

HPSC Lab1

Luna McBride, Noki Cheng

September 2023

1 Parallel Design

The process of implementing nPE processors to run *lookup* function on subsets can be split into following several steps:

1. **Initialize MPI:** we initialize the MPI process by assigning values for *the number of PE* and *myPE* (also known as rank);
2. **Define data array:** we define and claim space for the arrays as the inputs for the function *lookupVal*;
3. **Scatter the data:** we use *MPI_Scatter* function to distribute input data x to different processors, and each one of them should receive a subset of array;
4. **Conduct computation:** each processor should perform operations on their assigned access to the input array;
5. **Gather results:** we apply *MPI_Gather* function to collect results from different processors;
6. **Finalized MPI:** at the end we terminate the MPI process by calling function *MPI_Finalize*.

Overall, the parallel implementation in this example is mostly built on functions *lookupVal*, *MPI_Scatter*, and *MPI_Gather*. By assigning different processors working on different parts of sub-arrays, the total computational efficiency gets significantly improved.

2 Self Evaluation

2.1 Luna

I feel like I could have done better with this lab. I had gone into it without realizing my `rc` account had been suspended from not doing the survey, so I had been waiting on a response about reactivation (right before a holiday weekend no less). It was entirely my fault. Because of that, I feel like I pushed what ended up being the harder section (parallelization) onto Noki. We had only barely finished the first part before the end of lab, so it only made sense to split the two remaining tasks based off of his ability to do the parallel programming sooner while I was waiting for access.

This issue did lead to something interesting, being pair programming with one computer in the lab itself. As two Python programmers trying to code some c++, it was funny how often one would make a mistake by using Python logic while the other would bring up the missing double, ;, or . I had the advantage in terms of programming knowledge, but that does not mean we were not on equal footing.

As for results, this was not a crazy lab at all. I do point out in Appendix C the differing results between two jobs, which is expected in parallelization. That does not mean it is not interesting to see which core will win the race, so to speak.

2.2 Noki

This lab assignment didn't go very smoothly. As a student with a background in theoretical analysis, I found it challenging to understand the demo command lines and C++ code shared in class. I had trouble with some details of the C language during the lab, which is a programming language learned seven years ago and never used. I also had difficulties with MPI when implementing the lookup function in parallel. I'm relieved that I managed to figure everything out before the lab report deadline, and I consider myself lucky to be taking this class because facing these challenges pushes me out of my comfort zone and helps me learn new things.

3 Appendix A: Parallel Code

```
1 // This is the code for parallel implementation
2 // of lookup file
3
4 #include <mpi.h>      // Required for MPI
5 #include <stdio.h>
6
7 #include "main.h"
8
9 void lookupVal(int n, int N, double *x, double *y,
10              double *xval, double *values)
11 {
12     for ( int j = 0 ; j < N ; ++j){
13         double xval_single = xval[j];
14         for ( int i = 0 ; i < n ; ++i){
15             if ( xval_single >= x[i] && xval_single <= x[i+1] ){
16                 values[j] = y[i] + (xval_single - x[i])
17                     * (y[i+1]-y[i]) / (x[i+1]-x[i]);
18                 break;
19             }
20         }
21     }
22 }
23
24 int main(int argc, char* argv[]) {
25     // Generate values for x and y
26     int n = 100;
27     double x[n], y[n];
28
29     for ( int i = 0 ; i < n ; ++i)
```

```

30 {
31     x[i] = i;
32     y[i] = i*i;
33 }
34
35 // Begin MPI process
36 MPI_Init(&argc, &argv);
37
38 int numPE = 4;
39 int myPE;
40 MPI_Comm_size(MPI_COMM_WORLD, &numPE);
41 MPI_Comm_rank(MPI_COMM_WORLD, &myPE);
42
43 int arraySize = 20;
44 int subArraySize = arraySize / numPE;
45
46 double* globalArray = new double[arraySize];
47 double* localArray = new double[subArraySize];
48
49 // Only root process initializes the global array
50 if (myPE == 0) {
51     for (int i = 0; i < arraySize; ++i) {
52         globalArray[i] = 1.0 * i;
53     }
54 }
55
56 // Scatter the data to each processor
57 MPI_Scatter(globalArray, subArraySize, MPI_DOUBLE,
58            localArray, subArraySize, MPI_DOUBLE, 0, MPI_COMM_WORLD);
59
60 // Perform function evaluation on the local sub-array
61 double localValues[subArraySize];
62 lookupVal(n, subArraySize, x, y, localArray, localValues);
63
64 for (int i = 0; i < subArraySize; ++i) {
65     cout << localArray[i] << ": " << localValues[i] << endl;
66 }
67
68 // Gather the results to the root processor
69 double* globalValues = nullptr;
70 if (myPE == 0) {
71     globalValues = new double[arraySize];
72 }
73 MPI_Gather(localValues, subArraySize, MPI_DOUBLE,
74            globalValues, subArraySize, MPI_DOUBLE, 0, MPI_COMM_WORLD);
75
76 // Only the root process prints the result
77 if (myPE == 0) {
78     cout << "Squared array: ";
79     for (int i = 0; i < arraySize; ++i) {
80         cout << globalValues[i] << " ";
81     }
82     cout << endl;
83 }
84

```

```

85     MPI_Finalize();
86     return 0;
87 }

```

4 Appendix B: Slurm Code

```

1  #!/bin/bash
2
3  # -
4  # |
5  # | This is a batch script for running a MPI parallel job on Alpine
6  # |
7  # | (o) To submit this job, enter: sbatch slurm.bat
8  # |
9  # | (o) To check the status of this job, enter: squeue -u <username>
10 # |
11 # -
12
13 # -
14 # |
15 # | Part 1: Directives
16 # |
17 # -
18
19 #SBATCH --nodes=4
20 #SBATCH --ntasks=4
21 #SBATCH --time=00:01:00
22 #SBATCH --partition=amilan
23 #SBATCH --output=slurm-%j.out
24
25 # -
26 # |
27 # | Part 2: Loading software
28 # |
29 # -
30
31 module purge
32 module load intel
33 module load impi
34 module load gcc
35 module load openmpi
36
37 # -
38 # |
39 # | Part 3: User scripting
40 # |
41 # -
42
43 echo "=="
44 echo "||"
45 echo "|| Begin Execution of fd in slurm batch script."
46 echo "||"
47 echo "=="
48
49 srun -n 4 ./lookup_mpi > output.out

```

```

50
51 echo "=="
52 echo "||"
53 echo "|| Execution of ./main in slurm batch script complete."
54 echo "||"
55 echo "=="

```

5 Appendix C: Outputs

5.1 Parallel

```

(base) [nuch1756@c3cpu-c15-u1-1 lookup]$ mpic++ lookup_mpi.cpp -o lookup_mpi
(base) [nuch1756@c3cpu-c15-u1-1 lookup]$ mpirun --oversubscribe -n 4 lookup_mpi
0: 0
1: 1
2: 4
3: 9
4: 16
5: 25
6: 36
7: 49
8: 64
9: 81
10: 100
11: 121
12: 144
13: 169
14: 196
15: 225
16: 256
17: 289
18: 324
19: 361
Squared array: 0 1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361
(base) [nuch1756@c3cpu-c15-u1-1 lookup]$ 

```

Figure 1: Output from the parallel implementation part.

5.2 Slurm

```

1
2 Lmod is automatically replacing "intel/2022.1.2" with "gcc/11.2.0".
3
4
5 Inactive Modules:
6   1) impi
7
8 ==
9 ||
10 || Begin Execution of fd in slurm batch script.
11 ||
12 ==
13 15: 225

```

```

14 16: 256
15 17: 289
16 18: 324
17 19: 361
18 5: 25
19 6: 36
20 7: 49
21 8: 64
22 9: 81
23 10: 100
24 11: 121
25 12: 144
26 13: 169
27 14: 196
28 0: 0
29 1: 1
30 2: 4
31 3: 9
32 4: 16
33 Squared array: 0 1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361
34 ==
35 ||
36 || Execution of ./main in slurm batch script complete.
37 ||
38 ==

```

Listing 1: Output before sending output to output.out

```

1 15: 225
2 16: 256
3 17: 289
4 18: 324
5 19: 361
6 10: 100
7 11: 121
8 12: 144
9 13: 169
10 14: 196
11 5: 25
12 6: 36
13 7: 49
14 8: 64
15 9: 81
16 0: 0
17 1: 1
18 2: 4
19 3: 9
20 4: 16
21 Squared array: 0 1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361

```

Listing 2: output.out

The two files are very different runs, as shown by how the different runs parallelized differently, leading to a different order in the outputs.

6 Appendix D: Math

6.1 Questions 1, 3, and 5 (Luna)

As bad as this handwriting in paint looks, I feel like you would prefer to see this instead of my sloppy handwriting caused by wrist injury and pain. Pen/pencil to paper is just a terrible time for me. Plus, who can beat copy paste.

Math

$$\textcircled{1} \quad x_1 + 3x_2 + x_3 = 6$$

$$x_2 - x_3 = -3 \rightarrow x_2 = x_3 - 3$$

$$-x_1 - 3x_2 = 12 \rightarrow x_1 = -3x_2 - 12$$

$$(-3x_2 - 12) + 3x_2 + x_3 = 6$$

$$-3(x_3 - 3) - 12 + 3(x_3 - 3) + x_3 = 6$$

$$-3x_3 + 9 - 12 + 3x_3 - 9 + x_3 = 6$$

$$x_3 - 12 = 6$$

$$x_3 = 18$$

$$x_2 = 18 - 3$$

$$x_2 = 15$$

$$x_1 = -3(15) - 12$$

$$x_1 = -45 - 12$$

$$x_1 = -57$$

$$\begin{aligned} x_1 &= -57 \\ x_2 &= 15 \\ x_3 &= 18 \end{aligned}$$

Figure 2: Math 1

math

(3)

$$1x_1 + 3x_2 + 1x_3 = 6$$

$$0x_1 + 1x_2 - 1x_3 = -3$$

$$-1x_1 - 3x_2 + 0x_3 = 12$$

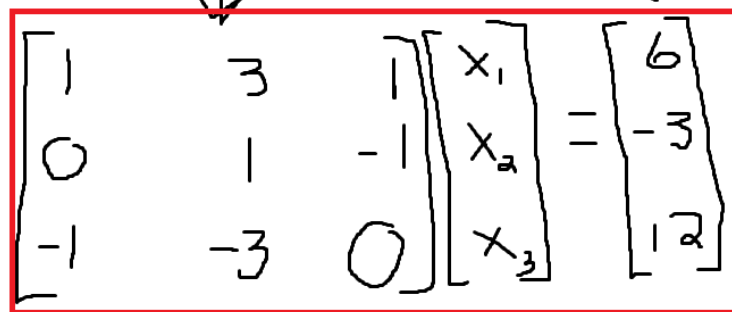

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & -1 \\ -1 & -3 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -3 \\ 12 \end{bmatrix}$$

Figure 3: Math 3

Math

⑤

$$\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 1 \\ 1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 1 \\ 0 \end{bmatrix}$$

$$1 \cdot 1 + 0 \cdot -1 + -2 \cdot 2 = -3$$

$$1 \cdot 0 + 1 \cdot -1 + 1 \cdot 2 = 1$$

$$1 \cdot 1 + 3 \cdot -1 + 1 \cdot 2 = 0$$

Figure 4: Math 5

6.2 Questions 2, 4, and 6 (Noki)

$$2. \begin{bmatrix} 1 & 0 & -2 \\ -2 & 1 & 6 \\ 3 & -2 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 7 \\ -3 \end{bmatrix}$$

$$\rightarrow \left[\begin{array}{ccc|c} 1 & 0 & -2 & -1 \\ 0 & 1 & 2 & 5 \\ 0 & -2 & 1 & 0 \end{array} \right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & -2 & -1 \\ 0 & 1 & 2 & 5 \\ 0 & 0 & 5 & 10 \end{array} \right]$$

$$\rightarrow x_3 = 2, x_2 = 1, x_1 = 3 \rightarrow \vec{x} = [3, 1, 2]^T$$

$$4. \begin{bmatrix} 1 & 0 & -2 \\ -2 & 1 & 6 \\ 3 & -2 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 7 \\ -3 \end{bmatrix}$$

$$6. \begin{bmatrix} 3 & -2 & 2 \\ 1 & 4 & -2 \\ 2 & -5 & 0 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 4 \\ -1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\Rightarrow y_1 = 3 \times 2 - 2 \times 4 - 2 \times 1 = -4$$

$$y_2 = 1 \times 2 + 4 \times 4 + 2 \times 1 = 20$$

$$y_3 = 2 \times 2 - 5 \times 4 - 0 \times 1 = -16$$