Introduction to the program

**Client.erl**:

client is used to initiate server and get inputs from the user. Firstly, getting the number of required 0’s, then getting the number of workers (processes), the program will run depends on the work unit set in the code. Also the user can enter the command of adding new workers to the server according to the Ip address of the server, which is given right after the client started. Finally, user can enter the command “shutdown all” to shut down the server as well as all the workers to finish the program.

**Server.erl**:

server is used to react to the command received from the client. Server works as the master which keeps receiving messages from all the workers and store result of hash strings in the file “data.txt”. Server itself can not mine coins, but there is one worker work permanently for the server even when there are no other workers. So, the server can keep getting coins when there is no other worker.

The server has the function to create workers and add new workers while mining the coin. After the server receiving the coin from the worker, it will send message to the worker via process identifier to tell the worker that keep on working until the worker finishes all the job assigned. When worker finishes its job, the server will delete the worker from the list which is used to store the names of all workers. After all the workers except one worker belong to the server itself finish their jobs, the server will count the CPU time and real time, then we can use the ratio of these two time to know the situation of parallelism of our program.

**Worker.erl**:

works as the worker to mine the coin. User can create several numbers of worker to work at the same time. Worker is the core of distributed implementation, using the erlang Actor Model to create a large number of processes (parallelism).

Size of work unit for best performance

To get the size of work unit that can result in the best performance, we set the required 0’s number is 4, the number of workers working at the same time is 20.

|  |  |
| --- | --- |
| Work unit | Ratio of CPU time to real time |
| 1 | 3.2382 |
| 10 | 6.1805 |
| 30 | 7.1010 |
| 40 | 7.0512 |
| 50 | 7.0003 |
| 100 | 6.7730 |
| 500 | 6.9805 |

According to the above table, the best work unit is approximately 30, but the work unit around 50 or more then 50 also have good performance.

Steps to run the program

1. Enter “client:start().”, to start the program.
2. Enter the number of required 0’s, for example, 4.
3. Enter the number of workers, for example, 10.
4. Enter the Name and ip address to add new worker while does not show “Finished”.
5. Enter “shutdown all” to finish the server and all the workers.

Setting of the running computer

CPU: Intel(R) Core(TM) i7-9700KF CPU @ 3.60GHz; RAM: 16GB

Results of running the program

**Text

Description automatically generated**

**The result of running your program for input 4**

A picture containing text

Description automatically generated

**The ratio of CPU time to Real time**

Graphical user interface, text, application, email

Description automatically generated

**The coin with the most 0s you managed to find**

The coin we find with the most 0’s has 7 zeros: （only find two coins in almost one hour）

qinxuan>Mk?@b 0000000a58a15d3bd11785bd12bc1e9e95c5d48ad6379040ca3c1f249039f217

qinxuan^?,A!Ckk 0000000160bead80dc14bc477db24aab293a758858c0cb41f322662902ffda46

Graphical user interface, text, application, email

Description automatically generated

**The largest number of working machines you were able to run your code with**

4 machines