

EECS201000

Introduction to Programming Laboratory

Homework 1: Odd-Even Sort

Due: July 10, 2017, 8AM

1 GOAL

This assignment helps you get familiar with MPI by implementing odd-even sort. We encourage you to optimize your program by exploring different parallelizing strategies.

2 PROBLEM DESCRIPTION

In this assignment, you are required to implement odd-even sort algorithm using MPI Library **under the restriction that MPI process can only send messages to its neighbor processes**. Odd-even sort is a comparison sort which consists of two main phases: *even-phase* and *odd-phase*.

In even-phase, all even/odd indexed pairs of adjacent elements are compared. If a pair is in the wrong order, the elements are switched. Similarly, the same process repeats for odd/even indexed pairs in odd-phase. The odd-even sort algorithm works by alternating these two phases until the list is completely sorted.

In order for you to understand this algorithm better, the execution flow of odd-even sort is illustrated step by step as below: (We are sorting the list into ascending order in this case)

1. [Even-phase] even/odd indexed adjacent elements are grouped into pairs.

Index	0	1	2	3	4	5	6	7
Value	6	1	4	8	2	5	9	3

2. [Even-phase] elements in a pair are switched if they are in the wrong order.

Index	0	1	2	3	4	5	6	7
Value	1	6	4	8	2	5	3	9

3. [Odd-phase] odd/even indexed adjacent elements are grouped into pairs.

Index	0	1	2	3	4	5	6	7
Value	1	6	4	8	2	5	3	9

4. [Odd-phase] elements in a pair are switched if they are in the wrong order.

Index	0	1	2	3	4	5	6	7
Value	1	4	6	2	8	3	5	9

5. Run even-phase and odd-phase alternatively until **no swap-work happens** in both even-phase and odd phase.

3 INPUT / OUTPUT FORMAT

1. Your programs are required to read an input file, and generate output in another file.
2. Your program accepts 3 input parameters, separated by space. They are:

- i 、 (Integer) the size of the list n ($0 \leq n \leq 2147483647$)
- ii 、 (String) the input file name
- iii 、 (String) the output file name

Make sure users can assign test cases through command line. For instance:

```
$ mpirun ./HW1_s106012345.exe 1000 in_file out_file
```

3. The input file lists n **32-bit floats** in binary format. Please refer to the sample input files.
4. The output file lists the n **32-bit floats** from the input file in ascending order. Please refer to the sample output files located at **/home/ipl2017/shared/hw1**

4 OPTIMIZATION HINTS

- You can send multiple items in a message to reduce swapping iterations.
- You are allowed to use gather and scatter before or after swapping iterations (not in between).
- You can try to overlap computation time and communication time as much as possible.
- You can try to overlap the operations between iterations.
- **If you are not sure whether your implementation follows the rules, please discuss with TA for approval.**

5 GRADING

1. Correctness (70%)

- A set of test cases will be given. You will receive the points for the test cases you pass.
- **Correctness check will be performed after the homework deadline and before the demo. The correctness results will be given at the demo time.**
- If you did not pass the test, you are given **3 days** to correct your code, but you will **only receive 80% of the points after correction.**
- Any correct result **delivered after 3 days will only receive 60% of the points**

2. Performance (20%)

- Performance is measured by the execution time of your program using ‘time’ Linux command.
- Points are giving according to the performance ranking of your program among all the students.

3. Demo (10%)

- Each student is given 5 minutes to explain your implementation followed by some questions from TA.
- No debugging or code modification is allowed during the demo.
- Points are given according to your understanding and explanation of your code, and your answers of the TA questions.

6 SUBMISSION

- Please upload the following files to **HW_submission/HW1** directory on **apollo31** under your home directory before **7/10(Mon) 8:00AM** (**The folder will be locked after deadline**)

i、 **HW1_{account_ID}.c**

Make sure your compile script can execute correctly and your code has no compile error in the **uploaded folder**

7 REMINDER

- We provide sample testcase, judge script and README under **/home/ipl2017/shared/hw1**, please refer to README to learn how to use.
- **You may write your own file reader to print out the values in the input and output files for verification.**
- Since we have limited resources, please **start your work ASAP**. Do not leave it until the last day!
- **Do NOT try to abuse the computing nodes by ssh to them directly**. If we ever find you doing that, you will get 0 point for the homework!
- **0 will be given to cheater** (even copying code from the Internet), but discussion on code is encouraged.
- Asking questions through iLMS is welcomed!