CS426

Summer 2021

Project 1 v3

Due 27/06/2021 – 23:59

Input files can be found on this link:

Data

PART A: Count the ACGT

In this problem you will be given a file containing DNA information with a vector per line. You will write a serial and a parallel code which counts the number of nitrogen-containing nucleobases in each vector and print out the total number of each type. So your code will go through the first line of the input data and count the number of each type then it goes to the second line and again counts the number of each type and then adds them up to the values from the first line. This procedure will continue until all the lines in the input file has been covered.

```
% input
CAAATCTACCTGAACATCAGCACCTACCGGCTAATTGCTTCTCTTGTTCTCCCTGTGGAA
ATCGTGTGGGATACGTGTGTAAAAGCTGATGAATTATTGGGATCAAATCCGCCATTACCT
ACAAAGCAAAGTGAGCGCGGATGGTTACGACAACTGGCTGAAATCAACGGCATTTGTCGG

% output
C:41
T:47
G:42
A:50
```

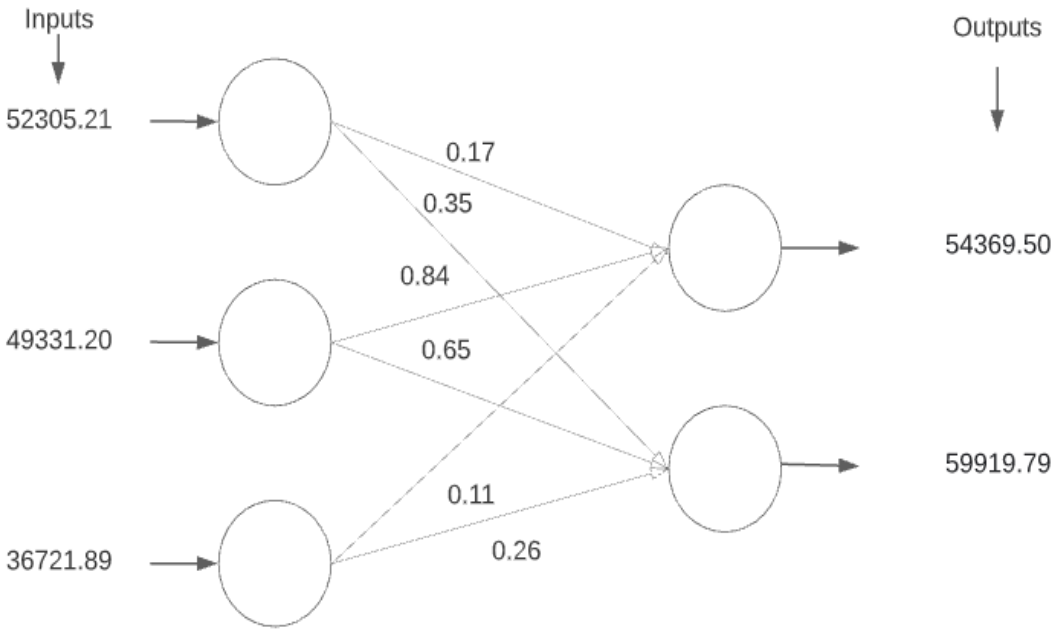
You will write three versions of this program in serial and in MPI.

- **Serial:** Write a serial program to solve this problem. Name the source code **dna-serial.c**.
- **MPIv1:** Write an MPI implementation of the above program in which a master process reads in the entire input file and then dispatches pieces of it to workers, where these pieces are of same size as much as possible. The master must also perform computation. Each processor counts the total number of each type in its local data and results are then collected by the master. Master then sums up the values for each type and prints out the total number of each type on the screen. This implementation should not use any collective communications, but only point-to-point. Name the source file **dna-mpi-v1.c**.
- **MPIv2:** This is similar to MPIv1 except that all processors should have the computed overall values. In this implementation, you are expected to use collective communication features of MPI.(MPI_Allreduce()/MPI_Bcast()) Name the source file **dna-mpi-v2.c**.

While the number of characters and lines can change, an example scenario could be to have 60 characters in each line and 100000 lines in total.

PART B: Neural Network

In this problem you are provided with input data and the hidden layer's weights of a neural network. The input data is Bitcoin's opening price in the last 100 days. Your code will calculate the multiplication of the weights and input data. Consider following example:



Inputs	Weights	
52305.21	0.17	0.35
49331.20	0.84	0.65
36721.89	0.11	0.26
Outputs	54369.50	59919.79

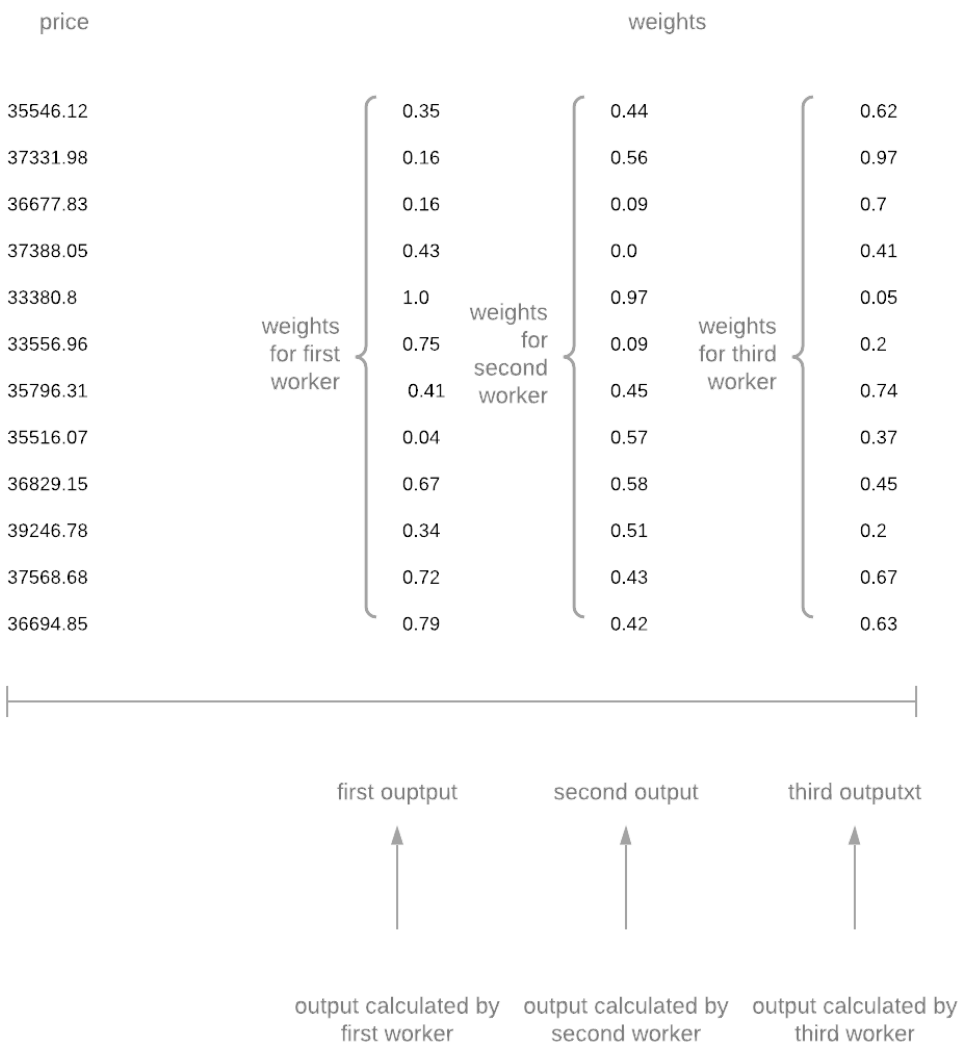
$$Firstoutput: 52305.21 * 0.17 + 49331.20 * 0.84 + 36721.89 * 0.11 = 54369.50$$
$$Secondoutput: 52305.21 * 0.35 + 49331.20 * 0.65 + 36721.89 * 0.26 = 59919.79$$

So, to calculate the first output you need to multiply the first column of weights with all the inputs. The first input will be multiplied by the first weight in the first column of weights, second input with second weight in the first column of inputs and so on. Then, you will add up all these values. Same will be applied for the second output, where you will choose the second column of the weights and all the inputs. This will be applied for all outputs accordingly as shown in the above figure.

Similar to the previous problem you will write several versions of this program in serial and in MPI.

- **Serial:** Write a serial program to solve this problem. Name the source code **NN-serial.c**. You will write the output into **NN-serial-output.txt**.
- **MPIv1:** Write an MPI implementation of the above program in which a master process reads in the entire input files and then dispatches columns of it to workers. In this part you will have exactly 10 new processes. The master process will send all price data to all the workers. Then it will send the first column of weights to the first worker. Second column to second worker and so on. Each processor makes the computation for its data and results are then collected by the master. Then you will write the results in a file named **NN-mpi-v1-output.txt**. This implementation should not use any collective communications, but only point-to-point. Name the source file **NN-mpi-v1.c**.

- **MPIv2**: In this part you will have different numbers of workers. The master process reads the entire input files and then dispatches the data to workers, where data is distributed in equal sizes. The number of workers can be set as any number less than the number of the weights' columns. For example, assume that we choose 3 workers. Consider the following example and implement your design the same way. The master process writes the outputs in a file named **NN-mpi-v2-output.txt**. Name the source file for this code **NN-mpi-v2.c**.


price
weights
35546.12
37331.98
36677.83
37388.05
33380.8
33556.96
35796.31
35516.07
36829.15
39246.78
37568.68
36694.85
weights for first worker
0.35
0.16
0.16
0.43
1.0
0.75
0.41
0.04
0.67
0.34
0.72
0.79
weights for second worker
0.44
0.56
0.09
0.0
0.97
0.09
0.45
0.57
0.58
0.51
0.43
0.42
weights for third worker
0.62
0.97
0.7
0.41
0.05
0.2
0.74
0.37
0.45
0.2
0.67
0.63
first ouptput
second output
third outputxt
output calculated by first worker
output calculated by second worker
output calculated by third worker

Submission

- Send a single zip file (ID-p1.zip) that includes:
- Your implementation:
    - dna-serial.c
    - dna-mpi-v1.c
    - dna-mpi-v2.c
    - NN-serial.c
    - NN-mpi-v1.c
    - NN-mpi-v2.c
- Run your code with various number of processes
- Your report (3 pages at most):
    - Explain the implementation and design choices
    - Plot a graph with various process numbers indicating the performance of your implementation. Use sequential implementation as a baseline.
    - Your observations about the performance of your implementation, how you interpret the results etc.
- We will not test your codes with the given inputs, so run your code with different dimensions.

Grading Criteria:

- dna-serial.c: 10
- dna-mpi-v1.c: 15
- dna-mpi-v2.c :15
- NN-serial.c: 10
- NN-mpi-v1.c: 15
- NN-mpi-v2.c: 15
- Report: 20

Email:pouria.hasani@bilkent.edu.tr
Email subject: CS426_p1 (Without this subject, your project will not be evaluated).
Zip File name: ID-p1.zip
Example:21701221-p1.zip (Without this name, will not be evaluated).
No Late Submission Allowed!