Navigating Ethereum Smart Contracts

Created by Vitalik Buterin

In 2013 (Toronto), launched in 2014 with Dr. Gavin Wood

~10K nodes in network

Bitcoin ~ 7K

~4 B$ market cap

Bitcoin ~18 B$

Main differences with the Bitcoin Protocol

Block Time & Size

- Block time ~**12.7**s (Bitcoin 10 minutes)
- Size: ~**2.5Kb** (bitcoin 1Mb)

Different Hashing Algorithm

- **Ethash** (memory intensive)
- Less centralization (bitcoin miners use ASICS)

Turing Completeness

- Quasi-turing complete state machine.
- Customizable Code
- “Global computer”

Block header

- Txn root
- logsBloom
- State root
- Receipts Root
- OmmersHash

Block

- Txn list
- Ommers list

What is a **Smart Contract**?

A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs. -**Szabo '94**

Key concepts in ethereum

Ether

The currency used in Ethereum. The smallest unit being the Wei.

10^{18} Wei = 1 Eth ~ 50 USD

Gas/Gas Price

- Solves the **Halting Problem**.
- Pay per computation
- 1 computation step = 1 gas, 1 byte/txn data = 5 gas

EVM

- 256 bit Virtual Machine that allows anyone to execute arbitrary EVM Byte Code.
- Stack based architecture, stack depth of **1024**.

Languages

- Main Language is **Solidity** (like JS)
- - Serpent (Python like)
- LLL (Lua like)

Miner reward

- **5 eth/block**
- Gas
- Uncle reward

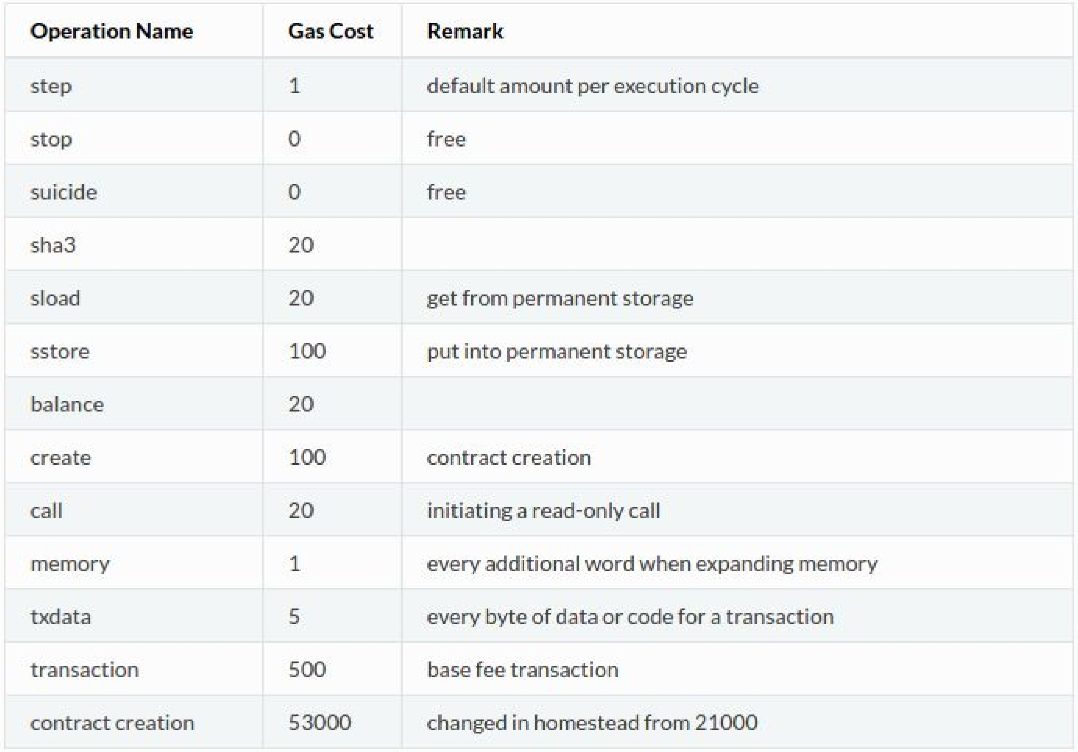
Issuance : 18 million/year

Storage

Each contract/account has its very own database. In the form of a key value pair (LevelDB)

Essentially endless amount of memory, if you are willing to pay for it!

Cost of Operations



Operation Name	Gas Cost	Remark
step	1	default amount per execution cycle
stop	0	free
suicide	0	free
sha3	20	
sload	20	get from permanent storage
sstore	100	put into permanent storage
balance	20	
create	100	contract creation
call	20	initiating a read-only call
memory	1	every additional word when expanding memory
txdata	5	every byte of data or code for a transaction
transaction	500	base fee transaction
contract creation	53000	changed in homestead from 21000

2 types of Accounts

Externally Owned Accounts (EOA)

- Just like in bitcoin, this is your “wallet” account (ctrl by private key)
- Contains :
  - Nonce
  - Balance
  - Storage (root)

Contracts (1st class citizens)

- Similar to EOA but they contain code (codeHash).
- Controlled by its code.
- **Whenever** a contract receives a transaction, its code is run
- Can also contain/control Ether!

When sending a transaction from EOA to EOA, the cost is 21000 gas, which at the current gas price is equivalent to 21000 * 22000000000 $4.62*10^{14}$ ~ 0.000462 Eth ~**0.02 USD**. But if you transfer Eth to a Contract, the contract will fire up its code and the gas cost will be higher. Keep this in mind when sending a transaction.

Transactions vs. Messages

Transactions

- Signed data package from an EOA to another account (can be a contract)
- Contains:
  - ➢ Recipient
  - ➢ Signature
  - ➢ Value
  - ➢ Data field (optional)
  - ➢ StartGas
  - ➢ Gas price

Message

- From contract to contract
- Virtual objects (can be seen as function calls)
- Contains:
  - ➢ Sender
  - ➢ Recipient
  - ➢ Data (function call, args)
  - ➢ StartGas
- Implemented when using the CALL or DELEGATECALL Opcode

Solidity

- Looks a lot like **JavaScript**
- Strongly Typed
- OOP ==> COP
- Contract is essentially an Object
- Events → Dapp's window into the blockchain
- `this` in a contract
- Important global variables (msg, tx, block, <address>)

  - `block.timestamp` (`uint`): current block timestamp (alias is `now`)
  - `msg.data` (`bytes`): complete calldata
  - `msg.gas` (`uint`): remaining gas
  - `msg.sender` (`address`): sender of the message (current call)
  - `msg.value` (`uint`): number of wei sent with the message
  - `tx.gasprice` (`uint`): gas price of the transaction
  - `tx.origin` (`address`): sender of the transaction (full call chain)

Not the same!!

Hello World in Solidity

```
contract HelloWorld {
  event Print(string out); // event that our Dapp will read
  string hello = “World”; // saved to storage
  uint sum = 5 + 2; // saved to storage too
  function() { // fallback function
     Print("Hello, World!"); // event saved in receipt logs
  }
}
```

Primitives

- ↦ Bool
- ↦ int/uint/real
- ↦ address
- ↦ bytesXX
- ↦ String
- ↦ Arrays (fixed/dynamic)
- ↦ Structs
- ↦ Enums
- ↦ Mappings (hashes)
- ↦ var (type inference)

Interesting Extras

Swarm

- P2P file sharing, similar to BitTorrent
- Incentivised with micropayments of ETH.
- Redundant storing
- Aims to be DDOS-resistant
- Aims to have zero downtime
- Similar to IPFS

Files are split into chunks, distributed and stored with participating volunteers. These nodes that store and serve the chunks are compensated with ETH from those storing and retrieving the data.

- ⏩ This is *file storage* without relying on a central server.

Whisper

- Encrypted messaging protocol
- Nodes/Dapps can send messages directly to each other in a secure way
- Allows to hide the sender and receiver from third party.
- Core objective:
  - hide location of sender and receiver
  - In transit, make it difficult if not impossible to establish one, the other or both.
- ⏩ This is *communications* without relying on a central server.

Code

Find the example at **https://github.com/Dev43/blockchain-bootcamp**