Smart Contracts and Scalability

By Raj Thimmiah

What are smart contracts?

- Goals:
  - Autonomous
    - Able to take input and make outputs to other smart contracts/users
  - Reliable
    - No downtime
  - Trustable
    - Don’t need to worry about the contract cheating you or doing something other than what the code specifies

How can we host or run these smart contracts?

- If we run them on a cloud server like AWS:
  - Need to trust AWS
  - Need to trust me (the person running code on the server)

How can we run smart contracts and trust the result?

- We *all* run the smart contract:
  - By running the smart contract we validate the result as well
  - If we all run same validation, any deviant showing a different result can be proved wrong

Issues

- Performance issues
- Not scalable in terms of
  - Computation
  - Storage
- Shows and stores all data publicly which wastes space

Computational Scaling

- How can we do work without parallelizing it?
- Answer:
  - Perform task on only a subset of the network

Truebit

- Task giver: has problem that they want solved
- Solver: computes this problem
- Verifier/Challenger: checks if the solver computes correctly

There are forced errors that the solver is compelled economically to add, the challenger/verifier is rewarded for finding these and the solver is penalized if they don’t include these

Split task into steps, if a challenger disagrees on some section then the blockchain is used to check which person is the cheater

Checking for cheater is expensive so there is a penalty for the challenger if there is no issue and penalty on the solver if there is an issue

Enigma/MPC/Plasma

Where does the efficiency come from?

- Don’t need to have all nodes validate everything
- Can limit computation (based on the trust/reliability of the base blockchain) to a subset