Chapter 26: Blockchain Databases

Database System Concepts, 7th Ed.

Outline

- Overview
- Blockchain Properties
- Achieving Blockchain Properties via Cryptographic Hash Functions
- Consensus
- Data Management in a Blockchain
- Smart Contracts
- Performance Enhancement
- Emerging Applications

History

- Blockchain technology's development was driven initially by **cryptocurrencies**.
  - But cryptocurrency is just one application of blockchain
- Cryptocurrencies:
  - Purely online
  - Maintained by a decentralized distributed ledger
- **Bitcoin**
  - One of the first successful cryptocurrencies, and the best known.
  - Published under the pseudonym Satoshi Nakomoto
  - Aims to be an anonymous, fully distributed and decentralized currency
  - There are other blockchains that are more appropriate for blockchain-based enterprise data

Types of Blockchain

- **Public**
  - Anyone can download the needed software and create a blockchain node
  - No trust assumed among participating nodes
- **Permissioned**
  - Permission to run a blockchain node is granted by a permissioning authority
  - Some degree of relaxation of the assumptions of trustlessness and autonomy
- The type of blockchain influences the choice of algorithm used by which nodes agree on the next block to be added to the blockchain

Blockchain Properties and Structure

- Linked list of blocks
  - Each block contains a pointer to the previous block plus a hash of the previous block
    - Except, of course, for the first block, called the **genesis block**
- **Tamper resistance**
  - The inclusion of the hash of the previous block makes tampering difficult
    - Changing the contents of a block means changes all newer blocks as well
  - Specific mathematical requirements for the hash function (see later)
  - Replication prevents replacement of the entire blockchain without gaining majority control

Blockchain Properties and Structure

- Node types
  - **Full** node – maintains copy of blockchain and participates in the consensus process
  - **Light** node – submits updates to the blockchain but does not participate in the consensus process
- Consensus algorithms to choose node to add next block:
  - **Proof of work** – first node to solve a certain hard math problem
  - **Proof of stake** – node selected based on amount of currency owned or held in reserve
  - **Byzantine consensus** – message-based protocol to reach consensus on next block
  - Other approaches – proof of activity, proof of burn, proof of capacity, proof of elapsed time

Blockchain Properties and Structure

- **Forks**
  - A fork occurs if a block is added to a block that is not the most recent one
  - Accidental if consequence of the consensus algorithm
    - One fork survives; others are eventually **orphaned**
  - Malicious if due to an attempt to damage the blockchain
  - Consensus-based if agreed to by majority of users
- Consensus-based fork types
  - **Hard** – old version of blockchain software does not recognize new blocks as valid
  - **Soft** – old version of blockchain software recognizes new blocks as valid

Blockchain Properties and Structure

- Digital signature
  - Public-key encryption is used to allow users to sign their transactions.
  - Ensures that users cannot deny submitting the transaction, a property called **irrefutability**.
- Anonymity
  - Users can remain anonymous unless there is a way to tie the user's ID (public key) to a real-world entity
- Summary of blockchain properties:
  - **Decentralization** – majority consensus with no central authority.
  - **Tamper resistance** – infeasibility of changing the contents of blocks on the blockchain.
  - **Irrefutability** – user cannot deny having submitted a transaction.
  - **Anonymity** – IDs not directly tied to any real-world entity

Cryptographic Hash Functions

- Let *h* denote a cryptographic hash function. Then *h* must satisfy the following properties:
  - **Collision resistant** – It is infeasible to find two distinct values *x* and *y* such that h(x) = h(y)
  - **Irreversible** – Given h(x), it is infeasible to find x.
- By ***infeasible***, we mean that there is strong mathematical evidence, if not an actual proof, that there is no approach to obtaining an answer that is better than guessing from the set of all possibilities.

Blockchain Transactions

- Exact transaction model is specific to each blockchain.
- Bitcoin
  - No account balances stored directly.
  - A transaction specifies:
    - Input transactions (whose output are spent by this transaction)
    - A set of outputs, each specifying the recipient and the amount
    - A digital signature from the user submitting the transaction
  - Additionally a Bitcoin transaction may:
    - Store a small amount of data on the blockchain
    - Specify a slightly more complex transaction using the Bitcoin scripting language
- Ethereum
  - Maintains account balances, which are modified by transactions
  - Has a more sophisticated, Turing-complete scripting language

Consensus

- All nodes must agree on additions to the blockchain
- In a decentralized system like a blockchain system, there is no central coordinator (unlike the case for 2PC and 3PC)
- Categorization of consensus algorithms:
  - **Proof of Work**
    - Node needs to solve a cryptographic puzzle in order to add a block (more on next slide)
  - **Proof of Stake**
    - Node is chosen to add next block based on amount of currency held, with probability proportionate to stake
  - **Byzantine Consensus**
    - Node is chosen to add next block based on a message-passing-based consensus algorithm (more later)

Proof of Work

- To add a block B, a node needs to find a **nonce**, *n*, such that the value of the hash function *h* applied to the concatenation of *n* and *B* (n || B) is less that some specified value.
- The function *h* must have the **puzzle-friendliness** property: Given *k* and an *n*-bit value *y*, it is infeasible to find *x* such that h(x || k) = y in time significantly less than 2^n.
- Forks
  - If more than one node solves the puzzle around the same time, two blocks could be added after the most recent block
  - The result is a fork
  - Since nodes attempt to add to the most recent block of the longest chain, eventually blocks on shorter forks are orphaned (slide 7)

Byzantine Consensus

- **Byzantine failure**: A failed node can behave in an arbitrarily bad manner, including taking the exactly correct set of steps to sabotage the system
  - Contrast with 2PC, 3PC, Paxos, and Raft (Chapter 23), where the only assumed type of failure is a crash. This is called the **fail-stop** model of failure.
- Achieving consensus with Byzantine failure that at most (n-1)/3 nodes fail, where *n* is the total number of nodes.
- Traditional Byzantine consensus algorithms assume that *n* does not change.
  - In a blockchain system, however, nodes can join and leave.
  - Modern blockchain Byzantine consensus algorithms are robust to the number of nodes changing

Sybil Attacks

- A **Sybil attack** is an attempt to overwhelm the consensus algorithm by adding a large number of nodes.
- Protection against Sybil attack:
  - Proof of work: Hard for an attacker to control a majority of the computing power in the network, thus making it hard to dominate success in solving the cryptographic puzzle.
  - Proof of stake: Costly to acquire a majority of all outstanding currency.
  - Byzantine consensus: Vulnerable to attack unless there is a permissioning mechanism for new nodes:
    - Trusted permission-granting agent.
    - A decentralized trust-based feature in the protocol itself.

Data Management in a Blockchain

- Validating a transaction requires data look-up in the blockchain:
  - Bitcoin: Look up input transactions to ensure that their output has not already been spend ("**double-spend**").
  - Ethereum: Look up account balances.
- Blockchains use the **Merkle-tree** data structure (Chapter 23):
  - Allows a node to store just the root-hash of the Merkle tree for verification purposes, rather than the entire blockchain.
  - But blockchain immutability means the tree structure can't be updated in place.
- **Merkle-Patricia-tree** structure:
  - Patricia-tree structure allows key-based search.
  - Updates performed by creating a new root that points to unchanged parts of the data structure.

Smart Contracts

- Transaction execution is specified by code
  - Bitcoin: a relatively simple stack-based scripting language with instructions designed for funds-transfer, including the **multisig** instruction: *m* of *n* users must approve to enable escrow transactions.
    - Infinite loops not possible due to the limited power of the language.
  - Ethereum: a scripting Turing-complete language:
    - Based on the **Ethereum virtual machine** (EVM)
    - **Solidity**: A high-level language compiled to EVM code
    - Greater expressive power but risk of infinite loops
      - Need mechanism for terminating such loops despite undecidability of the halting problem
      - A cost-per-instruction framework serves both to avoid infinite loops and to incentivize miners (next slide)

Concept of "Gas" in Ethereum

- Each Ethereum instruction has a fixed cost, denominated in **gas**.
- Each smart contract sets its own gas price in Ether defining the amount paid to the miner for one unit of gas.
- How to set price wisely:
  - Gas price too low: miners disincented to include transaction.
  - Gas price too high: user overpaid.
- Limits
  - Transaction gas limit: upper bound on transaction gas usage, set by user.
  - Block gas limit: upper bound on gas usage in a block, set by the system.
- Infinite loops impossible since the transaction gas limit is reached eventually, at which point the transaction fails and terminates, (though the miner keeps the payment received).

Smart Contracts

- External input:
  - Messages from other contracts
  - Input pertaining to the outside world from trusted sources called **oracles**.
- Autonomy:
  - An Ethereum smart contract may run indefinitely by receiving Ether from external sources to fund its continued operation. Such contracts are called **distributed autonomous organizations** (DAOs).
- A smart contract may be used to create a separate currency or token on top of the Ethereum blockchain.
  - The **ERC-20** standard is the most widely used
- Smart contracts may be used in the implementation **of cross-chain transactions**, allowing transactions between separate blockchain systems

Performance Enhancement

- The consensus mechanism is an important factor in blockchain performance, as discussed earlier.
- Other ways to enhance performance include
  - **Sharding**: parallelizing the mining of new blocks
  - **Off-chain transaction processing**: Creation of a separate channel for users who process transactions among themselves frequently.
    - Channel funded with funds from the underlying blockchain.
    - Routine transactions avoid mining overhead.
    - Users can terminate the agreement to process transactions off-chain at any point, with current channel funds balances refunded on the underlying blockchain.
  - Database-style blockchain data structures.

Emerging Applications

- Academic transcript distribution
- Accounting and audit
- Asset management
- E-Government
- Foreign-currency exchange
- Health care
- Insurance claims
- Internet of Things
- Loyalty programs
- Supply chain
- Ticket sales and re-sales
- Trade finance
- and many more

End of Chapter 26