Darryn Wong
CPSC 474
Dr.Bein

Calculating All Non-Zero Values in a Matrix Using MPI

Introduction

This project was built mainly to enhance my understanding of MPI. In this project the program calculates all the non-zero values in a given matrix and outputs the values and their position in the matrix to the terminal.

How to Run

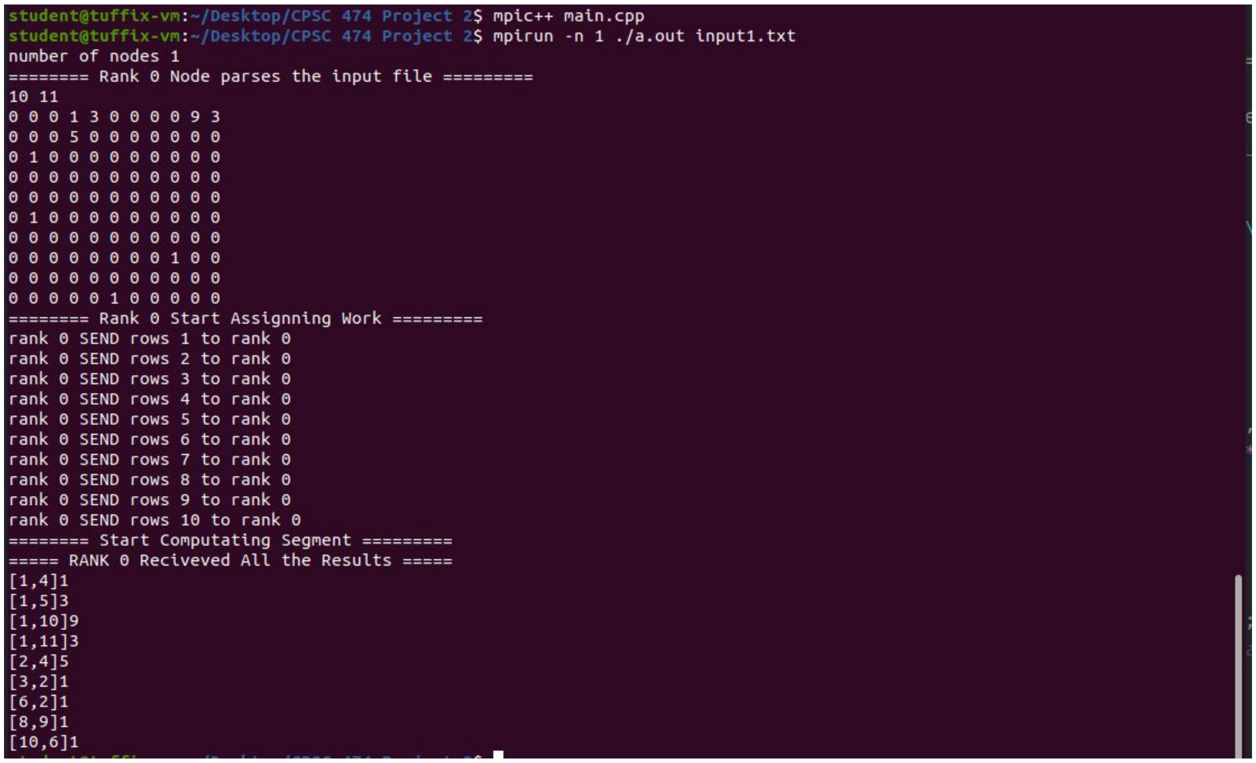
This program uses MPI and g++ compiler. It is tested and works on Tuffix. If you are running locally make sure to have the lates MPI library and g++ compiler installed. The compiling method provided uses g++ compiler supplied by Tuffix.

To compile:
```
mpic++ main.cpp
```
To run with default input file(input.txt) and 1 process:
```
mpirun -n 1 ./a.out
```
To run with default input file and 5 process:
```
mpirun -n 5 ./a.out
```
To run with any input file (using "input1.txt in this case") and 2 process:
```
mpirun -n 5 ./a.out input1.txt
```

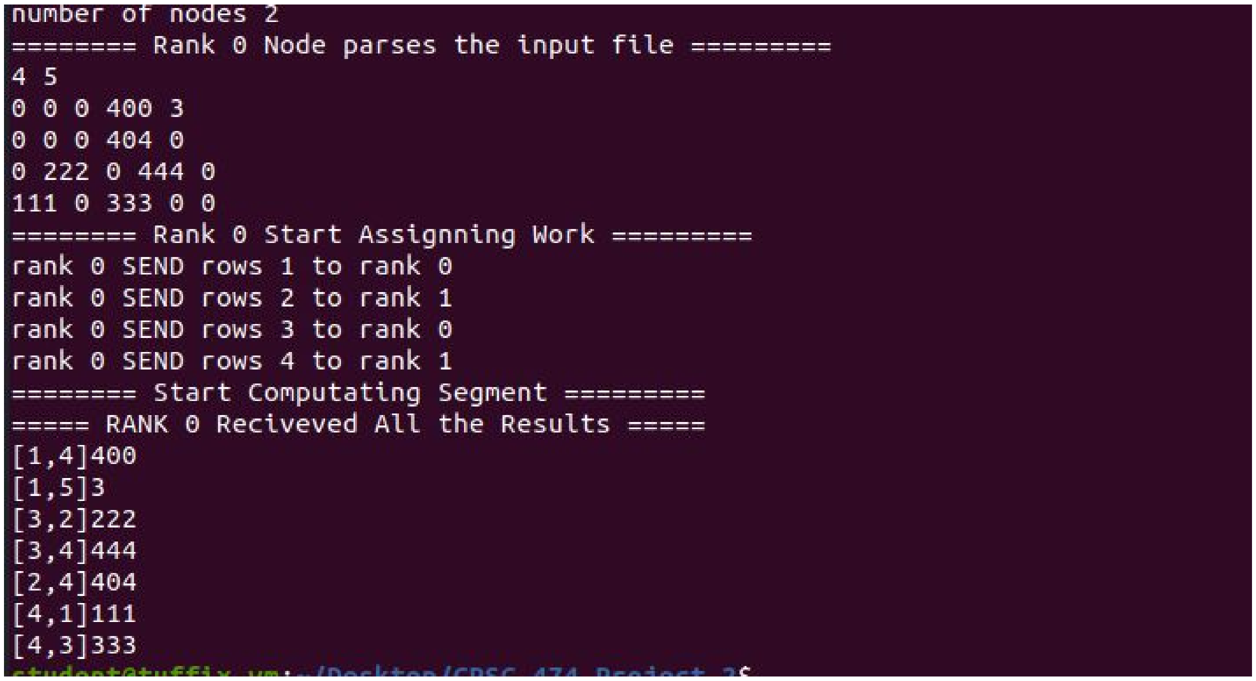
Pseudo code

1. Rank 0 will read the input file and parse it as a 2D array.
2. Rank 0 will take the row count and use a cyclic distribution to distribute them to all the processors (including itself)
3. All the procross will compute their portion and sends the results back to Rank 0
4. When Rank 0 receives the results it will output the in the format "[row,column] value"

Sample Output and Running The program



```
student@tuffix-vm:~/Desktop/CPSC 474 Project 2$ mpic++ main.cpp
student@tuffix-vm:~/Desktop/CPSC 474 Project 2$ mpirun -n 1 ./a.out input1.txt
number of nodes 1
======== Rank 0 Node parses the input file =========
10 11
0 0 0 1 3 0 0 0 0 9 3
0 0 0 5 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
======== Rank 0 Start Assignning Work =========
rank 0 SEND rows 1 to rank 0
rank 0 SEND rows 2 to rank 0
rank 0 SEND rows 3 to rank 0
rank 0 SEND rows 4 to rank 0
rank 0 SEND rows 5 to rank 0
rank 0 SEND rows 6 to rank 0
rank 0 SEND rows 7 to rank 0
rank 0 SEND rows 8 to rank 0
rank 0 SEND rows 9 to rank 0
rank 0 SEND rows 10 to rank 0
======== Start Computating Segment =========
===== RANK 0 Reciveved All the Results =====
[1,4]1
[1,5]3
[1,10]9
[1,11]3
[2,4]5
[3,2]1
[6,2]1
[8,9]1
[10,6]1
```



```
number of nodes 2
======== Rank 0 Node parses the input file =========
4 5
0 0 0 400 3
0 0 0 404 0
0 222 0 444 0
111 0 333 0 0
======== Rank 0 Start Assignning Work =========
rank 0 SEND rows 1 to rank 0
rank 0 SEND rows 2 to rank 1
rank 0 SEND rows 3 to rank 0
rank 0 SEND rows 4 to rank 1
======== Start Computating Segment =========
===== RANK 0 Reciveved All the Results =====
[1,4]400
[1,5]3
[3,2]222
[3,4]444
[2,4]404
[4,1]111
[4,3]333
```