CS426
Spring 2019
Project 1
Due 10/03/2019

PART A: Compute Sum

The objective of the problem is to write a serial, then a parallel program that takes as input an array of integers, stored in an ASCII file with one integer per line, and prints out the sum of the elements in the file.

```
% cat input
1
24
9
% my_program input
34
```

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 `sum-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, which these pieces being of as equal size as possible. The master must also perform computation. Each processor computes a local sum and results are then collected and aggregated by the master. This implementation should not use any collective communications, but only point-to-point. Name the source file `sum-mpi-ppv1.c`.
- **MPIv2:** This is similar to MPIv1 except that all processors should have the computed overall sum. In this implementation, you are expected to use collective communication features of MPI.(MPI_Allreduce()/MPI_Bcast()) Name the source file `sum-mpi-ppv2.c`.

PART B: MM

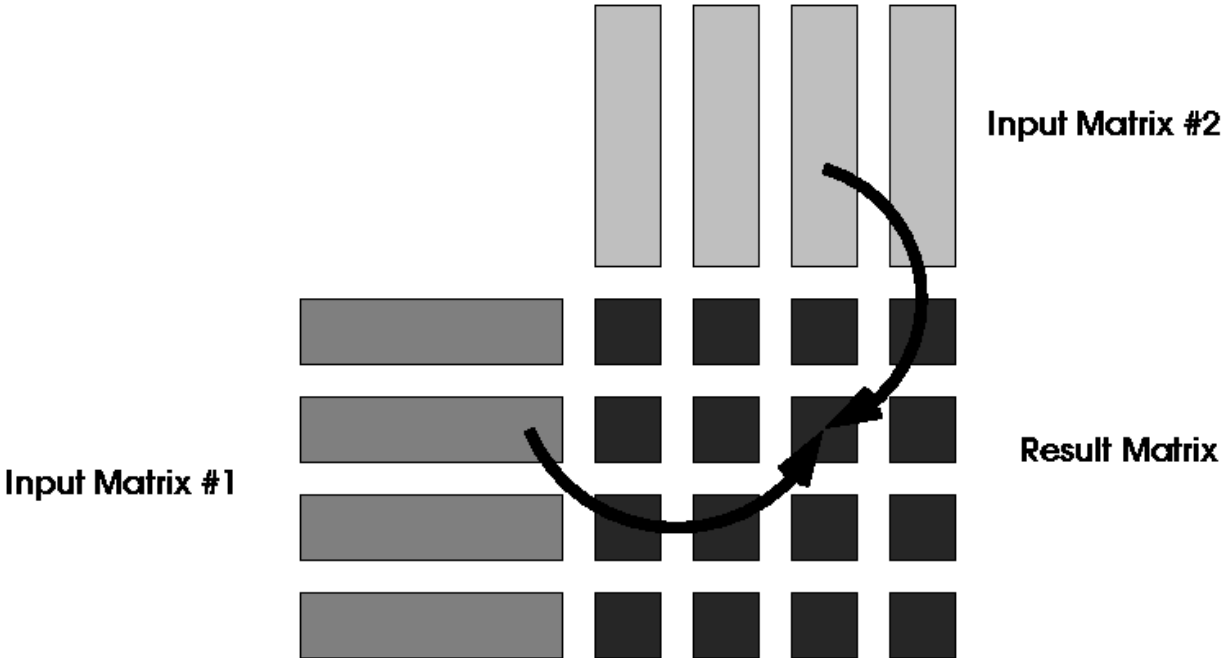
The objective of this problem is to write a serial, then parallel program that takes as input two integer **square** matrices of same dimension, stored in ASCII files whose first line gives the number of rows, and in which each following line lists space-separated integer elements of each row. The program writes the matrix that is the product of the two input matrices to a file, in the same format as the input.

```
% cat mat1
3
2 4 5
5 2 3
1 4 2
% cat mat2
3
4 2 1
4 1 1
```

```
1 2 1
% my_program mat1 mat2 mat3
% cat mat3
3
29 18 11
31 18 10
22 10 7
```

- **Serial:** Write a serial program called `matmult-serial.c`.
- **MPI:** Write an MPI implementation of the matrix multiply program in which a master process read in both input files and then dispatches it to workers. **The master must perform computation as well**. Assume that the number of processors used is a perfect square (4, 9, etc.), and that the matrix dimensions are perfectly divisible by the square root of the number of processors (e.g., if matrices are 30x30, then we use 9 processors). The processors are thought of, logically, as organized in a 2-D "processor grid".

In this implementation, the master partitions the first input matrix in blocks of rows, and the second matrix in blocks of columns. Row-blocks (resp. column-blocks) should contain the same number of rows (resp. columns). This is called a 1-D block distribution, and is illustrated on the figure below.



As seen in the figure, each processor computes the product of one block-row by one block-column, and returns the result to the master. The master then writes the output file. Name this source code `matmult-mpi-1d.c`.

Submission

Send a single zip file (**yourname_lastname_p1.zip**) that includes:

- Your implementation with source files and necessary inputs for the following
  - sum-serial.c
  - sum-mpi-ppv1.c
  - sum-mpi-ppv2.c
  - matmult-serial.c
  - matmult-mpi-1d.c
- Your output generated by your implementation
- Run your code with various number of threads
- Your report (3 pages at most):
  - Explain the implementation and design choices
  - Plot a graph with various thread numbers indicating the performance of your implementation. Use sequential implementation as a baseline.
  - Your observations about the performance of your implementation.

Email: kaan.akyol@bilkent.edu.tr

Email subject: CS426_Project1 (Without this subject, your project will not be evaluated).

Zip File name: yourname_lastname_p1.zip (Without this name, will not be evaluated).

No Late Submission Allowed!