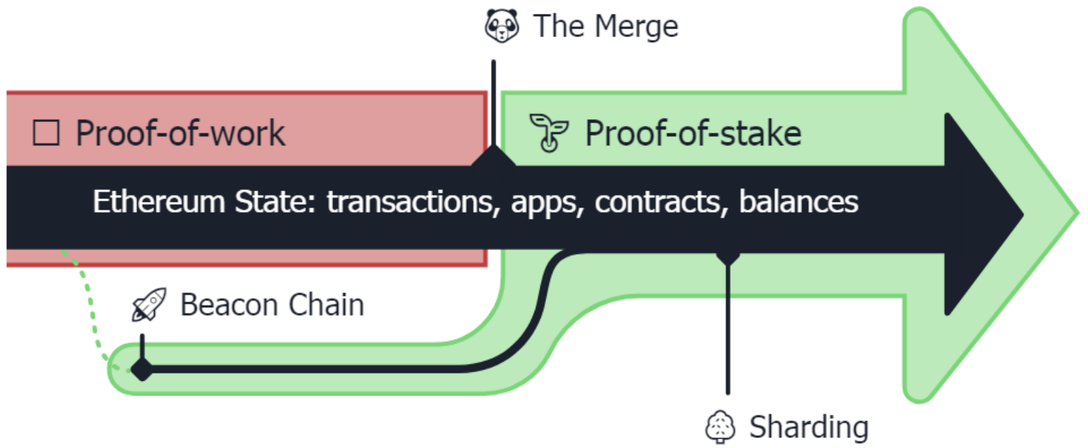
ETHEREUM:
UNA PIATTAFORMA PER SMART CONTRACTS

*Ethereum è una **piattaforma tecnologica decentralizzata** con una propria **blockchain** che, oltre a gestire lo scambio di **moneta digitale**, può eseguire programmi chiamati **"smart contracts"** all'interno della propria **virtual machine***

ROADMAP DI ETHEREUM

- **Novembre 2013:** Vitalik Buterin propone la specifica del **protocollo** di Ethereum
- **2014:**
  - Vitalik Buterin rilascia la **descrizione** del protocollo di Ethereum sotto forma di **White Paper**
  - Gavin Wood **formalizza** il protocollo di Ethereum sotto forma di **Yellow Paper**
- **Luglio 2015:** viene lanciata la **public network** (mainnet) di Ethereum basata sull'algoritmo di consenso **proof-of-work (PoW)**
- **Dicembre 2020:** viene lanciata, in parallelo alla Ethereum PoW, la **Beacon Chain,** la versione di Ethereum basata sull'algoritmo di consenso **proof-of-stake (PoS)**
- **15 Settembre 2022 → The Merge:** la **mainnet** di Ethereum **ingloba** il livello di consenso **PoS** della Beacon Chain ed abbandona definitivamente PoW



IN COSA ETHEREUM-POW È SIMILE A BITCOIN

i. Si basa su una **rete P2P di** partecipanti - **peer nodes** - ed un **algoritmo di consenso** per la creazione **(mining)** di nuovi **blocchi,** contenenti transazioni validate, da accodare alla blockchain (**append-only**)

ii. Ha una propria **moneta digitale** e si avvale di **firme digitali** per la generazione delle **transazioni**

iii. Il mining di un blocco dà diritto ad una ricompensa **(reward)** al **miner** che lo ha creato, così come la **validazione** e l'**esecuzione** delle **transazioni** incluse nel blocco

iv. Ingloba una **macchina a stati replicata** sui vari peer nodes, le cui **transizioni di stato** avvengono in risposta all'**esecuzione di transazioni**

v. Ha una **memoria permanente** in grado di **registrare le transazioni eseguite** ed associarle ai blocchi di appartenenza

IN COSA ETHEREUM È DIVERSA DA BITCOIN

i. Adotta un meccanismo per la **validazione delle transazioni** di tipo "**account-based**", associato al concetto di **balance**, in contrasto con il meccanismo "UTXO" di Bitcoin

ii. E' una **blockchain programmabile general-purpose**. Il linguaggio Ethereum è Turing-complete

iii. Il codice eseguito è costituito da **smart contracts** memorizzati all'interno della blockchain in apposite strutture dati

iv. Lo **stato** consiste non soltanto nel **balance** dei vari account, ma anche dello **storage** associato agli smart contracts

v. La **memoria permanente** registra, in un'apposita struttura dati associata ai blocchi, anche gli **effetti delle transazioni eseguite**, compresi gli **eventi emessi dagli smart contracts** eseguiti

vi. Esiste, inoltre, un'altra **struttura di memoria**, dove vengono registrati i **valori ephemeral delle variabili di stato** degli smart contracts

ETHEREUM PROOF-OF-STAKE

i. Introduce un meccanismo di consenso **più sicuro, maggiormente decentralizzato** e **meno energivoro** (-99,95%) e più adatto per l'implementazione di **nuove soluzioni di scalabilità,** rispetto alla precedente versione basata su proof-of-work

ii. In PoS, i validatori (**stakers** o **validators)** investono del capitale sotto forma di ETH (**staking**) su Ethereum; il validatore è responsabile del **controllo della validità** dei nuovi blocchi propagati sulla rete e, occasionalmente, della **creazione** e della **propagazione** di nuovi blocchi

iii. Il **capitale investito** funge da garanzia e **può essere distrutto** se il validatore si comporta in modo disonesto

iv. Mentre in proof-of-work **la tempistica per la creazione dei blocchi** è determinata dalla difficoltà di mining, in proof-of-stake **è predeterminata**. Il tempo per la creazione di nuovi blocchi nel proof-of-stake di Ethereum è suddiviso in **slot (12 secondi)**

v. A causa del basso fabbisogno energetico è richiesta una **minore emissione di ETH** per incentivare la partecipazione

MONETA DIGITALE: ETHER

- La **moneta digitale** di Ethereum è chiamata **Ether**, indicata con ETH o con il simbolo Ξ (dalla lettera greca Xi)
  - Un Ether: 1 Ether, 1 ETH, 1 Ξ
  - Un Ether è diviso in unità più piccole, fino alla più piccola detta **wei**
    1 Ether = 10^{18} wei
  - 1 Ether = 1133,37 USD (21 novembre 2022)

ACCOUNT: TIPI, INDIRIZZI E BALANCE

- Esistono **due tipi** di account:
  - gli account **EOA** (Externally Owned Account), ad ognuno dei quali è associata una **chiave privata, *possono generare transazioni (esterne)***
  - i **contract account,** ai quali non è associata alcuna chiave privata, ***non possono generare transazioni (esterne)****;* questo tipo di account è controllato dalla logica dello smart contract associato, cioè il programma che è registrato sulla blockchain al momento della creazione dell'account
- Ambedue i tipi di account hanno **indirizzi** e **balance** associati e possono **mandare e ricevere Ether**
- Le **transizioni di stato** sono sempre **attivate dalle transazioni generate dagli account,** che implicano trasferimenti diretti di valori monetari (**value**) e/o da informazioni fra account (**input data**)
- L'autorizzazione all'esecuzione di una transazione (monetaria e non) dipende dal **balance collegato all'account,** il cui valore è memorizzato nello storage della blockchain

TRANSAZIONI E MESSAGGI

- Le transazioni (esterne) sono **messaggi firmati** digitalmente, **originati solo dagli EOA**, trasmessi sulla rete Ethereum per la loro validazione ed **inclusi nei blocchi** di Ethereum per la loro registrazione nella blockchain
- Le transazioni sono gli unici elementi che possono innescare una **transizione di stato** o causare l'**esecuzione di uno smart contract**
- Una transazione (esterna) **da un EOA ad un altro EOA** può contenere solo dei trasferimenti di **Ether**
- Una transazione (esterna) **da un EOA ad un contract account** viene chiamata messaggio **(message)** e contiene, oltre ad un eventuale trasferimento di Ether, una **chiamata parametrica a delle funzioni** contenute nello smart contract associato che, a sua volta, può invocare altre chiamate verso altri account (transazioni interne), sia di tipo contract che EOA
- Il problema del **double-spending** nelle transazioni viene risolto, in Ethereum, dalla presenza, **in ogni account**, di un valore scalare (**nonce**) equivalente al numero delle transazioni valide generate

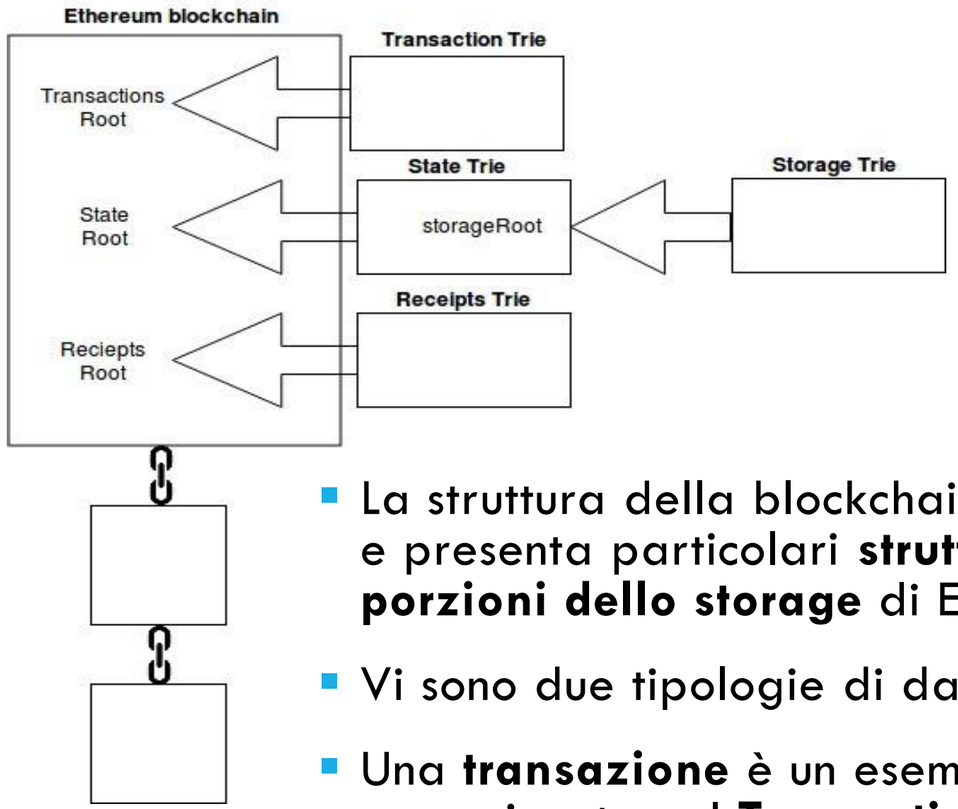
ETHEREUM VIRTUAL MACHINE

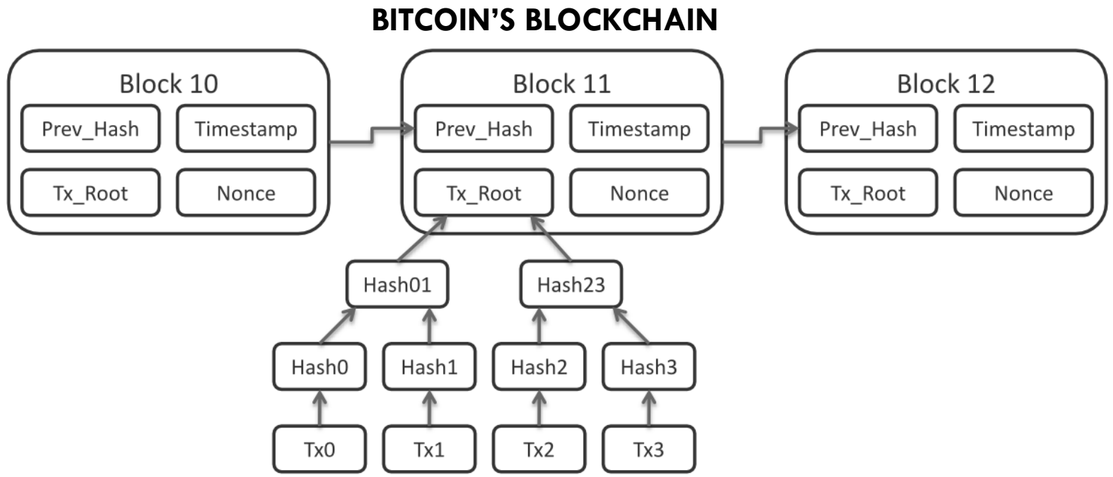
- Ethereum è una **blockchain programmabile general-purpose** ed il compito della EVM è di **aggiornare lo stato globale** della blockchain in seguito all'esecuzione di transazioni, che possono causare solo trasferimento di valuta e/o invocare uno smart contract
- Ad alto livello, la EVM può essere considerata un **computer globale decentralizzato** di tipo "singleton" che contiene milioni di oggetti eseguibili, ciascuno dei quali con il proprio storage; nella pratica, ogni nodo di Ethereum esegue una **copia locale della EVM**
- Il **linguaggio** della EVM è **Turing-complete**
- La EVM non ha capacità autonome di **scheduling**, in quanto l'ordine di esecuzione delle transazioni viene organizzato, al suo esterno, dai miners (validators). In tal senso, la EVM può essere classificata come una macchina **single-threaded**
- Tutte le operazioni sono eseguite su un'area dati virtuale denominata "stack", formata da 1024 words di 256 bits. La EVM non è infatti una "register machine" bensì una "**stack machine**"

SICUREZZA: GAS FEE

- Per proteggere la propria blockchain da attacchi e abusi, Ethereum prevede il pagamento di una tassa (**gas fee**) per l'esecuzione di ogni transazione, attraverso una **moneta virtuale** chiamata **gas,** le cui quantità vengono espresse in **gas unit**
- La **gas fee** (in gas unit) è costituita da una **parte fissa** per tutte le transazioni e, in caso di esecuzione di **smart contract,** di una **parte variabile,** proporzionale alle istruzioni eseguite e alla memoria usata
- Le transazioni hanno un campo **gas limit** che specifica l'**ammontare massimo della gas fee** che l'account sorgente intende pagare per la loro esecuzione
- Se il gas usato in una transazione **eccede questo limite** la computazione viene bloccata e **lo stato della EVM riportato a quello precedente** l'inizio dell'esecuzione della transazione. Questa caratteristica di sicurezza rende di fatto il linguaggio della EVM **quasi-Turing complete**
- La **gas fee** viene **incassata dal miner/validator** che ha inserito la transazione in un nuovo blocco. La ***tassa*** in Ethereum è ***pagata in Ether***, in accordo ad un valore di cambio variabile chiamato **gas price**
- Esempio per una **transazione base** che trasferisce Ether da EOA a EOA (gas fee = 21000 gas unit):
  - il **gas price** è, alla data del 21/11/2022, di 12 gwei
  - il **gas limit** di una transazione base è di 21000 unità
  - con questi dati: **21000** * **12 [gwei]** = 2,52 * 10^5 * 10^9 [wei] = 2,52 * 10^{14} [wei] * 10^{-18} [ETH/wei] = = **0,000252 ETH** * 1133,37 [USD/ETH]= **0,287 [USD]**

ETHEREUM'S BLOCKCHAIN



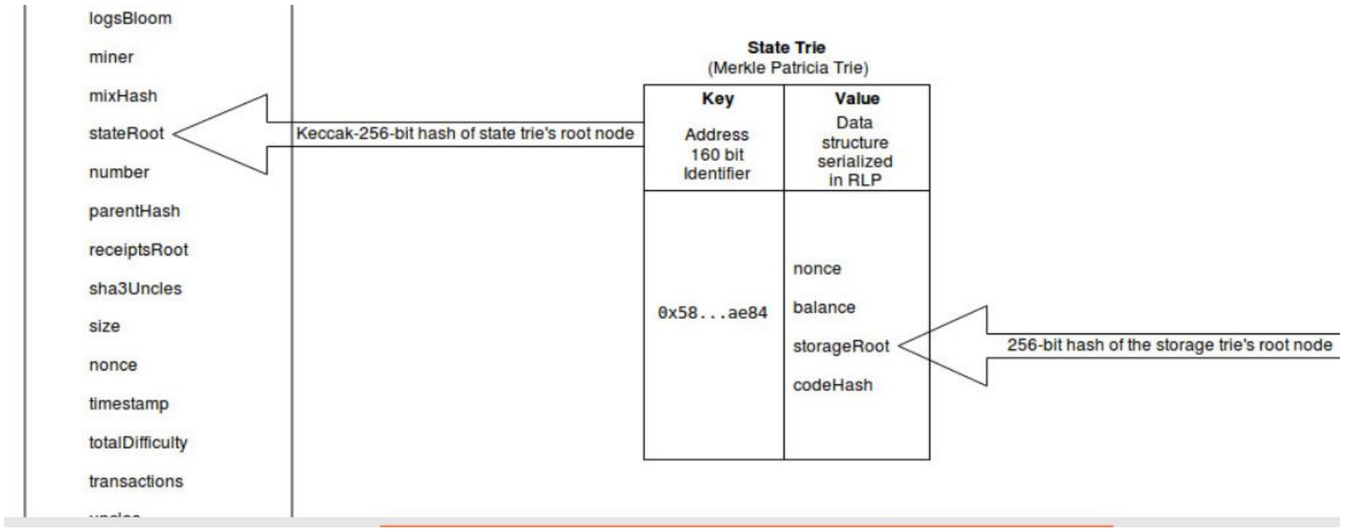


- La struttura della blockchain di Ethereum è ben più complessa rispetto a quella di Bitcoin e presenta particolari **strutture dati,** denominate **Trie,** ognuna delle quali gestisce **porzioni dello storage** di Ethereum
- Vi sono due tipologie di dati in Ethereum: **permanent data** e **ephemeral data**
- Una **transazione** è un esempio di **permanent data**: una volta confermata viene memorizzata nel **Transaction Trie** e non verrà mai alterata. Anche il risultato della esecuzione della transazione viene memorizzato permanentemente, ma in una struttura diversa denominata **Receipt Trie**
- Un esempio di **ephemeral data** è il **balance** di un account, che viene memorizzato nello **State Trie** e viene modificato per ogni transazione che lo riguarda. Anche lo **Storage Trie** contiene I **dati modificabili** relativi agli **smart contracts**

STATO GLOBALE DELLA EVM: STATE TRIE

- A differenza del Transaction Trie e del Receipt Trie (uno di ciascuno per blocco), esiste **soltanto uno State Trie** in Ethereum, che viene **costantemente aggiornato**. Mantiene lo **stato globale della EVM replicata** nei vari peer nodes. Contiene una coppia **key-value** per ogni account in rete.
- La "**key**", equivalente all'**indirizzo di un account**, è un identificatore unico a 20 byte.
- Il "**value**" è creato codificando i seguenti campi relativi all'account:
  - nonce
  - balance
  - storageRoot (contiene l'hash del root node dello Storage Trie relative all'account) (*)
  - codeHash (contiene l'hash del codice dello smart contract associate) (*)

(*) solo in caso di account relativi a smart contract



SMART CONTRACTS

- **Programmi immutabili** che vengono eseguiti in maniera determistica all'interno della EVM
- Gli smart contracts sono tipicamente **scritti in liguaggi d'alto livello,** come Solidity, ma per poter essere **eseguiti** devono essere compilati **nel bytecode di basso livello** che va in esecuzione nella EVM
- Una volta compilati vengono **depositati** sulla piattaforma Ethereum, utilizzando una **transazione speciale** di tipo "**contract creation**"
- Ogni smart contract è **identificato da un indirizzo Ethereum,** che ne identifica il relativo account, che è derivato da alcune informazioni dell'account di chi crea e deposita il contratto
- Il **creatore** dello smart contract **non ottiene** alcun privilegio, a livello di protocollo Ethereum, sul **possesso del contratto depositato**. Non riceve la private key del contract account, che di fatto non esiste. Si può affermare che i contract account posseggono se stessi
- Il codice di uno smart contract **non può essere modificato** benché possa essere rimosso dalla blockchain assieme al suo storage
- Un smart contract può chiamare un altro smart contract e così via, ma il **primo smart contract** della catena di esecuzione sarà sempre **chiamato da una transazione proveniente da un EOA**

ESEMPIO DI SMART CONTRACT

```
contract valueChecker
{
    address payable public owner; //state variable
    uint price = 10; //state variable

    event valueEvent(bool returnValue); //contract event: logs the boolean result of the check

    constructor() {
       owner = msg.sender; //the contract creator becomes the contract owner
    }

    function Matcher (uint8 x) public returns (bool){ //checker function
        if(x>=price){
            emit valueEvent(true);
            return true;
        }
        else {
            emit valueEvent(false);
            return false;
        }
    }
    function destroy() external { //contract destroy function
        require(msg.sender == owner); //only the owner can destroy the contract
        selfdestruct(owner); // system call to remove the contract from the blockchain
    }
}
```

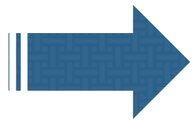
EVENTI

- Quando una transazione viene eseguita essa produce una ricevuta (**transaction receipt** memorizzata nel Receipts Trie del blocco corrispondente) contenente **log entries** che forniscono informazioni sulle azioni occorse durante l'esecuzione
- Gli eventi (**events**) sono oggetti di **Solidity** che vengono utilizzati per **costruire i log entries** memorizzati nella ricevuta della transazione eseguita
- I **logs** possono essere considerati come una **forma alternativa di storage** a più basso costo
- Gli **eventi** sono utili in quanto permettono alle **applicazioni esterne alla blockchain** di:
  - osservarli, catturarli e **riportarli nell'interfaccia**
  - **operare delle scelte** a seguito della loro valorizzazione
- Gli eventi prendono degli **argomenti** che possono essere indicizzati (**indexed**) all'interno di una **hash table** di tipo **key-value,** dove la key è l'indice assegnato all'argomento e value è il suo contenuto. L'indicizzazione è un meccanismo utile per **ricerche** o **filtraggi** di dati da parte di un'applicazione esterna

APPLICATION BINARY INTERFACE

- Una Application Binary Interface (**ABI**) è un'**interfaccia tra due moduli di programma**, spesso tra il sistema operativo ed un programma utente. Una ABI **definisce l'accesso alle strutture dati e alle funzioni del codice macchina** degli smart contracts
- In **Ethereum**, la **ABI** è un **array JSON** che espone il **prototipo delle funzioni** contenute negli smart contracts e la **struttura degli eventi** generati dagli smart contracts a beneficio dei linguaggi di frontend, che la utilizzano mediante l'uso di librerie specializzate

```
contract valueChecker
{
    uint price = 10;
    event valueEvent(bool returnValue);
    function Matcher (uint8 x) public returns (bool)
    {
         if(x>=price){
             emit valueEvent(true);
             return true; }
         else {
             emit valueEvent(false);
             return false; }
     }
}
```



```
[ {"anonymous": false,
    "inputs": [ {
        "indexed": false,
        "name": "returnValue",
        "type": "bool" } ],
    "name": "valueEvent",
    "type": "event" } ,
  {"constant": false,
    "inputs": [ {
        "name": "x",
        "type": "uint8" } ],
    "name": "Matcher",
    "outputs": [ {
        "name": "",
        "type": "bool" } ],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"} ]
```

DECENTRALISED APPLICATIONS (DAPP)

- Ethereum è una piattaforma per programmare Applicazioni Decentralizzate (**Dapp***), ovvero al **minimo uno smart contract** depositato in rete ed **una interfaccia utente** come front-end
- Affinché il front-end di una Dapp possa interagire con uno smart contract sono necessari:
  - l'**ABI** e l'**indirizzo** dello smart contract
  - un software per la gestione degli account EOA (**wallet**)
  - un **punto di accesso** ai nodi di Ethereum
  - una serie di **API per accedere** alle funzionalità messe a disposizione dai nodi della blockchain

Attraverso le API, una **Dapp può inoltre avere accesso ai log** degli eventi generati dagli smart contracts e depositati permanentemente nei Receipt Trie dei vari blocchi

(*) anziché DApp si può trovare ÐApp, dove il simbolo Ð rappresenta il carattere insulare chiamato 'eth'