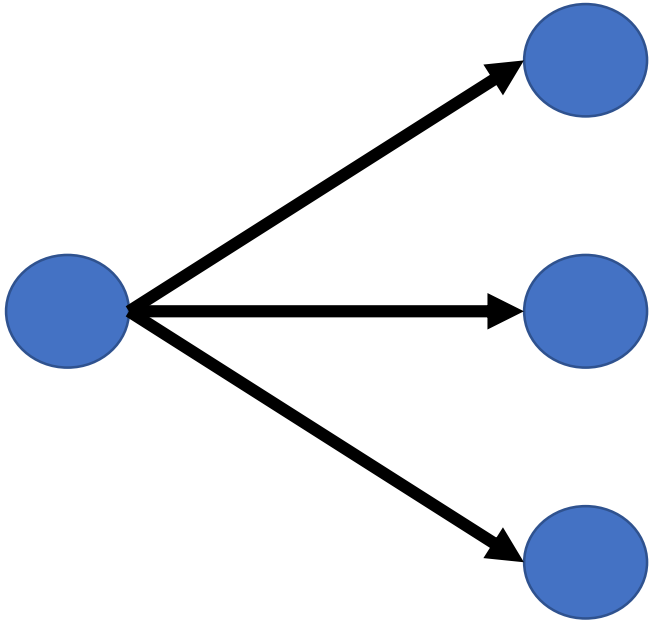
Collective Communication

Feb 9, 2021

P2P and Collective



Collective Communication

- Must be called by all processes that are part of the communicator

Types

- Synchronization (MPI_Barrier)
- Global communication (MPI_Bcast, MPI_Gather, …)
- Global reduction (MPI_Reduce, ..)

Barrier

- MPI_Barrier (comm)
- Every rank needs to call this function (for true synchronization)
- Caller returns only after all processes have entered the call

```
printf("Before barrier");
MPI_Barrier (MPI_COMM_WORLD);
printf("After barrier");
```

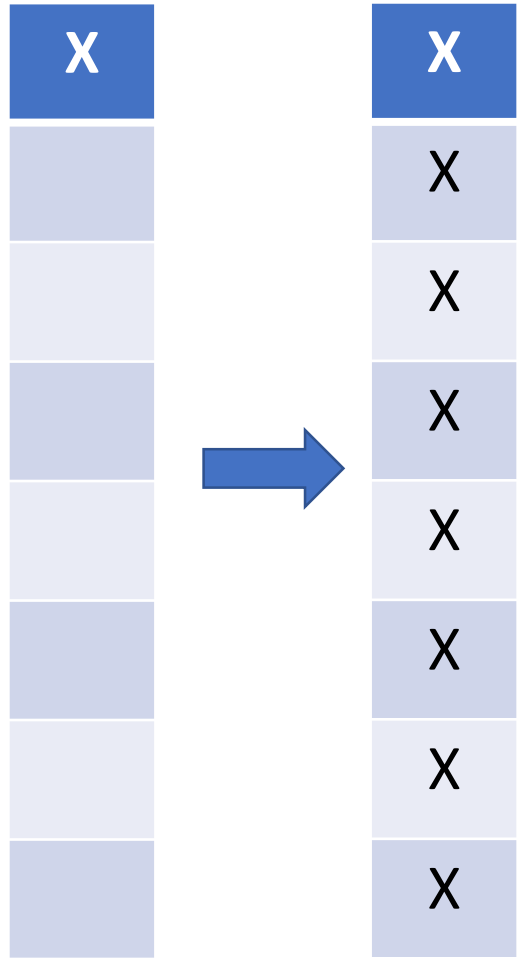
Barrier

n=4

```
if (myrank != 0)
    MPI_Barrier (MPI_COMM_WORLD);
printf("%d\n", rank);
```

Broadcast

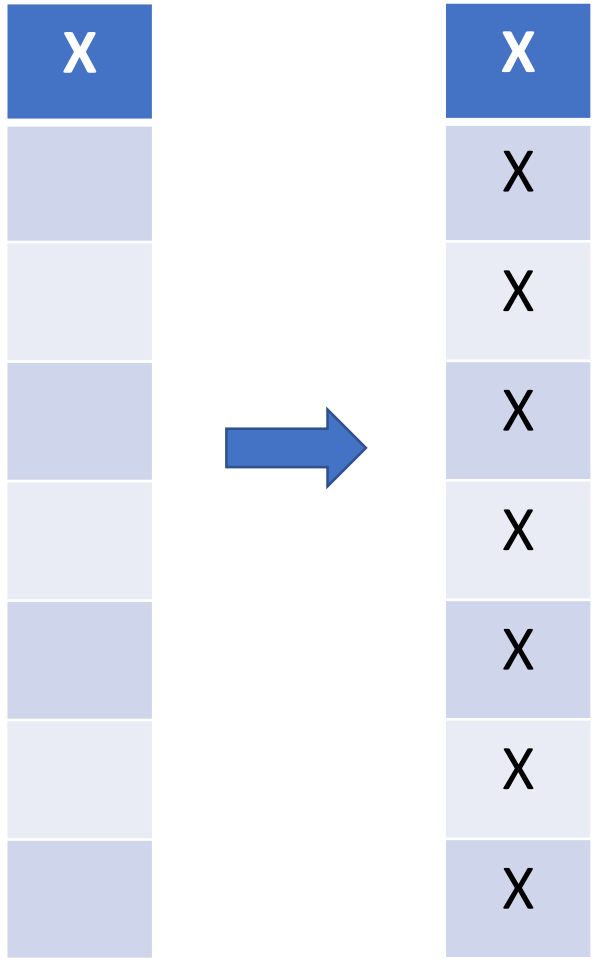
- Root process sends message to all processes
- int MPI_Bcast (buffer, count, datatype, root, comm)
- “count” is the number of elements in “buffer” – must match
  - “message sizes do not match across processes in the collective routine: Received 400 but expected 4000”
- “buffer” is input at root process
- Any process can be a root process but has to be the same process when MPI_Bcast is called
- Buffer size of “buf” array is not known apriori at non-root processes
- Tag?



Broadcast

Q1: Point-to-point communication for the same?

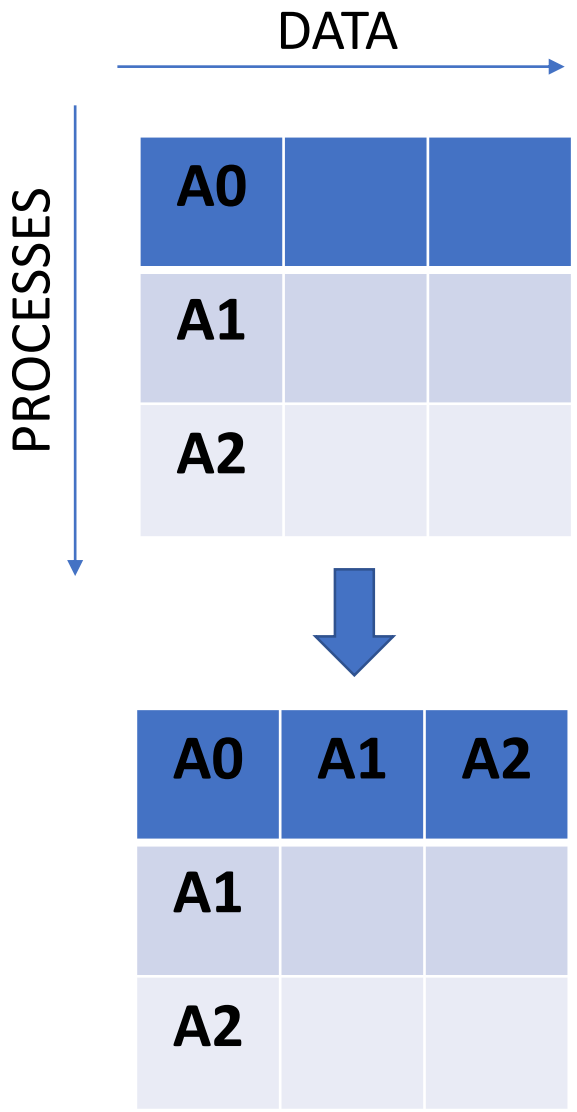
Q2: Buffer size of "buf" array is not known apriori at non-root processes, how should root broadcast buf?



Bcast Demo

Gather

- Gathers values from all processes to a root process
- int MPI_Gather (sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, root, comm)
- Arguments recv* not relevant on non-root processes
- recvcount → size of any (single) receive
- Distinct values (may be same) received from non-root processes at the root process
- Example: Reading a file

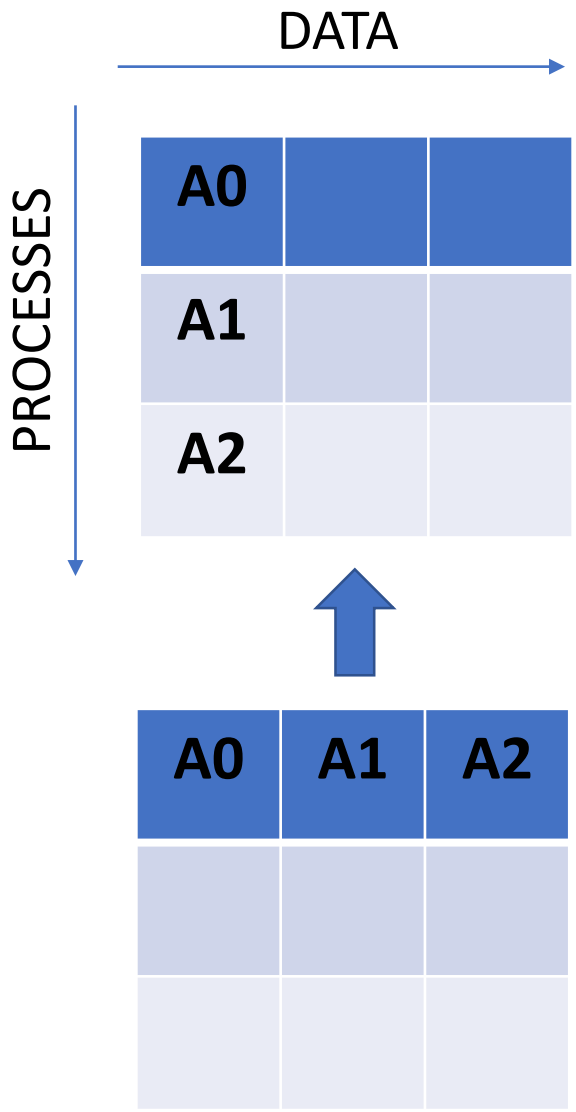


Q: Equivalent point-to-point communications for the same?

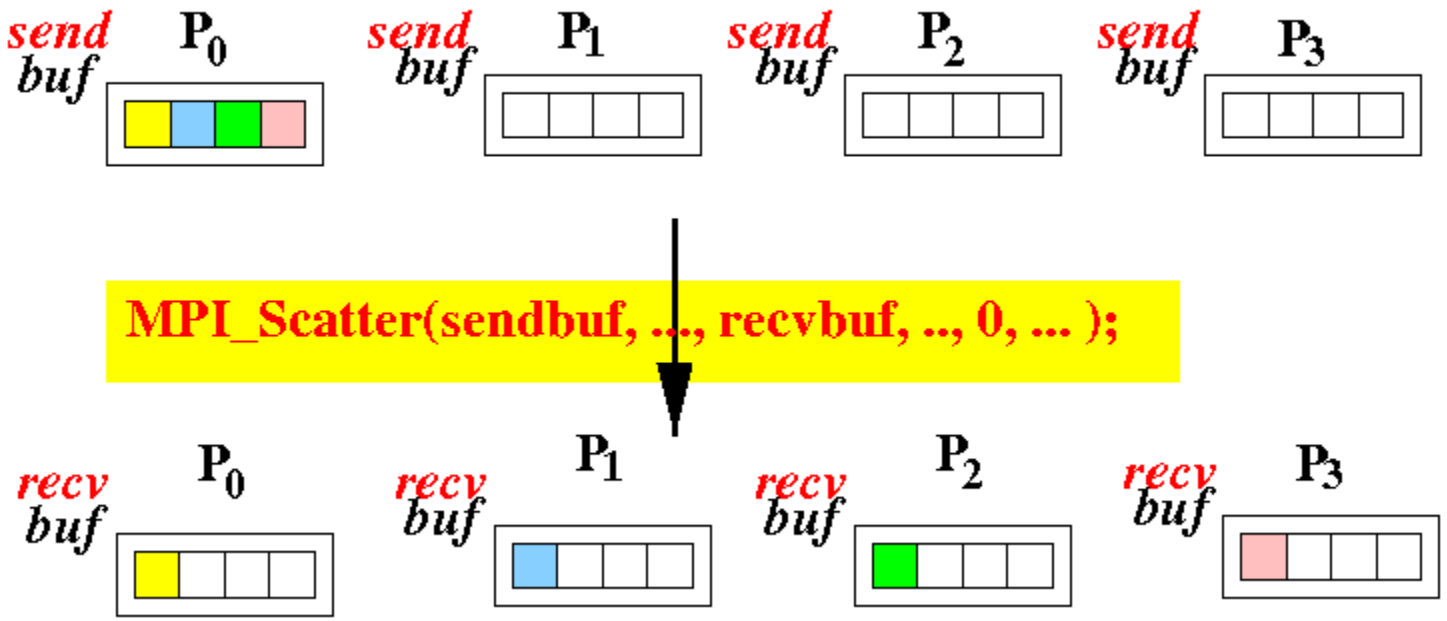
- MPI_Recv at root
- MPI_Send at non-root

Scatter

- Scatters values to all processes from a root process
- int MPI_Scatter (sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, root, comm)
- Arguments send* not relevant on non-root processes
- Output parameter – recvbuf



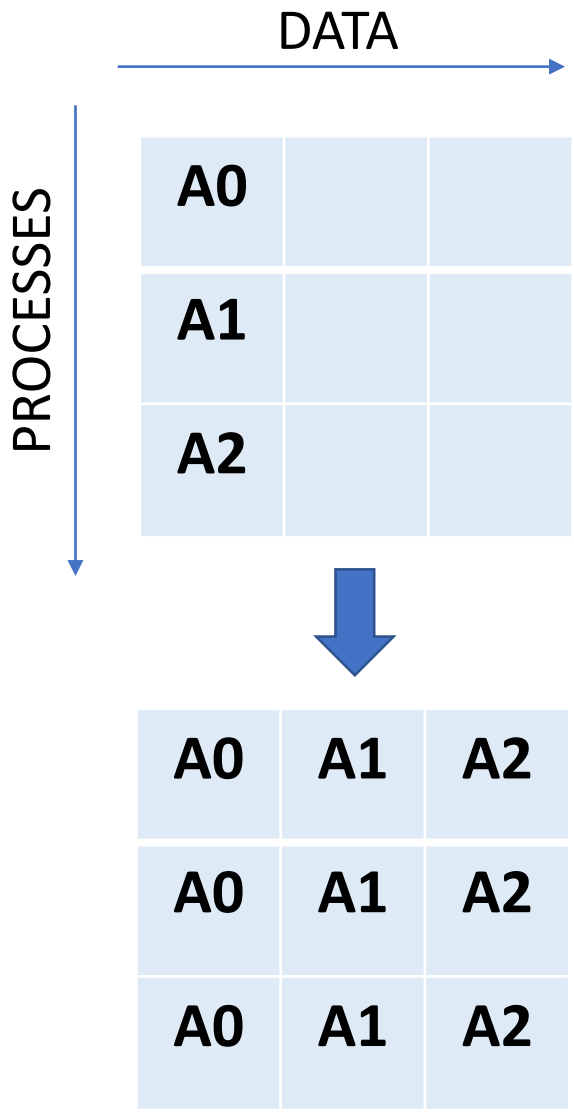
MPI_Scatter Illustration



Credit: Shun Yan Cheung

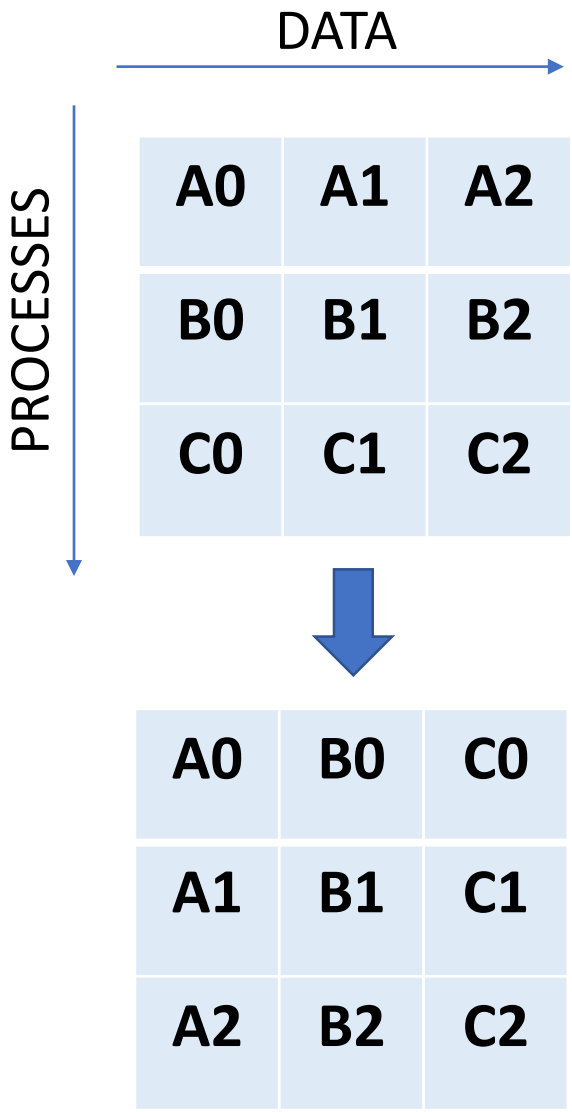
Allgather

- All processes gather values from every other process
- int MPI_Allgather (sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, comm)



Alltoall

- Send data from all processes to all processes
- int MPI_Alltoall (sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, comm)
- Output parameter – recvbuf

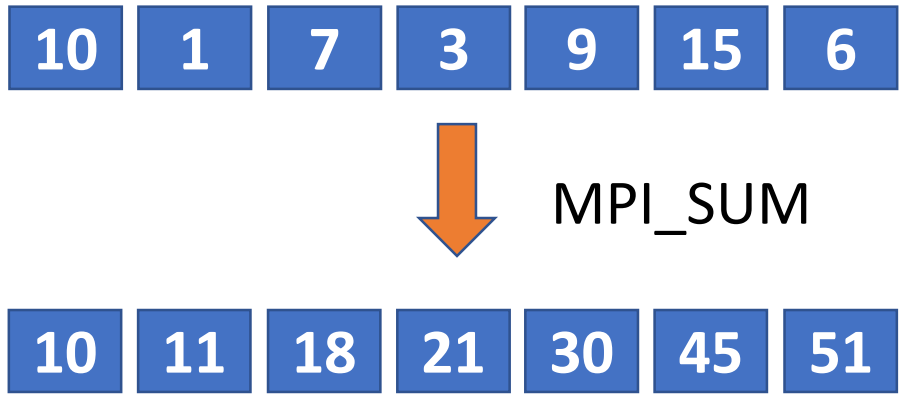


Equivalent collective?

- MPI_Scatter at all processes

Scan

- MPI_Scan (inbuf, outbuf, count, datatype, op, comm)
- op: MIN, MAX, SUM, PROD, …
- Perform a prefix reduction on distributed data
- Reduction of values in the send buffers of processes with ranks 0:i-1 is returned in receive buffer of rank i



DEMO

scan.c

```
#include <stdio.h>
#include <stdlib.h>
#include "mpi.h"

int main( int argc, char *argv[])
{
  int myrank, size, sendval, val;
  MPI_Status status;

  MPI_Init( &argc, &argv );
  MPI_Comm_rank( MPI_COMM_WORLD, &myrank );
  MPI_Comm_size( MPI_COMM_WORLD, &size );

  sendval = myrank; // initialization
  MPI_Scan (&sendval, &val, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);

  // verify - print from all ranks
  printf ("%d val=%d\n", myrank, val);

  MPI_Finalize();
  return 0;

}
~
~
~
~
~
~
~
~
~
~
~
"scan.c" [dos] 25L, 506C
```

Reduce

- MPI_Reduce (inbuf, outbuf, count, datatype, op, root, comm)
- Combines element in inbuf of each process
- Combined value in outbuf of root
- op: MIN, MAX, SUM, PROD, …
- Example:

MPI_Reduce (…, MPI_MAX, …) Output: 21

MPI_Reduce (…, MPI_MIN, …) Output: 1

Allreduce

- MPI_Allreduce (inbuf, outbuf, count, datatype, op, comm)
- op: MIN, MAX, SUM, PROD, …
- Combines element in inbuf of each process
- Combined value in outbuf of each process

Homework

1. Broadcast P doubles from rank 0 to all ranks (consider total #processes = 10). You can run on any number of hosts or cores. P = 10^3, 10^4, 10^5, 10^6, 10^7.

2. Let total number of processes be P. Write MPI_Allgather using P MPI_Bcast calls (i.e. every process broadcasts its data to the other processes). Compare the performance of both.

3. Compare performance of MPI_Alltoall with multiple MPI_Scatter for different process sizes and data sizes.