Assignment 3 Report

Jui-yang Cheng
CPSC 524

Development Envirnoment

I used an IDE called *CodeRunner* to implement, test and debug the code on my local computer.
The module I loaded on Grace is the `iomkl` module. I loaded it with command `module load intel`.

Part 1

Task 1 (General)

Terminal Commands

- Compile and link to the code: `make task1`
- Build and execute the code: `sbatch ./runtask1.sh`
- View and collect the data: `vi MPI_Hello-X.out` where X should be replaced by the corresponding job number.

Sample Output

```
Currently Loaded Modules:
  1) StdEnv                                            (S)
  2) GCCcore/7.3.0
  3) binutils/2.30-GCCcore-7.3.0
  4) icc/2018.3.222-GCC-7.3.0-2.30
  5) ifort/2018.3.222-GCC-7.3.0-2.30
  6) iccifort/2018.3.222-GCC-7.3.0-2.30
  7) zlib/1.2.11-GCCcore-7.3.0
  8) numactl/2.0.11-GCCcore-7.3.0
  9) XZ/5.2.4-GCCcore-7.3.0
 10) libxml2/2.9.8-GCCcore-7.3.0
 11) libpciaccess/0.14-GCCcore-7.3.0
 12) hwloc/1.11.10-GCCcore-7.3.0
 13) OpenMPI/3.1.1-iccifort-2018.3.222-GCC-7.3.0-2.30
 14) iompi/2018b
 15) imkl/2018.3.222-iompi-2018b
 16) iomkl/2018b

  Where:
   S:  Module is Sticky, requires --force to unload or purge





Working Directory:
/home/cpsc424_jc3768/p3/part1


Making task1
rm -f task1 task1.o task2 task2.o task3 task3.o
mpicc -g -O3 -xHost -fno-alias -std=c99 -c task1.c
mpicc -o task1 -g -O3 -xHost -fno-alias -std=c99 task1.o timing.o rwork.o


Node List:
c01n07,c03n02
ntasks-per-node =  2


Run 1
Message printed by manager: Total elapsed time is 0.001588 seconds.
From process 1: I worked for 5 seconds after receiving the following message:
         Hello, from process 0.
From process 3: I worked for 10 seconds after receiving the following message:
         Hello, from process 0.
From process 2: I worked for 15 seconds after receiving the following message:
         Hello, from process 0.

real    0m15.582s
user    0m0.752s
sys     0m0.561s


Run 2
Message printed by manager: Total elapsed time is 0.002037 seconds.
From process 1: I worked for 5 seconds after receiving the following message:
         Hello, from process 0.
From process 2: I worked for 10 seconds after receiving the following message:
```

```
         Hello, from process 0.
From process 3: I worked for 15 seconds after receiving the following message:
         Hello, from process 0.

real    0m15.592s
user    0m0.782s
sys     0m0.643s


Run 3
Message printed by manager: Total elapsed time is 0.004703 seconds.
From process 3: I worked for 5 seconds after receiving the following message:
         Hello, from process 0.
From process 2: I worked for 10 seconds after receiving the following message:
         Hello, from process 0.
From process 1: I worked for 15 seconds after receiving the following message:
         Hello, from process 0.

real    0m15.572s
user    0m0.551s
sys     0m0.441s
```

Task 1.1

The elapsed time reported by the manager is so different from the elapsed time reported by the `time` command is because the elapsed time reported by the manager is the amount of time the manager took to send out messages to the workers, while the elapsed time reported by the `time` command is the amount of time it used to finish running the whole program (which also consider the time used by the workers).

It could be a useful measurement when you need to know the communication time of the manager. However, the time used to send out messages is trivial in this case compared with the spend needed for the worker to do their work or the amount of time needed to run the whole program. As the manager, it would be better to measure the computation time used to by each worker to finish their works, or the total amount of time the workers spend.

Task 1.2

There could be competition for resources between MPI program, the MPI runtime system, and the operating system for this program such as cache memory and CPU.

Task 2 (General)

Terminal Commands

- Compile and link to the code: `make task2`
- Build and execute the code: `sbatch ./runtask2.sh`
- View and collect the data: `vi task2-X.out` where X should be replaced by the corresponding job number.

Sample Output

```
Currently Loaded Modules:
  1) StdEnv                                            (S)
  2) GCCcore/7.3.0
  3) binutils/2.30-GCCcore-7.3.0
  4) icc/2018.3.222-GCC-7.3.0-2.30
  5) ifort/2018.3.222-GCC-7.3.0-2.30
  6) iccifort/2018.3.222-GCC-7.3.0-2.30
  7) zlib/1.2.11-GCCcore-7.3.0
  8) numactl/2.0.11-GCCcore-7.3.0
  9) XZ/5.2.4-GCCcore-7.3.0
 10) libxml2/2.9.8-GCCcore-7.3.0
 11) libpciaccess/0.14-GCCcore-7.3.0
 12) hwloc/1.11.10-GCCcore-7.3.0
 13) OpenMPI/3.1.1-iccifort-2018.3.222-GCC-7.3.0-2.30
 14) iompi/2018b
 15) imkl/2018.3.222-iompi-2018b
 16) iomkl/2018b

  Where:
   S:  Module is Sticky, requires --force to unload or purge





Working Directory:
/home/cpsc424_jc3768/p3/part1



Making task2
rm -f task1 task1.o task2 task2.o task3 task3.o
mpicc -g -O3 -xHost -fno-alias -std=c99 -c task2.c
mpicc -o task2 -g -O3 -xHost -fno-alias -std=c99 task2.o timing.o rwork.o
```

```
Node List:
c01n[06-07]
ntasks-per-node =  2


Run 1
Message from process 1: Hello manager, from process 1 after working 15 seconds.
Message from process 2: Hello manager, from process 2 after working 10 seconds.
Message from process 3: Hello manager, from process 3 after working 5 seconds.
Message printed by manager: Total elapsed time is 24.003851 seconds.

real	0m24.592s
user	0m15.424s
sys	0m0.359s


Run 2
Message from process 1: Hello manager, from process 1 after working 10 seconds.
Message from process 2: Hello manager, from process 2 after working 5 seconds.
Message from process 3: Hello manager, from process 3 after working 15 seconds.
Message printed by manager: Total elapsed time is 19.009272 seconds.

real	0m19.582s
user	0m10.363s
sys	0m0.393s


Run 3
Message from process 1: Hello manager, from process 1 after working 15 seconds.
Message from process 2: Hello manager, from process 2 after working 10 seconds.
Message from process 3: Hello manager, from process 3 after working 5 seconds.
Message printed by manager: Total elapsed time is 24.017927 seconds.

real	0m24.606s
user	0m15.399s
sys	0m0.379s
```

Task 2.1

Observation

The elapsed time reported by the manager in this task is very close to the "real" time reported by the `time` command and it is qualitatively different from task 1 because this time it takes the worker's computation time into account rather than simply measuring the communication time of the manager.

Expanation

The comparison is qualitatively different from Task 1 because the elapsed time reported by the manager this time not only includes the time spent to send messages to the workers, but also includes the time spend by the workers to finish their works. Therefore, the elapsed time reported by the manager is much closer to the wall clock time.

Task 2.2

Explanation

The total elapsed time among the three runs has variation because, based on my implementation, the manager will always wait for message from worker rank#1 first even when worker rank#2 or worker rank#3 finish before worker rank#1, causing unpredictable waiting time (the waiting time is shorest when worker rank#1 finishes first, followed by worker rank#2, followed by worker rank#3, and is the longest when worker #1 finishes last)

Fix

To fix this time variation, instead of receiving message with a particular order (in my case, message from rank#1, then from rank#2, then from rank#3), it should receive message from any worker regardless of their rank, and store their message contents and corresponding rank numbers. Finally, after receives messages from every worker, print them out in order of increasing rank.

Task 3

Terminal Commands

- Compile and link to the code: `make task3`
- Build and execute the code: `sbatch ./runtask3.sh`
- View and collect the data: `vi task3-X.out` where X should be replaced by the corresponding job number.

Sample Output

```
Currently Loaded Modules:
  1) StdEnv                                           (S)
  2) GCCcore/7.3.0
  3) binutils/2.30-GCCcore-7.3.0
  4) icc/2018.3.222-GCC-7.3.0-2.30
  5) ifort/2018.3.222-GCC-7.3.0-2.30
  6) iccifort/2018.3.222-GCC-7.3.0-2.30
  7) zlib/1.2.11-GCCcore-7.3.0
  8) numactl/2.0.11-GCCcore-7.3.0
  9) XZ/5.2.4-GCCcore-7.3.0
 10) libxml2/2.9.8-GCCcore-7.3.0
 11) libpciaccess/0.14-GCCcore-7.3.0
 12) hwloc/1.11.10-GCCcore-7.3.0
 13) OpenMPI/3.1.1-iccifort-2018.3.222-GCC-7.3.0-2.30
 14) iompi/2018b
 15) imkl/2018.3.222-iompi-2018b
 16) iomkl/2018b

  Where:
   S:  Module is Sticky, requires --force to unload or purge





Working Directory:
/home/cpsc424_jc3768/p3/part1


Making task3
rm -f task1 task1.o task2 task2.o task3 task3.o
mpicc -g -O3 -xHost -fno-alias -std=c99 -c task3.c
mpicc -o task3 -g -O3 -xHost -fno-alias -std=c99 task3.o timing.o rwork.o


Node List:
c01n[06-07]
ntasks-per-node =  2


Run 1
Message from process 1: Hello manager, from process 1 after working 5 seconds.
Message from process 2: Hello manager, from process 2 after working 15 seconds.
Message from process 3: Hello manager, from process 3 after working 10 seconds.
Message printed by manager: Total elapsed time is 18.000271 seconds.

real    0m18.608s
user    0m9.535s
sys     0m0.378s


Run 2
```

```
Message from process 1: Hello manager, from process 1 after working 5 seconds.
Message from process 2: Hello manager, from process 2 after working 10 seconds.
Message from process 3: Hello manager, from process 3 after working 15 seconds.
Message printed by manager: Total elapsed time is 18.005259 seconds.

real    0m18.634s
user    0m9.470s
sys     0m0.407s



Run 3
Message from process 1: Hello manager, from process 1 after working 5 seconds.
Message from process 2: Hello manager, from process 2 after working 15 seconds.
Message from process 3: Hello manager, from process 3 after working 10 seconds.
Message printed by manager: Total elapsed time is 18.000335 seconds.

real    0m19.101s
user    0m9.516s
sys     0m0.388s
```

Part 2

Task 4

Terminal Commands

- Compile and link to the code: `make serial`
- Build and execute the code: `sbatch ./build-run-serial.sh`
- View and collect the data: `vi slurm-X.out` where X should be replaced by the corresponding job number.

Summary timing/error table

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   1000       0.1558      0.000000000000
   2000       1.3504      0.000000000000
   4000      16.1754      0.000000000000
   8000     132.4886      0.000000000000
```

Task 5A

Terminal Commands

- Compile and link to the code: `make task5`
- Build and execute the code: `sbatch ./build-run-mpi.sh`
- View and collect the data: `vi slurm-X.out` where X should be replaced by the corresponding job number.

Summary timing/error tables

p = 1

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   1000       0.1591      0.000000000000
   2000       1.2142      0.000000000000
   4000      15.7156      0.000000000000
   8000     128.8955      0.000000000000
```

p = 2

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   1000       0.1296      0.000000000014
   2000       0.8459      0.000000000000
   4000      10.3632      0.000000000000
   8000      87.2888      0.000000000000
```

p = 4

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   1000       0.2119      0.000000000016
   2000       0.5912      0.000000000045
   4000       7.1425      0.000000000254
   8000      62.6764      0.000000000000
```

p = 8

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   1000       0.2539      0.000000000015
   2000       0.3288      0.000000000056
   4000       3.0485      0.000000000181
   8000      32.8495      0.000000000000
```

Raw Performance

Aside from p = 1 where the raw performance doesn't make much difference from the serial version, all the results shows significant improvement of raw performance compared to the serial version: with p = 2, the wall clock runtime is shorten by around 40 seconds; with p = 4, the wall clock runtime is shorten by around 60 seconds; with p = 8, the wall clock runtime is shorten by around 90 seconds.

Scalability

By comparing the average wall clock time spent on computing one cell of matrix with a fixed p, I noticed that the average time spent to compute one cell in the matrix increases as the N gets larger. More specifically, with p = 2, the time spent to compute one cell is approximately 0.00000013 when N = 1000, is approximately 0.00000021 when N = 2000, is approximately 0.00000065 when N = 4000 and is approximately 0.00000136 when N = 8000. Similar pattern occurs for p = 1, 4 and 8.

Load Balance

As shown in below computation times, it seems that the computation time spent by process rank 0 and process rank (p-1) are generally smaller than the rest of the process if there are any, indicating that the load balance is not very balanced. (we do not include the case where p = 1 because all the job is done by one process)

p = 2

```
N = 1000
rank 0 work time: 0.108886
rank 1 work time: 0.105316
N = 2000
rank 0 work time: 0.823418
rank 1 work time: 0.811776
N = 3000
rank 0 work time: 10.201013
rank 1 work time: 9.897544
N = 4000
rank 0 work time: 86.277387
rank 1 work time: 85.296899
```

p = 4

```
N = 1000
rank 0 work time: 0.076741
rank 1 work time: 0.085049
rank 2 work time: 0.089070
rank 3 work time: 0.075641
N = 2000
rank 0 work time: 0.464374
rank 1 work time: 0.525910
rank 2 work time: 0.583157
rank 3 work time: 0.451922
N = 3000
rank 0 work time: 4.734780
rank 1 work time: 5.204063
rank 3 work time: 4.663346
rank 2 work time: 5.878035
N = 4000
rank 0 work time: 46.321520
rank 1 work time: 52.524389
rank 2 work time: 58.035718
rank 3 work time: 45.270901
```

p = 8

```
N = 1000
rank 0 work time: 0.043796
rank 1 work time: 0.047947
rank 2 work time: 0.047591
rank 3 work time: 0.048029
rank 4 work time: 0.048320
rank 5 work time: 0.048406
rank 6 work time: 0.048359
rank 7 work time: 0.041858
N = 2000
rank 0 work time: 0.255596
rank 1 work time: 0.271601
rank 2 work time: 0.277227
rank 3 work time: 0.282574
rank 4 work time: 0.288670
rank 5 work time: 0.293281
rank 6 work time: 0.296383
rank 7 work time: 0.224501
N = 3000
rank 0 work time: 2.687928
rank 1 work time: 2.773051
rank 2 work time: 2.822772
rank 3 work time: 2.872821
rank 4 work time: 3.038214
rank 6 work time: 3.076310
rank 5 work time: 3.114609
rank 7 work time: 2.252144
N = 4000
rank 0 work time: 29.158483
rank 1 work time: 29.696731
rank 2 work time: 30.856594
rank 3 work time: 31.550415
rank 4 work time: 33.311853
rank 6 work time: 33.492798
rank 5 work time: 33.514177
rank 7 work time: 24.777866
```

Task 5B

Note

If need to see the breakdown between computation and communication time for each rank, uncomment the `printf()` and `MPI_Barrier` near the end of my code.

Terminal Commands

- Compile and link to the code: `make task5`
- Build and execute the code: `sbatch ./p4n2.sh` , `sbatch ./p4n4.sh` , `sbatch ./p8n2.sh` , `sbatch ./p8n4.sh`
- View and collect the data: `vi slurm-X.out` where X should be replaced by the corresponding job number.

Summary Tables and Breakdown Tables

Summary Table (1 node and p = 4)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      60.5561      0.000000000000
```

Breakdown Table (1 node and p = 4)

```
rank 0 computation time: 12.250514, communication time: 48.305582
rank 1 computation time: 30.843751, communication time: 25.782491
rank 2 computation time: 42.071201, communication time: 20.158602
rank 3 computation time: 47.548171, communication time: 14.776649
```

Summary Table (1 node and p = 8)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      37.9264      0.000000000000
```

Breakdown Table (1 node and p = 8)

```
rank 0 computation time: 4.243804, communication time: 33.682561
rank 1 computation time: 11.159165, communication time: 24.507948
rank 2 computation time: 16.386893, communication time: 20.189646
rank 3 computation time: 20.402873, communication time: 17.007901
rank 4 computation time: 23.447193, communication time: 14.744269
rank 5 computation time: 25.536326, communication time: 13.380103
rank 6 computation time: 27.027786, communication time: 12.599931
rank 7 computation time: 26.868258, communication time: 12.807033
```

Summary Table (2 nodes and p = 4)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      67.2519      0.000000000000
```

Breakdown Table (2 nodes and p = 4)

```
rank 0 computation time: 7.624226, communication time: 59.627590
rank 1 computation time: 26.098134, communication time: 34.139500
rank 2 computation time: 42.098931, communication time: 26.827652
rank 3 computation time: 51.164909, communication time: 17.838212
```

Summary Table (2 nodes and p = 8)

```
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      36.2806      0.000000000000
```

Breakdown Table (2 nodes and p = 8)

```
rank 0 computation time: 1.890759, communication time: 34.389833
rank 1 computation time: 5.441649, communication time: 27.693225
rank 2 computation time: 11.994629, communication time: 22.566108
rank 3 computation time: 17.073432, communication time: 17.684037
rank 4 computation time: 16.247775, communication time: 19.140545
rank 5 computation time: 19.138912, communication time: 17.085864
rank 6 computation time: 23.845161, communication time: 14.151105
rank 7 computation time: 25.811711, communication time: 12.222238
```

Summary Table (4 nodes and p = 4)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      56.6563      0.000000000000
```

Breakdown Table (4 nodes and p = 4)

```
rank 0 computation time: 7.631105, communication time: 49.025167
rank 1 computation time: 26.170258, communication time: 26.356781
rank 3 computation time: 43.209493, communication time: 15.198551
rank 2 computation time: 37.770101, communication time: 20.638034
```

Summary Table (4 nodes and p = 8)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      38.8317      0.000000000000
```

Breakdown Table (4 nodes and p = 8)

```
rank 0 computation time: 6.933333, communication time: 31.898181
rank 1 computation time: 4.807281, communication time: 35.646440
rank 2 computation time: 8.785979, communication time: 32.136422
rank 3 computation time: 12.772196, communication time: 28.960106
rank 4 computation time: 26.420375, communication time: 18.535564
rank 5 computation time: 21.201455, communication time: 23.985526
rank 6 computation time: 23.036603, communication time: 22.150417
rank 7 computation time: 23.823791, communication time: 21.363187
```

Raw Performance

Based on the tables above, it seems that, with a certain fixed number of node, the more process there are, the faster it runs; and also, with a fixed number of processes p, the more nodes there are , the faster it runs (with one exception).

Load Balance

The load balance seems to be pretty unbalanced based on the tables above, it seems that the lower rank processes has less loads and more communication times, and as the rank increases, the computation times increases and communication time decreases.

Suggestion

1) Make variable C into a shared variable across processes: If C becomes a shared variable, all the processes can insert the result of their computations into C right away instead of having to send their individual C's back to the manager at the end and have manager to do the job. In this way, we can save some communication time at the end, which will improve raw performance.

2) Use nonblocking communicaiton function: with nonblocking communication functions, each process can overlap some communication time with computation time, which can improve the raw performance.

Task 6A

Terminal Commands

- Compile and link to the code: `make task6`
- Build and execute the code: `sbatch ./task6.sh`
- View and collect the data: `vi slurm-X.out` where X should be replaced by the corresponding job number.

Summary timing/error tables

p = 1

```
Matrix multiplication times:
     N      TIME (secs)    F-norm of Error
   -----   -------------  -----------------
    1000       0.1576      0.000000000000
    2000       1.2055      0.000000000000
    4000      14.3031      0.000000000000
    8000     120.3617      0.000000000000
```

p = 2

```
Matrix multiplication times:
     N      TIME (secs)    F-norm of Error
   -----   -------------  -----------------
    1000       0.1290      0.000000000014
    2000       0.8460      0.000000000000
    4000      10.2955      0.000000000000
    8000      86.3305      0.000000000000
```

p = 4

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   1000       0.1236      0.000000000016
   2000       0.4818      0.000000000048
   4000       5.3779      0.000000000254
   8000      49.8268      0.000000000000
```

p = 8

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   1000       0.1093      0.000000000015
   2000       0.2676      0.000000000056
   4000       2.5423      0.000000000182
   8000      27.0158      0.000000000000
```

Raw Performance

By comparing with Task 5A where blocking MPI communication operations and blocking collecting operations are used, the raw performance now generally improves by several seconds.

Scalability

Similar to the result in 5A, the overall trend is still that the average time to compute one cell in matrix gets larger when N gets larger. However, by comparing the average wall clock time spent on computing one cell of matrix with a fixed p with Task 5A, the average time spent to compute one cell is less than that of 5A, showing a slightly better scalability.

Load Balance

As shown below, even though the load seems to be more balanced when p = 8 when compared with the result from 5A, the overall load balance doesn't seem to improve much.

p = 2

```
N = 1000
rank 0 work time: 0.125211
rank 1 work time: 0.151323
N = 2000
rank 0 work time: 0.830639
rank 1 work time: 0.932848
N = 4000
rank 0 work time: 10.230217
rank 1 work time: 10.388876
N = 8000
rank 0 work time: 86.057049
rank 1 work time: 87.048547
```

p = 4

```
N = 1000
rank 0 work time: 0.103294
rank 1 work time: 0.146966
rank 2 work time: 0.148968
rank 3 work time: 0.144476
N = 2000
rank 0 work time: 0.444205
rank 1 work time: 0.558562
rank 2 work time: 0.589678
rank 3 work time: 0.553918
N = 4000
rank 0 work time: 4.755887
rank 1 work time: 5.199393
rank 2 work time: 5.805883
rank 3 work time: 5.198923
N = 8000
rank 0 work time: 46.330843
rank 1 work time: 47.344585
rank 2 work time: 51.534316
rank 3 work time: 47.898590
```

p = 8

```
N = 1000
rank 0 work time: 0.074273
rank 1 work time: 0.129184
rank 2 work time: 0.127568
rank 3 work time: 0.130837
rank 4 work time: 0.135085
rank 5 work time: 0.136343
rank 6 work time: 0.137445
rank 7 work time: 0.128941
N = 2000
rank 0 work time: 0.230181
rank 1 work time: 0.343485
rank 2 work time: 0.345413
rank 3 work time: 0.339444
rank 4 work time: 0.351062
rank 5 work time: 0.368263
rank 6 work time: 0.375128
rank 7 work time: 0.335644
N = 4000
rank 0 work time: 2.146368
rank 1 work time: 2.600438
rank 2 work time: 2.605242
rank 3 work time: 2.615377
rank 4 work time: 2.630178
rank 5 work time: 2.915720
rank 6 work time: 2.977507
rank 7 work time: 2.596297
N = 8000
rank 0 work time: 23.310599
rank 1 work time: 24.888523
rank 2 work time: 24.962921
rank 3 work time: 25.043972
rank 4 work time: 25.764581
rank 5 work time: 28.255623
rank 6 work time: 28.753622
rank 7 work time: 25.095305
```

Task 6B

Note

If need to see the breakdown between computation and communication time for each rank, uncomment the `printf()` and `MPI_Barrier` near the end of my code.

Terminal Commands

- Compile and link to the code: `make task6`
- Build and execute the code: `sbatch task6.sh` , `sbatch ./p4n2.sh` , `sbatch ./p4n4.sh` , `sbatch ./p8n2.sh` , `sbatch ./p8n4.sh`
- View and collect the data: `vi slurm-X.out` where X should be replaced by the corresponding job number.

Summary Tables and Breakdown Tables

Summary Table (1 node and p = 4)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      55.9484      0.000000000000
```

Breakdown Table (1 node and p = 4)

```
rank 0 computation time: 10.123046, communication time: 45.825327
rank 1 computation time: 28.948075, communication time: 25.235618
rank 2 computation time: 43.168801, communication time: 14.552279
rank 3 computation time: 48.560991, communication time: 5.674407
```

Summary Table (1 node and p = 8)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      30.8539      0.000000000000
```

Breakdown Table (1 node and p = 8)

```
rank 0 work time: 26.473607, computation time: 1.883281, communication time: 28.97056
6
rank 1 work time: 28.060418, computation time: 5.233994, communication time: 23.13290
5
rank 2 work time: 28.237518, computation time: 11.968264, communication time: 16.4323
75
rank 3 work time: 28.045962, computation time: 15.990114, communication time: 12.4105
30
rank 4 work time: 28.575785, computation time: 19.782175, communication time: 8.82250
5
rank 5 work time: 31.550547, computation time: 21.663476, communication time: 9.91499
2
rank 6 work time: 32.597580, computation time: 25.068665, communication time: 7.55788
0
rank 7 work time: 28.282523, computation time: 25.493567, communication time: 2.915213
```

Summary Table (2 nodes and p = 4)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      59.0701      0.000000000000
```

Breakdown Table (2 nodes and p = 4)

```
rank 0 computation time: 8.419573, communication time: 50.571758
rank 1 computation time: 27.636127, communication time: 28.277306
rank 2 computation time: 43.468149, communication time: 17.348241
rank 3 computation time: 52.292606, communication time: 3.621729
```

Summary Table (2 nodes and p = 8)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      38.0594      0.000000000000
```

Breakdown Table (2 nodes and p = 8)

```
rank 0 computation time: 1.891943, communication time: 36.167433
rank 1 computation time: 6.138104, communication time: 28.703244
rank 2 computation time: 13.284781, communication time: 23.035836
rank 3 computation time: 18.560463, communication time: 18.193664
rank 4 computation time: 18.006323, communication time: 19.257460
rank 5 computation time: 20.514698, communication time: 17.512743
rank 6 computation time: 25.460114, communication time: 14.312030
rank 7 computation time: 27.287705, communication time: 12.522858
```

Summary Table (4 nodes and p = 4)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      60.7584      0.000000000000
```

Breakdown Table (4 nodes and p = 4)

```
rank 0 work time: 48.636194, computation time: 9.194363, communication time: 51.56401
9
rank 1 work time: 55.779682, computation time: 29.769039, communication time: 28.1398
62
rank 2 work time: 60.301432, computation time: 38.322211, communication time: 24.2051
37
rank 3 work time: 47.338547, computation time: 47.251033, communication time: 15.2762
63
```

Summary Table (4 nodes and p = 8)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      38.0437      0.000000000000
```

Breakdown Table (4 nodes and p = 8)

```
rank 0 computation time: 1.872291, communication time: 36.171417
rank 1 computation time: 5.170179, communication time: 29.758078
rank 2 computation time: 9.123489, communication time: 26.481491
rank 3 computation time: 14.420393, communication time: 22.246830
rank 4 computation time: 19.795707, communication time: 18.281887
rank 5 computation time: 23.044367, communication time: 15.961276
rank 6 computation time: 23.467928, communication time: 16.326590
rank 7 computation time: 25.398930, communication time: 14.395678
```

Observation

By comparing the summary tables from 5B and the summary tables above, there is a improvement in raw performance in call cases (where most processes spend more fraction of the total time on computation and less fraction of total time on communication). There is also an improvement in scalability. However, load balance doesn't seem to have changed much in general.

Explanation

There is an improvement in scalability and raw performance is probably because some processes won't have to spend a lot of time waiting for data to be sent from the lower rank processes. Instead, they are using that waiting time to do computation beforehand until they really need that data from lower rank processes to proceed. It can also be supported from the breakdown tables above where it shows significantly lower communication time for most of the processes. Load balance doesn't change much is probably because that each process still has the same amount of work to do as before.

Task 7

Note

If need to see the breakdown between computation and communication time for each rank, uncomment the `printf()` and `MPI_Barrier` near the end of my code.

Terminal Commands

- Compile and link to the code: `make task7`
- Build and execute the code: `sbatch task7.sh`, `sbatch ./p4n2.sh`, `sbatch ./p4n4.sh`, `sbatch ./p8n2.sh`, `sbatch ./p8n4.sh`
- View and collect the data: `vi slurm-X.out` where X should be replaced by the corresponding job number.

Idea

From above tables, in terms of computation time, it seems process rank 0 always has the least work to do and process rank (size-1) often has the most work to do (the amount of work other process need to do seems to be balanced). Therefore, I think it would improve the load balance if I move some of the work from process rank (size-1) to process rank 0. More specifically, since the very first multiplication work requires most work for process rank (size-1), I want to let process rank 0 do it instead. So ideally, the computation time of process rank 0 will go up a bit and the computation time of process rank (size-1) will go down a bit, making both of their computation time more close the the average computation time.

Tables

Summary Table (1 node and p = 4)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000     127.0518      0.000000007402
```

Breakdown Table (1 node and p = 4)

```
rank 0 computation time: 93.330135, communication time: 21.524862
rank 1 computation time: 30.556137, communication time: 95.369622
rank 2 computation time: 40.679861, communication time: 88.079306
rank 3 computation time: 28.072778, communication time: 2.358405
```

Summary Table (1 node and p = 8)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      52.1012      0.000000004072
```

Breakdown Table (1 node and p = 8)

```
rank 0 computation time: 26.028612, communication time: 22.052082
rank 1 computation time: 10.783051, communication time: 39.495694
rank 2 computation time: 16.026381, communication time: 34.378822
rank 3 computation time: 20.837132, communication time: 29.684453
rank 4 computation time: 23.534644, communication time: 27.130735
rank 5 computation time: 25.936885, communication time: 26.933781
rank 6 computation time: 26.923941, communication time: 26.916606
rank 7 computation time: 21.735925, communication time: 2.150612
```

Summary Table (2 nodes and p = 4)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000     126.9018      0.000000007402
```

Breakdown Table (2 nodes and p = 4)

```
rank 0 computation time: 93.348096, communication time: 25.376034
rank 1 computation time: 25.648524, communication time: 98.306792
rank 2 computation time: 39.889598, communication time: 88.707702
rank 3 computation time: 26.831173, communication time: 2.552535
```

Summary Table (2 nodes and p = 8)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      49.9274      0.000000004072
```

Breakdown Table (2 nodes and p = 8)

```
rank 0 computation time: 25.725106, communication time: 22.119080
rank 1 computation time: 6.227554, communication time: 41.401723
rank 2 computation time: 10.530138, communication time: 36.835508
rank 3 computation time: 16.781751, communication time: 30.520935
rank 4 computation time: 16.754856, communication time: 30.521480
rank 5 computation time: 20.097173, communication time: 30.110653
rank 6 computation time: 22.022731, communication time: 29.629901
rank 7 computation time: 19.347053, communication time: 2.058097
```

Summary Table (4 nodes and p = 4)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000     122.2609      0.000000007402
```

Breakdown Table (4 nodes and p = 4)

```
rank 0 computation time: 93.296066, communication time: 20.781308
rank 1 computation time: 25.814058, communication time: 94.946826
rank 2 computation time: 36.374935, communication time: 87.581864
rank 3 computation time: 23.449534, communication time: 2.524410
```

Summary Table (4 nodes and p = 8)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   8000      49.6063      0.000000004072
```

Breakdown Table (4 nodes and p = 8)

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
rank 0 computation time: 25.683303, communication time: 21.847841
rank 1 computation time: 4.868430, communication time: 40.737996
rank 2 computation time: 9.203814, communication time: 36.560429
rank 3 computation time: 13.518383, communication time: 32.293305
rank 4 computation time: 17.742659, communication time: 28.227957
rank 5 computation time: 21.238818, communication time: 29.103505
rank 6 computation time: 23.063019, communication time: 28.246910
rank 7 computation time: 17.710813, communication time: 2.027598
```

Observation

From the tables above, it seems that I did achieve what I was initially planning for, which was to increase the amount of computation time of process rank 0 and decrease the computation time of process rank (size-1). However, both the raw performance and scalability decreases, and the load balance between processes without process rank 0 does become more balanced. I think this is method does what I wanted it to do, but failed in general because such slight improvement in load balance cost such high performance and scalability.

Explanation

As shown in the tables above, the computation time of process rank 0 increases significantly after the change I made, and, because of the increase in the amount of computation time of process rank 0, subsequent processes have to wait longer and results in a increase in communication times. After analyzing the time spend on each section of the code, I noticed that the extra work assigned to process rank 0 seems to be too heavy even though I am using the exactly same approach for process rank 0 to do the extra work as process rank (size-1) would initially. The reason for this not going as much as I hoped is probably that process rank 0 is running out of memory on the heap since I had to assign extra space for process rank 0 to store the corresponding data for the extra work.

Task 8

Terminal Commands

- Compile and link to the code: `make task8`
- Build and execute the code: `sbatch ./task8.sh`
- View and collect the data: `vi slurm-X.out` where X should be replaced by the corresponding job number.

Summary Table

```
Matrix multiplication times:
    N      TIME (secs)    F-norm of Error
  -----   -------------  -----------------
   1000       0.1466      0.000000000016
   2000       0.4759      0.000000000044
   4000       5.1068      0.000000000098
   7633      49.7784      0.000000000704
   8000      57.1916      0.000000000000
```