CP631 Advanced Parallel Programing Assignment 1

Qing Dai, David (Student Number: 235821890)

# File Layout

Assignment1

| -- question 1 : Answers to the question 1

| -- Eggholder.c : The function that parallel programs working on

| -- q1p1.c : Answer to 1st part of question 1

| -- q1p2.c : Answer to 2nd part of question 1

| -- q1p3.c : Answer to 3rd part of question 1

| -- question 2 : Answer to question 2

| -- q2.c : Compound answer to question 2, which comprises:

| -- Gather\_save\_matrix : answer to section a of question 2

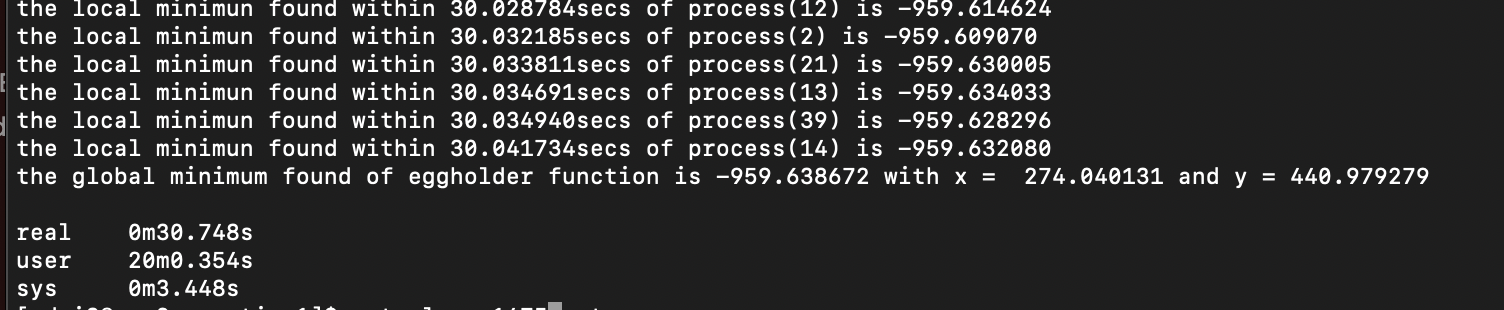
| -- Read\_scatter\_matrix : answer to section b of question 2

# Test result

## Question 1

### Snapshots or logs of result of running

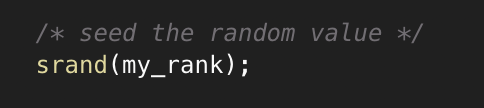
* Part 1
  + Job script: job\_q1p1.sh
  + Log file: slurm-1468(q1p1).out
  + Snapshot: A computer screen with numbers and a black background

    Description automatically generated
* Part 2
  + Job script: job\_q1p2.sh
  + Log file: slurm-1475(q1p2).out
  + Snapshot:
  + 
* Part 3
  + Job script: job\_q1p3.sh
  + Log file: slurm-1476(q1p3).out
  + Snapshot:
  + A screen shot of a computer

    Description automatically generated

### Key pionts of code

* The code ensuring that each process generates a different random sequence.



* Take parameters to ease execution of program, which enable change parameter by simply passing different initial parameter without recompiling the file!
  + Q1p1: [-n total Iteraion Per process]
  + Q1p2: [-s second to keep process running]
  + Q1p3: [-s seconds for differenc check interval] [-d difference(threshold) ]

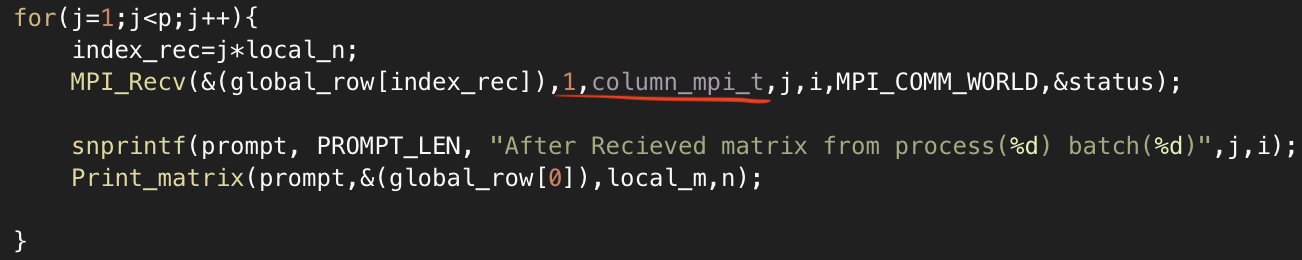
## Question 2

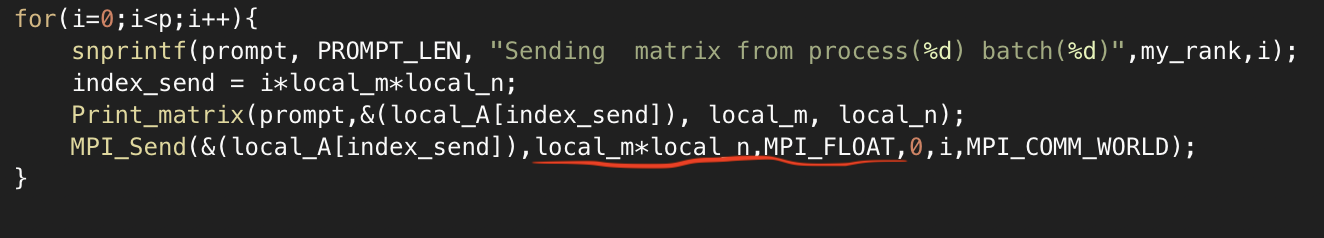
### Snapshots and logs of result of running

* Source code: q2.c
* Job script:
* Log file:
* Snapshot:

### Key pionts of code

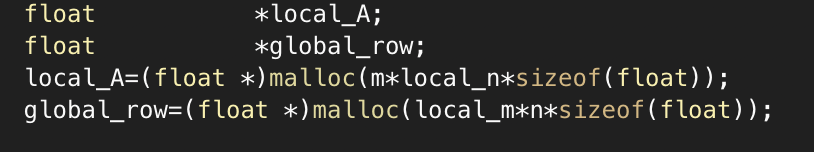
* Using MPI\_Type\_vector to increate readability code, interesting thing is global\_row and local\_A utilize different type to make the code more concise:



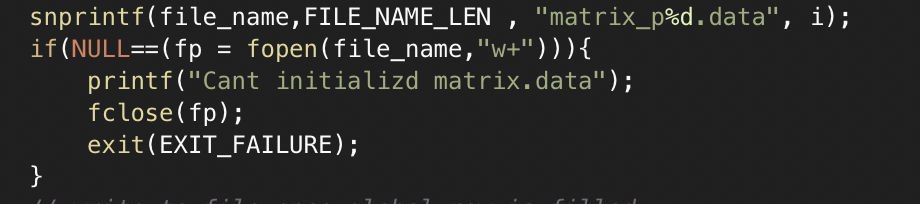


It works perfect in : Gather\_save\_matrix but can’t make it in Read\_scatter\_matrix, still struggle to know why.

* Isolation of memory of two functions: Gather\_save\_matrix vs. Read\_scatter\_matrix



However, such methods caused blank readout which I can’t handle in time.

* Create different files for each process: to maximize parallelization and minimize interference between processes, each process has its own file
* 
* A screen shot of a computer code

  Description automatically generated

### Performance Analysis and findings

* Keep the process no.(p) and increase the size of matrix(m\*n)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trail No. | Size of matrix(m\*n) | Process no.(p) | Time (Gather\_save\_matrix ) | Time (Read\_scatter\_matrix) |
| 1 | 64 | 32 | <1s | <1s |
| 2 | 256 | 32 | <1s | <1s |
| 3 | 1024 | 32 | <2s | <2s |

* Keep the ratio of matrix(m\*n) / process no.(p) and increase the size of matrix(m\*n)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trail No. | Size of matrix(m\*n) | Process no.(p) | Time (Gather\_save\_matrix ) | Time  (Read\_scatter\_matrix) |
| 1 | 256 | 16 | <2s | <2s |
| 2 | 1024 | 32 | <2s | <2s |
| 3 | 2048 | 64 | <5s | <5s |

Above statistics is done on test server1 on the mid-night, which can be assumed the same result as on test server.

Since the test server is so powerful, most of operations will not last more than 2s, but we can also have interesting observation:

* Expanding size of matrix will be relatively is more obvious, especially the first time of reach that number. The system buffer will be effective immediately for later operations
* Even the same ratio, the more size of matrix the more detail it will be.