COL730 Assignment 3: Sorting with MPI

Problem statement:

Design and implement a distributed sorting algorithm using the MPI framework. The total size of data to be sorted is expected to be larger than memory available on any single node. The data consists of key-value pairs of 64 bytes each. The key consists of a 2-character code in [a-z] (say a & b) and an unsigned integer (say x), and the value is a byte string of 58 characters, with a combined key-value size of 64 bytes. The algorithm is expected to sort the dataset by their keys while preserving the contents of the values corresponding to each key. The sorts are not expected to be stable, i.e., any two equal records can come in any order in the result. At the end of the procedure, each process should contain locally sorted data and for each process with rank k, the keys should be less than or equal to keys contained in processes with ranks higher than k. The order of keys is defined as follows:

A key (a_1, b_1, x_1) is relatively smaller than (a_2, b_2, x_2) if and only if:

Either "a_1b_1" < "a_2b_2" or ("a_1b_1" = "a_2b_2" and $x_1 < x_2$),

Where "ab" represents a character string consisting of the characters a & b and character strings are ordered lexicographically.

The algorithm is expected to scale to 64 processes and a total of 2^{32} elements.

Submission Instruction:

Submit a single zip file named [Your Entry Number].zip on moodle with the following:

1. An outline of the designed algorithm, the choices made and why.
2. Graphs between Number of processes v. Execution time.
3. Sources implementing the function signatures provided in header "psort.h".
4. A makefile that builds a library named "psort" (libpsort.a or libpsort.so) with the implementations.

Check course webpage for last date of submission.

Note:

A significant weightage will be given to performance of the algorithm.

All global MPI configuration should be performed by init() and reverted by close() (if any).

The function sort() is called on each process with its local dataset and number of records in its local dataset. The number of records on each process should be kept unchanged with the results being written directly into data. Note that each process may receive different number of records in its local dataset.

You are expected to write your own driver program for testing and documentation for different input configurations as necessary, but it may not be included in your submission.