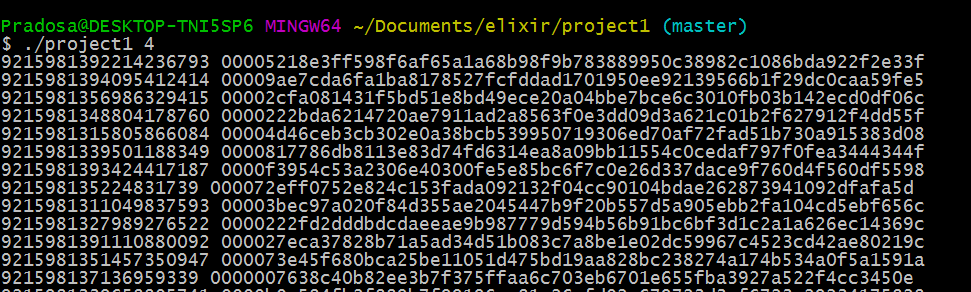
Project 1

Distributed Systems

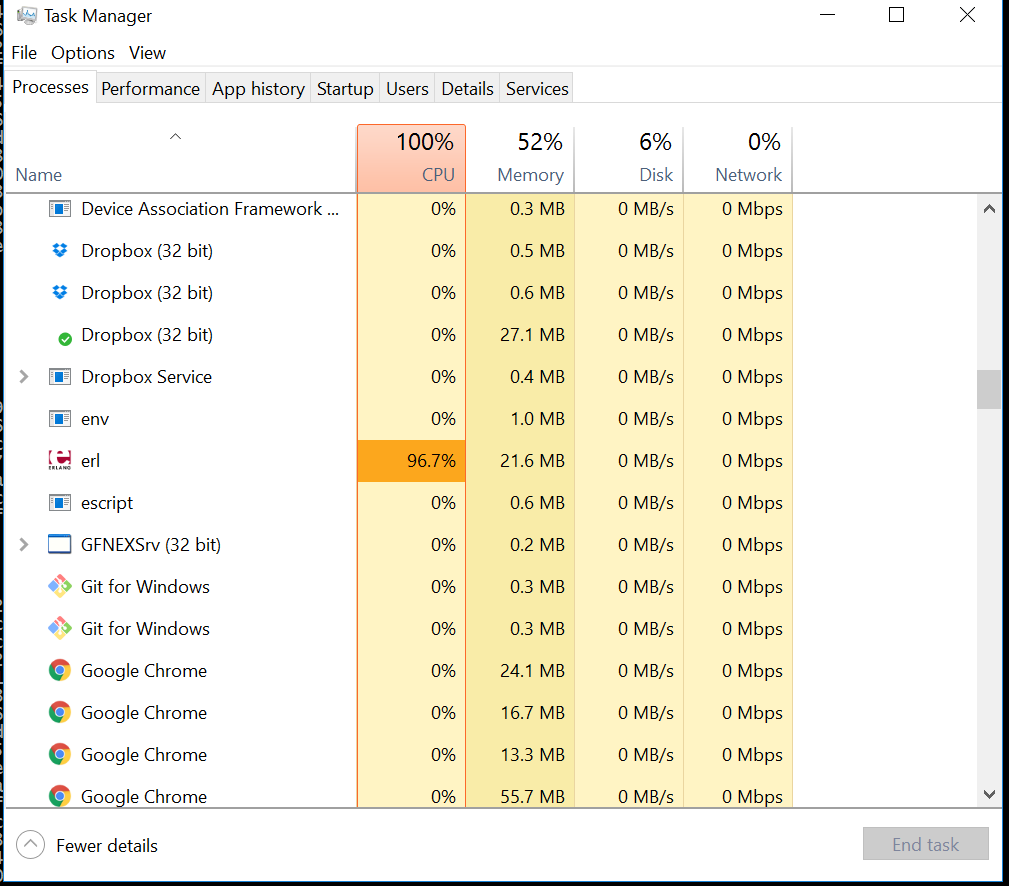
Pradosa Patnaik (UFID :12889584), Supraba Muruganantham (UFID : 92159813)

The project includes a bit coin mining program which runs in Server client model.In the server mode, the program generates random numbers and appends with a pre-defined string (UFID) and generates hash-codes using the SHA256 hash code generator. The output of the hash-code is then verified for the required number of leading zeros, the codes are printed on the console only when it generates code with the required or more number of leading zeros. To exploit the full computing power of the server we spawn multiple actors (more than the number of cores available) and ensure 100% concurrency is achieved. As the elixir process are the lightweight process we spawned, four times more process than the available cores.In this mode along with mining coin, server also has the capacity to accommodate clients and assign tasks. Whenever a Client (different machine with own computing power) connects with the Server, Server assigns works to the clients in chunks called work-units.

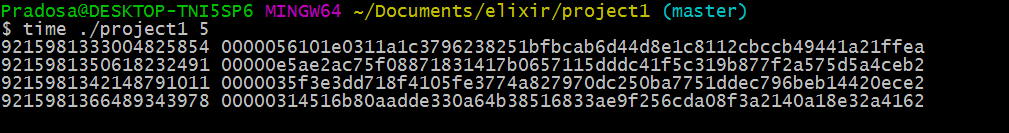
A work-unit is defined as the number of strings assigned by the server to the client for one request. Upon request from the client, Server spawns multiple process to generate the random and strings and sends to the client. In this way server takes care that no client mining the same coin. To run the program in Client mode we need to provide the server’s ip address from the command line. E.g ./project1 192.168.0.9.Then the Client machine requests for task from the server. Client also exploits the actor model design and spawns’ multiple processes locally. Each process then obtains their share of work (batches of string called Work-Uints) from the server and start generating the hash-code. The verification of the validity of the code happens at the client side locally, the server is only communicated as soon as a valid hash-code is found to be printed. Note : Client does not generates strings or print hash code on it’s end .Client keeps hashing the strings until one work-unit is finished and requests for more work-units after the current batch is finished.

1. Result of running ./project 4

CPU utilization monitored in windows for mining coins with 4 leading zeros



Result of running ./project 5:



1. Calculation of CPU utilization:

Calculation for the cup performance in the above case is done with the time command in linux.

Running time ./project1 5 Obtained the three time outputs Real time, user time and System time, where user time reflects the cup time.

Real time : 0m12.680s

User time : 0m48.010s

System time: 0m0.54s

CPU utilization = 48.010/12.380 = 3.87

Which indicates almost 4 out of 4 cores were fully utilized during the execution.

1. Concept of Work-Unit:

The best work-unit we obtained for mining 5 leading zeros is by sending work-unit of size around 50000 strings per work-unit. We determined this number by gradually increasing the work-unit size from small to large number of random strings .For small work units the performance was very poor because of the communication overhead. And for very large work unit also the performance was poor as most of the worker had to sit idle waiting for work units to be assigned.

1. Hash-code with maximum number of zeros:

We could generate hash code with maximum 11 zeros mining with one server with intel i7 4cores and 3 clients with 4 cores. It took around 4hr.26m to generate the code.

1. Largest number of machines used:

The largest number of machines we could use for our project is 4 intel i7 quadcore machines. Where one machine served as the Server and others served as Client.