Home Assignment 3

Yuxuan Jing

March 2020

Question 3.2

Replace line 12:

```
1 int main(void){
2 int my_rank, comm_sz, n = 1024, local_n;
3 double a = 0.0, b = 3.0, h, local_a, local_b;
4 double local_int, total_int;
5 int source;
6
7 MPI_Init(NULL, NULL);
8 MPI_Comm rank(MPI_COMM_WORLD, &my_rank);
9 MPI_Comm size(MPI_COMM_WORLD, &comm_sz);
10
11 h = (b-a)/n;
12 if (local_b>b) { local_b = b; local_n=(local_b-local_a)/h; }
    \*replace the original code*\
13
14 local_a = a + my rank*local_n*h;
15 local_b = local_a + local_n*h;
16 local_int = Trap(local_a, local_b, local_n, h);
17
18 if (my_rank != 0) {
19 MPI_Send(&local_int, 1, MPI_DOUBLE, 0, 0,
20 MPI_COMM_WORLD);
21 } else {
22 total_int = local_int;
23 for (source = 1; source < comm sz; source++) {
24 MPI_Recv(&local_int, 1, MPI_DOUBLE, source, 0,
25 MPI_COMM_WORLD, MPI_STATUS_IGNORE);
26 total_int += local_int;
27 }
28 }
29
30 if (my_rank == 0) {
31 printf("With n = %d trapezoids, our estimate\n", n);
32 printf("of the integral from %f to %f = %.15e\n",
33 a, b, total_int);
```

```
}
MPI_Finalize();
return 0;
}
```

Question 3.5

We have that:

Table 1: complete binary tree with n leaves will have depth $log2n$.

Leaves	Depth
2	1
4	2
8	3
16	4
32	5

A complete binary tree with n leaves will have depth $log2n$, table 1.

Question 3.6

3.6.a

Table 2: Block.

processor_0	0	1	2	3
processor_1	4	5	6	7
processor_2	8	9	10	11
processor_3	12	13	NA	NA

3.6.b

Table 3: Cyclic.

processor_0	0	4	8	12
processor_1	1	5	9	13
processor_2	2	6	10	NA
processor_3	3	7	11	NA

3.6.c

Table 4: Block-Cyclic, BS = 2.

processor_0	0	1	8	9
processor_1	2	3	10	11
processor_2	4	5	12	13
processor_3	6	7	NA	NA

Question 3.7

Calling MPI_Reduce just copies the value input from process 0 to process 0. For a single process MPI_AllReduce is equalt to a call to MPI_Reduce or MPI_Bcast.

Question 3.8

3.8.a

See figure 1

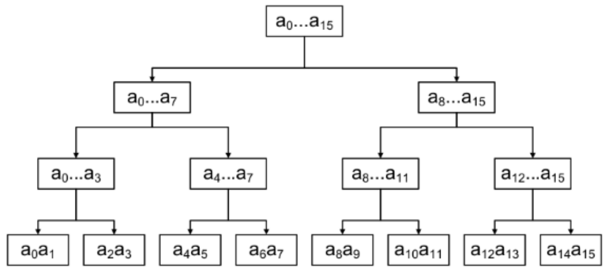

Figure 1: MPI Scatter

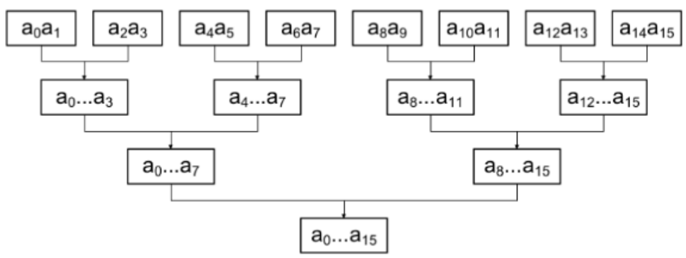

Figure 2: MPI Gather

3.8.b

See figure 2

Question 3.9

See the codes

Our target is to implement:

$$R = [x_1, ..., x_n] * [y_1, ..., y_n] * z \tag{1}$$

We separate x, and y into p segment, where p is the number of processors.

For each node, they compute:

$$local\_sum = [x_1, ..., x_{local\_n}] * [y_1, ..., y_{local\_n}] * z \tag{2}$$

,where

$$local\_n = n/p \tag{3}$$

The we use MPI_Gather to sum up the total sum from every local_sums.

$$R = totol\_sum = MPI\_Gather(local\_sum). \tag{4}$$

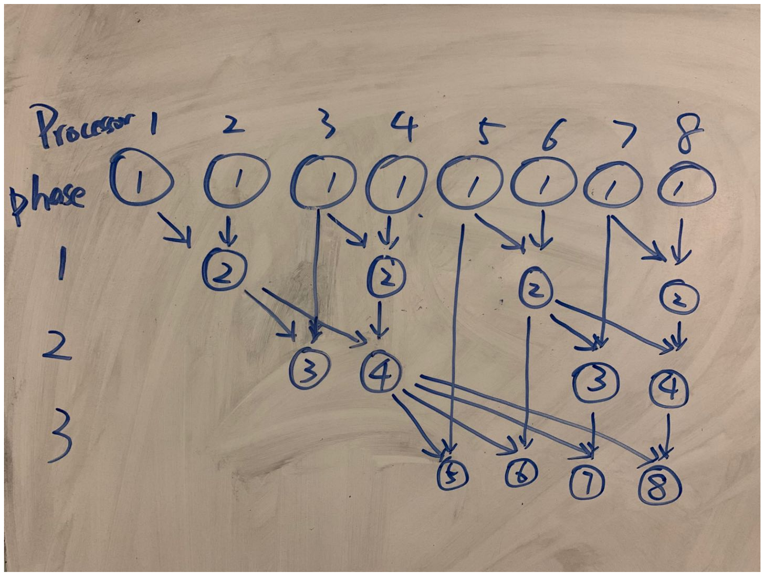

Figure 3: A tree-structured algorithm

Question 3.11

See the codes

The serial algorithm is trivial, and the code is self-explained.

The parallel algorithm is show in figure 3.

3.11.c

Using a tree structured algorithm. Only k communicate phases are needed. In this algorithm, for each nodes, only one local value need to be stored.

Question 3.12

See the codes

Assume we have p nodes. The ring-pass structure, compared with butterfly-structure:

- ring-pass structure need p phases to implement the MPI_Allreduce.
- butterfly-structure need $log_2 p$ phases to implement the MPI_Allreduce.