Question 1

Write a parallel MPI program which generates a sequence of N random numbers, each between 0 and 1, on each of its p processes.

Then implement a sorting algorithm where each process sorts the sequence of numbers it has, then distributes parts of it to the other processes so that at the end process 0 has the random numbers between 0 and 1/p, process 1 has numbers between 1/p and 2/p, and so on until the last process has numbers between (p-1)/p and 1

Logic

framework of program

The overall framework of the program can be as simple as following:

1. Initialize the sequence of floating numbers and sort within process
2. Distribute numbers to corresponding processes according to the requirement
3. Merge distributed number from different processes and print the result

key code segment

There are major challenges I encountered or trickes i used in coding

- utilized the merchanism of asynchronized communication: `MPI_Isend` and `MPI_Irecv` I tried to use `MPI_Waitany` to maximize concurrency without waiting for recieving next process. However, it doesn't work well, so I change back to `MPI_Waitall` and found the communicate between processes is very fast partly due to limited number of processes.
- use merge to maximize performance on sorted data(will analyze in detial in next section)
- Print gracefully To print gracefully, the program only print first and last few float numbers for large number of N. One easy to neglect details is that c program don't initialize string during delaration, so that it should be intialized with a simple line of code `result[0] = '\0'`, which took me some time to find out this bug that only appeared on large number of N.

Performance analysis

Theoratical Calculation

The overall time can be devided into(excluding num_array initialziation and misc thing, such as printing):

Time	Description	Complexity
T(sort_local)	Sort(quick sort) of num_array in each process	O(n*log(n))
T(comm)	Inter-process communication	O(n * p) **
T(merge)	Merge portion of sorted num_array from different processes	O(np(3/2))

** For asynchronize communicaiton, it is hard to estimate the communication speed, just using synchronized communication as simulation.

To make the parallelism worthwile, according to "Isoefficiency function($N = KTo(N , p)$)", `Ts = n*p*log(n*p)*T(sort) Tp = n*log(n)*T(sort) + n*p*T(comm) + n*p*(3/2)T(sort)` .where `T(sort)` is the unit time for sort and `T(comm)` is the time for single communication

To archieve different level of efficienty(use spead-up ratio for easy to understand), N need to be big enough as described in a table as follows, where `S=Ts/Tp`, and `log2(n) = (T(comm)/T(sort)+3*S/2-log2(p))/(1-S/p)`:

S\P	P=10	P=20	P=40
S=8	N=2^30	N= 2^8.6	N= 2^5.2
S=16	N/A	N= 2^26	N= 2^6.9
S= 32	N/A	N/A	N= 2^20.9

let T(comm)/T(sort) = 8

Lab Experiment on Server

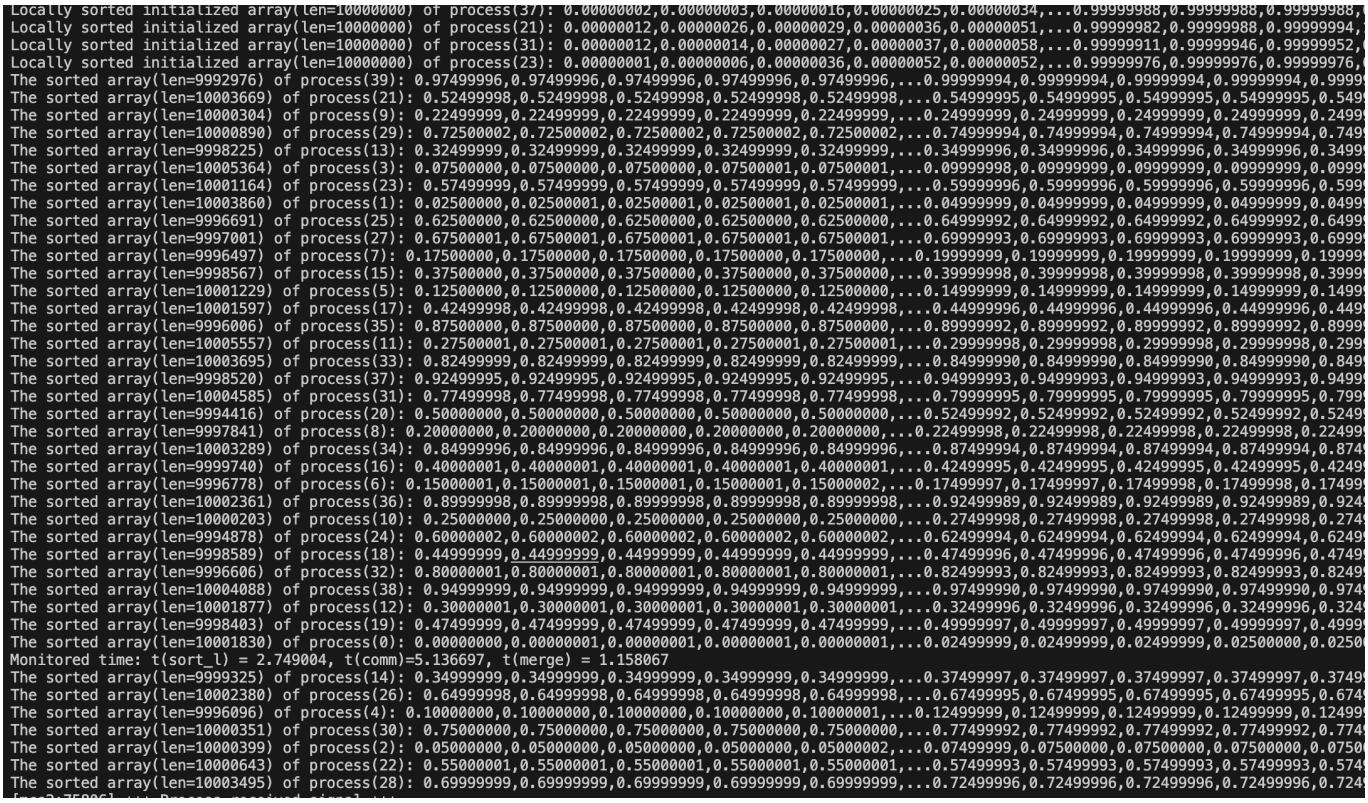
P=10

```
^C[qdai3@mcs2 question1]$ mpirun -np 10 sort_in_parallel.o
Please enter the number of random array length n:
100000000
Locally sorted initialized array(len=100000000) of process(3): 0.00000000,0.00000001,0.00000001,0.00000004,0.00000007,...0.99999994,0.99999994,0.99999994,0.99999994,1.00000000
Locally sorted initialized array(len=100000000) of process(7): 0.00000001,0.00000003,0.00000004,0.00000004,0.00000006,...0.99999994,0.99999994,1.00000000,1.00000000,1.00000000
Locally sorted initialized array(len=100000000) of process(2): 0.00000001,0.00000001,0.00000002,0.00000002,0.00000003,...1.00000000,1.00000000,1.00000000,1.00000000,1.00000000
Locally sorted initialized array(len=100000000) of process(8): 0.00000000,0.00000000,0.00000000,0.00000001,0.00000002,...0.99999988,0.99999988,0.99999988,0.99999994,0.99999994
Locally sorted initialized array(len=100000000) of process(1): 0.00000000,0.00000001,0.00000004,0.00000004,0.00000004,...0.99999994,1.00000000,1.00000000,1.00000000,1.00000000
Locally sorted initialized array(len=100000000) of process(9): 0.00000002,0.00000003,0.00000003,0.00000003,0.00000003,...0.99999994,0.99999994,1.00000000,1.00000000,1.00000000
Locally sorted initialized array(len=100000000) of process(4): 0.00000004,0.00000004,0.00000006,0.00000006,0.00000007,...1.00000000,1.00000000,1.00000000,1.00000000,1.00000000
Locally sorted initialized array(len=100000000) of process(0): 0.00000000,0.00000000,0.00000002,0.00000003,0.00000005,...0.99999994,1.00000000,1.00000000,1.00000000,1.00000000
Locally sorted initialized array(len=100000000) of process(6): 0.00000000,0.00000001,0.00000001,0.00000002,0.00000002,...0.99999994,0.99999994,1.00000000,1.00000000,1.00000000
Locally sorted initialized array(len=100000000) of process(5): 0.00000000,0.00000001,0.00000001,0.00000002,0.00000002,...0.99999994,1.00000000,1.00000000,1.00000000,1.00000000
The sorted array(len=100000786) of process(4): 0.40000001,0.40000001,0.40000001,0.40000001,0.40000001,...0.49999997,0.49999997,0.49999997,0.49999997,0.49999997
The sorted array(len=99998952) of process(5): 0.50000000,0.50000000,0.50000000,0.50000000,0.50000000,...0.59999996,0.59999996,0.59999996,0.59999996,0.59999996
The sorted array(len=99991827) of process(8): 0.80000001,0.80000001,0.80000001,0.80000001,0.80000001,...0.89999992,0.89999992,0.89999992,0.89999992,0.89999992
The sorted array(len=99989564) of process(7): 0.69999999,0.69999999,0.69999999,0.69999999,0.69999999,...0.79999995,0.79999995,0.79999995,0.79999995,0.79999995
The sorted array(len=99998570) of process(9): 0.89999998,0.89999998,0.89999998,0.89999998,0.89999998,...0.99999994,0.99999994,0.99999994,0.99999994,0.99999994
The sorted array(len=100003387) of process(6): 0.60000002,0.60000002,0.60000002,0.60000002,0.60000002,...0.69999993,0.69999993,0.69999993,0.69999993,0.69999993
The sorted array(len=100001453) of process(0): 0.00000000,0.00000000,0.00000000,0.00000000,0.00000000,...0.09999999,0.09999999,0.09999999,0.09999999,0.09999999
Monitored time: t(sort_l) = 24.790730, t(comm)=6.370868, t(merge) = 3.167309
The sorted array(len=100023688) of process(3): 0.30000001,0.30000001,0.30000001,0.30000001,0.30000001,...0.39999998,0.39999998,0.39999998,0.39999998,0.39999998
The sorted array(len=99993675) of process(2): 0.20000000,0.20000000,0.20000000,0.20000000,0.20000000,...0.29999998,0.29999998,0.29999998,0.29999998,0.29999998
The sorted array(len=99998066) of process(1): 0.10000000,0.10000000,0.10000000,0.10000000,0.10000000,...0.19999999,0.19999999,0.19999999,0.19999999,0.19999999
```

P=20

```
Locally sorted initialized array(len=50000000) of process(3): 0.00000001,0.00000012,0.00000012,0.00000013,0.00000017,...0.99999994,0.99999994,0.99999994,1.00000000,1.00000000
Locally sorted initialized array(len=50000000) of process(1): 0.00000000,0.00000001,0.00000003,0.00000003,0.00000005,...0.99999982,0.99999988,0.99999994,0.99999994,1.00000000
Locally sorted initialized array(len=50000000) of process(7): 0.00000005,0.00000007,0.00000013,0.00000015,0.00000016,...0.99999988,0.99999994,1.00000000,1.00000000,1.00000000
Locally sorted initialized array(len=50000000) of process(5): 0.00000003,0.00000003,0.00000004,0.00000009,0.00000009,...0.99999988,0.99999988,0.99999994,0.99999994,1.00000000
Locally sorted initialized array(len=50000000) of process(9): 0.00000001,0.00000002,0.00000003,0.00000004,0.00000005,...0.99999988,0.99999988,0.99999994,1.00000000,1.00000000
Locally sorted initialized array(len=50000000) of process(2): 0.00000007,0.00000011,0.00000012,0.00000014,0.00000014,...0.99999994,1.00000000,1.00000000,1.00000000,1.00000000
The sorted array(len=50007110) of process(2): 0.10000000,0.10000000,0.10000001,0.10000001,0.10000001,...0.14999999,0.14999999,0.14999999,0.14999999,0.14999999
The sorted array(len=49997645) of process(12): 0.60000002,0.60000002,0.60000002,0.60000002,0.60000002,...0.64999992,0.64999992,0.64999992,0.64999992,0.64999992
The sorted array(len=49999276) of process(3): 0.15000001,0.15000001,0.15000001,0.15000001,0.15000001,...0.19999999,0.19999999,0.19999999,0.19999999,0.19999999
The sorted array(len=50009936) of process(9): 0.44999999,0.44999999,0.44999999,0.44999999,0.44999999,...0.49999997,0.49999997,0.49999997,0.49999997,0.49999997
The sorted array(len=50004006) of process(11): 0.55000001,0.55000001,0.55000001,0.55000001,0.55000001,...0.59999996,0.59999996,0.59999996,0.59999996,0.59999996
The sorted array(len=49994858) of process(18): 0.89999998,0.89999998,0.89999998,0.89999998,0.89999998,...0.94999993,0.94999993,0.94999993,0.94999993,0.94999993
The sorted array(len=49990688) of process(7): 0.34999999,0.34999999,0.34999999,0.34999999,0.34999999,...0.39999998,0.39999998,0.39999998,0.39999998,0.39999998
The sorted array(len=49995733) of process(4): 0.20000000,0.20000000,0.20000000,0.20000000,0.20000000,...0.24999999,0.24999999,0.24999999,0.24999999,0.24999999
The sorted array(len=50000206) of process(1): 0.05000000,0.05000000,0.05000000,0.05000000,0.05000000,...0.09999999,0.09999999,0.09999999,0.09999999,0.09999999
The sorted array(len=49999574) of process(14): 0.69999999,0.69999999,0.69999999,0.69999999,0.69999999,...0.74999994,0.74999994,0.74999994,0.74999994,0.74999994
The sorted array(len=50001636) of process(5): 0.25000000,0.25000000,0.25000000,0.25000000,0.25000000,...0.29999998,0.29999998,0.29999998,0.29999998,0.29999998
The sorted array(len=50005385) of process(13): 0.64999998,0.64999998,0.64999998,0.64999998,0.64999998,...0.69999993,0.69999993,0.69999993,0.69999993,0.69999993
The sorted array(len=50004284) of process(10): 0.50000000,0.50000000,0.50000000,0.50000000,0.50000000,...0.54999995,0.54999995,0.54999995,0.54999995,0.54999995
The sorted array(len=49995640) of process(15): 0.75000000,0.75000000,0.75000000,0.75000000,0.75000000,...0.79999995,0.79999995,0.79999995,0.79999995,0.79999995
The sorted array(len=50010119) of process(17): 0.84999996,0.84999996,0.84999996,0.84999996,0.84999996,...0.89999992,0.89999992,0.89999992,0.89999992,0.89999992
The sorted array(len=50007536) of process(0): 0.00000000,0.00000000,0.00000000,0.00000000,0.00000001,...0.04999999,0.04999999,0.05000000,0.05000000,0.05000000
The sorted array(len=49989871) of process(8): 0.40000001,0.40000001,0.40000001,0.40000001,0.40000001,...0.44999996,0.44999996,0.44999996,0.44999996,0.44999996
The sorted array(len=49999340) of process(19): 0.94999999,0.94999999,0.94999999,0.94999999,0.94999999,...0.99999994,0.99999994,0.99999994,0.99999994,0.99999994
The sorted array(len=49999255) of process(16): 0.80000001,0.80000001,0.80000001,0.80000001,0.80000001,...0.84999990,0.84999990,0.84999990,0.84999990,0.84999990
The sorted array(len=49987867) of process(6): 0.30000001,0.30000001,0.30000001,0.30000001,0.30000001,...0.34999996,0.34999996,0.34999996,0.34999996,0.34999996
Monitored time: t(sort_l) = 12.704827, t(comm)=7.176141, t(merge) = 2.685458
```

P=40



Lab Experiment on Server

After lab experiment, we can found the communication time rises with more processes involved in the calculation.