IEEE ICBC'24 1570979826

Web3DB: Towards Decentralized Relational DBMS with Blockchain-Based AccessControl

***Abstract*—In the wake of Web 3.0, the need for decentralized data management solutions is becoming increasingly apparent. The current global infrastructure predominantly relies on centralized databases, posing significant concerns regarding data ownership, security, and single points of failure. Despite the emergence of NoSQL as a successor to relational databases, most databases worldwide still operate on structured data, and many users and developers are more familiar with SQL-like queries. The research in decentralized database management systems (DBMS) remains in its nascent stages, particularly in the context of structured data and SQL-based operations. This paper situates itself at this intersection, aiming to address the gap in the current research on decentralized databases capable of handling structured data and complex queries.**

This paper presents Web3DB, a pioneering decentralized relational DBMS that uniquely integrates blockchain technology for access control. The system employs smart contracts and decentralized key management to implement refined access control. Smart contracts are utilized to verify user permissions, providing secure and precise control over data access down to the row level in database tables. The architecture involves a seamless integration of front-end user interactions via MetaMask for authentication and a back-end supported by interconnected database engines. Each engine functions as a node within an IPFS network, facilitating distributed data storage and retrieval.

Our findings indicate that Web3DB successfully merges the reliability of a decentralized architecture with the functionality of SQL query processing, offering a novel and effective solution in the DBMS domain. This system not only improves data security and user control but also aligns with the decentralized, user-centric vision of Web 3.0. By bridging a significant gap in the market, Web3DB marks a substantial progression in decentralized data management, paving the way for future innovations in the field.

I. Introduction

Centralized database management system (DBMS) architectures dominate the current landscape, leading to several challenges. The concentration of data control in a single entity raises concerns about data privacy and security, and it results in single-point-of-failure issues, leading to significant data loss or downtime. The primary motivation for this paper is to forge a path toward a more decentralized and secure DBMS architecture that aligns with the principles of Web 3.0 [1], which we call Web3DB. Unlike previous web iterations, Web 3.0 emphasizes decentralized data networks, reducing reliance on centralized repositories and emphasizing individual data ownership. This is often achieved through blockchain technology, enhancing security, privacy, and control over personal data. The need for user ownership of personal data and eliminating single points of failure in database management are critical driving forces behind this research. Furthermore, the existing gap in the literature for a decentralized DBMS capable of processing traditional relational databases and SQL-like queries presents a unique opportunity for innovation. Moreover, there is no existing decentralized relational DBMS that supports multi-tenancy [2].

There has been a shift towards decentralized databases in the domain of Web 3.0, but most aim to support NoSQL-based DBMS [3] [4] [5]. There is one decentralized DBMS supporting SQLite [6]. Unfortunately, none supports relational DBMS with `JOIN` operations in a decentralized manner, which is essential due to the role of data integration and fusion in the data industries nowadays. Considering that traditional relational DBMS with structured data still dominates many industries [7], it is important to design a decentralized relational DBMS architecture that is compatible with legacy systems.

We navigate through several intricate challenges. The first is to design a decentralized DBMS which can deal with the complexity of querying data that is stored on IPFS [8]. IPFS stores data in a Content Addressable aRchive (CAR) format, unsuitable for SQL query execution, primarily due to its un/semi-structured nature. Secondly, enforcing true individual data ownership in DBMS poses a significant challenge. Traditional (and even distributed) DBMS models typically centralize data control where DBMS administrators decide who can access data, thereby limiting actual user ownership. Lastly, implementing fine-grained access control in a decentralized DBMS environment is particularly daunting, which also makes multi-tenancy incredibly challenging. This challenge stems from the decentralized nature of data across a wide network without a central control point. This complicates the establishment of trust and consensus for fine-grained access control policies among varied and possibly anonymous nodes, and policy enforcement is nontrivial in such a decentralized setting.

In our paper, we propose a groundbreaking architecture for Web3DB, a decentralized relational DBMS designed for Web 3.0. This modular architecture integrates MetaMask [9], a crypto wallet, for user authentication, decentralized key management, and blockchain-based access control, enhancing security and user data sovereignty. As described in section *IV*, the system architecture includes an end-user layer interfacing with users, an API layer facilitating communication, a database engine layer for query processing, and a data storage layer that uses IPFS for decentralized storage. The system flow, as described in section *V*, demonstrates the integration of these components, highlighting the seamless data management process. Additionally, in section *VI*, we describe our decentral-

ized access control list, managed via smart contracts, which provides granular asset control, underpinning the system's innovative approach to decentralized data management in line with Web 3.0 principles. We have implemented a revolutionary system according to the architecture to blend it with advanced blockchain technologies. This design enhances flexibility and scalability, segmenting functionalities into layers for ease of updates and maintenance. A key element of this architecture is the integration of MetaMask for managing user authentication and key management, ensuring secure, user-centric interactions. Complementing this, our protocol uses decentralized key management and decentralized access control for relational DBMS, providing each user a unique, verifiable identity and enforcing access control without central authority reliance. While our prototype features some centralization in the API layer, the modular nature of Web3DB allows for incorporating decentralized APIs (dAPIs), pointing towards a future where the entire system is fully decentralized [10]. This approach, using a combination of IPFS, a decentralized database engine layer that uses a gossip network, and smart contracts on Hyperledger Fabric, positions Web3DB at the forefront of decentralized database management, adhering to the core tenets of Web 3.0.

Web3DB embodies a significant leap in the field of database management, addressing critical challenges in data security, privacy, and decentralization. This paper delves into the architectural design, system workflow, operational mechanics, and the potential impact of Web3DB, setting a new precedent for future developments in decentralized database systems.

Our contributions are multifold:

1) A Novel, Layered, and Modularized Design: We introduce a novel framework for DBMS, characterized by its layered and modularized design. This novel architecture facilitates the distribution of data across multiple nodes, leveraging the InterPlanetary File System (IPFS) for the storage of databases. A key aspect of this design is its support for SQL queries on relational data, marking a significant advancement in the realm of decentralized database systems.

2) Fine-grained Decentralized Access Control: We present a protocol that integrates Decentralized key management with blockchain-based smart contracts for access control. This innovative approach empowers users with unprecedented control over their data within database tables, enabling granular management down to the row level. Moreover, it offers the flexibility to dynamically delegate or revoke access, ensuring that users retain complete sovereignty over their data.

3) Multi-Tenancy Integration: This paper introduces the capability of multi-tenancy in Web3DB, a feature that allows multiple users or tenants to use the same DBMS infrastructure simultaneously while maintaining logical data isolation. Alongside multi-tenancy, our system introduces the concept of shared database ownership. This is a significant shift from traditional DBMS models, where a single entity typically owns a database.

4) Reproducible and Testable Outcomes: We provide an anonymized source code repository for reproducibility [11] [12], and we also provide an anonymized link to the fully functional prototype demo of Web3DB [13].

II. Related Work

Distributed DBMS. The evolution of Distributed Database Management Systems (D-DBMS) has been pivotal in the development of scalable and reliable data storage solutions. Early works, such as Özsu and Valduriez [14], provide foundational insights into the principles of distributed databases, highlighting their advantages in terms of scalability, reliability, and efficiency over centralized systems. Recent studies, like Corbellini et al. [15], delve into advancements in distributed systems, focusing on the challenges and solutions in data consistency and fault tolerance.

Several contemporary distributed DBMS architectures have been proposed and implemented. Apache Cassandra, discussed in Lakshman and Malik [16], offers a decentralized structure that excels in handling large volumes of data across many commodity servers. Google's Spanner, as explored in Corbett et al. [17], is another landmark system that combines the benefits of traditional relational databases with the scalability of NoSQL system.

Decentralized DBMS. With the advent of Web 3.0, the importance of decentralized databases has gained unprecedented attention. Tapscott and Tapscott [18], in their work on blockchain technology, illustrate how decentralized systems underpin the principles of Web 3.0, offering enhanced data security and user sovereignty. These systems, as argued by Swan [19], are crucial in the shift towards a more open, interconnected, and user-centric web.

The landscape of decentralized DBMS is still evolving. Technologies like BigchainDB, explored in McConaghy et al. [20], represent early attempts to integrate blockchain features into database systems, focusing on scalability and decentralized control. However, these solutions predominantly revolve around NoSQL paradigms, as discussed by Dinh et al. [21] in their evaluation of blockchain-based databases.

Blockchain-Based Access Control. Blockchain-based access control mechanisms have gained attention in recent years for their potential to address security and privacy concerns in the Internet of Things (IoT) [22]. These mechanisms leverage the decentralized and immutable nature of blockchain to provide integrity and security without relying on a central authority [23]. Several research studies have explored the use of blockchain for access control in IoT, highlighting its applicability and benefits [24] [25] [26]. These studies discuss various access control methods, including certificate-based, certificate-less, and blockchain-based approaches, and compare their pros and cons . They also emphasize the importance of verifying the authenticity of IoT devices before allowing them to communicate with the network and discuss the use of blockchain in establishing secure communications and session keys . Additionally, these studies provide a comprehensive description of IoT applications, comparative analysis of access control protocols, and highlight open research issues and challenges in blockchain-driven IoT networks .

The concept of decentralized access control in DBMS is a relatively new area of research. The integration of blockchain for access control in databases, as examined by Xu et al. [27], offers a glimpse into the potential of immutable, fine-grained access control mechanisms. These studies, however, are in their infancy and have yet to be fully realized in practical, scalable systems.

Despite these advancements, there is a significant research gap in decentralized DBMS, especially concerning systems that support SQL queries and possess robust access control mechanisms. Most existing decentralized databases are based on NoSQL technology [3] [4], as highlighted by previous studies, and lack fine-grained access control which is imperative in the era of Web 3.0. Furthermore, the familiarity and widespread use of SQL in the database community are not addressed in current decentralized systems.

Our work, Web3DB, seeks to fill this gap by introducing a decentralized DBMS that not only supports SQL-like queries but also integrates blockchain-based access control, offering a unique combination of user empowerment, data security, and familiarity with SQL. This represents a significant step forward in the field, aligning with the decentralized, user-centric vision of Web 3.0, and it sets a new precedent for future research and development in decentralized database systems.

III. Backgrounds

A. Distributed DBMS

A Distributed Database Management System (DDBMS) governs the storage, processing, and retrieval of data distributed across multiple physical locations. The inception of DDBMS was primarily driven by the necessity for organizations to access and manipulate data across various sites synchronized to operate cohesively. This system is an evolution of the traditional centralized database system, designed to address the growing challenges associated with large-scale data handling, such as bottlenecks in data processing and limitations in data storage. By enabling a distributed model, DDBMSs offer an architecture where data is not only stored on multiple servers but also where these servers, potentially located across different geographic regions, can autonomously process data requests [28].

The architecture of DDBMS is predominantly categorized into two types: homogeneous and heterogeneous systems [29]. In homogeneous DDBMS, all nodes run identical DBMS software, which simplifies the process of data management and system administration. Conversely, heterogeneous DDBMS deals with different DBMS software, necessitating a middleware to provide uniformity in data management. The core of DDBMS functionality rests on the principles of data independence and transactional integrity, ensuring that the system provides a unified view of the database to the user while maintaining the accuracy and consistency of transactions across the network. These transactions are typically managed through protocols such as the Two-phase Commit Protocol, ensuring that each transaction is either completely executed or entirely canceled, thus preserving the database's integrity.

Despite its significant advantages, such as scalability, improved data access and sharing across the network, and increased reliability, DDBMS also poses unique challenges. The complexities of maintaining data consistency and integrity become magnified in a distributed environment. Moreover, the system's security is paramount, as data is more susceptible to breaches when spread across different nodes. As part of a DDBMS's operational challenges, the trade-offs between consistency, availability, and partition tolerance—often conceptualized in the CAP Theorem—must be carefully balanced to meet system requirements [30].

B. IPFS

The InterPlanetary File System (IPFS) is a peer-to-peer hypermedia distribution protocol aimed at making the web faster, safer, and more open [8]. It was developed with the vision of creating a decentralized and distributed file system that connects all computing devices with the same system of files. In contrast to traditional HTTP/HTTPS protocols, which are based on a centralized, client-server relationship, IPFS operates on a distributed network where users not only consume content but also host and distribute it. This system forms a unique fingerprint called a content identifier (CID) for each file, and files are retrieved based on their content, not their location, which is a paradigm known as content-addressable storage.

IPFS's architecture enables various resilient features: it reduces the redundancy of data storage across the network, enhances performance by retrieving data from the nearest nodes, and preserves versions of historical content through immutable data structures. As a distributed network, IPFS can run on a variety of devices and scales naturally with the size of its network. The protocol's resistance to censorship comes from its architectural design, which avoids reliance on centralized points of control or failure. This makes it particularly useful for preserving information in a permanent and tamper-proof manner, which is crucial for applications such as historical archives or digital preservation.

While IPFS offers advantages in terms of data permanence and retrieval efficiency, it requires a paradigm shift in how users interact with and think about data. Overcoming the inertia of established web protocols and integrating with the broader ecosystem of internet technologies are ongoing efforts by the IPFS community. With its potential to revolutionize data sharing and storage, IPFS is an area of active research and development, with implications for a wide array of fields, from distributed applications (dApps) to decentralized finance (DeFi), and beyond.

C. Web 3.0

Web 3.0 represents the next evolutionary phase of the internet, often characterized by the shift from centralized to decentralized web applications and services [1]. It is built upon the core principles of decentralization, openness, and greater user utility. This new iteration of the web incorporates

technologies such as blockchain, machine learning, and decentralized data networks like the InterPlanetary File System (IPFS), aiming to create a more autonomous and intelligent internet. Web 3.0 enables direct peer-to-peer interactions without intermediaries, promoting a transparent and user-centric online environment. Integrating artificial intelligence and semantic web technologies allows machines to understand and interpret the meaning of data, facilitating more relevant and contextual user experiences.

In the Web 3.0 paradigm, users have greater control over their personal data, thanks to the use of blockchain technology, which provides secure and immutable data management. This contrasts with the previous iteration of the web—Web 2.0—where data is primarily controlled by centralized platforms, often resulting in privacy concerns and data monopolies. Moreover, Web 3.0's emphasis on semantic web technologies empowers a more connected and intelligent internet, where information is linked in a way that is easily processable by machines, enabling sophisticated services like automated reasoning and intelligent agents that can perform tasks on behalf of users.

IV. System Architecture

Web3DB's modularized design promotes decentralization at its core, distributing data across multiple nodes and using a decentralized storage solution. The proposed decentralized relational database management system (DBMS) is designed to operate on top of the Inter Planetary File System (IPFS).This reduces the risks associated with central points of failure and enhances data security. Traditional DBMS often struggle with scalability, especially when handling large volumes of data or a high number of simultaneous requests. The layered architecture of Web3DB, being modular, allows for better scalability. Each component, or module, can be scaled independently, enabling the system to handle increased loads efficiently. The modular nature of Web3DB's design makes it adaptable to future technological advancements. Components of the system, such as the API layer, can be updated or replaced without overhauling the entire system. This ensures longevity and relevance in the rapidly evolving field of database technology.

The system architecture comprises four layers (Figure 1): the End-User layer, the API layer, the Database Engine layer, and the Data Storage layer. Each layer plays a critical role in enabling decentralized query processing, data storage, and retrieval. Such a layered design provides modularization to Web3DB, and it also allows one to selectively decentralize any layer/component of Web3DB at their choice, thereby providing maximal flexibility.

A. End-User Layer

The end-user layer represents the interface through which users interact with the decentralized DBMS. Users issue queries through API requests on a web page, which are then forwarded to the API layer for processing. This layer provides a user-friendly experience, shielding the complexities of the underlying infrastructure while enabling seamless data access.

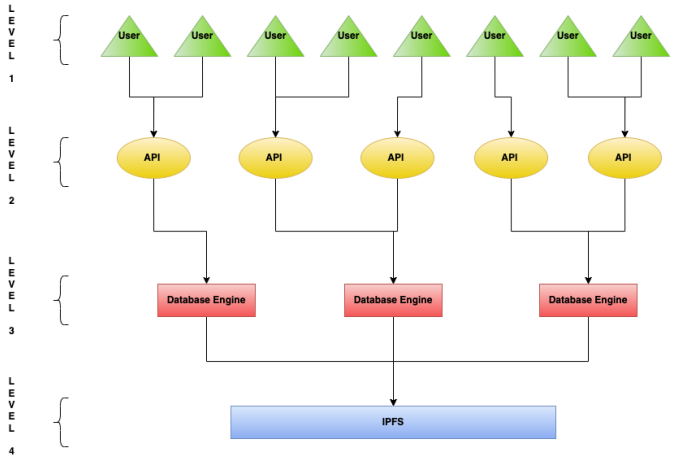

Fig. 1. Web3DB System Architecture

The end-user layer is also responsible for interacting with Metamask. MetaMask is a software cryptocurrency wallet used primarily for interacting with the Ethereum blockchain. It also provides key features like key vault, secure login, token wallet, and token exchange—everything needed to manage digital assets.

B. API Layer

The API layer serves as the intermediary between the end-users and the database engine layer. It hosts web pages for the end-users and handles API request routing. When a query is received from an end-user, the API routes the request to the appropriate database engine in the database engine layer based on user association. It acts as a communication bridge, relaying the queries to the database engines for processing, and it also receives the query results to be sent back to the end-users.

C. Database Engine Layer

The database engine layer in Web3DB is a crucial component of its architecture, playing a pivotal role in the system's ability to process queries and manage data. This layer is intricately designed to harness the benefits of decentralized technology while maintaining the functionality and efficiency expected of a robust database management system (DBMS).

The database engine layer utilizes Apache Spark for its in-memory data processing capabilities, enabling fast and efficient handling of large datasets. Apache Hive is integrated to provide support for SQL-like queries on relational data. This allows the system to execute complex queries typically associated with relational databases. Each database engine instance acts as a node within a private InterPlanetary File System (IPFS) network. This design choice leverages IPFS's decentralized and distributed nature for storing and retrieving database files, ensuring data availability and fault tolerance. Data is stored in a decentralized manner across the IPFS network, which the database engine accesses as required. When a query is received, the engine retrieves the necessary

data (if not already cached) from IPFS, utilizing the system's decentralized storage solution.

The database engine layer is enhanced with a gossip protocol to facilitate communication and synchronization among different instances (nodes). This protocol ensures that all nodes are consistently updated and aware of the network's state, enhancing the system's reliability and efficiency. The modular nature of Web3DB allows for scaling the database engine layer horizontally by adding more nodes, thereby distributing the workload and improving query response times. Load balancing is inherently managed within this layer, directing queries to nodes based on their current load and computational capacity. The layer handles the execution of SQL queries received from the API layer. It ensures that the queries comply with the access control rules set via the blockchain-based smart contracts, thus enforcing security and privacy. Post-query execution, any changes made to the data are updated back to the IPFS network, maintaining consistency. The layer manages these updates efficiently to ensure that the latest version of the data is always available across the network.

The database engine layer in Web3DB is thus a testament to the innovative blending of decentralized storage, advanced data processing technologies, and robust query execution mechanisms. It stands as a critical enabler for Web3DB's functionality, balancing the need for decentralized data management with the performance demands of contemporary DBMSs.

D. Data Storage Layer

The Data Storage layer serves as the decentralized and fault-tolerant storage layer for the DBMS. It utilizes the IPFS protocol, which enables distributed file storage and content addressing. The IPFS layer provides a robust and scalable storage solution, allowing the database engines to store and retrieve data in a decentralized manner. Each database engine interacts with the data storage layer to store tables and retrieve them as needed. The data storage layer ensures data availability, replication, and fault tolerance, enhancing the overall reliability of the system.

The system architecture facilitates seamless data flow and interaction between the layers. When an end-user submits a query through the web server layer, it is forwarded to the appropriate database engine in the database engine layer. If the table required for query execution is not already loaded, the database engine retrieves it from the IPFS. The query is then processed by the embedded PostgreSQL engine, and the results are sent back to the web server layer. The web server layer, in turn, relays the results to the end-user, completing the query processing cycle.

The proposed system architecture offers several advantages over existing decentralized DBMS solutions. By leveraging IPFS, it provides a decentralized and fault-tolerant storage layer, ensuring data availability even in the presence of network failures or node outages. The integration of PostgreSQL enables efficient and powerful SQL-based query processing, allowing for complex data manipulations and analysis. The parallel computation and load balancing capabilities of the database engine layer further enhance query processing efficiency.

V. Web3DB Implementation and System Flow

Web3DB embodies a sophisticated architecture that integrates several key components to facilitate secure and efficient data management. Here, we have described the workflow of the entire system, as shown in Figure 2.

End-User Layer: The user interface, developed in JavaScript, interacts with the Ethereum blockchain via Metamask. This interaction is crucial for managing user keys and credentials securely. When a user initiates a session, the front end retrieves the user's keys from Metamask and sends them to the API layer.

The user interface of Web3DB is developed using JavaScript, providing a responsive and interactive experience. The API layer, crucial for the communication between the front end and the database engines, is implemented in C#. C# was chosen for the API layer due to its robustness and performance in handling back-end operations. It offers a strong type system, comprehensive framework libraries, and seamless integration with database systems, making it ideal for managing the complex data flow between the front end and the database engines.

Integrating MetaMask for user key management leverages its widespread adoption and familiarity within the cryptocurrency community. It provides a secure and user-friendly interface for authentication, aligning with the decentralized ethos of Web3DB.

API Layer: The API, implemented in C#, acts as a bridge between the end-user and the backend. Upon receiving keys from the front end, the API communicates with the Access Control List (ACL) managed on the blockchain. Smart contracts within the ACL verify these keys, ensuring the user's authenticity and returning the most recent data hashes corresponding to the user's permissions. The API then compiles this information into a dictionary containing the names, hashes, and keys of the tables accessible to the user and sends it back to the front end.

When the user submits a SQL query, the end-user forwards it to the API. The API's first step is to validate the user's authorization to execute the query. This validation is performed by the smart contract on the blockchain, which checks the user's key against the permissions stored in the ACL. If the user is authorized, the API proceeds to send the query along with the relevant data hashes to the database engine.

Database Engine Layer: Within the database engine, Apache Spark maintains an open connection with a private IPFS network, with each database engine functioning as a node. Upon receiving a query, the engine retrieves the corresponding database using the hash provided by the API. The IPFS network returns the raw SQL file to the database engine, where Apache Spark hands it over to Apache Hive. Hive then recreates the entire database locally and executes the SQL query.

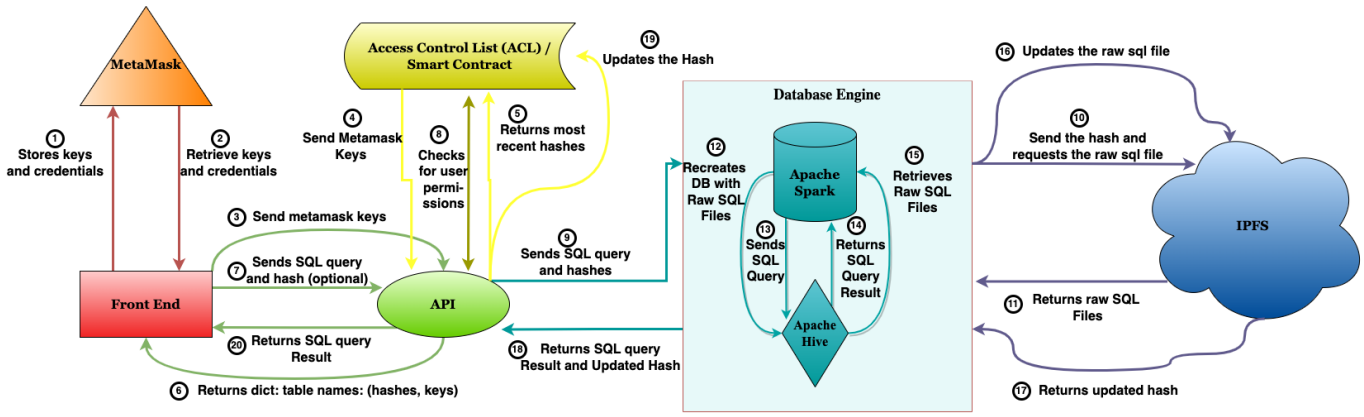

Fig. 2. Web3DB System Flow

A key feature of Web3DB's database engine is its connectivity through a gossip network. This network ensures that all Hive instances are connected, enabling them to communicate and synchronize. When a new database engine joins the network, it establishes connections with pre-determined, stable Hive nodes. These stable nodes provide the new engine with a list of all active Hive nodes in the network. The new engine then attempts to connect with these active nodes, and once successful, it is added to the list, expanding the network's reach and resilience.

After Hive executes the query, Spark retrieves the updated SQL file of the database and sends it back to IPFS, obtaining an updated hash. Spark then sends the query result and the new hash back to the API. The API updates the ACL with the new hash, ensuring that the blockchain reflects the latest state of the database. Finally, the API sends the query results back to the front end, completing the user's request.

The core of Web3DB's database engine is an amalgamation of Apache Spark and Apache Hive. The combination of Apache Spark and Apache Hive harnesses Spark's fast data processing capabilities with Hive's proficiency in managing large-scale data warehousing. Hive's support for ACID transactions underpins the system's data consistency, a vital attribute for reliable CRUD operations. Another crucial advantage of our system is that HIVE can use its own SQL-like DBMS as the DB engine and other traditional DBMSs as the DB engine. Our prototype is using Postgresql as the engine in the HIVE framework.

Data Storage Layer: The final layer of Web3DB is built on a public InterPlanetary File System (IPFS) network. Opting for a public IPFS network underlines the commitment to decentralization, ensuring distributed, resilient data storage and access across network nodes, fundamental to Web3DB's architecture.

This detailed workflow of Web3DB demonstrates a seamless integration of blockchain technology, a decentralized file system (IPFS), and a distributed database architecture, ensuring a robust, secure, and user-friendly system for managing and querying data in a decentralized environment.

VI. Decentralized Access Control List

In the Web3DB system, we implement an innovative Access Control List (ACL) on the blockchain, a critical component in managing permission rights for accessing specific assets, such as table rows. This ACL is managed through a smart contract designed for token management, encompassing access token creation, delegation, revocation, and verification.

When an asset owner, User A, wishes to grant access to another user B, he creates a token within the smart contract. This token creation process involves using a specific function that encrypts the token with User A's secret key, leveraging Ethereum's cryptographic functions for security. The smart contract updates an `accessList` mapping to reflect this new access right, indicating that User B is now authorized to access User A's asset. An event is emitted upon the creation of this token, serving as a notification for User B. The `accessList` is a two-dimensional mapping, where the first key represents the asset owner's address (User A), and the second key is the address of the user granted access (User B). The value stored in this mapping is a boolean, indicating whether User B is authorized to access User A's asset. Each mapping entry correlates to a unique token generated by User A, granting User B specific access rights.

When User A decides to grant access to their asset to User B, they utilize the `grantAccess` function. This function is responsible for updating the `accessList` mapping to reflect the new permission settings. Specifically, it sets the value of the mapping entry, identified by User A and User B's addresses, to true. This update signifies that User B now holds a valid token, enabling them to access User A's asset.

For access verification, User B initiates a function call to the smart contract when attempting to access the asset. This public verification function checks the `accessList` for the presence and validity of the token corresponding to User B's access rights. The function returns a boolean value, indicating the success or failure of this access verification.

In scenarios where User A needs to revoke User B's access, the smart contract facilitates this through a revocation function. This function, accessible only by the token owner (User A),

updates the smart contract's state, setting User B's token in the `accessList` to false, effectively revoking their access rights. Event logging is incorporated post-revocation to maintain a transparent record of these changes.

This decentralized access control list, facilitated through blockchain technology, offers a secure, transparent, and efficient means of managing permissions. It empowers users with granular control over their assets, ensuring that access rights are accurately represented and enforced on the blockchain. This mechanism is particularly advantageous in scenarios where trustless and verifiable access control is paramount.

Access control is a pivotal aspect of Web3DB, implemented using Hyperledger Fabric. Hyperledger Fabric is utilized for its advanced features in permissioned blockchain environments, offering modular architecture for implementing complex access control mechanisms crucial for fine-grained permissions in Web3DB.

In summary, the `accessList` mapping within our smart contract framework forms the backbone of our decentralized access control system in Web3DB, embodying the principles of security, efficiency, and user empowerment inherent in Web 3.0 technology.

VII. Prototype Demo and Experiments

A. Testable Prototype System

We have developed a testable prototype of Web3DB, embodying the proposed architecture and demonstrating its feasibility and effectiveness. This prototype serves as a tangible proof of concept for our novel decentralized relational Database Management System (DBMS) suited for the Web 3.0 ecosystem.

We commit to regular maintenance and updates to ensure the prototype remains functional, secure, and aligned with the latest technological advancements. This includes updating the codebase to address any emerging security vulnerabilities, improving system performance based on user feedback, and incorporating new features that align with the evolving landscape of decentralized databases and Web 3.0 technologies.

Looking ahead, we plan to enhance the prototype by fully decentralizing the API layer using decentralized APIs (dAPIs). This will bring the system closer to a fully decentralized model, eliminating any central points of failure and further aligning with the principles of Web 3.0. We have provided an anonymized link for our prototype system [13]. We have also provided anonymized GitHub repository links for the front-end and back-end codebases of our system [11] [12].

B. Experiments

This section aims to evaluate the performance of the Web3DB system. The system architecture incorporates a public InterPlanetary File System (IPFS) network for data storage, Apache Spark and Hive for query processing, and smart contracts on Hyperledger Fabric for access control. The database engine layer of Web3DB was hosted on an AWS EC2 instance, providing a scalable and reliable cloud computing environment.

TABLE I
TPC-H Benchmark test for execution time(Seconds)

TPC-H Query	1 Node(s)	2 Nodes(s)	3 Nodes(s)	4 Nodes(s)
Query 1	38.02	35.86	30.37	26.11
Query 2	32.16	30.21	25.69	22.11
Query 3	36.87	34.68	29.45	25.31
Query 4	34.64	32.49	27.70	23.93
Query 5	52.85	49.65	42.02	36.27
Query 6	35.57	33.62	28.54	24.52
Query 7	37.10	34.88	29.77	25.90
Query 8	36.89	34.57	29.33	25.67
Query 9	35.81	33.65	28.78	24.21
Query 10	35.83	33.38	27.47	24.13
Query 11	36.53	34.83	29.01	24.94
Query 12	37.70	35.15	30.04	26.14
Query 13	35.71	33.48	29.01	24.60
Query 14	36.90	34.47	29.19	24.77
Query 15	42.65	40.12	34.44	29.20
Query 16	30.81	28.65	22.80	20.86
Query 17	53.14	50.25	43.42	35.66
Query 18	49.08	45.70	39.38	34.67
Query 19	36.70	34.51	27.59	26.43
Query 20	37.05	34.85	29.72	25.60
Query 21	43.64	40.82	34.03	29.86
Query 22	33.82	31.81	26.98	23.03

The TPC-H benchmark, a standard for evaluating the performance of DBMS, was chosen for testing. We imported TPC-H data with a scale factor of 10, ensuring a standardized dataset for comparison. The tests were conducted across different node configurations - specifically, 1, 2, 3, and 4 parallel nodes - to evaluate the system's scalability and query processing efficiency. Each node represented an instance of the database engine within the network, and the tests aimed to observe the impact of parallel processing on query execution times.

The TPC-H queries, consisting of 22 standardized queries, were executed on each node configuration. These queries were designed to cover a wide range of DBMS functionalities, including join operations, aggregations, and nested queries. The execution times for each query were recorded in all node setups. This approach allowed us to measure the performance under varying parallelism and workload distribution levels. Throughout the testing phase, care was taken to maintain environmental consistency. This included controlling for variables such as network latency, server load, and background processes on the AWS EC2 instances. Such control was essential to ensure that the results were attributable solely to the system's performance and not external factors.

The experimental results are shown in `Table I`. The results indicated a clear trend: as the number of nodes increased, the query execution times decreased. This improvement can be attributed to the distributed nature of the system, where parallel processing and efficient workload distribution played a significant role. For instance, the system had no parallel processing capabilities with a single node, leading to longer execution times. However, as we added more nodes, the system could distribute the query processing load, significantly reducing the execution times.

The results demonstrates approximately a 6% improvement in execution times with two nodes, 20% with three nodes, and

up to 32% with four nodes. This scaling efficiency showcases the capability of Web3DB to handle increasing loads and complex queries effectively.

We can extrapolate that distributing the query among 2 nodes is not enough to fully utilize parallel processing capabilities, leading to longer execution times for complex queries. A significant improvement in execution times is observed when we have 3 parallel database engines, as queries can be efficiently parallelized. Further improvements in execution times are observed when we have 4 parallel nodes, but the rate of improvement decreases due to increased communication overhead and complexity in managing more nodes.

The experiment with the TPC-H benchmark on the Web3DB system successfully demonstrated the system's scalability and efficiency in processing complex SQL queries in a decentralized environment. The results validate the effectiveness of the system's architecture and its potential in managing large-scale data in line with the principles of Web 3.0.

VIII. Discussion and Future Work

A key assumption in our design is the stability of the database engines, which are presumed not to go offline unexpectedly. This stability is crucial for the reliability and consistency of the system. Additionally, in the context of the CAP theorem – which posits that a distributed system can only simultaneously satisfy two out of the three properties: Consistency (C), Availability (A), and Partition Tolerance (P) – our system aligns with the CA paradigm. This means that while we prioritize consistency and availability, the system may be less resilient to network partitions compared to CP or AP systems.

The overhead between the database engine layer and the IPFS network is pretty high. This leads to the increase of the overall execution time. The reason behind this overhead is that we can not directly execute queries on IPFS due to its usage of Content Addressable aRchive (CAR) format, which is an an open problem. As a result we are required to load the entire database everytime we execute the query. Resolving this problem is out of scope for this paper, although there is ongoing research regarding this problem.

Looking forward, there are several avenues for enhancing Web3DB. To bolster security, we plan to integrate more sophisticated encryption techniques. This would enhance data privacy and security, making the system robust against potential cyber threats. A fascinating prospect is to enable the selling of databases via smart contracts. This approach could transform data storage into an economic model where databases are traded, with transactions and ownership transfers securely managed and recorded on the blockchain. This functionality would also facilitate the reception of cryptocurrency payments, aligning with the decentralized ethos of Web 3.0. Given our system's current alignment with the CA paradigm, exploring ways to improve partition tolerance without significantly compromising consistency or availability would be a valuable area of research. This could involve developing algorithms or mechanisms that enable the system to maintain operations during network partitions.

IX. Conclusion

In this paper, we have presented Web3DB, a novel decentralized database management system (DBMS) designed for the evolving Web 3.0 landscape. Our system addresses several critical challenges in the realm of database management, particularly in the context of decentralization and the need for fine-grained access control. Web3DB's layered and modular architecture is a significant contribution to the field. It demonstrates how traditional SQL-based query processing can be seamlessly integrated with decentralized storage solutions like the InterPlanetary File System (IPFS).

One of the standout features of Web3DB is its decentralized access control mechanism. This feature is crucial in a landscape where data security and user sovereignty are of paramount importance. Our approach allows users to retain complete control over their data, a critical aspect in the age of data privacy concerns. Web3DB's support for multitenancy and its ability to facilitate databases with multiple owners represent a significant advancement. This functionality caters to the needs of collaborative environments and joint ventures, where shared data management is vital. The TPC-H benchmark tests on Web3DB, conducted across various node configurations, have validated the system's efficiency and scalability. The results demonstrated the system's capability to handle increasing loads and complex queries effectively, underscoring the benefits of our decentralized approach.

In conclusion, Web3DB stands as a testament to the potential of decentralized systems in managing and processing data. This work not only contributes a significant architectural and functional model but also opens up new avenues for research and development in the field of decentralized data systems

References

[1] W. Gan, Z. Ye, S. Wan, and P. S. Yu, "Web 3.0: The future of internet," in *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, (New York, NY, USA), p. 1266–1275, Association for Computing Machinery, 2023.

[2] O. Matthew, C. Dudley, and R. Moreton, "A review of multi-tenant database and factors that influence its adoption.," 2014.

[3] "Orbitdb - peer-to-peer databases for the decentralized web." https://github.com/orbitdb. Accessed: 2023-04-01.

[4] "Gundb - a realtime, decentralized, offline-first, graph database engine." https://gun.eco/. Accessed: 2023-10-10.

[5] "Apache couchdb." https://docs.couchdb.org/en/stable/. Accessed: 2023-10-10.

[6] "Icefiredb-sqlite - a branch of icefiredb project on github." https://github.com/IceFireDB/IceFireDB/tree/main/IceFireDB-SQLite. Accessed: 2023-04-01.

[7] J. Kammerath, "Relational database systems are becoming a problem - but what to do about it," 2023. Accessed: 2023-10-10.

[8] J. Benet, "IPFS - content addressed, versioned, P2P file system," *CoRR*, vol. abs/1407.3561, 2014.

[9] W.-M. Lee, "Using the metamask crypto-wallet," in *Beginning Ethereum Smart Contracts Programming: With Examples in Python, Solidity, and JavaScript*, pp. 111–144, Springer, 2023.

[10] solo.io, "Decentralized approach to api gateways for openshift on your way to service mesh." https://www.solo.io/blog/decentralized-approach-to-api-gateways-for-openshift-on-your-way-to-service-mesh/, 2023. Accessed: 2023-04-01.

[11] "Web3db front end." https://anonymous.4open.science/r/Web3DB-frontend-ICBC-5886/. Accessed: 2023-28-11.
[12] "Web3db back end." https://anonymous.4open.science/r/Web3DB-Backend-ICBC-F53C/. Accessed: 2023-28-11.
[13] "Run query - web3db." https://www.web3db.org/run-query. Accessed: 2023-28-11.
[14] M. T. Özsu and P. Valduriez, "Distributed database systems: where are we now?," *Computer*, vol. 24, pp. 68–78, 1991.
[15] A. Corbellini, C. Mateos, A. Zunino, D. Godoy, and S. N. Schiaffino, "Persisting big-data: The nosql landscape," *Inf. Syst.*, vol. 63, pp. 1–23, 2017.
[16] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, p. 35–40, apr 2010.
[17] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, "Spanner: Google's globally distributed database," *ACM Trans. Comput. Syst.*, vol. 31, aug 2013.
[18] D. Tapscott and A. Tapscott, *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World*. Portfolio, 2016.
[19] M. Swan, *Blockchain: Blueprint for a New Economy*. O'Reilly Media, Inc., 1st ed., 2015.
[20] T. McConaghy, R. Marques, A. Müller, D. D. Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto, "Bigchaindb whitepaper." https://gamma.bigchaindb.com/whitepaper/, 2023. Accessed: 2023-04-01.
[21] S. Nathan, C. Govindarajan, A. Saraf, M. Sethi, and P. Jayachandran, "Blockchain meets database: Design and implementation of a blockchain relational database," *Proc. VLDB Endow.*, vol. 12, p. 1539–1552, jul 2019.
[22] R. Singh, D. Kukreja, and D. K. Sharma, "Blockchain-enabled access control to prevent cyber attacks in iot: Systematic literature review," *Frontiers in Big Data*, vol. 5, 2023.
[23] K. Patel, R. Modi, S. Sharma, and M. Patel, "A survey: Secure cloud data storage and access control system using blockchain," in *Soft Computing for Security Applications* (G. Ranganathan, X. Fernando, and S. Piramuthu, eds.), (Singapore), pp. 195–207, Springer Nature Singapore, 2023.
[24] P. Bagga, A. K. Das, V. Chamola, and M. Guizani, "Blockchain-envisioned access control for internet of things applications: a comprehensive survey and future directions," *Telecommunication Systems*, vol. 81, pp. 125–173, 9 2022.
[25] A. Bashir Dar, A. Hamid Lone, R. Naaz, A. Iqbal Baba, and F. Wu, "Blockchain driven access control mechanisms, models and frameworks: A systematic literature review," *Journal of Information Security and Cybercrimes Research*, vol. 5, pp. 05–34, Jun. 2022.
[26] R. Agrawal and N. Gupta, "Blockchain based access control systems," *Blockchain Applications in Cybersecurity Solutions*, vol. 1, p. 117, 2023.
[27] L. Xu, I. Markus, S. I, and N. Nayab, "Blockchain-based access control for enterprise blockchain applications," *International Journal of Network Management*, vol. 30, no. 5, p. e2089, 2020. e2089 nem.2089.
[28] B. Johnsirani and M. Natarajan, "An overview of distributed database management system," 2015.
[29] S. Ram and C. L. Chastain, "Architecture of distributed data base systems," *Journal of Systems and Software*, vol. 10, no. 2, pp. 77–95, 1989.
[30] F. D. Muñoz-Escoí, R. de Juan-Marín, J.-R. García-Escrivá, J. R. González de Mendívil, and J. M. Bernabéu-Aubán, "Cap theorem: Revision of its related consistency models," *The Computer Journal*, vol. 62, no. 6, pp. 943–960, 2019.