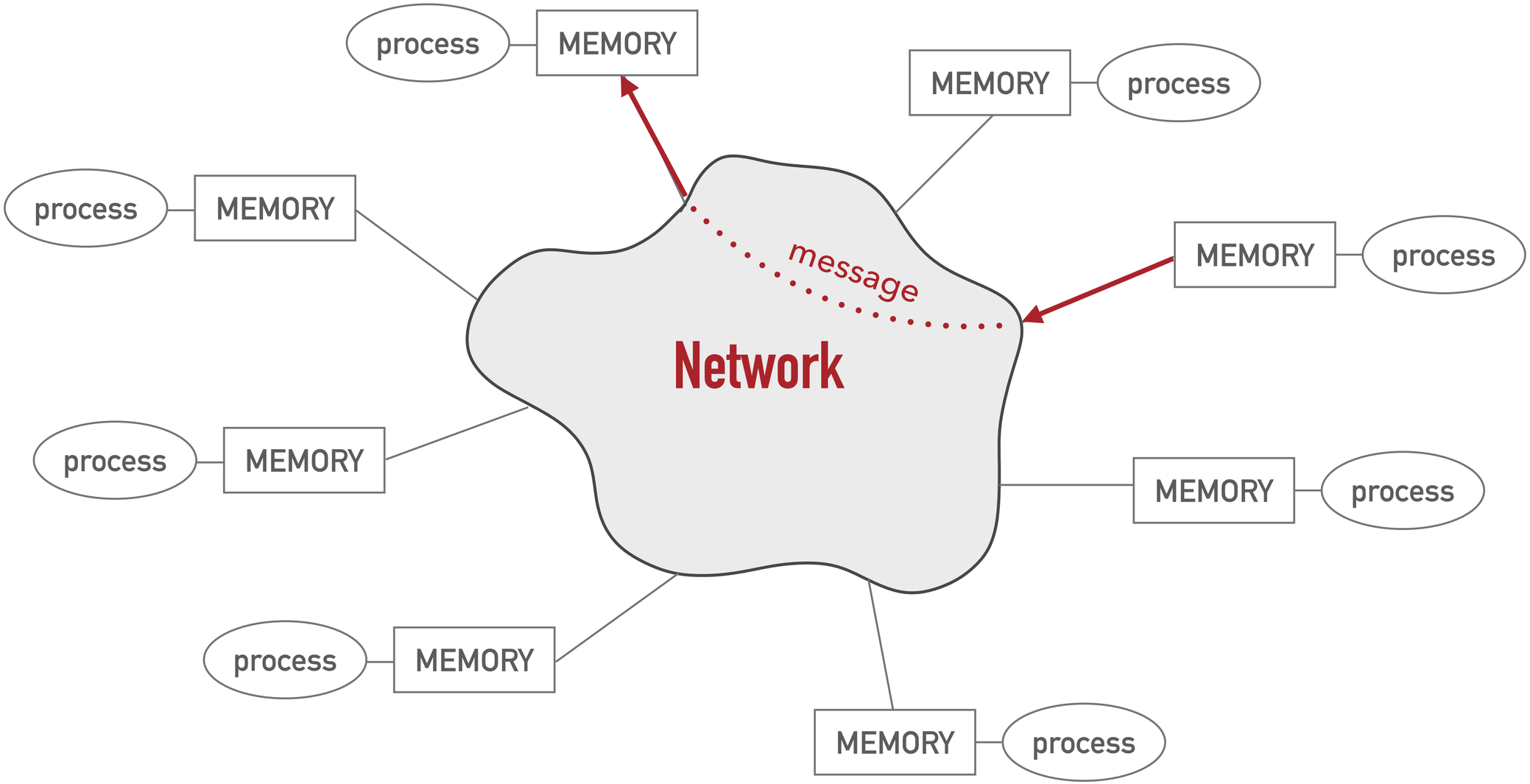
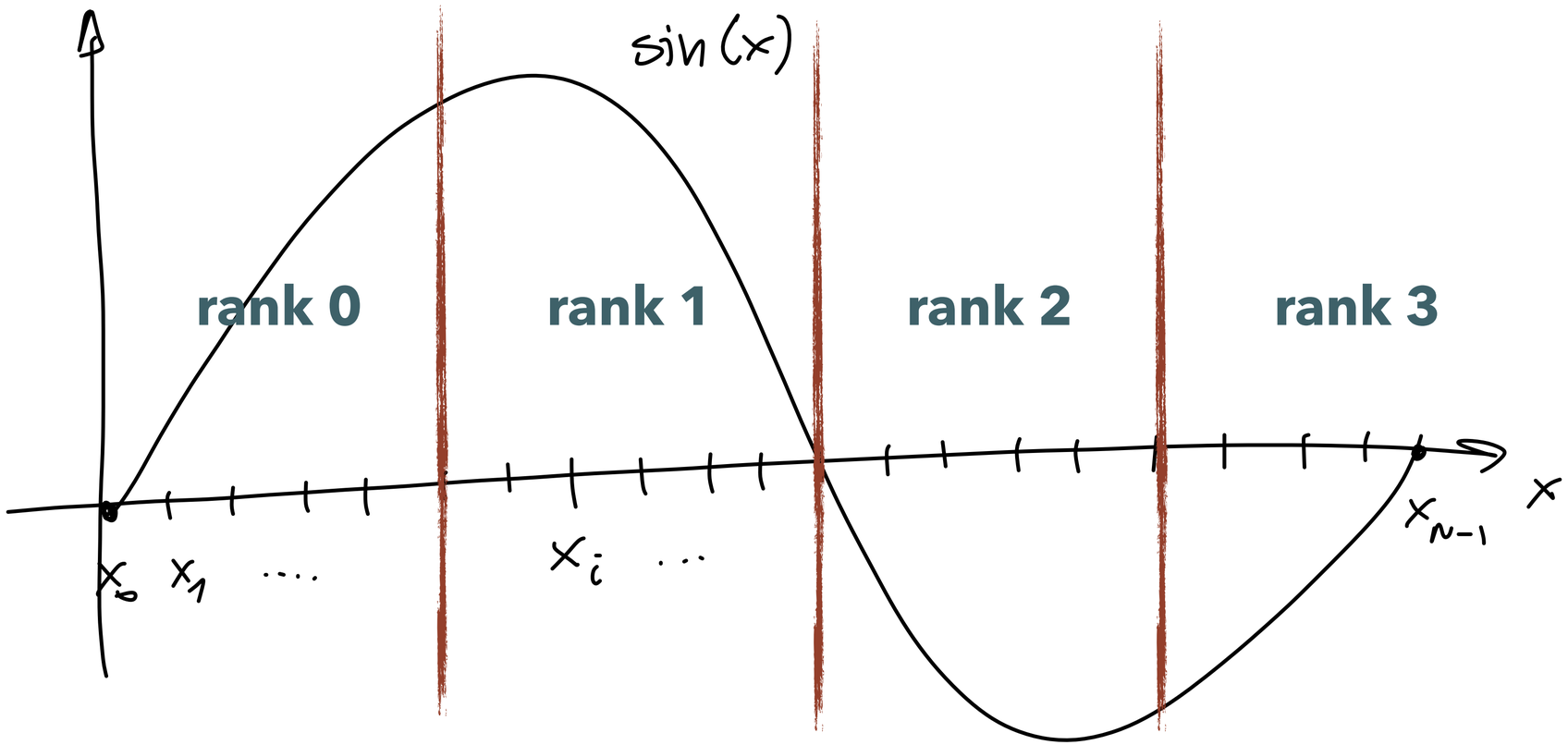
ME 471/571

Week 4 - Collective Communication

MESSAGE PASSING MODEL

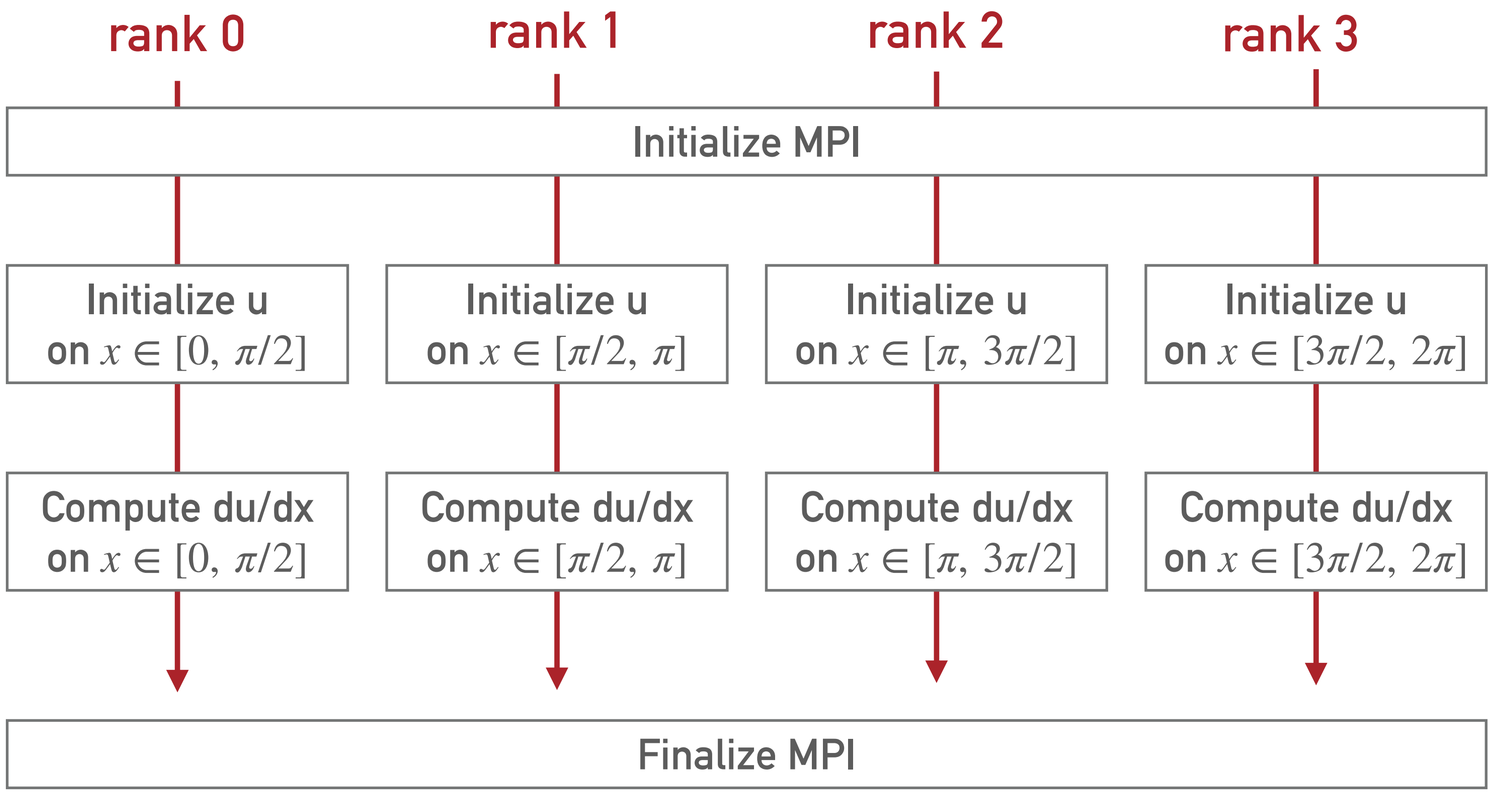


Example of data parallelism - compute the derivative of sin(x)



Divide and Conquer: *We can divide the work among available ranks, where each rank performs a fraction of the work*

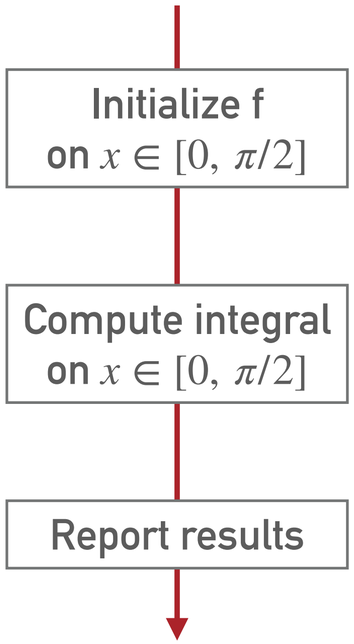
DERIVATIVE EXAMPLE

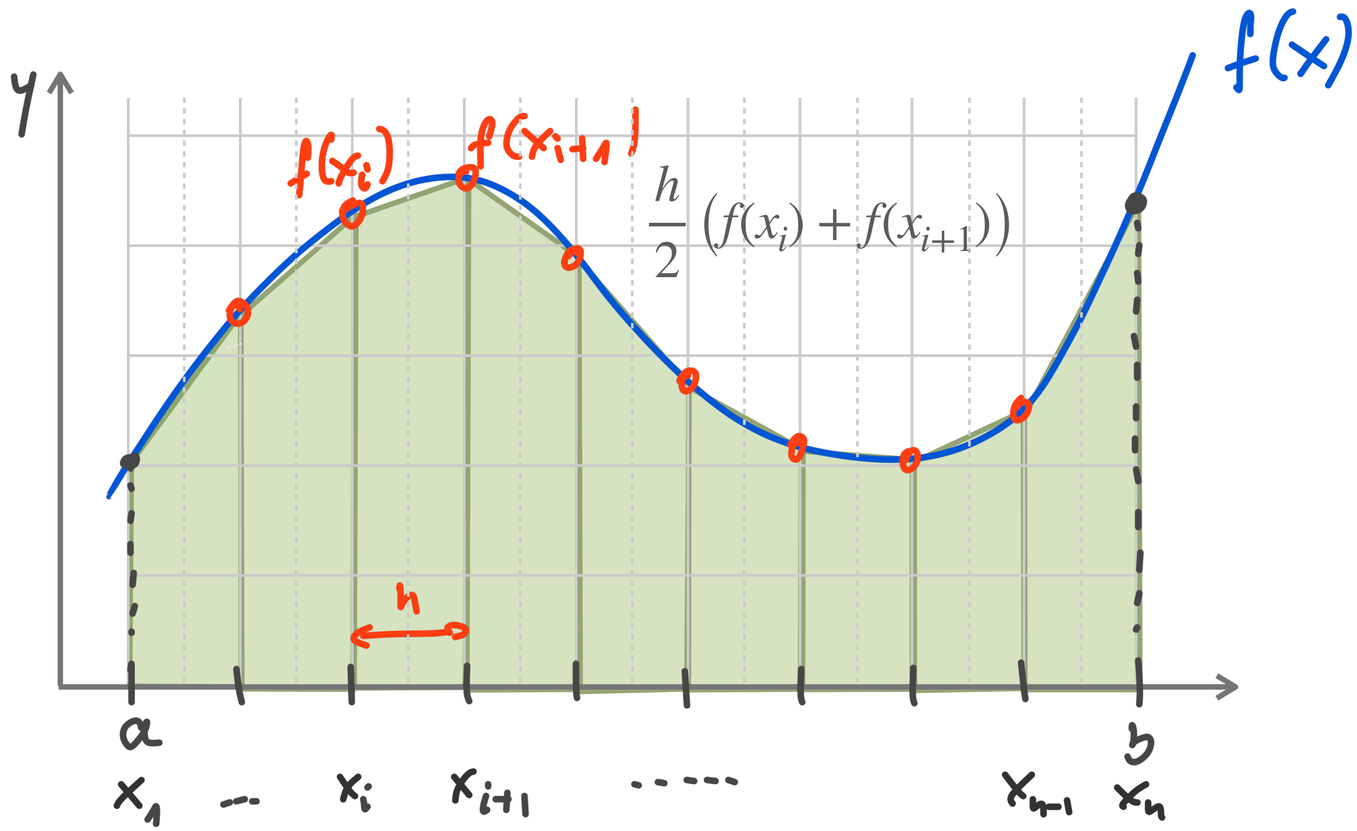


EXAMPLE – INTEGRATE A FUNCTION

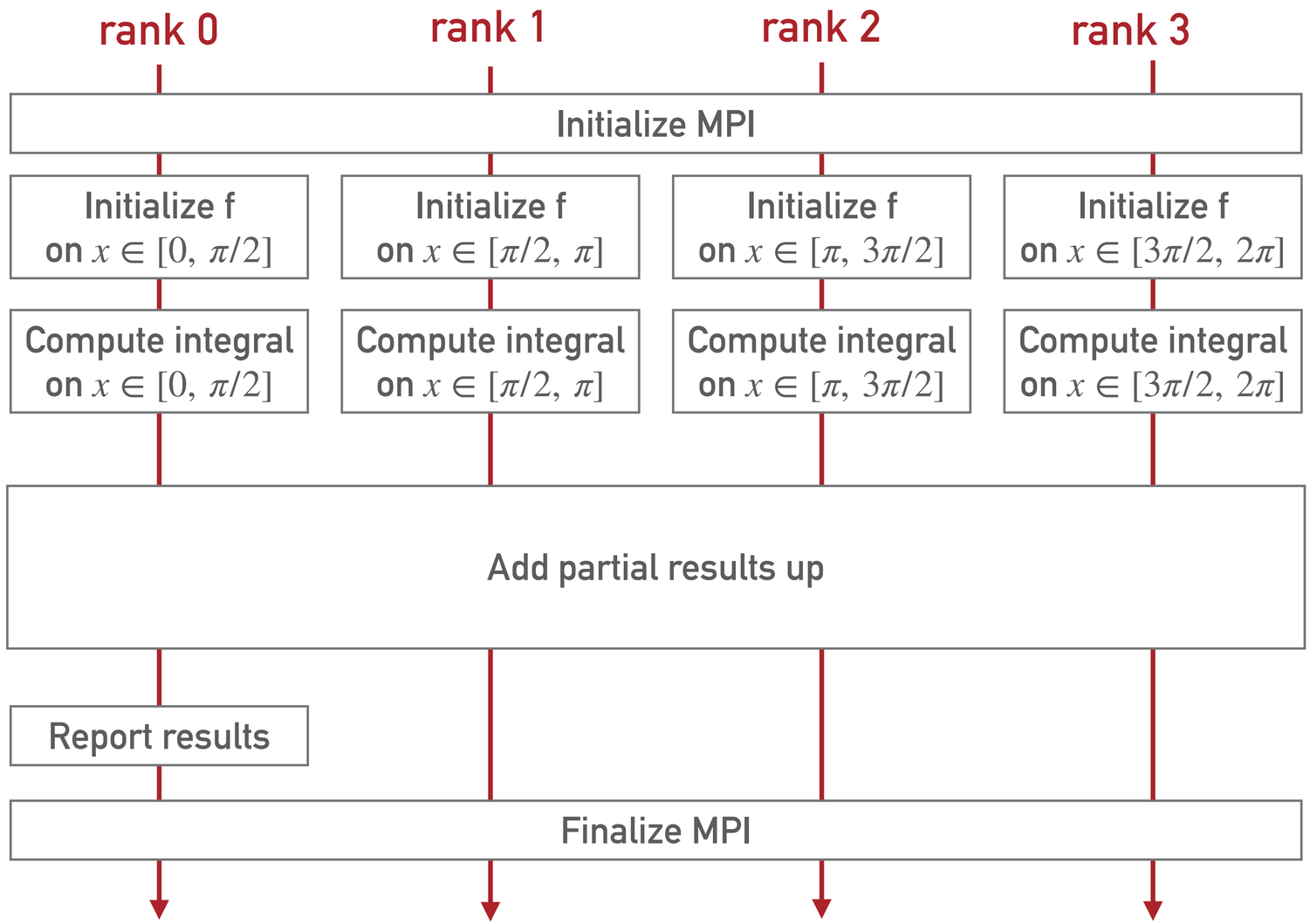
Integrate $f(x) = \sin(x)$ *on* $x \in [0,\ 2\pi]$ *using trapezoidal rule:*

$$\int_a^b f(x)\ dx \approx \sum_{i=0}^{N-1} \frac{1}{2} h \left(f(x_i) + f(x_{i+1}) \right)$$

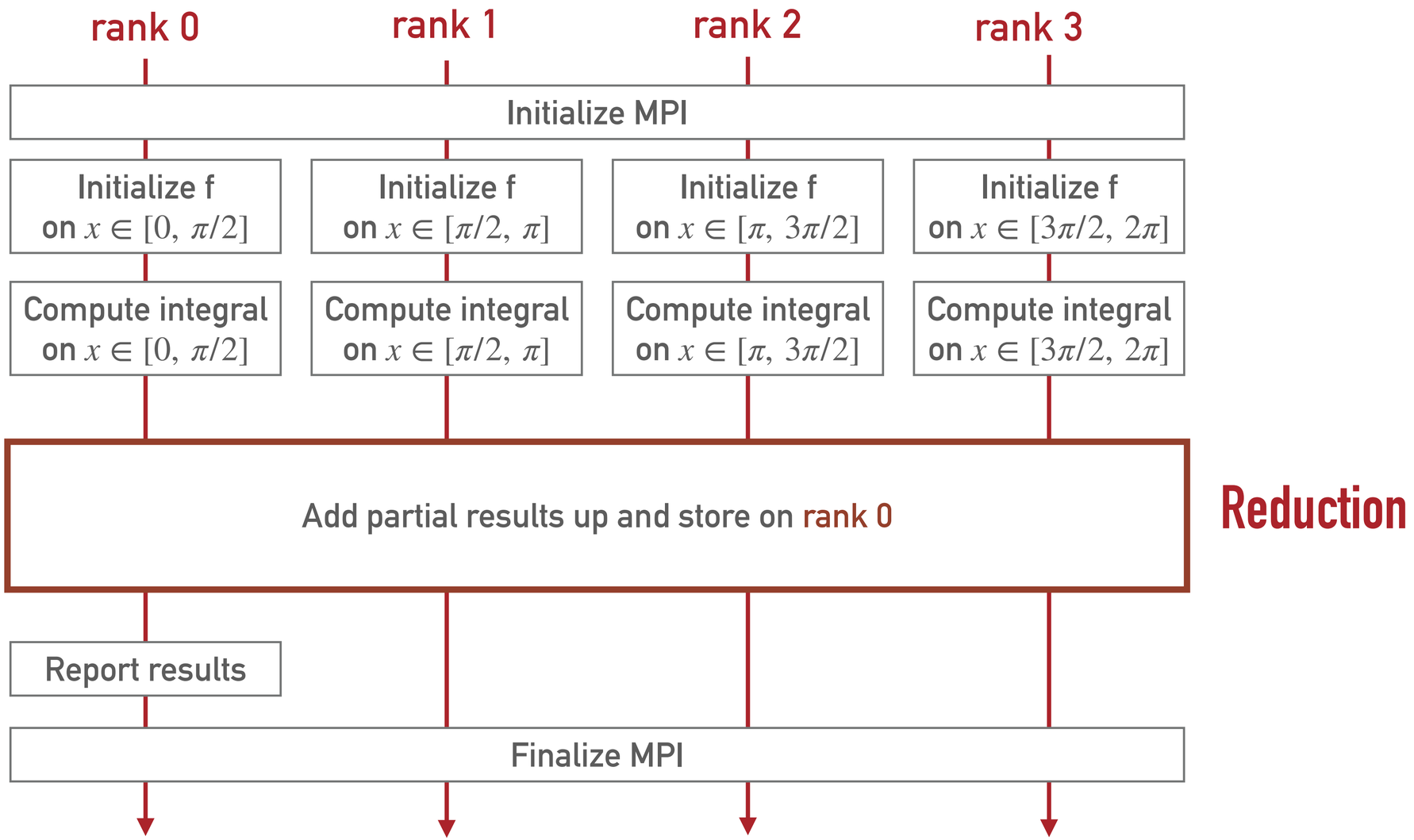




TRAPEZOIDAL RULE EXAMPLE

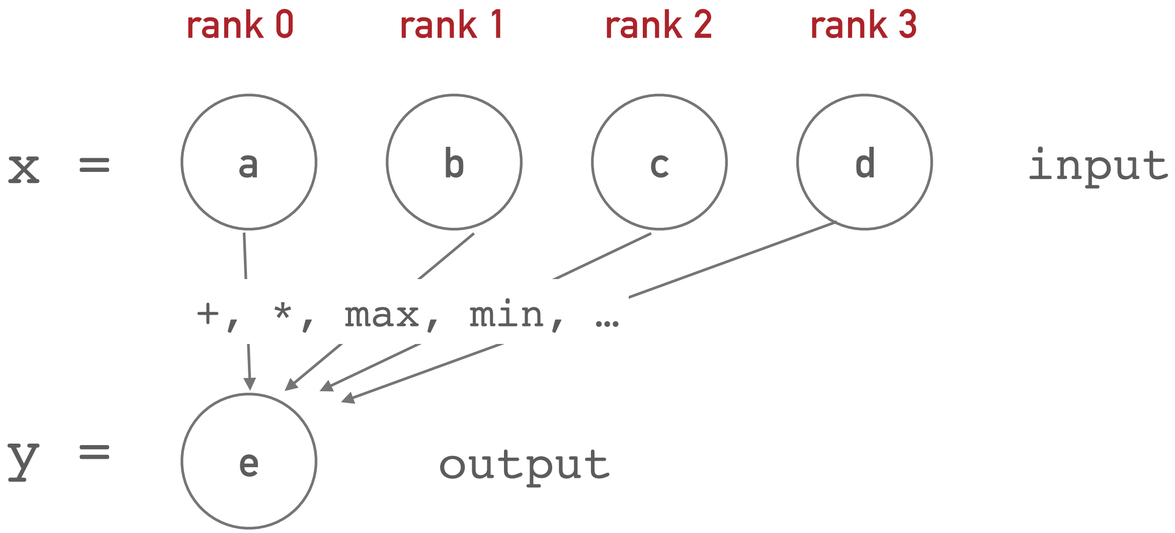


TRAPEZOIDAL RULE EXAMPLE



Reduction

COLLECTIVE COMMUNICATION PATTERNS



Reduction

```
MPI_Reduce(input, output, count, datatype, operation, root, communicator);
```

which variable to reduce

variable for the result

how many data elements to send per rank

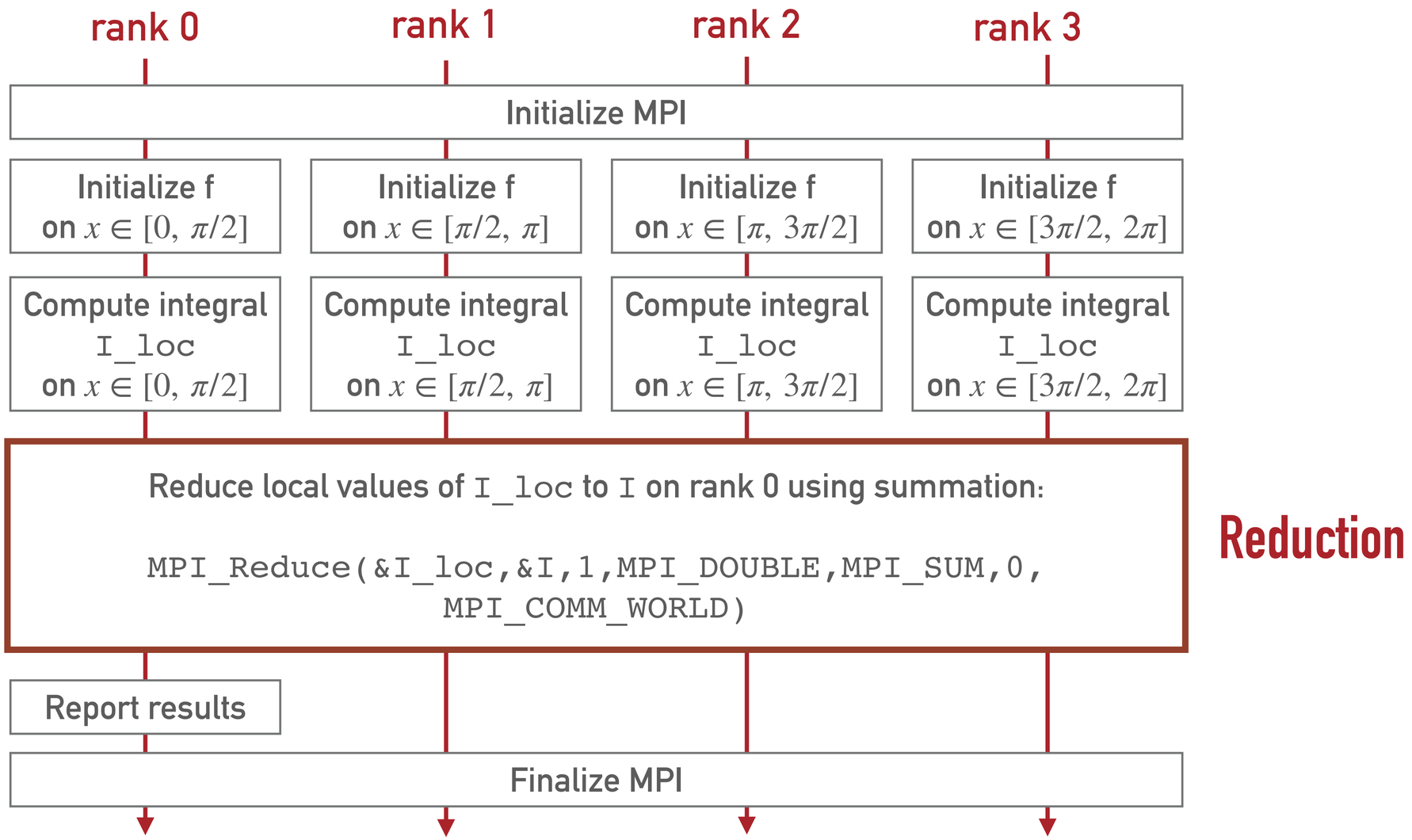
what type?

```
MPI_SUM
MPI_PROD
MPI_MAX
MPI_MIN
…
```

which rank is receiving?

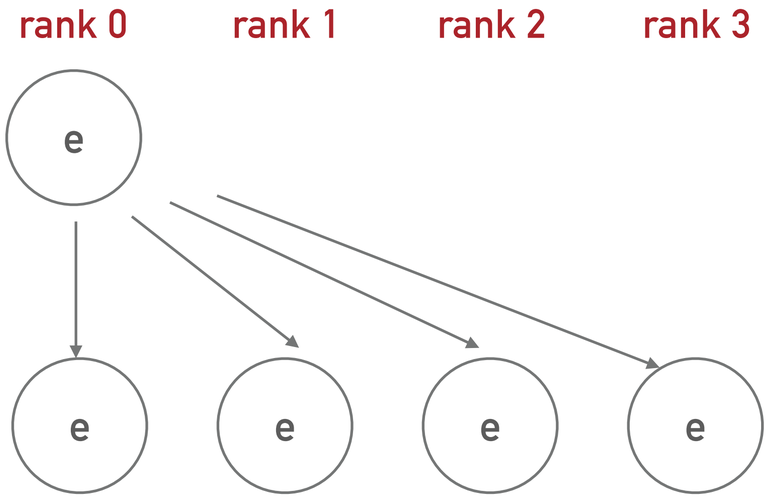
which communicator to use?

TRAPEZOIDAL RULE EXAMPLE



Reduction

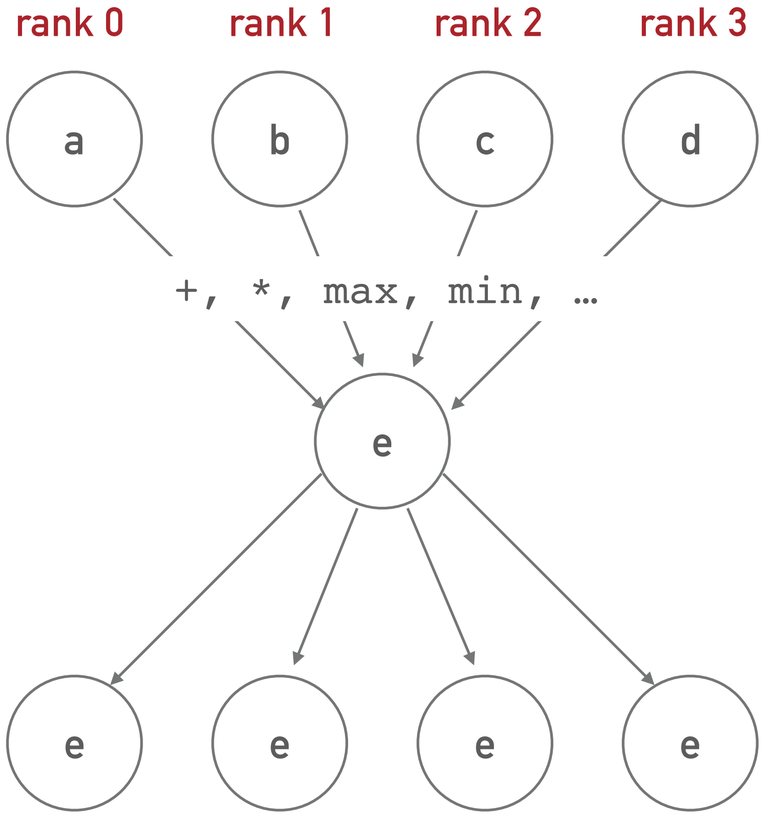
COLLECTIVE COMMUNICATION PATTERNS



Broadcast

```
MPI_Bcast(data, count, datatype, root, communicator);
```

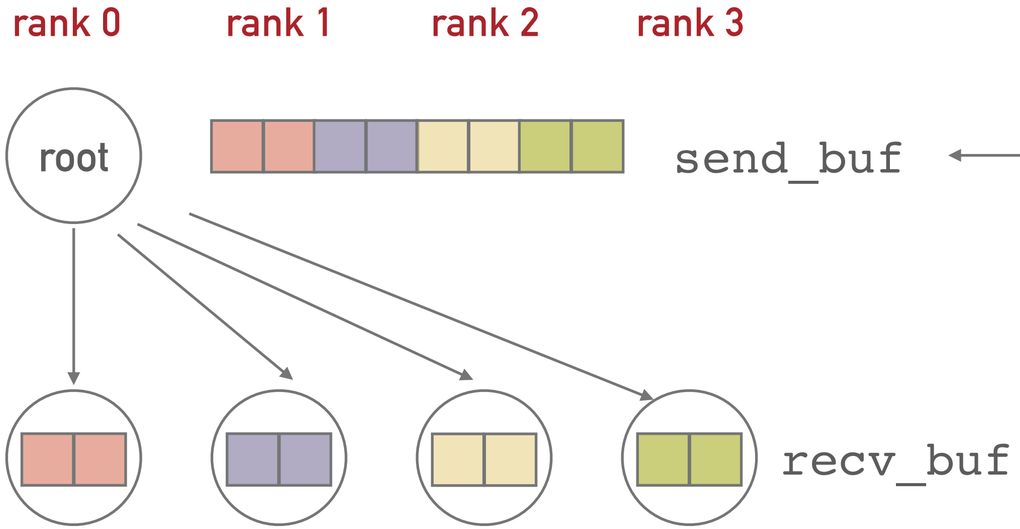
COLLECTIVE COMMUNICATION PATTERNS



Reduction + Broadcast = Allreduce

```
MPI_Allreduce(input, output, count, datatype, operator, communicator);
```

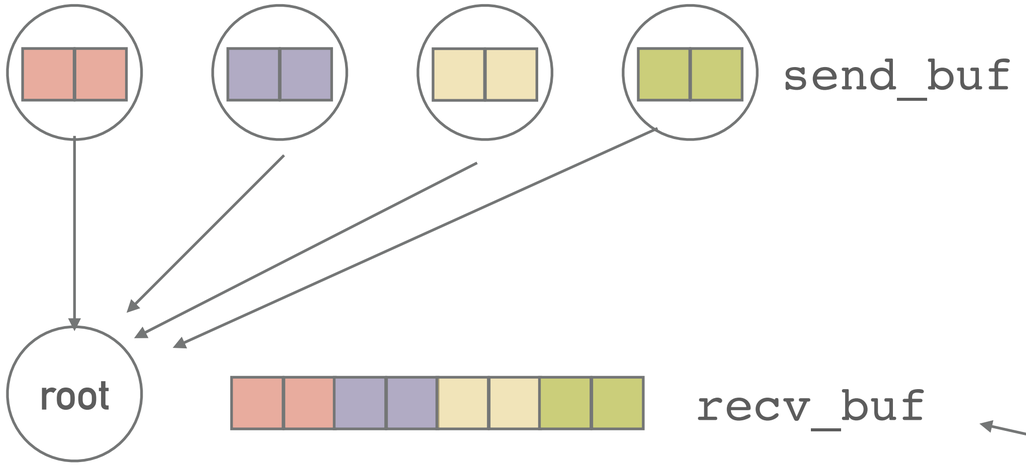
COLLECTIVE COMMUNICATION PATTERNS



Scatter

```
MPI_Scatter(send_buf, send_count, send_type,…
            recv_buf, recv_count, recv_type,…
            root, communicator);
```

COLLECTIVE COMMUNICATION PATTERNS

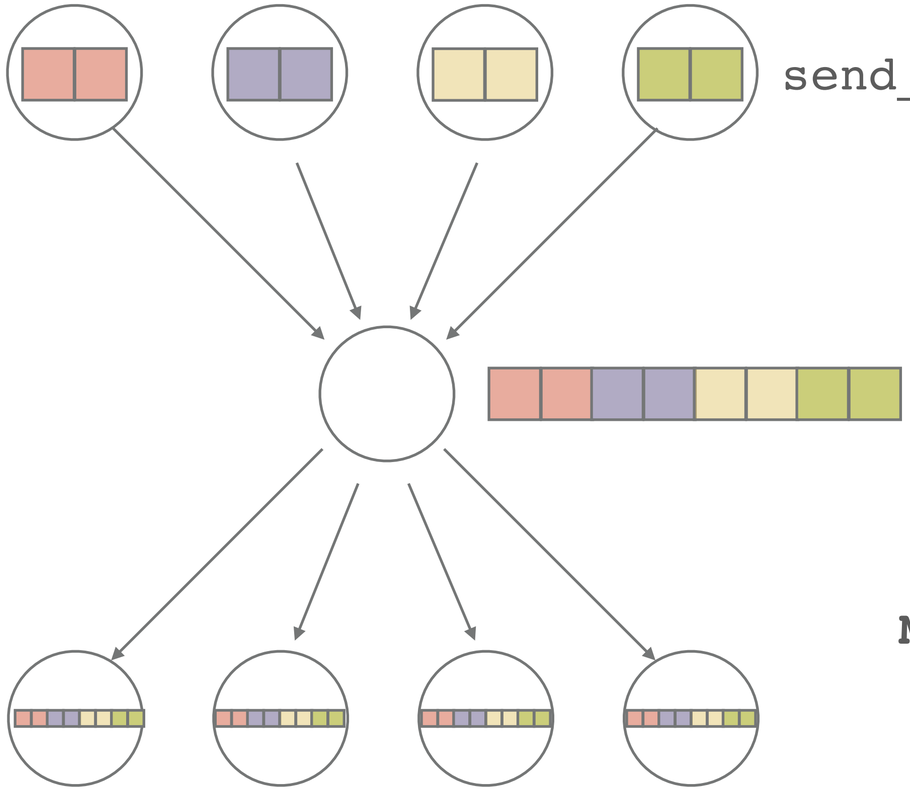


Gather

exists only on root!

```
MPI_Gather(send_buf, send_count, send_type,…
           recv_buf, recv_count, recv_type,…
           root, communicator);
```

COLLECTIVE COMMUNICATION PATTERNS

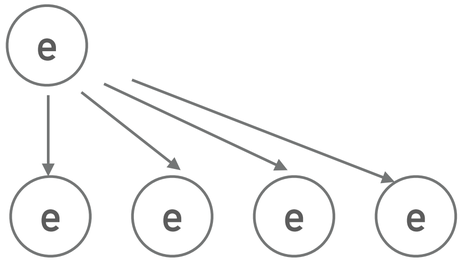


Gather + Scatter = Allgather

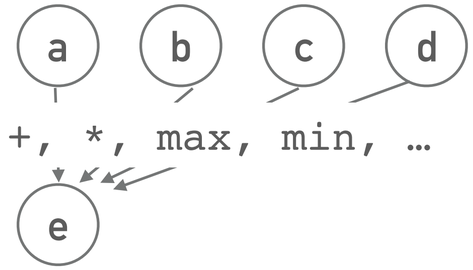
```
MPI_Allgather(send_buf, send_count, send_type,…
              recv_buf, recv_count, recv_type,…
              communicator);
```

CLASSROOM EXAMPLE

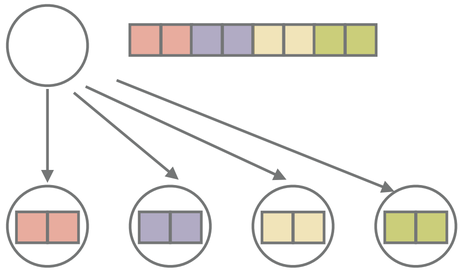
1. *An instructor is announcing the date of the test.* — broadcast
2. *The students write the test and the instructor is collecting their individual papers.* — gather
3. *The instructor gives the papers back to students.* — scatter
4. *The instructor computes the average score.* — reduction



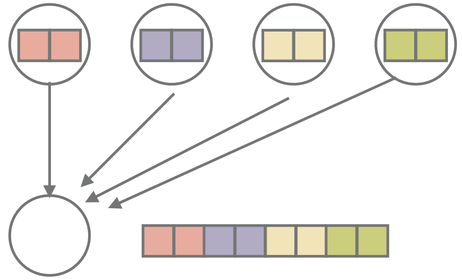
Broadcast



Reduction

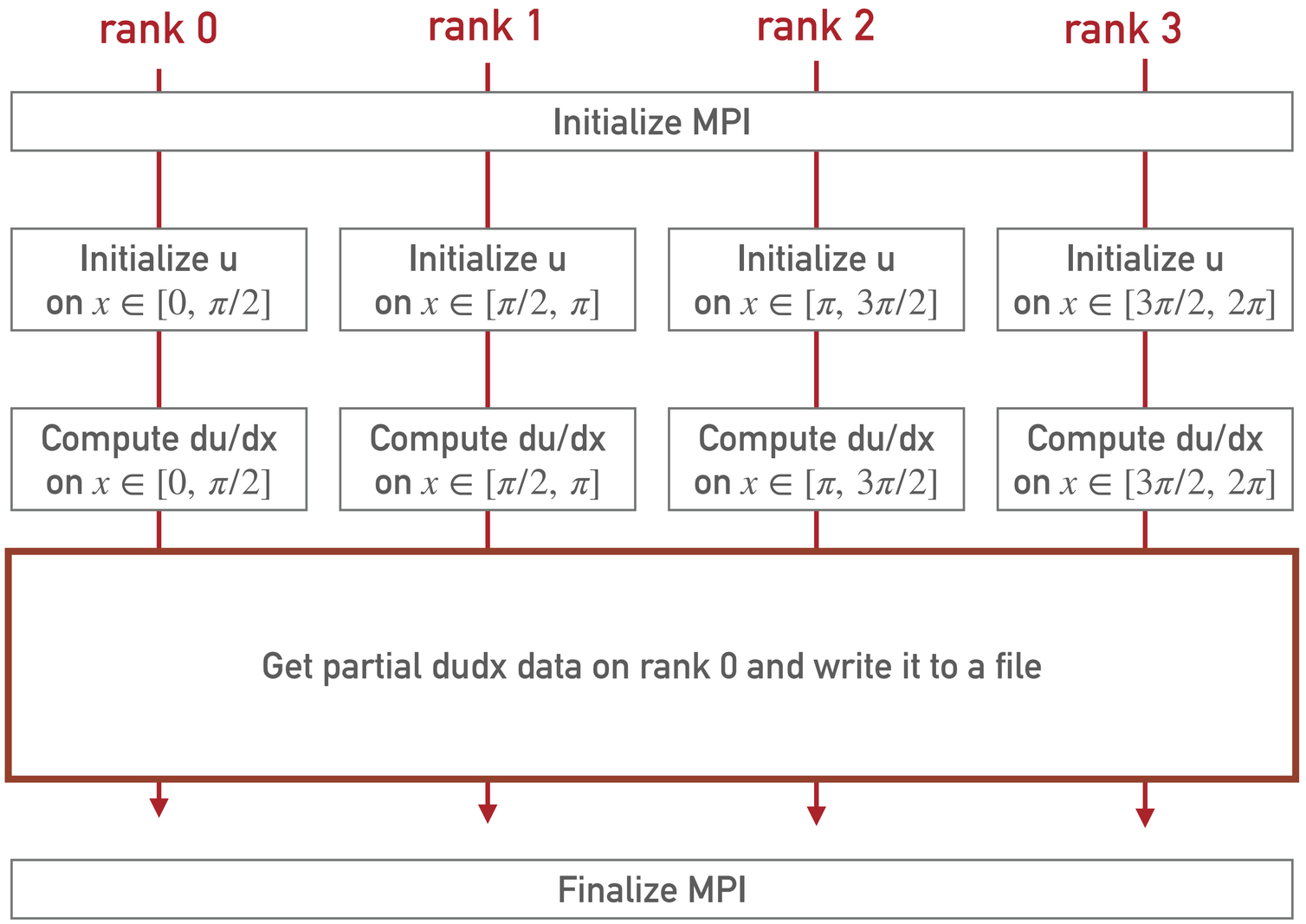


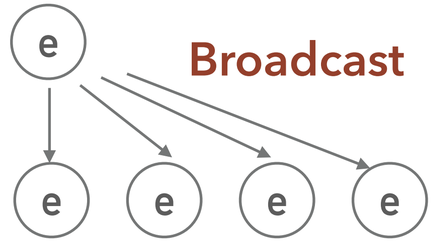
Scatter

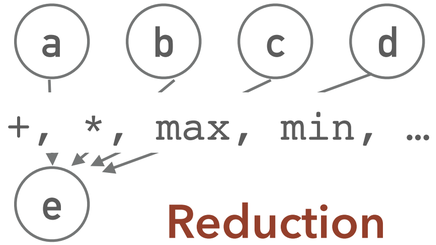


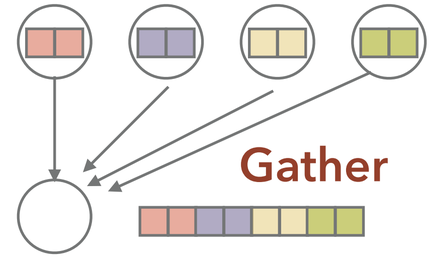
Gather

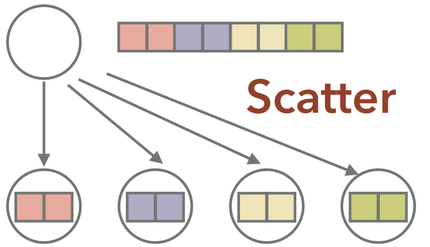
DERIVATIVE EXAMPLE



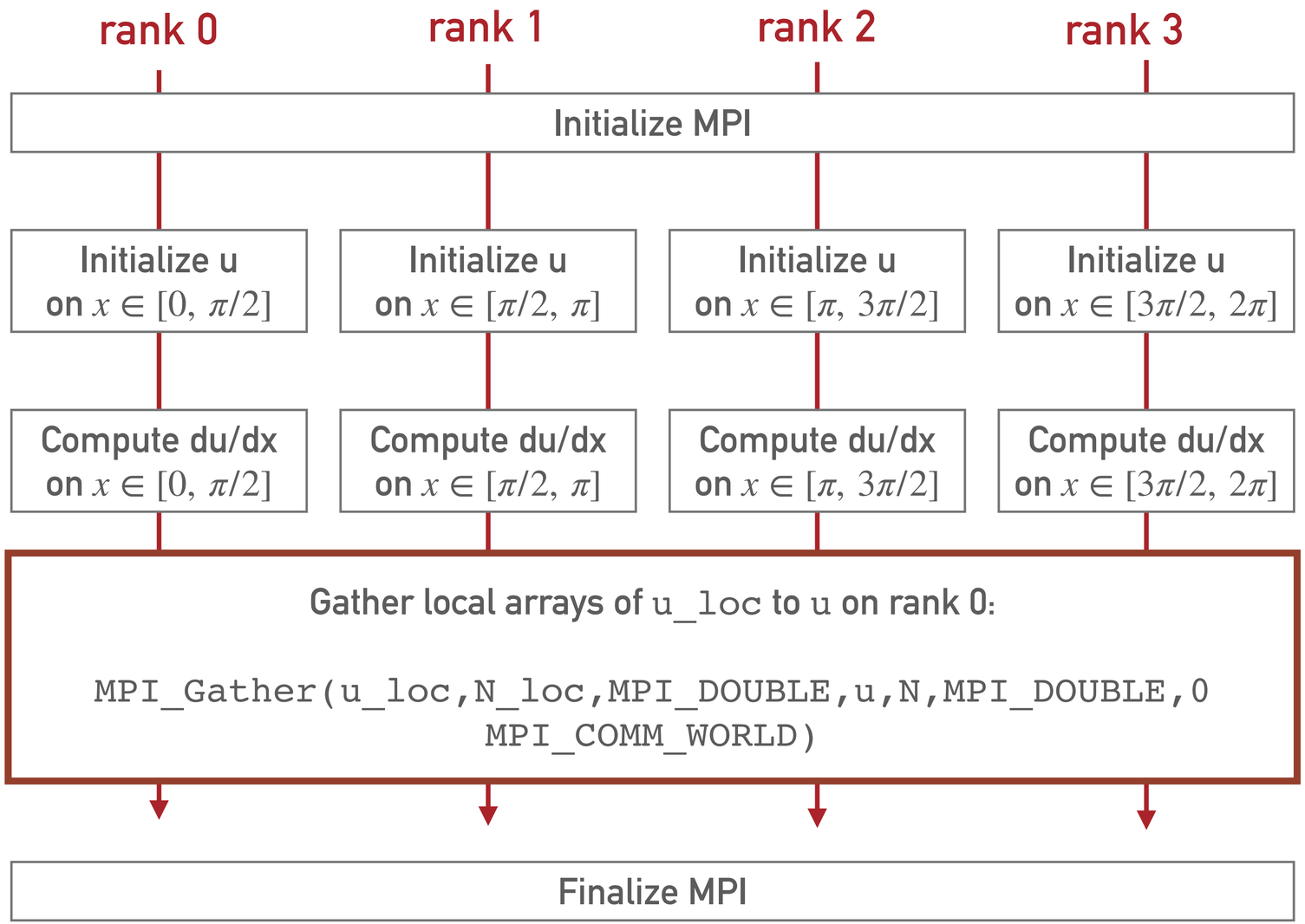








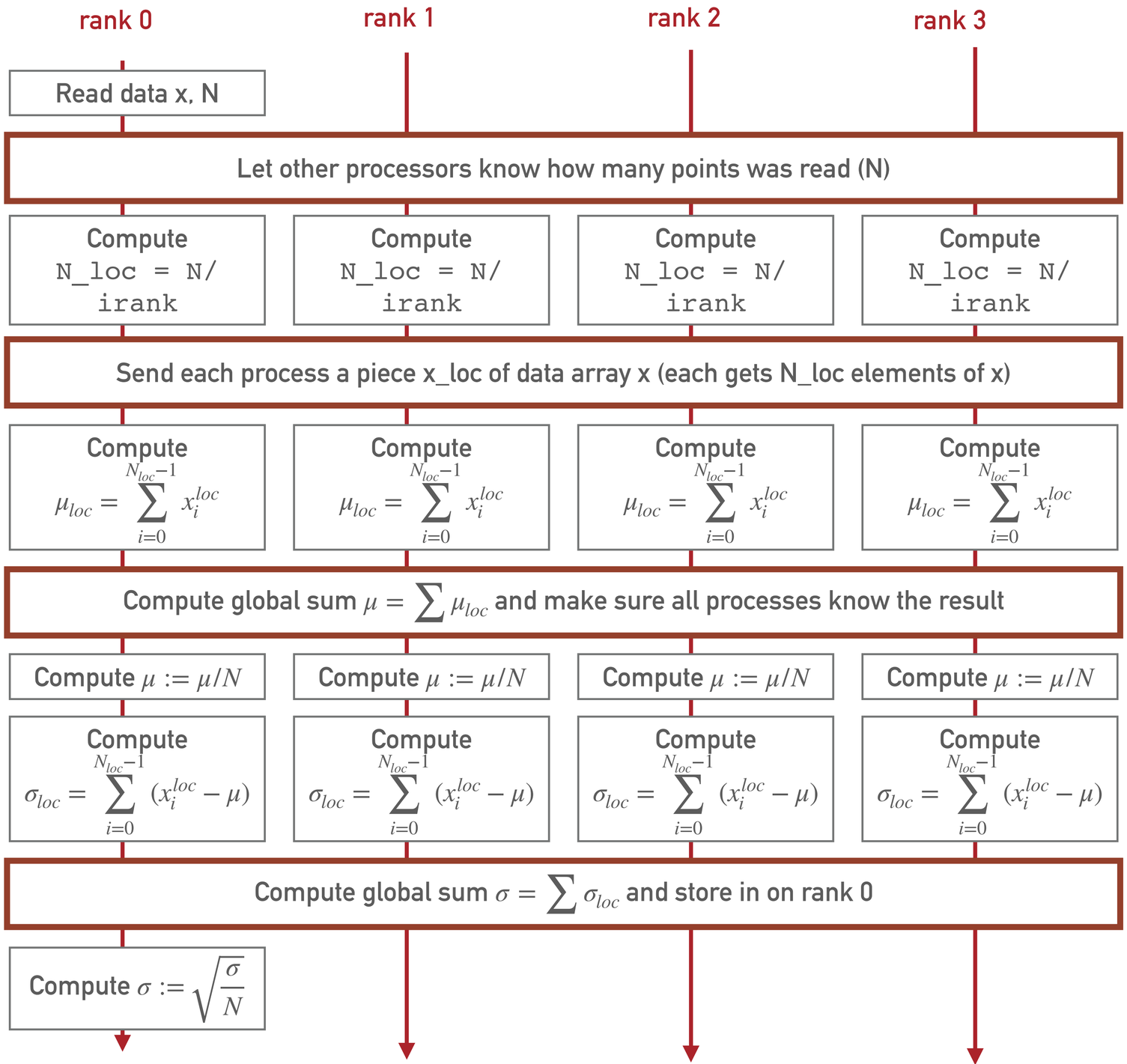
DERIVATIVE EXAMPLE

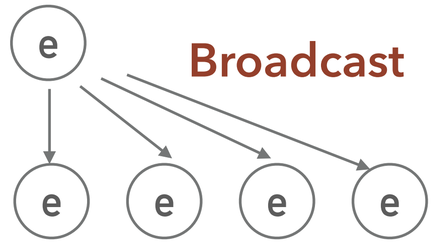


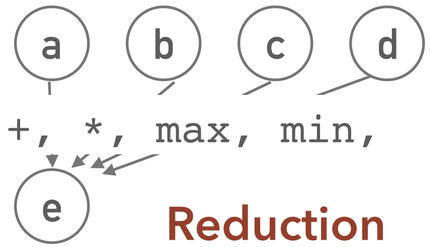
EXAMPLE

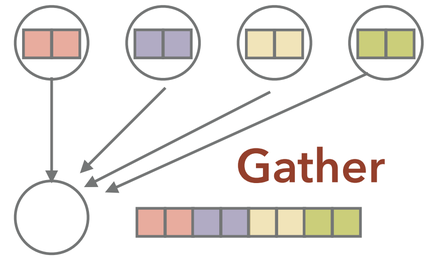
Imagine you need to compute a standard deviation of a large set of numbers saved in a file. How would you design a parallel algorithm using Reduce, Broadcast, Scatter, Gather, … functionalities?

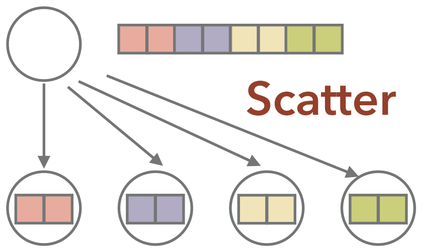
$$\sigma = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(x_i - \mu)} \qquad \textit{where} \qquad \mu = \frac{1}{N}\sum_{i=0}^{N-1} x_i$$


rank 0
rank 1
rank 2
rank 3
Read data x, N
Let other processors know how many points was read (N)
Compute N_loc = N/ irank
Compute N_loc = N/ irank
Compute N_loc = N/ irank
Compute N_loc = N/ irank
Send each process a piece x_loc of data array x (each gets N_loc elements of x)
Compute $\mu_{loc} = \sum_{i=0}^{N_{loc}-1} x_i^{loc}$
Compute $\mu_{loc} = \sum_{i=0}^{N_{loc}-1} x_i^{loc}$
Compute $\mu_{loc} = \sum_{i=0}^{N_{loc}-1} x_i^{loc}$
Compute $\mu_{loc} = \sum_{i=0}^{N_{loc}-1} x_i^{loc}$
Compute global sum $\mu = \sum \mu_{loc}$ and make sure all processes know the result
Compute $\mu := \mu/N$
Compute $\mu := \mu/N$
Compute $\mu := \mu/N$
Compute $\mu := \mu/N$
Compute $\sigma_{loc} = \sum_{i=0}^{N_{loc}-1} (x_i^{loc} - \mu)$
Compute $\sigma_{loc} = \sum_{i=0}^{N_{loc}-1} (x_i^{loc} - \mu)$
Compute $\sigma_{loc} = \sum_{i=0}^{N_{loc}-1} (x_i^{loc} - \mu)$
Compute $\sigma_{loc} = \sum_{i=0}^{N_{loc}-1} (x_i^{loc} - \mu)$
Compute global sum $\sigma = \sum \sigma_{loc}$ and store in on rank 0
Compute $\sigma := \sqrt{\frac{\sigma}{N}}$


e
e
e
e
e
Broadcast


a
b
c
d
+, *, max, min,
e
Reduction


Gather


Scatter