HPC Tools 2024, Exercise 7 solutions

Bruno Oliveira Cattelan, Laurent Chôné

In this problem, we examine the "fairness" of MPI communications, ie whether there is a preferred order. To do so, we write a program in which all processes send a message to rank 0, which receives them without order constraints (using `MPI_ANY_SOURCE` and `MPI_ANY_FLAG`). The answer turns out to be dependent on the particular setup, so we do the following experiments:

1. The program does 100 communication batches in one execution.
2. The program does 100 communication batches in one execution again.
3. The program does 1 communication batch in 100 executions.

We plot the order of communications below. We observe that, during experiments 1 and 2, there can significant correlation of the receiving order from one batch to the next. From one execution to the next however, the order is shuffled and the patterns are not reproduced. In experiment 3, we see that the order of communications appears essentially random from one execution to the next.

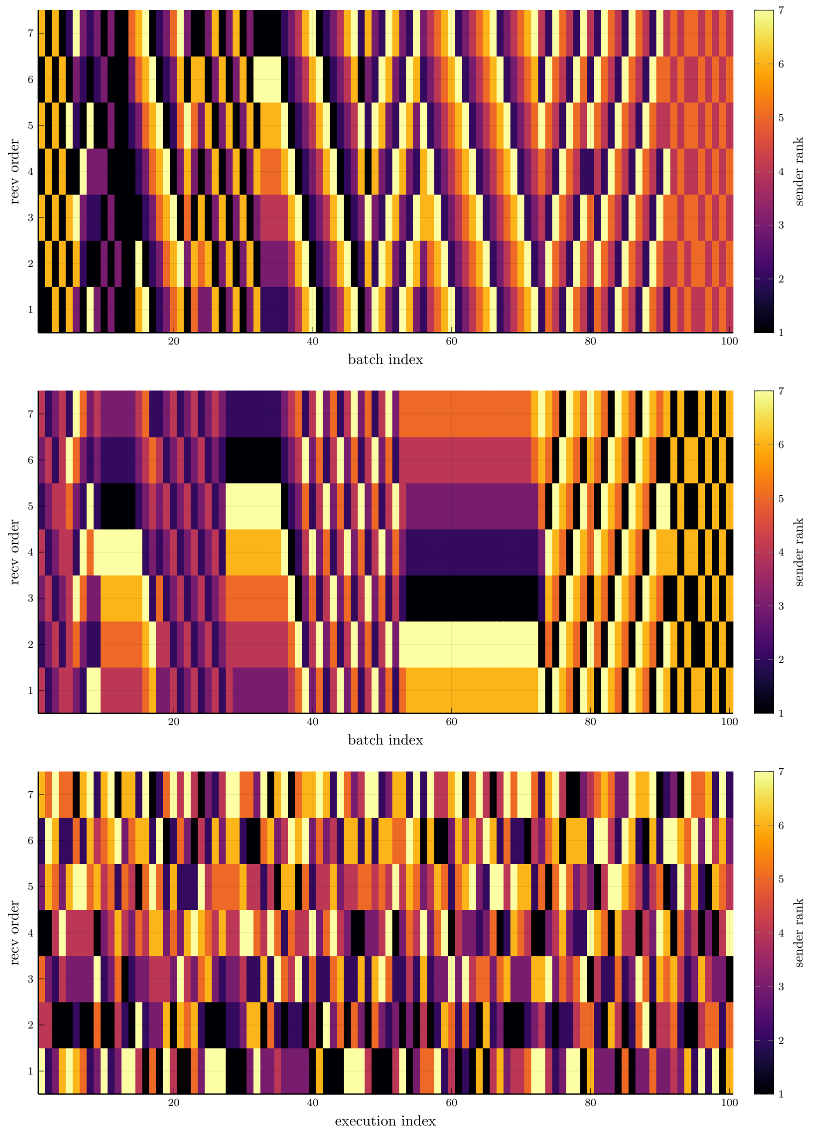
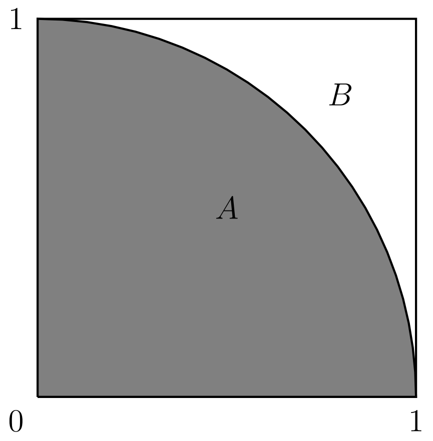


Figure 1: Order of message arrival for experiments 1 (top), 2 (middle), and 3 (bottom).

Problem 2

In this problem, we use MPI to parallelize the prototypical application of the Monte-Carlo integration method: calculating the value of π.

The method is illustrated in the figure below: the gray area A (under the curve $y = \sqrt{1 - x^2}$) is one fourth of that of a unit circle: $A_A = \frac{\pi}{4}$.



Let us pick N random points in the unit square $A \cup B$. Let N_A be the number of points in the region A. The ratio of point in A over points in $A \cup B$ tends to the ratio of areas of A and $A \cup B$:

$$\lim_{N\to\infty} \frac{N_A}{N_{A\cup B}} = \frac{A_A}{A_{A\cup B}}$$
$$\Rightarrow \lim_{N\to\infty} \frac{N_A}{N} = \frac{\pi}{4}$$

Hence, we have an approximation of π:

$$\tilde{\pi} = 4\frac{N_A}{N}$$

Since the random samples are independent, the value of $\tilde{\pi}$ can be evaluated simultaneously in several parallel processes, then averaged over all processes to obtain a more precise result. Using collective communications, this is easily achieved. Eg, with `localNumA` being the value of N_A evaluated by each process, and `globalNumA` the sum of all on process 0:

```
...
MPI_Reduce(&localNumA, &globalNumA, 1, MPI_INT, MPI_SUM, 0,
    MPI_COMM_WORLD);
double approxPi = (4.0 * globalNumA) / numSamples;
```

Where `MPI_Reduce` using `MPI_SUM` sums the values of `localNumA` over all ranks and stores the result on rank 0 in `globalNumA`.

The following figure illustrates the resulting approximation of π:

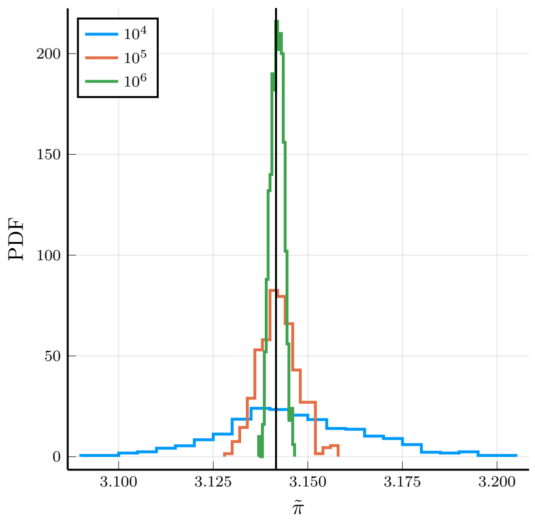


Figure 2: Distributions of $\tilde{\pi}$ for several values of N. The vertical line indicates the true value of π.

Problem 3

For this exercise we were asked to have one process read a file and send pieces of it to its worker processes by using MPI_Scatterv. Each worker then computes the mean and standard deviation and all values are collected by the main process.

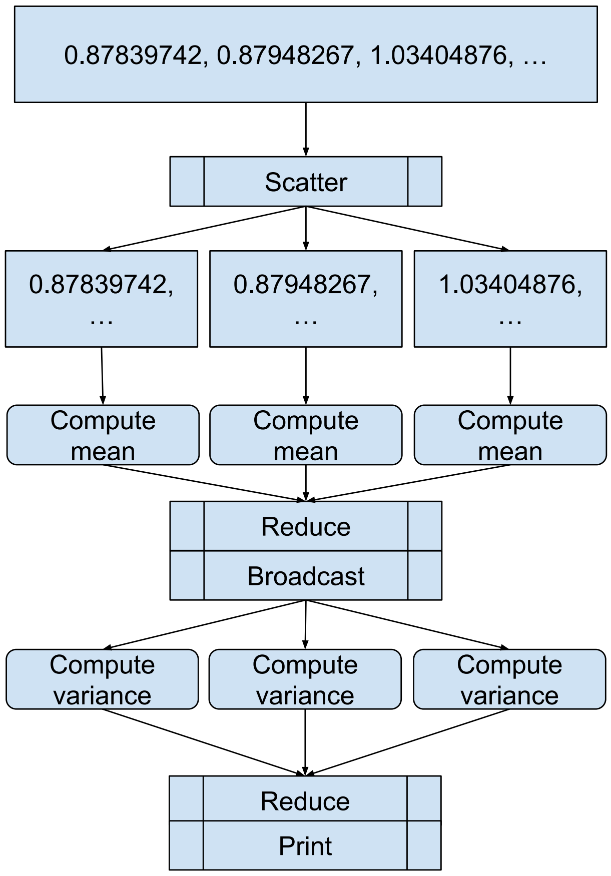


Figure 3: Flow chart for problem 3

$$Mean = 0.99932$$
$$Variance = 0.03973$$

Problem 4

For this exercise we were shown a tree like pattern. The idea was to generalize it for larger powers of two.

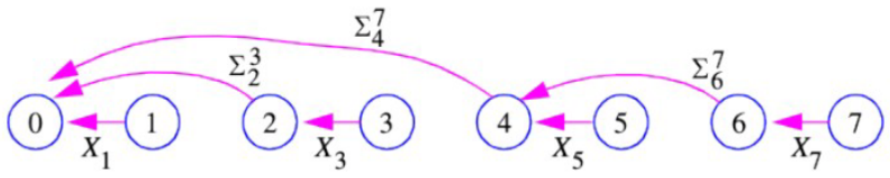


Figure 4: Problem 4 tree

In text, we could have

- If power of 2 send to process 0
- If even (but not power of 2) send to previous even process
- If odd send to previous process

where previous means in relation to their ID number. And as a pseudo-code,

```
1: $n \leftarrow$ ID
2: if $n \% 2 = 0$ then
3:     $proc_n \rightarrow proc_0$
4: else if $n \div 2 == 0$ then
5:     $proc_n \rightarrow proc_{n-2}$
6: else if $n \% 2 \neq 0$ then
7:     $proc_n \rightarrow proc_{n-1}$
8: end if
```

.