MPI — An example

Salvatore Filippone

salvatore.filippone@uniroma2.it

An example problem

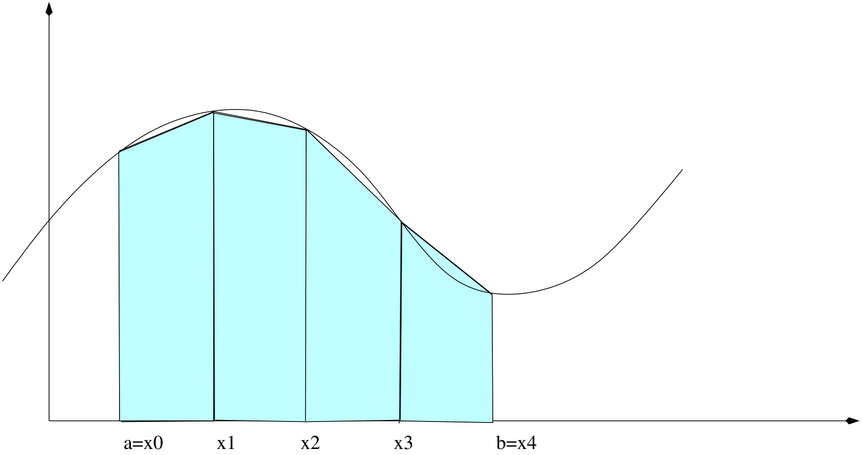
The trapezoidal rule for computing a definite integral

If we have a sequence of points $a = x_0, x_1, x_2, \dots, x_{n-1}, b = x_n$, with constant $h = x_k - x_{k-1}$, and a function $f(x)$, we can approximate

$$\int_a^b f(x) \approx \sum_{i=1}^{n} \frac{1}{2} h \left[f(x_i) + f(x_{i-1}) \right] = h \cdot \left\{ \frac{f(x_0)}{2} + \frac{f(x_n)}{2} + \sum_{i=1}^{n-1} f(x_i) \right\}.$$

An example problem

The trapezoidal rule for computing a definite integral



Definite integral — serial

```c
#include <stdio.h>

float f(float x);

main(int argc, char* argv[]) {
  float integral;
  float a, b;
  int n;
  float h;
  int i;

  printf("Enter a, b and n\n");
  scanf("%f %f %d",&a,&b,&n);

  h=(b-a)/n;
  integral = (f(a)+f(b))/2.0;
  for (i=1; i<n; i++){
    x = a+i*h;
    integral = integral + f(x);
  }
  integral *= h;
  printf("With n=%d trapezoids we estimate integral from %f to %f:  %f\n",
         n,a,b,integral);
}
```

Definite integral — parallel

Issues we need to figure out:

- Who gets the input;
- Who writes the output;
- How does everybody know what to do;

- How do you distribute the workload;
- What exactly is each process supposed to do;

- How is the team going to complete the process.

Definite integral — parallel

Issues we need to figure out:

- Who gets the input; Process 0
- Who writes the output; Process 0
- How does everybody know what to do; Based on process rank; receive information from process 0
- How do you distribute the workload; As uniformly as possible;
- What exactly is each process supposed to do; Identify which portion it will compute, then compute
- How is the team going to complete the process. Send the results to process 0 which will assemble the final result

Definite integral — parallel

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

float f(float x);
float Trap(float local_a, float local_b,
           int local_n, float h);

main(int argc, char* argv[]) {
  int p, my_rank;
  MPI_Status  status;
  int i,tag;

  MPI_Init(&argc, &argv);

  MPI_Comm_rank(MPI_COM_WORLD, &my_rank);
  MPI_Comm_size(MPI_COM_WORLD, &p);
```

Definite integral — parallel

Initial communication

```
tag = 0;
if (my_rank == 0) {
  printf("Enter a, b and n\n");
  scanf("%f %f %d",&a,&b,&n);
}
if (my_rank == 0) {
  for (i=1; i<p; i++) {
    MPI_Send(&a,1,MPI_FLOAT,i,tag,MPI_COMM_WORLD);
    MPI_Send(&b,1,MPI_FLOAT,i,tag,MPI_COMM_WORLD);
    MPI_Send(&n,1,MPI_INT,i,tag,MPI_COMM_WORLD);
  }
} else {
  MPI_Recv(&a,1,MPI_FLOAT,0,tag,MPI_COMM_WORLD);
  MPI_Recv(&b,1,MPI_FLOAT,0,tag,MPI_COMM_WORLD);
  MPI_Recv(&n,1,MPI_INT,0,tag,MPI_COMM_WORLD);
}
```

Definite integral — parallel

Local work setup & execution

```
float h, local_a, local_b, local_int;
int local_n,

h = (b-a)/n;
local_n = n/p;

// Start of local integration
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);
```

Definite integral — parallel

Collect data, print & close

```
float temp;

tag = 1;
if (my_rank == 0) {
  integral = local_int;
  for (i=1; i<p; i++){
    MPI_Recv(&temp,1,MPI_FLOAT,i,tag,
             MPI_COMM_WORLD,&status);
    integral = integral + temp;
  }
  printf("With n=%d trapezoids we estimate integral",n);
  printf(" from %f to %f:  %f\n",a,b,integral);
} else {
  MPI_Send(&local_int,1,MPI_FLOAT,0,tag,
           MPI_COMM_WORLD,&status);
}

MPI_Finalize();
```

Definite integral — parallel

What is the main problem with the previous code?

Definite integral — parallel

What is the main problem with the previous code?

We have implicitly assumed that `n` is exactly divisible by `p`! How do we fix this?

Definite integral — parallel

What is the main problem with the previous code?
We have implicitly assumed that n is exactly divisible by p! How do we fix this?

```
int r;
r=n%p;
if (r == 0) {
  local_n = n/p;
  local_a = a+my_rank*local_n*h;
 } else {
  if (my_rank < r) {
    local_n = n/p +1;
    local_a = a+my_rank*local_n*h;
  } else {
    local_n = n/p+1;
    local_a = a+r*local_n*h;
    local_n = n/p;
    local_a = local_a + (my_rank-r)*local_n*h;
  }
  local_b = local_a+local_n*h;
 }
```

Definite integral — parallel

Better yet

```
int r;
r=n%p;
 if (my_rank < r) {
   local_n = n/p +1;
   local_a = a+my_rank*local_n*h;
 } else {
   local_n = n/p+1;
   local_a = a+r*local_n*h;
   local_n = n/p;
   local_a = local_a + (my_rank-r)*local_n*h;
 }
 local_b = local_a+local_n*h;
```

(Why does it work?)

Definite integral — parallel

Anything else wrong with the previous code?

Definite integral — parallel

Anything else wrong with the previous code? Yes, the initial communication and the results collection

```
tag = 0;
if (my_rank == 0) {
  printf("Enter a, b and n\n");
  scanf("%f %f %d",&a,&b,&n);
 }
if (my_rank == 0) {
  for (i=1; i<p; i++) {
    MPI_Send(&a,1,MPI_FLOAT,i,tag,MPI_COMM_WORLD);
    MPI_Send(&b,1,MPI_FLOAT,i,tag,MPI_COMM_WORLD);
    MPI_Send(&n,1,MPI_INT,i,tag,MPI_COMM_WORLD);
  }
 } else {
  MPI_Recv(&a,1,MPI_FLOAT,0,tag,MPI_COMM_WORLD);
  MPI_Recv(&b,1,MPI_FLOAT,0,tag,MPI_COMM_WORLD);
  MPI_Recv(&n,1,MPI_INT,0,tag,MPI_COMM_WORLD);
 }
```

Definite integral — parallel

Next topic: Collective Communications

Exercise

Define and implement an MPI code that performs the Matrix-Vector product

$$y \leftarrow Ax;$$

consider as many variations as possible for the parallel data layout