Ethereum

Main difference with Bitcoin

# Ethereum v 1.0

**Main differences:**

* Programs can be stored on the blockchain
* Apparition of **smart contract**: piece of code that run on the blockchain and are guaranteed to produce the same result for everyone run them
* Creation of dApps (DeFi, Gaming) without any downtime, fraud, control or interference from a third party
* Non-Fungible Tokens (**NFTs**)
* 12/14 seconds to mine a block
* Current mining reward: 2ETH + gas fees + **uncle block**

**Uncle block:** is similar to orphan block during bitcoin blockchain forks. It is valid block that isn’t included in the blockchain due to 2 miners finding the solution almost instantaneously. In the Ethereum they are 2 types of rewards for miners:

* Uncles Reward is awarded to the miner that **creates an uncle**. The value of reward depends on how old the uncle block before it is included in the blockchain: max 1.75 ETH. The older the block is the lower the reward.
* Uncle Inclusion Reward is awarded to the miner that **includes the uncle block in the blockchain**: 0.0625 ETH.

## Ether

* Ether (ETH) is the cryptocurrency used for many things on the Ethereum network.
* Ethereum allows developers to **create decentralized applications** (dapps), which all **share a pool of computing power**.
* When users want to make a transaction, they must pay ether to have their transaction recognized on the blockchain.
* Ether burn occurs in every transaction on Ethereum depending on network demand.
* **Wei** is the smallest possible amount of ether.
* **Gwei** is the human readable form.
* When the recipient address is a smart contract, this transferred ether may be used to pay for gas when the smart contract executes its code.

## Dapps

* Application built on a decentralized network that combines a smart contract and a frontend user interface.
* Has its backend code running on a decentralized peer-to-peer network, the EVM.
* A dapp can have frontend code and user interfaces written in any language to make calls to its backend. Furthermore, its frontend can get hosted on decentralized storage such as IPFS.
* Zero downtime: the network as a whole will always be able to serve clients looking to interact with the contract.
* Dapps can be harder to maintain because the code and data published to the blockchain are harder to modify.
* Performance issues.
* Network congestion: one dapp may use too many computational resources. Currently, the network can only process about 10-15 transactions per second. If transactions are being sent in faster than this, the pool of unconfirmed transactions can quickly balloon.

## Account

### General

* An account is not a wallet. An account is the keypair.
* Accounts and account balances are stored in a big table in the EVM.
* 2 types of account: externally-owned and contract
* 4 fields:
  + Nonce: counter that indicates the number of transactions sent.
  + Balance: the number of wei owned by this address.
  + codeHash, immutable:
    - Hash of empty string for externally owned account.
    - Hash of the code stored in the state database for contract.
  + storageRoot: hash of the root node of Merkle Patricia Tree that encode the storage content of the account.

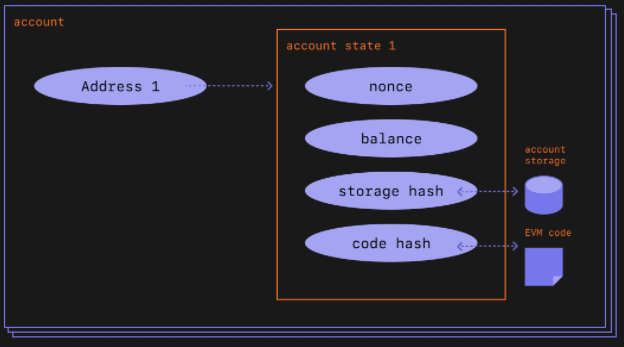


Figure a: Architecture of an account, [source](https://ethereum.org/en/developers/docs/accounts/).

### Externally-owned

* Creating an account cost nothing
* Can initiate transactions
* Transactions between externally-owned accounts can only be ETH/token transfers.
* made up of a cryptographic pair of keys: public and private.
* The contract address is given when a contract is deployed formed by the creator’s address and the nonce of that address.

### Contract

* Creating a contract has a cost because you're using network storage
* Can only send transactions in response to receiving a transaction
* Transactions from an external account to a contract account can trigger code which can execute many different actions, such as transferring tokens or even creating a new contract

## Transaction

* The formal term for a request for code execution on the EVM.
* **Different** types of transactions, e.g.:
  + Send X ether from my account to Alice's account.
  + Publish some smart contract code into EVM memory.
  + Execute the code of the smart contract at address X in the EVM, with arguments Y.
* **Contract creation**: results in the creation of a new contract account **containing compiled smart contract** bytecode.
* **Message calls:** **execute** contract’s bytecode.
* A **transaction** requires **gas** fee use for **reward miner** and for **ETH burn**.
* Ethereum originally had one format for transactions.
* Has evolved to support multiple types of transactions to allow for new features

## Blocks

* The volume of transactions is very high, so transactions are "committed" in batches, or blocks.
* Blocks generally contain **dozens to hundreds** of transactions.
* In addition to the Bitcoin standard blocks, Ethereum block contains stateRoot: the hash of the entire state of the system: account balances, contract storage, contract code and account nonces are inside.
* Blocks are bounded in size. Target of 15 million gas. But the size of blocks will increase or decrease in accordance with network demands, up until the block limit of 30 million gas (2x target block size). The total amount of gas expended by all transactions in the block must be less than the block gas limit to insure performance of full nodes.

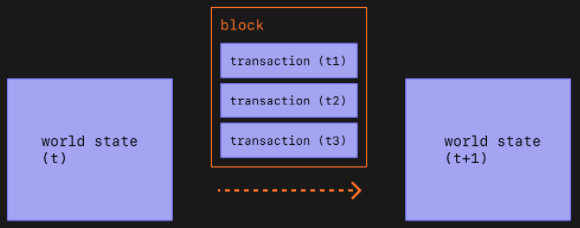


Figure b: block content, [source](https://ethereum.org/en/developers/docs/blocks/).

## EVM

* Stands for Ethereum Virtual Machine

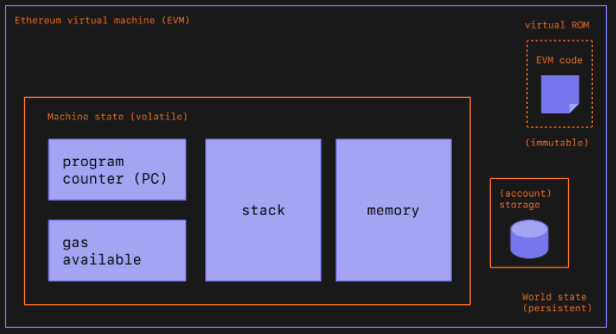


Figure 3: Diagram adapted from EVM, [source](https://ethereum.org/en/developers/docs/evm/)

* **Instead of a distributed ledger**, Ethereum is **a distributed state machine**.
* EVM behaves as a mathematical function would: **Given an input**, it produces a **deterministic output**.
* Y (S, T) = S'.
  + S: old valid state
  + T: new set of valid transactions
  + Y: the Ethereum state transition function
  + S’: new valid state
* EVM is the global virtual computer whose state every participant on the Ethereum network stores and agrees on.
* Any participant can request the execution of arbitrary code on the EVM.
* Code execution changes the state of the EVM.
* **State:** enormous data structure called a modified **Merkle Patricia Tree**. **Keeps all accounts** linked by hashes and reducible to a single root hash stored on the blockchain.
* The EVM executes as a stack machine.
* Compiled smart contract bytecode executes as a number of EVM opcodes, which perform standard stack operations like XOR, AND, ADD, SUB, etc.
* The EVM also implements a number of blockchain-specific stack operations, such as ADDRESS, BALANCE, BLOCKHASH, etc.

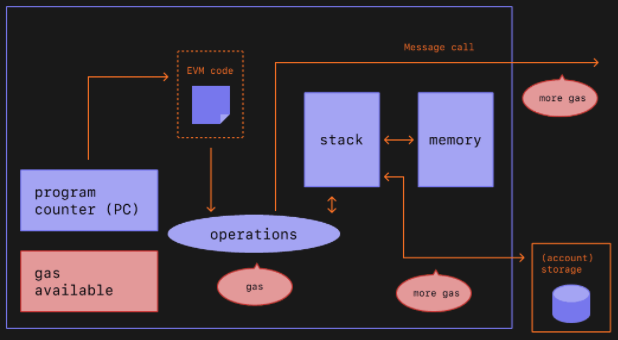


Figure 4: Diagrams adapted from Ethereum EVM, [source](https://ethereum.org/en/developers/docs/evm/).

## Gas

* Gas fees help keep the Ethereum network secure.
* Need fee for every computation executed on the network.
* Calculating gas fees:
  + (Base fee + tips) \* gas unit
  + (Current market price of fuel/liter + tips) + amount of fuel need for computation
* E.g. transfer ETH need 21.000 gas unit, if base fee was 100 gwei and tipped 10 gwei, the gas fee would be (100 + 10) \* 21,000 = 2,310,000 gwei ETH.
* Base fee is burnt. Here 2,100,000 gwei ETH burnt.
* Tips fee go to miners. Here 210,000 gwei ETH tipped.
* To be eligible for inclusion in a block the offered price per gas must at least equal the base fee.
* The base fee is calculated by a formula that compares the size of the previous block (the amount of gas used for all the transactions) with the target size (15 million gas unit).
* The base fee will increase by a maximum of 12.5% per block if the target block size is exceeded.

## Nodes

* Full node:
  + Stores full blockchain data.
  + Participates in block validation, verifies all blocks and states.
  + All states can be derived from a full node.
  + Serves the network and provides data on request
* Light node
  + Stores the header chain and requests everything else.
  + Can verify the validity of the data against the state roots in the block headers.
  + Useful for low capacity devices, such as embedded devices or mobile phones, which can't afford to store gigabytes of blockchain data
* Archive node
  + Stores everything kept in the full node and builds an archive of historical states.
  + Needed if you want to query something like an account balance at block #4,000,000.
  + Test your own transactions set without mining them
  + Handy for services like block explorers, wallet vendors, and chain analytics.
  + Units of terabytes

## Networks

* Ethereum is a protocol, this means there can be multiple independent "networks" conforming to this protocol that do not interact with each other.
* Networks are different Ethereum environments you can access for development, testing, or production use cases.
* Public networks:
  + Mainnet: he primary public Ethereum production blockchain, where actual-value transactions occur on the distributed ledger.
  + Testnets: test both protocol upgrades as well as potential smart contracts in a production-like environment before deployment to Mainnet. Most testnets use a proof-of-authority consensus mechanism. It's hard to incentivize mining on a proof-of-work testnet which can leave it vulnerable. Examples of testnets: Görli, Kovan, Rinkeby, Ropsten.
* Private networks:
  + Development networks: Similar to how you create a local server on your computer for web development, you can create a local blockchain instance to test your dapp. This allows for much faster iteration than a public testnet.
  + Consortium networks: controlled by a pre-defined set of nodes that are trusted.
  + For example, a private network of known academic institutions that each govern a single node, and blocks are validated by a threshold of signatories within the network.

## Smart contracts

* A **reusable** snippet of code (a **program**) which a developer publishes into EVM memory.
* Anyone can request that the smart contract code be **executed** by making a **transaction request**.
* Because developers can write **arbitrary executable applications** into the EVM (games, marketplaces, financial instruments, etc.) by publishing smart contracts, these are often also called **dapps**, or **Decentralized Apps**.
* Once smart contracts are deployed on the network you can't change them.
* Smart contracts alone cannot get information about "real-world" events: a workaround is to use oracles.
* They have a balance and they can send transactions over the network.
* They're not controlled by a user, instead they are deployed to the network and run as programmed.
* Perhaps the best metaphor for a smart contract is a vending machine: money + snack selection = snack dispensed.

## Storage

* Ethereum itself can be used as a decentralized storage system, and it is when it comes to code storage in all the smart contracts.
* However, it isn’t designed to store a large amount of data: every node on the network needs to be able to store all of the data. The nodes need to continue to run.
* The cost of deploying this much data would be expensive due to gas fees.
* We need a different chain or methodology to store large amounts of data in a decentralized way, like IPFS.
* IPFS is a distributed, decentralized system for storing and accessing files, websites, applications, and data.

## Standards

* The Ethereum community has adopted many standards that help keep projects (such as Ethereum clients and wallets) interoperable across implementations, and ensure smart contracts and dapps remain composable.
* Standards are introduced as Ethereum Improvement Proposals (EIPs), which are discussed by community members
* Certain EIPs relate to application-level standards (e.g. a standard smart-contract format), which are introduced as Ethereum Requests for Comment (ERC).
* Most notable standards are:
  + ERC-20 - A standard interface for fungible (interchangeable) tokens, like voting tokens, staking tokens or virtual currencies.
  + ERC-721 - A standard interface for non-fungible tokens, like a deed for artwork or a song.
  + ERC-777 - A token standard improving over ERC-20.
  + ERC-1155 - A token standard which can contain both fungible and non-fungible assets.

## MEV

* Miner Extractable value: Searchers (independent network participants) run complex algorithms on blockchain data to detect profitable MEV opportunities and have bots to automatically submit those profitable transactions to the network.
* MEV examples:
  + DEX arbitrage: if two DEXes are offering a token at two different prices, someone can buy the token on the lower-priced DEX and sell it on the higher-priced DEX in a single, atomic transaction, a flash loan. [Link](https://etherscan.io/tx/0x5e1657ef0e9be9bc72efefe59a2528d0d730d478cfc9e6cdd09af9f997bb3ef4).
  + Sandwich trading: if someone when a make a large trade on a token pair it will fluctuate the price. So, searcher execute an optimal buy order immediately before the large trade and sell just after.
  + NFT MEV: if there's a popular NFT drop, searchers can program a transaction such that they are the first in line to buy the NFT, or they can buy the entire set of NFTs in a single transaction

## Oracles

* Bridge between the blockchain and the real world.
* This could be anything from price information to weather reports or price president.
* Architecture:
  + Emit a log with your smart contract event
  + An off-chain service has subscribed (usually using something like the JSON-RPC eth\_subscribe command) to these specific logs.
  + The off-chain service proceeds to do some tasks (like http request) as defined by the log.
  + The off-chain service responds with the data requested in a secondary transaction to the smart contract

## Scaling

* Increase transaction speed (faster finality), and transaction throughput (high transactions per second), without sacrificing decentralization or security.

### On-chain scaling

* Sharding: creating new chains, known as “shards”, reduce network congestion and increase transactions per second.

### Off-chain scaling

* Layer 2 scaling:
  + Most layer 2 solutions are centered around a server or cluster of servers.
  + Transactions are submitted to these layer 2 nodes instead of being submitted directly to layer 1 then batches them into groups before anchoring them to layer 1, after which they are secured by layer 1 and cannot be altered.
* Rollups
  + **ZK-rollups** (Zero-knowledge rollups): bundle (or "roll-up") hundreds of transfers off-chain and generate a cryptographic proof, known as a SNARK (succinct non-interactive argument of knowledge). This is known as a validity proof and is posted on layer 1.
  + **Optimistic rollups:** assumes transactions are valid by default and only runs computation, via a fraud proof, in the event of a challenge.
* State channels: utilize multisig contracts to enable participants to transact quickly and freely off-chain, then settle finality with Mainnet.
* Sidechains: is an independent EVM-compatible blockchain which runs in parallel to Mainnet.
* Plasma: is a separate blockchain that is anchored to the main Ethereum chain, and uses fraud proofs.

# Ethereum v 2.0

## Beacon chain - alive

* Upgrade from PoW to PoS
* Shipped separately from the main net
* Create committees of validators randomized at the end of every epoch (32 blocks)

## The merge - 2022

* Main net under PoS
* Bring the ability to run smart contract on main net with PoS
* History and current state of Ethereum

## Shard chains - 2023

* Improve Ethereum’s scalability and capacity
* Spread the network’s load across 64 new chains
* Run a node by keeping requirement low
* Increase the TPS with rollups transactions