FARABI UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE



Lecture 6.
Organize Parallel Scientific Computing Using MPI

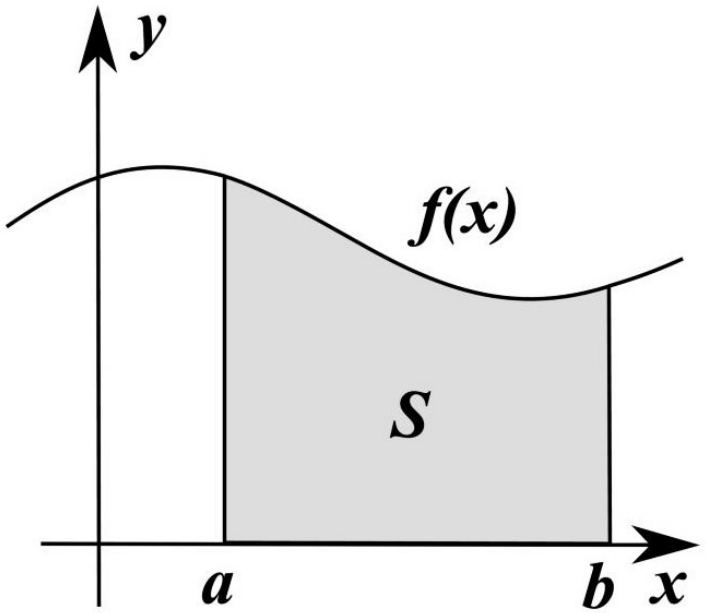
Dr. Bazargul Matkerim, Professor-Researcher

Outline

1. **Developing parallel programs**
   think about the big picture

2. **Example 1: global sum**
   partial sums, broadcast, send and receive

3. **Example 2: trapezoidal rule**
   numerical integration in parallel



Source: Integral as area, commons.wikimedia.org (CC BY-SA 3.0)

Developing Parallel Programs

Automatic conversion of serial programs is **difficult** and often **inefficient**

Parallelizing each step of a serial program may not be the best way

Look at the **big picture** and devise a new algorithm

$$S = \sum_{i=1}^{n} x_i$$

Example: compute n values and add them (serial code)

```
sum = 0;
for (i = 0; i < n; i++) {
    x = Compute_next_value(...);
    sum += x;
}
```

Source: Lecture 11 material, slide 3

Example 1: Partial Sums

We have **p** cores, **p ≪ n**

Each core computes a **partial sum** of n/p values

Each core uses its own **private variables**
and executes the block independently of the other cores

```
my_sum = 0;
my_first_i = . . . ;
my_last_i = . . . ;
for (my_i = my_first_i; my_i < my_last_i; my_i++) {
   my_x = Compute_next_value( . . .);
   my_sum += my_x;
}
```

Each core uses it's own private variables and executes this block of code independently of the other cores.

Source: Lecture 11 material, slide 4

Global Sum Example

8 cores, **24 values**

Each core adds 3 values: my_sum

Core 0 collects all my_sum values and computes the **global sum**

$$8 + 19 + 7 + 15 + 7 + 13 + 12 + 14 = 95$$

1,4,3, 9,2,8, 5,1,1, 5,2,7, 2,5,0, 4,1,8, 6,5,1, 2,3,9

Core	0	1	2	3	4	5	6	7
my_sum	8	19	7	15	7	13	12	14

Global sum

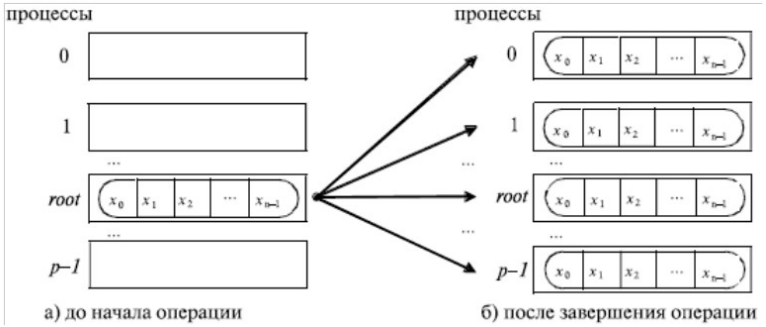
8 + 19 + 7 + 15 + 7 + 13 + 12 + 14 = 95

Core	0	1	2	3	4	5	6	7
my_sum	95	19	7	15	7	13	12	14

Source: Lecture 11 material, slide 5

Parallel Algorithm

1. **Distribute** the data to the processes
2. Each process adds the elements of its **own block**
3. Send the **local sum** to process 0
4. Process 0 receives the local sums and computes the **global sum**



Source: Lecture 11 material, slide 6

MPI Program

```
if (ProcRank == 0) DataInitialization(x, N);
MPI_Bcast(x, N, MPI_DOUBLE, 0, MPI_COMM_WORLD);
k  = N / ProcNum;
i1 = k * ProcRank;
i2 = k * (ProcRank + 1);
if (ProcRank == ProcNum-1) i2 = N;
for (int i = i1; i < i2; i++) ProcSum += x[i];
if (ProcRank == 0) {
  TotalSum = ProcSum;
  for (int i = 1; i < ProcNum; i++) {
    MPI_Recv(&ProcSum, 1, MPI_DOUBLE, MPI_ANY_SOURCE, 0,
             MPI_COMM_WORLD, &Status);
    TotalSum += ProcSum; }
} else
  MPI_Send(&ProcSum, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
```

Source: Lecture 11 material, slide 8 (main part)

MPI_Bcast: send data to all

i1, i2: my block

MPI_Send: local sum to 0

MPI_Recv with MPI_ANY_SOURCE

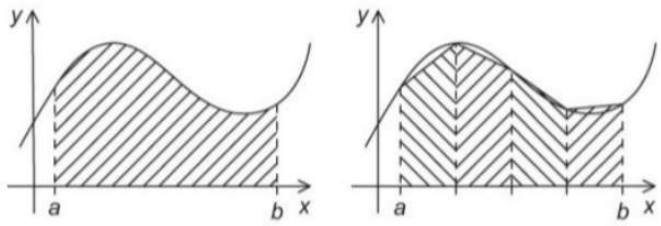
The last process takes the rest of the data

Example 2: The Trapezoidal Rule

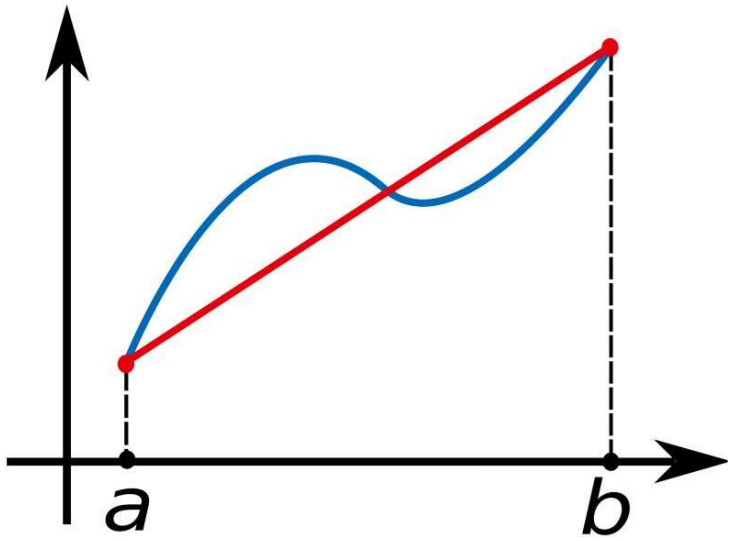
Integral = **area** under the curve y = f(x)

Divide [a, b] into **n** subintervals

Approximate the area by **trapezoids**



Source: Lecture 11 material, slide 10



Source: Trapezoidal rule, commons.wikimedia.org (public domain)

One Trapezoid

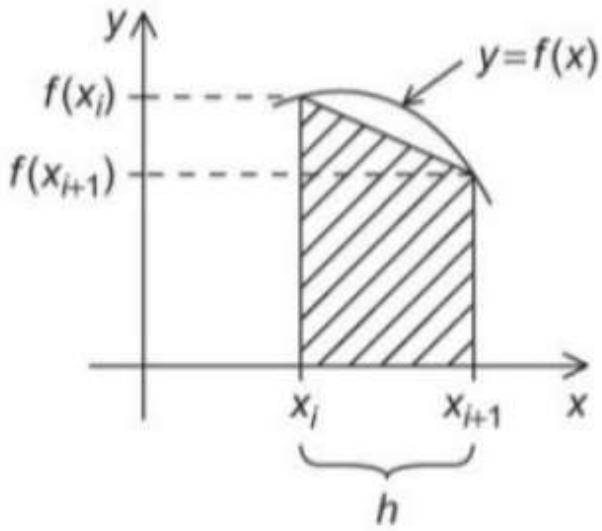
Width **h**, heights f(x_i) and f(x_i+1)

All points are equally spaced

Formulas

$$area = \frac{h}{2}[f(x_i) + f(x_{i+1})]$$

$$h = \frac{b - a}{n}, \qquad x_i = a + i \cdot h$$



Source: Lecture 11 material, slide 11

Trapezoidal Rule for Numerical Integration

Sum of trapezoid areas

$$\int_a^b f(x)\,dx \approx h\,[\, f(x_0)/2 + f(x_1) + \cdots + f(x_{n-1}) + f(x_n)/2 \,]$$

Serial program (pseudo-code)

```
/* Input: a, b, n */
h = (b-a)/n;
approx = (f(a) + f(b))/2.0;
for (i = 1; i <= n-1; i++) {
    x_i = a + i*h;
    approx += f(x_i);
}
approx = h*approx;
```

End points get weight **1/2**

Inner points get weight **1**

More trapezoids (larger n) give a **better** result

Source: Lecture 11 material, slide 13

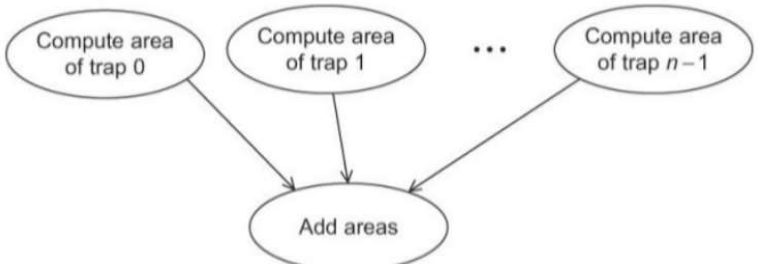
Parallelizing the Trapezoidal Rule

Partition the problem into tasks

Identify communication channels between tasks

Aggregate tasks into composite tasks

Map composite tasks to cores



Source: Lecture 11 material, slide 15

Tasks: compute the area of each trapezoid, then add the areas

Parallel Trapezoidal Rule with MPI

Work of one process

$$local_n = \frac{n}{p},$$
$$local_a = a + rank \cdot local_n \cdot h$$

Each process integrates its own **subinterval**

Local results are added with MPI_Reduce (MPI_SUM)

Result on process 0

```
local_n = n / p;
local_a = a + my_rank*local_n*h;
local_b = local_a + local_n*h;
local_int = Trap(local_a, local_b, local_n, h);
MPI_Reduce(&local_int, &total_int, 1,
           MPI_DOUBLE, MPI_SUM, 0,
           MPI_COMM_WORLD);
```

Code sketch of the task graph above