Master Course – High Performance Computing

Introduction to the Message Passing Interface (MPI)

Collective Communication

Dr. Juraj Kardos, Dr. Pratyuksh Bansal, Prof. Olaf Schenk
(TA: Tim Holt, Malik Lechekhab)

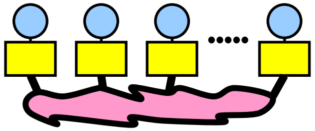
Faculty of Informatics
USI Lugano, Switzerland

November 3, 2021

MPI Outline

Chap. 5 Collective Communication

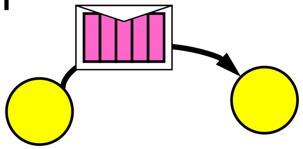
1. MPI Overview
   - one program on several processors work and data distribution



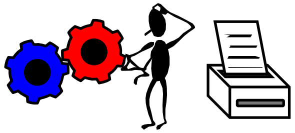
2. Process model and language bindings
   - starting several MPI processes

MPI_Init()
MPI_Comm_rank()

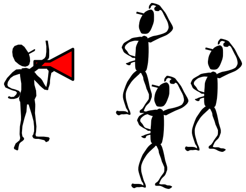
3. Messages and point-to-point communication
   - the MPI processes can communicate



4. Non-blocking communication
   - **to avoid idle time and deadlocks**

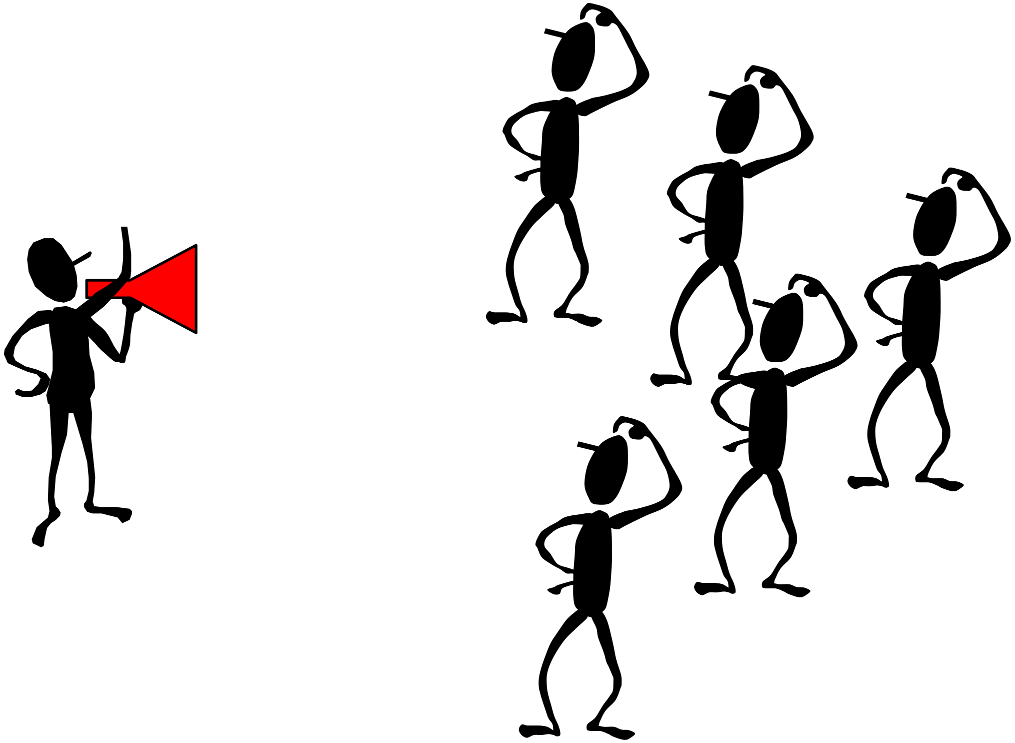


5. **Collective communication**
   - **e.g., broadcast**



Broadcast

- A one-to-many communication.

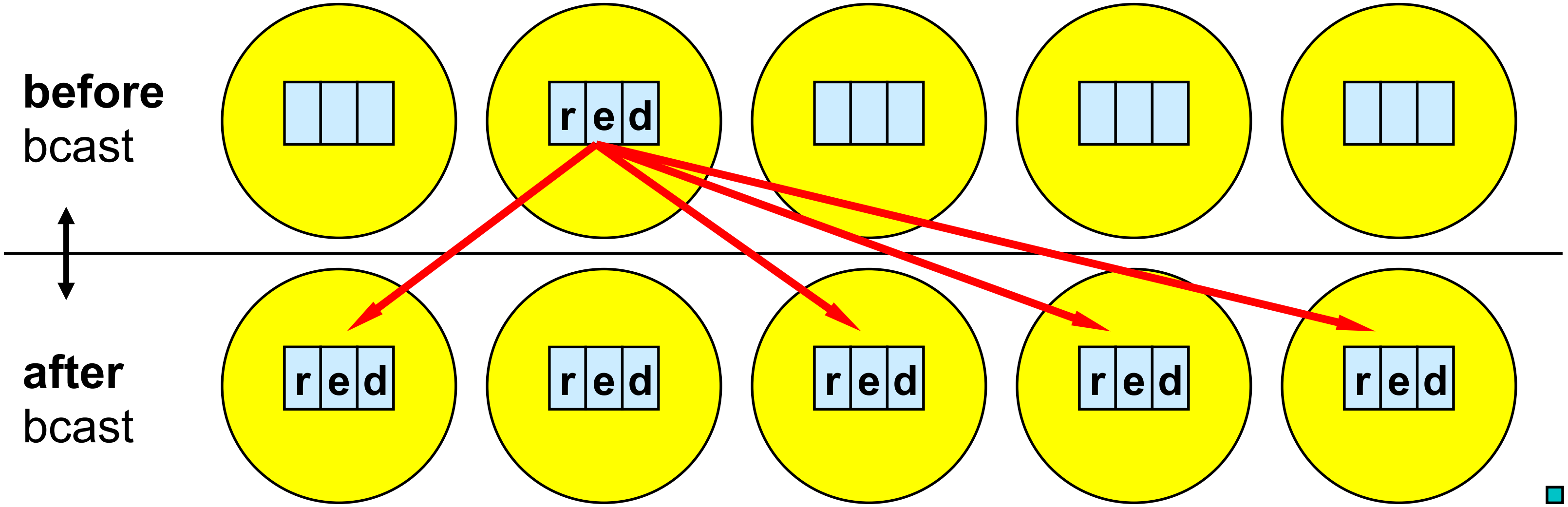


Collective Communication

- Communications involving a group of processes.
- Called by all processes in a communicator.
- Examples:
  - Broadcast, scatter, gather.
  - Global sum, global maximum, etc.

Broadcast

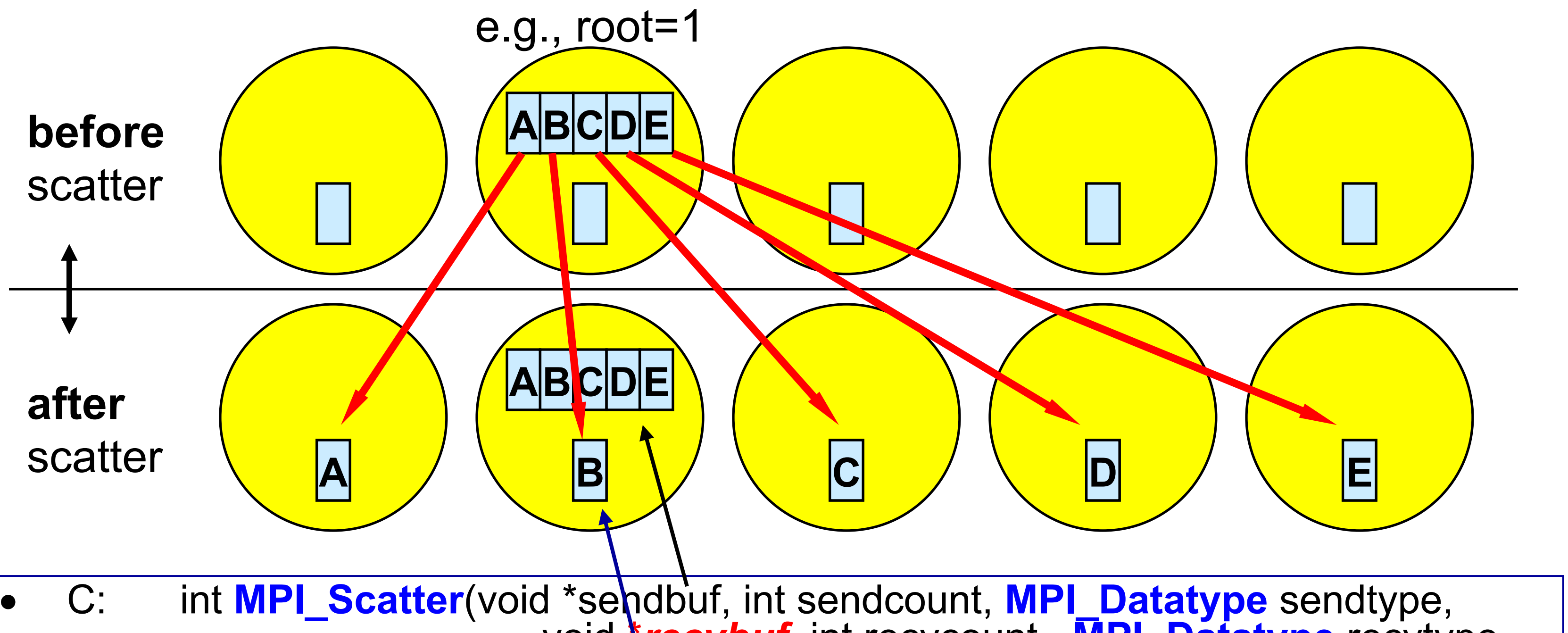
- C: int **MPI_Bcast**(void *buf, int count, **MPI_Datatype** datatype, int root, **MPI_Comm** comm)



e.g., root=1

- **rank of the sending process (i.e., root process)**
- **must be given identically by all processes**

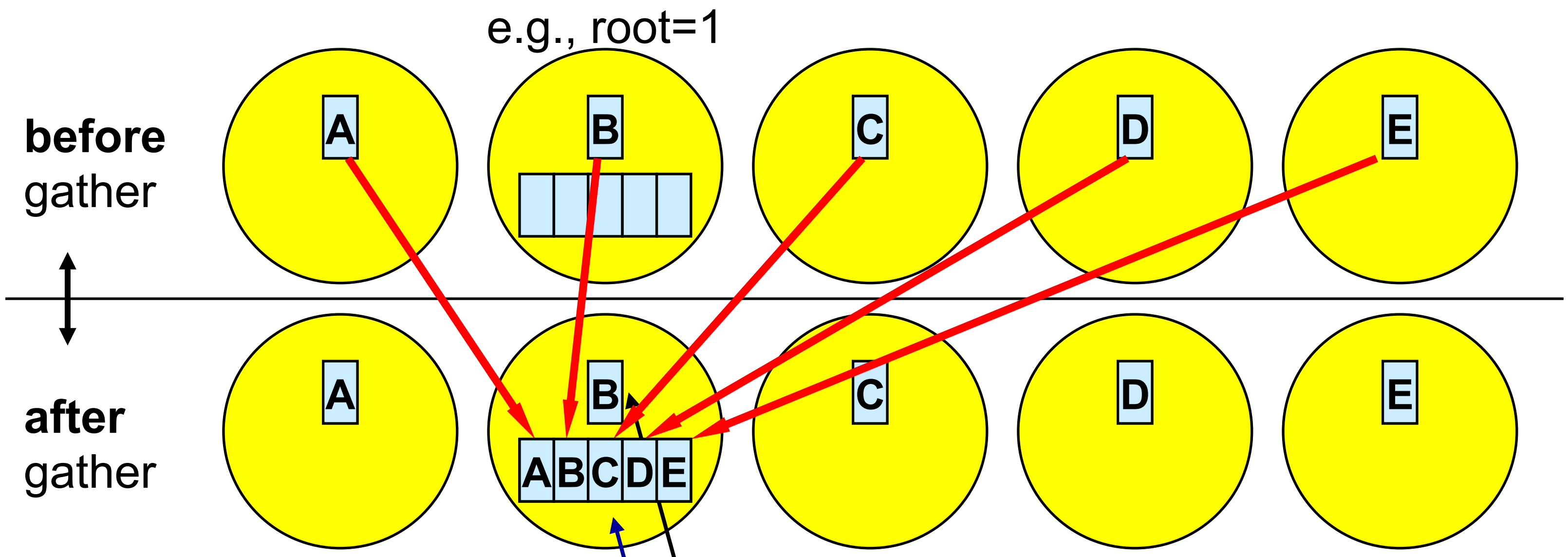
Scatter



- C: int **MPI_Scatter**(void *sendbuf, int sendcount, **MPI_Datatype** sendtype, void *recvbuf*, int recvcount, **MPI_Datatype** recvtype, int root, **MPI_Comm** comm)

Example:
MPI_Scatter(sbuf, 1, MPI_CHAR, *rbuf*, 1, MPI_CHAR, 1, MPI_COMM_WORLD)

Gather



- C: int **MPI_Gather**(void *sendbuf, int sendcount, **MPI_Datatype** sendtype, void ***_recvbuf_**, int recvcount, **MPI_Datatype** recvtype, int root, **MPI_Comm** comm)

Global Reduction Operations

- To perform a global reduce operation across all members of a group.
- $d_0 \circ d_1 \circ d_2 \circ d_3 \circ \ldots \circ d_{s-2} \circ d_{s-1}$
  - d_i= data in process rank i
    - **single variable, or**
    - **vector**
  - o = associative operation
  - Example:
    - **global sum or product**
    - **global maximum or minimum**
    - **global user-defined operation**
- floating point rounding may depend on usage of associative law:
  - $[(d_0 \circ d_1) \circ (d_2 \circ d_3)] \circ [\ldots \circ (d_{s-2} \circ d_{s-1})]$
  - $((((((d_0 \circ d_1) \circ d_2) \circ d_3) \circ \ldots) \circ d_{s-2}) \circ d_{s-1})$

Example of Global Reduction

- Global integer sum.
- Sum of all inbuf values should be returned in ***resultbuf.***
- C: int root=0

```
int root=0
MPI_Reduce(&inbuf, &resultbuf, 1, MPI_INT, MPI_SUM,
           root, MPI_COMM_WORLD);
```

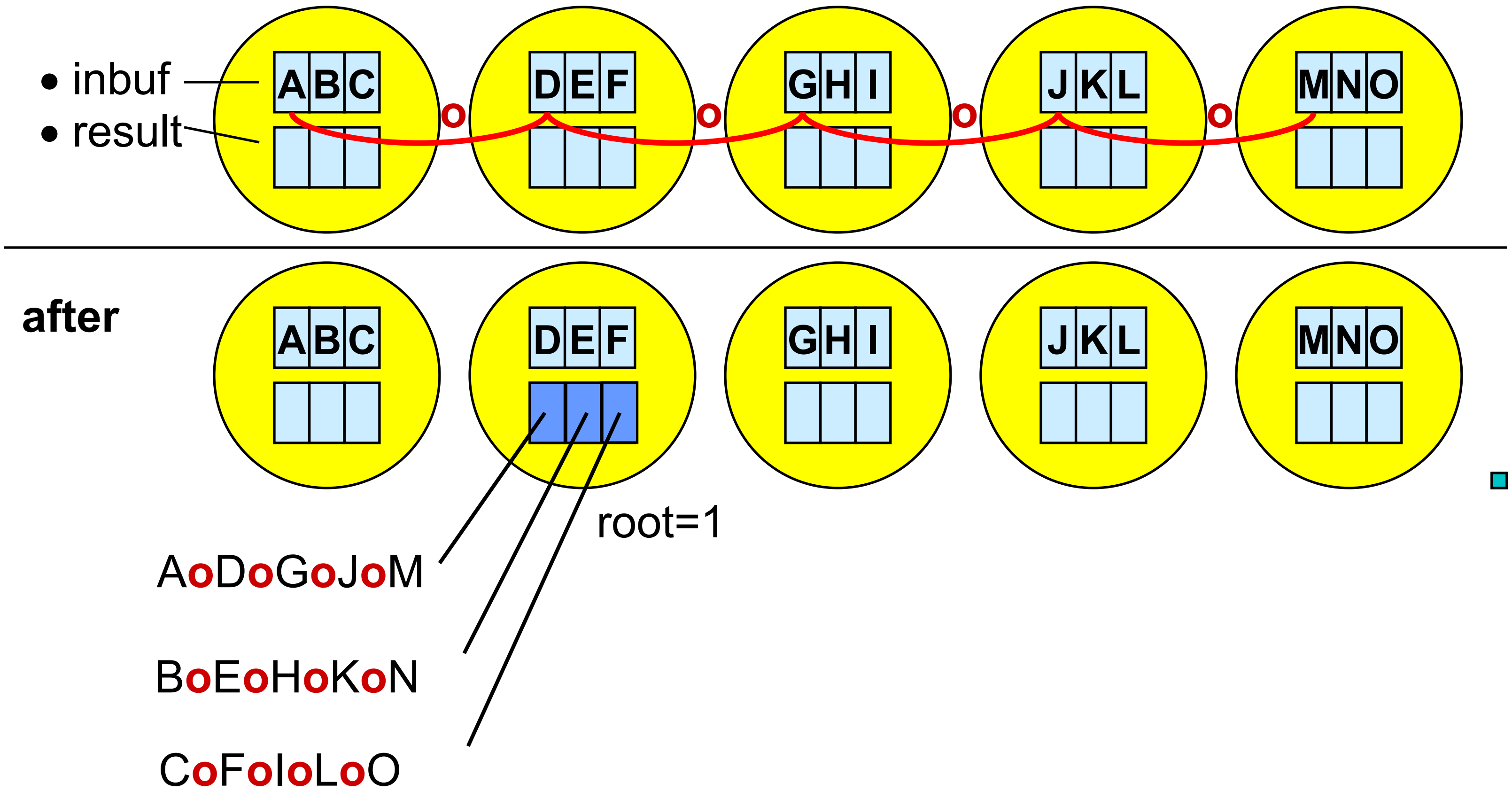
- The result is only placed in **resultbuf** at the root process.

Predefined Reduction Operation Handles

Predefined operation handle	**Function**
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Sum
MPI_PROD	Product
MPI_LAND	Logical AND
MPI_BAND	Bitwise AND
MPI_LOR	Logical OR
MPI_BOR	Bitwise OR
MPI_LXOR	Logical exclusive OR
MPI_BXOR	Bitwise exclusive OR
MPI_MAXLOC	Maximum and location of the maximum
MPI_MINLOC	Minimum and location of the minimum

MPI_REDUCE

before MPI_REDUCE



- inbuf
- result

after

root=1

AoDoGoJoM

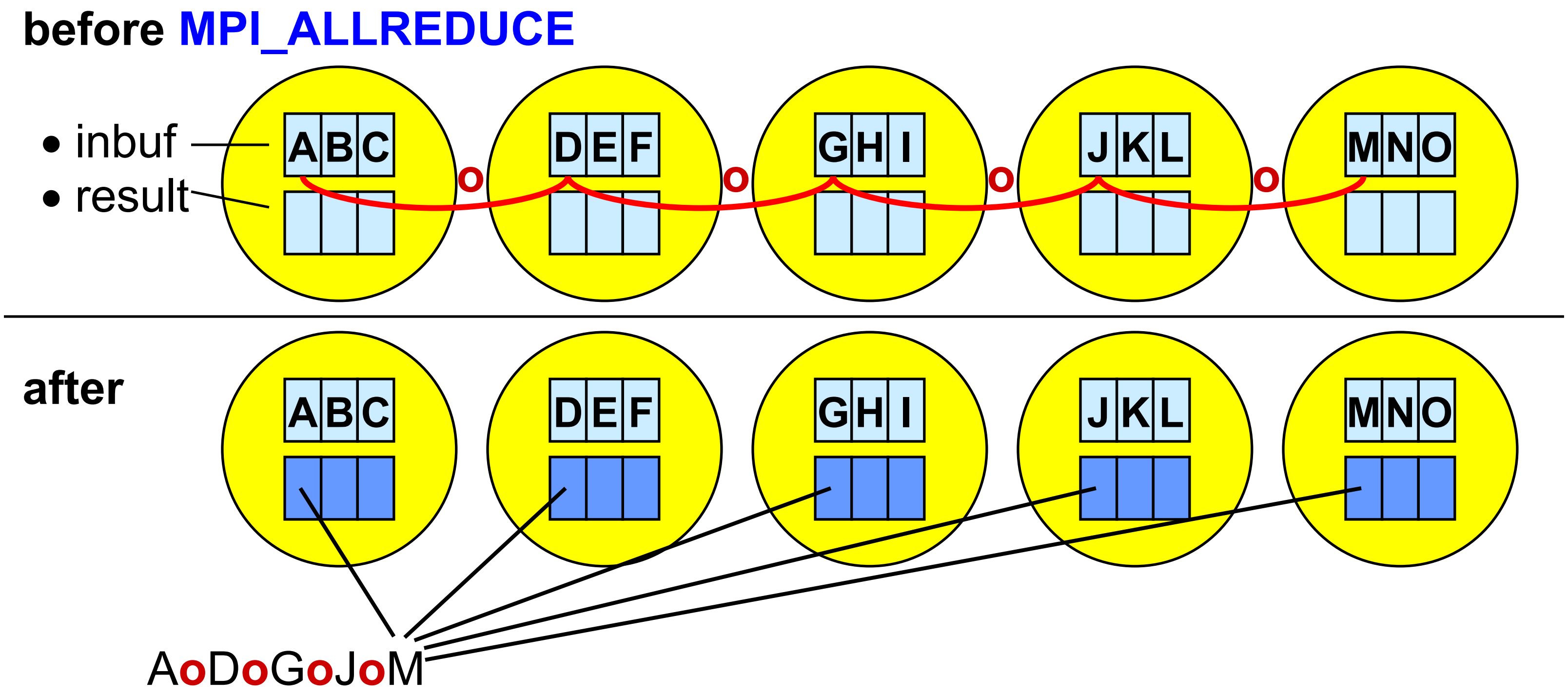
BoEoHoKoN

CoFoIoLoO

Variants of Reduction Operations

- **MPI_ALLREDUCE**
  - no root,
  - returns the result in all processes
- **MPI_SCAN**
  - prefix reduction
  - result at process with rank i :=
    reduction of inbuf-values from rank 0 to rank i
- **MPI_EXSCAN**
  - result at process with rank i :=
    reduction of inbuf-values from rank 0 to rank **i-1**

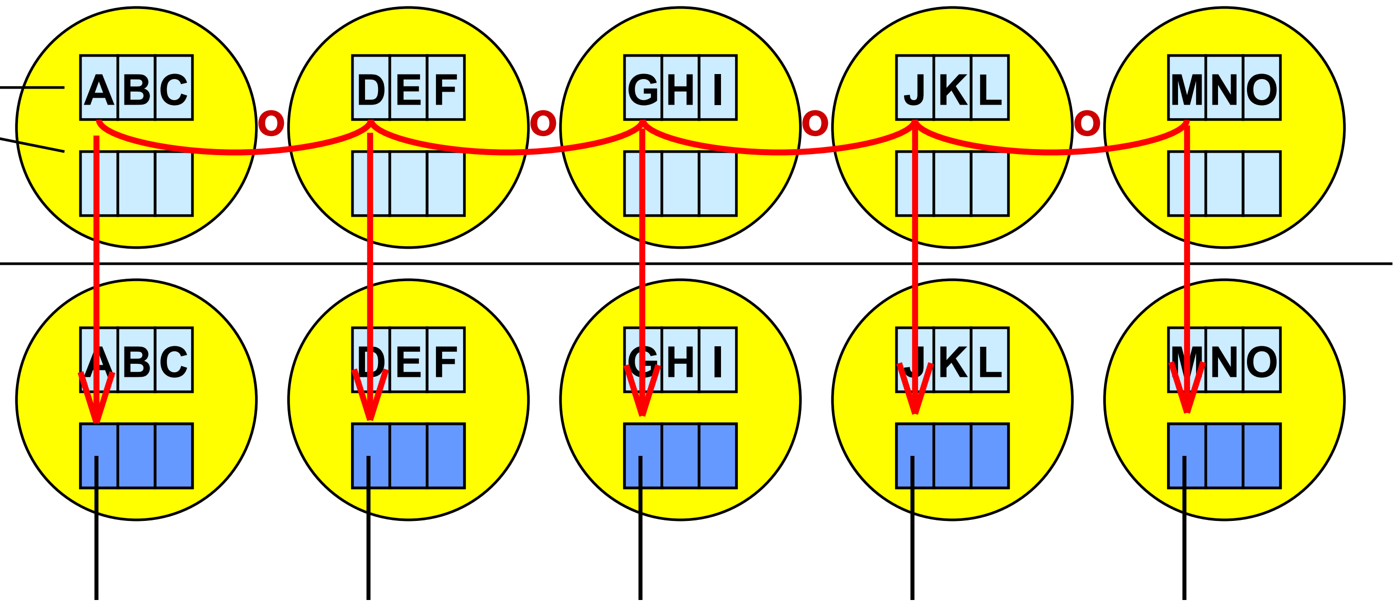
MPI_ALLREDUCE



MPI_SCAN and MPI_EXSCAN

before the call

- inbuf
- result



after

MPI_SCAN:	A	AoD	AoDoG	AoDoGoJ	AoDoGoJoM
MPI_EXSCAN:	-	A	AoD	AoDoG	AoDoGoJ

done in parallel

In Class Exercise: PI

- You are given the code template, that calculates PI value in pi.c
- The PI value is calculated by numerically solving the integral of arctangent
  - Each process calculates approximate area of its part of integral
  - In the end, partial results shall be summed together

- Use MPI reduction call to collect the grand total integral sum on one rank
- Print out the resulting approximation of PI value

In Class Exercise: PI

- show solution pi.c