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

```
qinxuan|/%.w>q= 00008002bd87c1a3c9cdc9ebcf73a9c51c97bf7b765e8b35b6e56e3f1fd1f029
qinxuanhZ(vz0 0000b1ef6a4222b88765aa992e557b2c5ab2e73a4717cf91037b32aa99c918e6
qinxuan^$m[!5<t 0000d8f86c876c7729ec9064c5653c12c768ad9660cb437cf0cc06d6304d47da
qinxuan4FzVx 0000ec67802873a6198eb8c35f608ed2415f7397a8e81806a66a7253d740e5ad
qinxuanIvW7+ 0000aef91bb4544958b53fd7b9afff4361dcaab0fd834d51e9db3604e49fcefd
qinxuant"!&vV1M 000059f3df006948dfada22f784e9d4ae6bb3ba8fcc28d47973fd720727cbf05
qinxuan-'Qp-i9 00003096976366 6a2e5085d408b8978c25b1c7c9c351848998c1fcf60a121ae1
qinxuan#fxbWd 0000d2d4f79b3a757481b55679484357ca9f5c2179f8b9187ca703746cfcbb6f
qinxuanFrJSrN^3 00005a8a3885d314356b31ea2853c4f2568b70291c6b676420df0850f5c4c643
qinxuanROdss 0000c09b8f9c9ed9802528cca5ad7bad351a3cce8703a38a5cef37573b936740
qinxuanTl1#1F 00001fe6a6d490f784f63818db313458b9e23497d94eaff80c6bcf5d960091b9
qinxuan0!3kc02 000044fd82c0f70bd03930ef05a58a7d758a55935062c2db12ce562b681ac594
qinxuan{[Bg3 00009d04b5d551e11b46ed1bb95c4f3e7f6fdde7dd7bda2d1e1e6813f1cf1672
qinxuan3Z#hcQ1Q 00006befb0910f5b8b0c2ff9fc3b4eddfc54c1de782e427643cbf65f7cb80676
qinxuan-wxGJb/ 0000d9df0c1d55805057f09da7346905c96db2cc07116fd6055860a8becbf03d
qinxuan'BT3#Q&' 0000f4c523fdab51aca2d0c26d684ee7818398f2a17cd8759ef7859afc415a9b
qinxuan\~2B%6P* 0000f7353974b07e0f8255325fa6acd4c9afec1268734412ac400c51afbcbf9c
qinxuanHcO[7Sl" 0000b815f921ce0ab45f03089c908a4985b7a73bde54c9bc77b7489e420f7f11
qinxuan3h7,A_# 00001a302efd88886f3d56cafd4bcf1f38a67f8cd4899d7e6dee6ebc4fd768cf
qinxuan+e5D.?D 0000d566b3c234e37ae40c067baad0837bd010c28226a2921c041e85716b8fc6
qinxuanB?.Gh 0000a3b7f6598ac9a3ab015d1d01efdfb2d73b65590a40692520464867405cb5
qinxuan8dt&ka 00008f3a418e3a51c29aacfdba47f4e66c13d7923ea4a803800fcdfc253b5cfd
qinxuanVTRws&_ 000083c6c1027b263735c2f1e06865694908dff3a32a78f9afda548efe6f8c7d
qinxuanR.J=o 0000890803fa949dba7530d3978494e841e7739435ce69d041548ede7271f2d7
qinxuanbuOBX 0000ff496fd7f7a925a748d7bf59ef63b7f5508b285211405bbe57829de4d777
qinxuan1gNc~-> 0000b2132319d4ff553fc6df6339302784b7a21f9104e7090396421078b7abdd
qinxuan'C.40 00009ecc364226fe087e18d558d6336fd7e28569193af22c7082d2678394811b
qinxuan0+FL_hp2 00002fc2101e1e6c4701479f0e9db5d9d565e19a9af88d934a7c054388aafe8c
qinxuan.9/z<UX 0000b0a817365bb1f313f4bee63097c834e43e2446a97524c02ff2f467d7b51f
qinxuan:ZHiA/R 0000ad9c3184389f0655689b69610 6d1c42d67ac2fa8aba8cfe488357b459e48
qinxuanY40o'; 00005403f78bc4c702f85f7974fa86c46d98af2c25c4f1a8c8e362ca8127acd1
qinxuan<4[{) 0000d3a9b4615bb11694572c1fc3ebe213fe30d9f93a1f9f1c1013706facc042
qinxuann?L}>ebo 0000e23c3ce41e3f267d940e4fcf409f97304d583660fd7f621f091f1d722bdf
qinxuanrth.?i_T 0000772edfd7f646024a4a5df4b522ffd9faebf4102212d31d1ea26eb69a5b2c
qinxuan1hH/Ekzq 0000e15b1aeb342d4b23a0a62bde04665bb57a519b1c5ae8eda069b489239cd6
qinxuanEB_-k88 00004029d2cf75eb884395c85f4ca50f04c4126361be152cd98769acfa7cc63b
qinxuan{7!qT 0000885659b6251e512b3371d0b7b2b7125a596b443d54a37551bdb16fbc3d0c
qinxuannGnP2[H 0000a05f5274c895bb8d9dba15a7ddf305fe87b53423910ae1f59f96797fa5f4
qinxuan!Dc_U) 00004e3b4c83bc04c5296ba2801dc9bbe5f1ed98fb5d2a5db20602882ecbc21a
qinxuance$ur1y 00004aaa169ef6101c47a5d281aee292cb9bf44b1578293637c35096118bc5e2
qinxuank{0r^V 0000fb655bb567497d855f13f71446ea5f480f365be1a3b1eb277955761b35fa
```

The result of running your program for input 4

The ratio of CPU time to Real time

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

```
"C:/Program Files/Erlang OTP/erts-13.0.4\bin\erl" -pa C:\UF\Fall-2022\DOSP\project1\_build\default\l
Eshell V13.0.4 (abort with ^G)
1> client:start().
server started, address: 10.2.0.252
Enter the required number of leading 0's>7
Enter the number of workers>3
Enter>qinxuan>Mk?@b 0000000a58a15d3bd11785bd12bc1e9e95c5d48ad6379040ca3c1f249039f217
Enter>
```

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

4 machines