Blockchain as a Back-End:
How to utilize current cryptocurrencies in your applications

Destry Saul (Unchained Capital)
4:25pm–5:05pm Wednesday, June 13, 2018

Web services and APIs
Location: 210 B/F
First product: USD loans using BTC and ETH as collateral
  ○ Banking site
  ○ Blockchain integration
  ○ Secure collateral storage
  ○ Custom Ethereum ‘vault’ smart contract
Blockchains Are More Than Code. Start Reading.
1. Bitcoin Transactions can:
   a. Transfer BTC
   b. Record SMALL data
2. You can lock the BTC behind a very small piece of code
   a. Pay per byte ($3-4 per kB today, has been much higher)
   b. No state store - No access to other transactions.
3. Extremely Secure
   a. Extremely Slow
   b. Most expensive blockchain
1. Ethereum Transactions can:
   a. Send ETH (no lock)
   b. Deploy contracts
   c. Call contract functions
2. Smart Contracts are:
   a. Small chunks of code
   b. Mutable state store
   c. Built-in Authentication
   d. Can call/create other contracts
   e. Can store and send ETH

```solidity
pragma solidity ^0.4.22;

contract Example {
    uint256 public myNumber;
    address public owner;

    constructor(uint256 initialValue) public {
        owner = msg.sender;
        myNumber = initialValue;
    }

    function double() public {
        require(msg.sender == owner);
        myNumber = myNumber * 2;
    }
}
```
1. Only consistent over long times - (Bitcoin ~1hr, Ethereum ~1-10min)
2. Expensive - every write costs money
3. Writes can fail
4. No Queries (more like key-value store)
5. All data and activity is publicly viewable

Would use use a database that is only consistent after 10mins, charges you a dollar per write, sometimes rejects you, makes you record the id of each record you want to track, has no group by, and lets the world see your data?
Pros:
- If you run a node, you have all the data
- ALL the data, through all time, yes. All of it.
- This includes (compiled) smart contract code.
- You can go back in time and re-execute a transaction/contract operation-by-operation

Cons:
- Designed for fast transaction validation and current-state lookup
- No query language, you have to manually track all addresses and contracts of interest
- Privacy concerns
https://etherscan.io/ (ethereum)
- Transaction details + metadata
- Contract source code (if submitted)
- Compiled contract bytecode (always)
Changing the state of the data on-chain requires a signed transaction.
Authentication is handled by the public-private crypto signature.
There are fees that scale relative to the transaction **datasize** and **computational complexity**.
Transactions are written to the chain highest-fee first.
Block times can be long.
  - Bitcoin 10min average
  - Ethereum 12sec average
You have to wait until you see the transaction accepted and written to the chain, you can’t assume it worked.
Blocktimes Can Vary. For ETH it’s 1sec - 1min

https://github.com/rolandkofler/blocktime
Bad on-chain:
● Large Data
● Complex computation
● Private data

Good on-chain:
● Data that should be known to all
● Logic that determines how that data can be manipulated
● Functions you want to expose globally
If security or uptime is important, run your own node.
The web3 library is an interface to ethereum’s standard RPC spec.
If you want to develop ethereum smart contracts, use Truffle.
Bitcoin has a json-rpc interface, different libraries are maintained by different companies (e.g. BitPay’s bitcore library)

For Ethereum, you can build an application that assumes the client has access to blockchain data, either through their wallet, or through a light-node.

For Bitcoin, this is less true. Either use a service, or host your own node.
1. Content, assets, etc. are stored on a decentralized file system
   ○ E.g. ipfs
2. Authentication handled by client-side nodes and wallets
   ○ E.g. metamask, parity, geth+mist
3. State and logic stored on blockchain.
4. Client pays for state-change
   ● Example: Cryptokitties

*Note: Remember that contracts and transactions are immutable…
You can put the blockchain behind your server

Client

Server

Blockchain Smart Contracts
Remember the Client can interact directly with Contracts
Serverless: You can Provide a Reference Front End

- Client
- **IPFS**
- Decentralized File System
- Blockchain Smart Contracts
1. Blockchains are still a very new technology
2. Many projects are just porting existing apps ‘onto the blockchain’.
   a. Title Registries
   b. Logistics
   c. Voting Systems
   d. Digital Asset Ownership
3. Blockchains have unique properties
   a. Strong Security
   b. Public & Trustless
   c. True Serverless Apps