Individual Report

Nikita Evsiukov
nikita.evsiukov@epfl.ch

January 22, 2024

1 Software Features and Contributions

1.1 Feature 1: Execution hash compilator adaptation

I worked on adapting the LLVM used by *FuzzCoin project* and created a bash script to facilitate the fuzzing process and to obtain the execution hash. The main technical challenge was to transfer the codebase to the new version of llvm-17 and make the execution hash generation thread-safe using temporary unique paths. One of the challenges with this task was the distribution of the binaries of the compiler, Milana proposed an idea to store binaries outside of the repo, so I made a new repository and a script that would download these binaries and other required libraries. For this part, Derya helped me a lot with debugging the execution hash generation script, and Milana pointed out the not-threadsafe initial implementation which saved me a huge amount of time. Milana and Derya also found the issue with my script and zero seed value that breaks the execution hash generation. It definitely saved me a night of debugging the integration tests.

1.2 Feature 2: Crypto utils

I also made a script that will take the execution hash from the previous module to generate a public key from it. I found multiple solutions, though I ended up deciding to keep the elliptic curve cryptography as it allows to use of arbitrary binary data as an initial private key. I decided to use *pem* format for this key to store them in the files persistently. I also initially used an elliptic curve that is not supported by the standard go library, which was pointed out by Derya, so I needed to change the base curve.

1.3 Feature 3: Integrating Proof of Work and time service

To keep a constant mining time between nodes, I also implemented a difficulty module to keep the system converged and stable. I implemented and tested the Network Time service. In this part, Milana helped me a lot with the idea of including timeservice messages in the Status message to synchronize the time between neighbouring nodes. This reduces the overhead of the module dramatically.

1.4 Feature 4: Integration tests

I worked on implementing the integration tests for the blockchain module so we could test it before introducing *Executor* and *Proposer* roles in the system. It allowed us to find different bugs on the blockchain level of the system. For this part, Derya helped me a lot with the validation part insights. And the whole team helped me to come up with test scenarios.

1.5 Feature 5: Proposer

I designed and implemented the *Proposer*, with detailed information available in the Project report. The main challenge of designing the *Proposer* was to think about how *Proposer* can best ensure that his *Fuzzle* gets eventually solved by the *Executors*. Together with Milana, I designed a proposing pool used by both *Proposer* and *Executor*. We also worked on the economic model of the *Proposer* and *Executor*.

1.6 Feature 6: Benchmarks

I worked on designing and performing the benchmarks, as well as interpreting the results. I created the graphs and wrote the section in the report related to this part. The biggest challenge with this task was small bugs that occurred non-deterministically from parts of the code base I was not very familiar with, there were many small bugs in different parts of our implementation that froze or stopped the benchmarks. I eventually overcame these errors and was able to run the benchmarks with the help of my teammates: Qifan helped me to debug the Fuzzle tracker, Milana helped me to debug the *Executor* part, Chau helped me to debug the *Validator* part, Derya helped me to debug the *Validation* part, and Vladimir helped me with the Fuzzer storage part.

2 Other Contributions

In the initial phases of the project, I researched the existing solutions for distributed fuzzing, what execution engines existing implementations are and how to make them decentralised.

I then spent time reading about the Bitcoin implementation to get inspiration for various modules such as Difficulty module, Proof of Fuzzing work etc.

At the end of the project, I estimated the probability of the fuzzing using our system with the help of Vladimir.

I was making sure things were moving along as they should by organising the meetings, and helping when people were stuck with their work. I spent some time writing the intermediate and the final reports.

I also reviewed everybody's code in the PR, providing feedback and sometimes finding bugs in the code. I also spent a lot of time debugging the code so we had a system as correct as possible.