COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

BLOCKCHAIN AND DATABASES

SUBMITTED BY HARSHA SALIM

([redacted])

IN PARTIAL FULFILLMENT FOR THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

KALAMASSERY

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

KOCHI 682022

CERTIFICATE

Certified that this bonafide record of seminar report titled Blockchain and Databases

Done by

Harsha Salim ([redacted])

of VIII Semester, Computer Science And Engineering in the year 2021 in partial fulfillment requirements for the award of degree of Bachelor of Technology in division of Computer Science and Engineering of Cochin University of Science and Technology.

Dr David Peter S	Ms Latha R Nair	Ms Mahima Mary Mathews
Head of Division	Class Coordinator	Seminar Guide

ABSTRACT

This seminar aims to display the relationship between blockchain technology and databases. First the mutual influence of database and blockchain is reviewed, and the CAP (Consistency, Availability, Partition tolerance) theorem is reviewed, and how the CAP theorem influenced the DCS (Decentralization, Consistency, Scalability) theorem for blockchain. Following this is an in depth study of BigChainDB. BigchainDB is a blockchain database offering decentralization, immutability and native assets. BigchainDB allows for the deployment of large-scale applications in a variety of use cases and industries from intellectual property and identity to supply chain, and Internet-of-Things.

ACKNOWLEDGEMENT

I take this occasion to thank God Almighty for blessing me with his grace and taking my endeavor to a successful culmination. I extend my sincere gratitude to my respected Head of Division, Dr. David Peter S for enabling me to enter this endeavor. I am highly indebted to Ms Latha R Nair, our batch co-coordinator and my guide Ms Mahima Mary Mathews, for their constant supervision and support for completing the seminar. I also take this opportunity to express my profound gratitude and heartfelt thanks for providing me with the right guidance and for their constant encouragement throughout the course of this seminar. I extend my sincere thanks to all other faculty members for sharing their valuable time and knowledge with me. Lastly, I would like to thank my friends and family for their constant encouragement without whom, this seminar would not have been possible.

Harsha Salim ([redacted])

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Blockchain has gained immense popularity from the last decade, acting as a distributed ledger for peer to peer transactions in a secure and immutable way. The development of blockchain pushed the market of decentralized applications in varied enterprises such as financial markets, insurance industries, supply chain industry. In this decentralized network of peers, each peer has a replica of data. This decentralization of blockchain raises many questions on “How and where to store data”. The problem where to store blockchain data has been somehow solved by using decentralized cloud storage solutions, but these solutions suffer from limited capability and user privacy matters.

Fig 1 : Bitcoin

After the invent of bitcoin[1], many questions about the blockchain have been carried out as “blockchain as a database” or “difference between blockchain and database” [2]. Blockchain differs from traditional databases in numerous ways like its decentralization, cryptographic security using chained hashes, no administration control, immutability, freedom to transfer without the permission of any central authority. To cherish these differences, many enterprise applications upgraded their traditional database storage solution with blockchain to make their implementation more secure, involving less trust among the parties of the industry.

Despite having the features mentioned above, blockchain still lacks some features which traditional database has. Blockchain can leverage the traditional database features by either integrating the traditional database with blockchain or, to create a blockchain-oriented distributed database. The inclusion of the database features will leverage the blockchain with low latency, high throughput, fast scalability, and complex queries on blockchain data. Thus having the features of both blockchain and database, the application enhances its efficiency and security.

Many of the blockchain platforms are now integrating with a database. In recent years, many blockchain databases have been developed and introduced. These distributed databases have their consensus mechanism for the joint agreement on a data block by the network parties. These blockchain databases support features like complex data types, rich query structure, ACID compliant [3], low latency, fast scalability, and cloud hosting. The adoption of database features in blockchain or vice-versa is an interesting research topic. Few industries have already built their blockchain database with all the required features. Many companies, including database giants IBM, Oracle, and SAP, as well as startups such as FlureeDB, BigchainDB, have devoted their efforts to develop blockchain database solutions to support SQL-like queries.

Fig 2 : BigChainDB

BigchainDB is software that has blockchain properties (e.g. decentralization, immutability, owner-controlled assets) and database properties (e.g. high transaction rate, low latency, indexing & querying of structured data), hence it was called a Blockchain Database. It was first released—open source—in February 2016 and has been improved continuously ever since. BigchainDB version 2.0 makes significant improvements over previous versions. In particular, it is Byzantine fault tolerant (BFT), so up to a third of the nodes can fail in any way, and the system will continue to agree on how to proceed. BigchainDB 2.0 is also production-ready for many use cases.

CHAPTER 2

LITERATURE REVIEW

In Permissioned blockchains and distributed databases: A performance study, by S. Bergman, M. Asplund, and S. Nadjm-Tehrani [4] a performance study on distributed databases for blockchain has already been done, but that involves very few databases. Also, in Trends in Development of Databases and Blockchain by Mayank Raikwar, Danilo Gligoroski, Goran Velinov, a systematized study of those development trends has been made. It provided a detailed summary of traditional databases that are used or can be used in the design of blockchain platforms or applications. It is also about a detailed explanation of different decentralized solutions that use traditional databases but provide blockchain-enabled solutions. Finally, it described the DCS theorem and the trade-off properties present in the blockchain systems in a similar way to the CAP theorem for the database systems [6].

In BigchainDB: A Scalable Blockchain Database by BigchainDB GmbH [7], it describes BigchainDB. BigchainDB fills a gap in the decentralization ecosystem: a decentralized database, at scale. BigchainDB inherits characteristics of modern distributed databases : linear scaling in throughput and capacity with the number of nodes, a full featured NoSQL query language, efficient querying, and permissioning. Being built on an existing distributed DB, it also inherits enterprise-hardened code for most of its codebase. Scalable capacity means that legally binding contracts and certificates may be stored directly on the blockchain database. The permissioning system enables configurations ranging from private enterprise blockchain databases to open, public blockchain databases.

In BigchainDB 2.0, The Blockchain Database by BigchainDB GmbH [7], BigchainDB is complementary to decentralized processing platforms like Ethereum, and decentralized file systems like InterPlanetary File System (IPFS) [11]. The paper describes technology perspectives that led to the BigchainDB design: traditional blockchains, distributed databases, and a case study of the domain name system (DNS). A concept called blockchain pipelining was introduced, which is key to scalability when adding blockchain-like characteristics to the distributed DB. Also a thorough description of BigchainDB, an analysis of latency, and preliminary experimental results were given.

The paper concluded with a description of use cases.

BigchainDB version 2.0 makes significant improvements over previous versions. In particular, it is Byzantine fault tolerant (BFT), so up to a third of the nodes can fail in any way, and the system will continue to agree on how to proceed. BigchainDB 2.0 is also production-ready for many use cases. The paper reviews the design goals of BigchainDB 2.0 and how they were achieved, some use cases, how BigchainDB fits into the overall decentralization ecosystem, the life of a transaction to understand how BigchainDB 2.0 works, and ways to try BigchainDB.

CHAPTER 3

BLOCKCHAIN AND DATABASES BASICS

3.1. BLOCKCHAIN

A blockchain is a growing list of records, called blocks, that are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. Here, modification of data is not permissible by design. It allows decentralized control and eliminates the risks of modification of data by other parties with sufficient access to the system.

It is a distributed ledger technology that enables a set of peers to work together to create a unified, decentralized network. The peers can communicate and share information or data with the help of the consensus algorithm. Also, there is no need for a centralized authority, which makes the whole network trustworthy when compared to other networks.

When one peer sends information to another, a transaction is generated. When this happens, the transactions need to be validated using the consensus algorithm. Proof of Work is an example of a consensus algorithm. It ensures that no invalid transactions are passed into the blockchain.

Timestamps are created to ensure that each transaction can be traced, backed, and verified by anyone. The whole system adds value and brings in new features such as transparency, immutability, and security.

If we measure the Bitcoin blockchain by traditional DB criteria: throughput is just a few transactions per second (tps), latency before a single confirmed write is 10 minutes, and capacity is a few dozen GB. Furthermore, adding nodes causes more problems: with a doubling of nodes, network traffic quadruples with no improvement in throughput, latency, or capacity. It also has essentially no querying abilities.

The following are the important properties of Blockchain:

- Immutability : Every node on the system has a copy of the digital ledger. To add a transaction every node needs to check its validity. If the majority thinks it's valid, then it's added to the ledger. This promotes transparency and makes it corruption-proof.

- Decentralized : The network is decentralized meaning it doesn't have any governing authority or a single person looking after the framework. Rather a group of nodes maintains the network making it decentralized.

- Enhanced Security : As it gets rid of the need for a central authority, no one can just simply change any characteristics of the network for their benefit. Using encryption ensures another layer of security for the system. Cryptography is a rather complex mathematical algorithm that acts as a firewall for attacks.

- Distributed Ledgers : Usually, a public ledger will provide every information about a transaction and the participant. It's all out in the open, nowhere to hide. Although the case for private or federated blockchain is a bit different. But still, in those cases, many people can see what really goes on in the ledger. That's because the ledger on the network is maintained by all other users on the system. This distributes computational power across the computers to ensure a better outcome.

- Consensus : Every blockchain thrives because of the consensus algorithms. The architecture is cleverly designed, and consensus algorithms are at the core of this architecture. Every blockchain has a consensus to help the network make decisions. When millions of nodes are validating a transaction, a consensus is absolutely necessary for a system to run smoothly. The consensus is responsible for the network being trustless. Nodes might not trust each other, but they can trust the algorithms that run at the core of it. That's why every decision on the network is a winning scenario for the blockchain. It's one of the benefits of blockchain features.

3.2. DATABASE

The database, unlike blockchains, are a centralized ledger that is run by an administrator. Databases also exhibit unique features, including the ability to read and write. Here, only the parties with proper access can do Write and Read actions. Databases also exhibit the ability to store multiple copies of the same data and their history. This is done with the help of a trusted, centralized authority who manages the server.

Centralization brings many benefits to the database. For example, it is easy to manage databases as the data is centralized. Accessing and storing data is not only easy but also fast. However, they also have drawbacks.

One of the biggest drawbacks is the chance of the data getting corrupted. To overcome the disadvantage, multiple backups are taken. But, that’s not always the case, as most of the entities always trust their owner and hence skip the backup data option.

Another big drawback is how the data can be modified by anyone who is in control of the database itself. This can happen as the database is centralized in nature.

A database utilizes data structure to store information. All the data that is stored in a database can be queried using a special querying language known as Structured Query Language (SQL). A database can work with almost every type of data and help support all modern enterprises. Also, it can be scaled to support millions of records.

The history of the database is also rich. It started with just hierarchical file systems. It had severe limitations, and hence it then later adapted to the relational model. The relational model is useful and gives the owner the ability to work with different databases at the same time. Database management systems are used to organize databases effectively.

At the core, data elements are stored in tables. The table consists of fields that can record a different type of data, known as attributes.

There are many different types of blockchain, out of which there is a private blockchain that works in a closed ecosystem. This may sound similar to what databases are about, but they are fundamentally different. Private blockchain inherits all the properties that a blockchain has to offer, but it works in a closed environment. Only people who are allowed by the administrator can participate in the blockchain. The only similarity between private blockchain and database is the centralized aspect.

CHAPTER 4

MUTUAL INFLUENCE OF BLOCKCHAIN AND DATABASES

Blockchain and database both can achieve many functionalities and features by coping with each other. If we frame blockchain as a database to provide a storage mechanism, then we can analyze how it differs from actual database systems. The following are the key points where blockchain and database differ in their properties, but both can leverage and enhance the characteristics of each other.

- Traditional blockchain throughput decreases when the processing capacity of nodes participating in the blockchain increases. Yet, in the case of the distributed database, the throughput increases when the nodes increase. Hence throughput can be enhanced.

- The latency of transactions in blockchain is usually high compared to the latency in databases. Thus, the latency can be made low as desired with the use of a database.

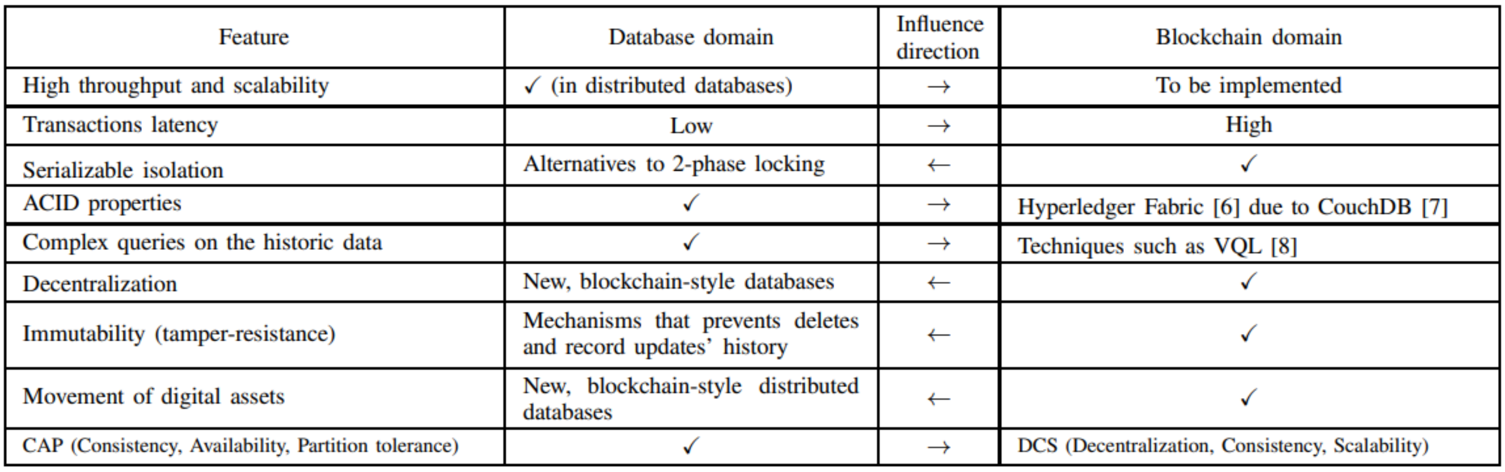
- Transactions in blockchain require serializable isolation, which can be achieved by consensus algorithms providing strong consistency. For the databases, there is a well-understood mechanism called 2-phase locking and concurrency-control [8]. However, new blockchain databases such as BlockchainDB based on MongoDB start to offer new transaction mechanisms based on blockchain.

- Most of the blockchain platforms do not support complex queries in its historic data. These queries are needed in many applications to retrieve the desired information. The complex query feature is available in most of the databases, but the provenance queries on historic data can be supported by the use of Multi-Version Concurrency Control [9].

- The decentralization feature of blockchain has rewired most of the financial systems and industries from the last decade. Decentralization is not available in the traditional distributed database. With the advent of new blockchain style databases, decentralization is now possible and leads to promising growth to be used in many applications.

- One of the other excellent features of blockchain is immutability or tamper-resistance of transactions. This tamper-resistance can be achieved in database systems by mechanisms that disallow the deletes and updates in the database.

- Blockchain allows the creation and movement of digital assets, which is not allowed in a classical database. But, a blockchain-style distributed database can have this feature as a built-in feature.



Feature	Database domain	Influence direction	Blockchain domain
High throughput and scalability	✓ (in distributed databases)	→	To be implemented
Transactions latency	Low	→	High
Serializable isolation	Alternatives to 2-phase locking	←	✓
ACID properties	✓	→	Hyperledger Fabric [6] due to CouchDB [7]
Complex queries on the historic data	✓	→	Techniques such as VQL [8]
Decentralization	New, blockchain-style databases	←	✓
Immutability (tamper-resistance)	Mechanisms that prevents deletes and record updates' history	←	✓
Movement of digital assets	New, blockchain-style distributed databases	←	✓
CAP (Consistency, Availability, Partition tolerance)	✓	→	DCS (Decentralization, Consistency, Scalability)

Fig 3 : A Summary of the mutual influence and the entangled development of Databases and Blockchain

CHAPTER 5

CAP THEOREM AND DCS THEOREM

5.1. CAP THEOREM

CAP was introduced 20 years ago by Brewer as a principle or conjecture, and two years later, it was proven in the asynchronous network model as a CAP theorem by Gilbert and Lynch. Similar impossibility results were proven for a partially synchronous network model. Additionally, by weakening the consistency conditions, they showed that it is possible to achieve all three properties in the so-called t-Connected Consistency model. In more details, CAP theorem identifies the three specific system properties for any distributed/decentralized system. These properties are Consistency, Availability and Partition Tolerance.

- Consistency : Any read in the distributed system gives the latest write on the nodes.
- Availability : A Client always receives a response at any point of time irrespective of whether the read is the latest write.
- Partition Tolerance : In case of partition between nodes in the distributed system, the system should still be functioning.

CAP theorem states that it is possible to achieve two of these three properties as guaranteed features in a distributed network, but it is impossible to achieve all three features at the same time. In practice, a distributed system always needs to be partition tolerant, thus leaving us to choose one property from Consistency or Availability. Hence, there is a trade-off between consistency and availability.

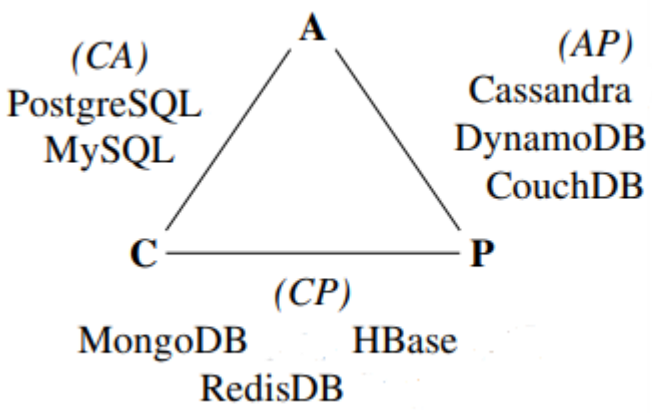


Fig 4 : CAP Triangle for Database Systems

CAP theorem has also made its influence in the blockchain realm. If we pick Availability over Consistency, any reads are not guaranteed to be up-to-date, and we call the system as AP. However, if we choose Consistency over Availability, the system, called CP, would be unavailable at the time of partition and might disrupt the consensus. Thus in blockchain systems, both properties are desirable.

5.2. DCS THEOREM

Though blockchain does not always require strong consistency, eventual consistency can serve the purpose and can be achieved through consensus. For example, in the case of bitcoin, the longest chain method brings eventual consistency, but there are no fixed methods to achieve eventual consistency and leaves this topic for debate. An analogy to the CAP theorem for blockchain has been proposed as the DCS theorem, where DCS abbreviation refers to Decentralization, Consistency, Scalability [10].
The DCS theorem states that a blockchain system can have at most two properties simultaneously out of the three estates of DCS. The DCS properties can be defined as follows:

- Decentralization - There is no trusted entity controlling the network, hence no single point of failure. Blockchains are inherently decentralized, but in the DCS triangle, consider the case of full decentralization, where any node can join the network and participate as a validator.
- Consistency - The blockchain nodes will read the same data at the same time. The query for the blockchain data on any blockchain node should fetch the same result. The consistency in blockchain should prevent double spending and should be brought from the consensus algorithm used.
- Scalability - The performance of blockchain should increase with the increase in the number of peers and the number of allocated computational resources. The throughput and the capacity of the system should be high, and latency should be low.

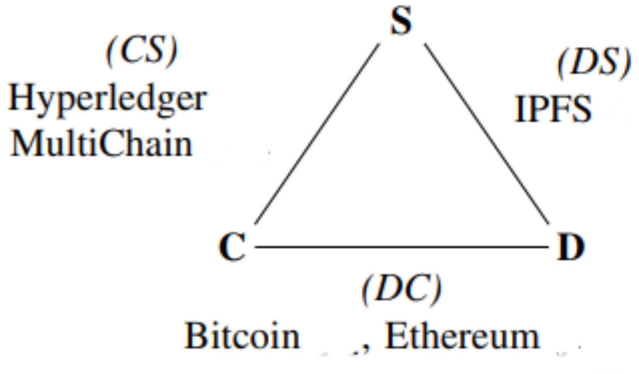


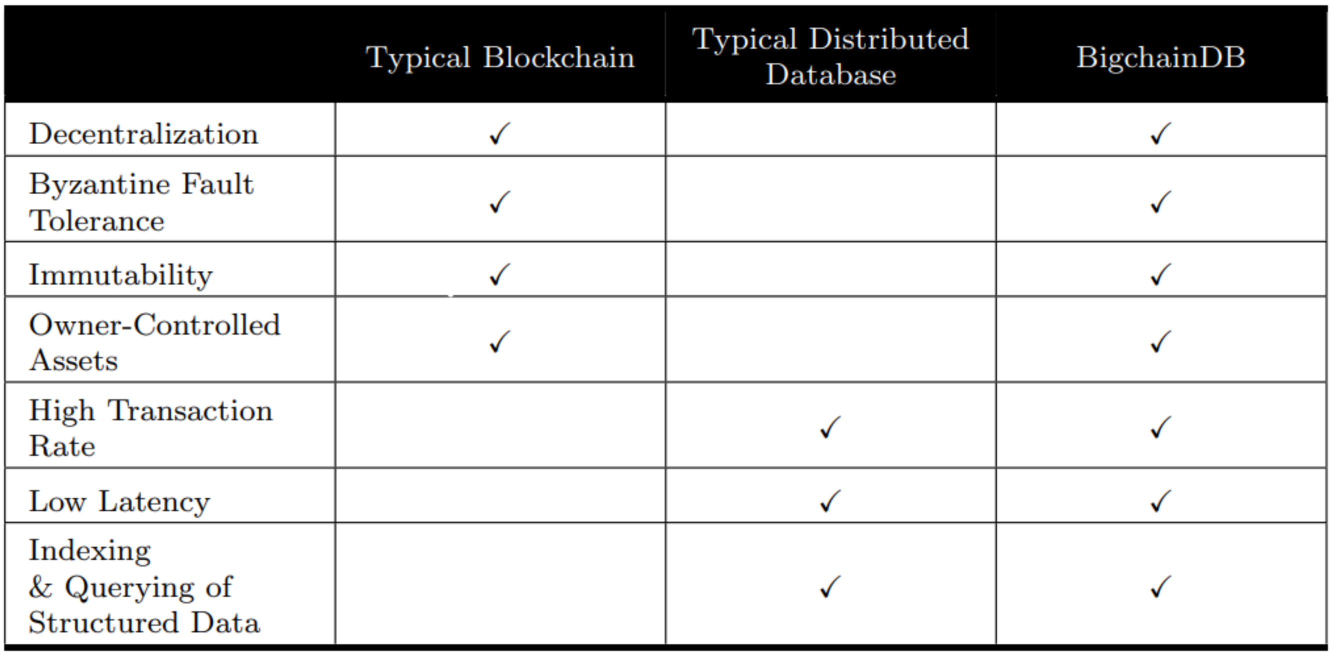
Fig 5 : DCS Triangle for Blockchain

In a similar way to CAP, one can also categorize the blockchain systems in DCS as DC, CS, and DS systems as trade-offs between the DCS properties. Most of the cryptocurrencies like Bitcoin can be considered as DC systems. Nevertheless, all the permissioned blockchains do not have full decentralization, hence should be regarded as CS systems.

CHAPTER 6

BIGCHAINDB

BigchainDB, is for database-style decentralized storage : a blockchain database. BigchainDB combines the key benefits of distributed DBs and traditional blockchains, with an emphasis on scale. The following are the design goals of BigChainDB 2.0 :



	Typical Blockchain	Typical Distributed Database	BigchainDB
Decentralization	✓		✓
Byzantine Fault Tolerance	✓		✓
Immutability	✓		✓
Owner-Controlled Assets	✓		✓
High Transaction Rate		✓	✓
Low Latency		✓	✓
Indexing & Querying of Structured Data		✓	✓

Fig 6 : BigChainDB Features

6.1. FULL DECENTRALIZATION AND BYZANTINE FAULT TOLERANCE

BigchainDB 2.0 uses Tendermint [12, 13] for all networking and consensus. Each node has its own local MongoDB database [14], and all communication between nodes is done using Tendermint protocols. One consequence is that the resulting system is BFT, because Tendermint is BFT. Another result is that if a malicious hacker manages to get admin privileges to one of the local MongoDB databases, then the worst they can do is corrupt or delete the data in that local database; the MongoDB databases in the other nodes won't be affected. If every node in a BigchainDB 2.0 network is owned and operated by a different person or entity, then it's a decentralized network because it has no single owner, no single point of control, and no single point of failure. Any node can fail and the rest of the network will continue to operate. In fact, up to one third of the nodes can fail in arbitrary ways, and the rest of the network will continue to work, i.e. the non-faulty nodes will agree on how to proceed.

6.2. IMMUTABILITY

Once data gets stored in a BigchainDB network, it can't be changed or erased, or at least not without great difficulty. If some data somehow manages to get changed or erased, then that is detectable. BigchainDB uses several strategies to achieve practical immutability. The simplest one is that there are no BigchainDB-provided APIs to change or erase stored data. Another strategy is that every node has a full copy of all the data in a standalone MongoDB database (i.e. there is no global MongoDB database). Even if one node gets corrupted or destroyed, the other nodes won't be affected and will still have a copy of all the data. Another strategy is that all transactions are cryptographically signed. After a transaction is stored, changing its contents will change the signature, which can be detected (unless the public key is also changed, but that should also be detectable because every block of transactions is signed by a node, and the public keys of all the nodes are all known).

6.3. OWNER-CONTROLLED ASSETS

Like most blockchains, BigChainDB has a concept of owner-controlled assets. Only the owner (or owners) of an asset can transfer that asset. (The owners are the holders of a particular set of private keys.) Not even a node operator can transfer an asset. In most blockchains, there's only one built-in asset (e.g. Bitcoin or Ether), but BigChainDB allows external users to create as many assets as they need. However, it's worth noting that a user can't create assets that appear to be created by someone else. BigchainDB checks every transaction to make sure it's not trying to transfer an output that was already transferred (spent) by another transaction, i.e. it prevents double-spending. It also checks many other things, all of which are listed in the BigchainDB Transaction Spec v2 [15].

6.4. HIGH TRANSACTION RATE

One design goal of BigChainDB has always been the ability to process a large number of transactions each second. That's still true with BigChainDB 2.0. BigChainDB 2.0 was still in Alpha at the time of writing. Performance tests were being written and started, but no concrete results were available yet. However, BigChainDB allows for quite complex conditions on the outputs. Tendermint consensus can process thousands of transactions per second, with commit latencies on the order of one to two seconds. Notably, performance of well over a thousand transactions per second is maintained even in harsh adversarial conditions, with validators crashing or broadcasting maliciously crafted votes.

6.5. LOW LATENCY AND FAST FINALITY

Tendermint-based networks (such as BigchainDB networks) take only a few seconds (or less) for a transaction to be included in a new committed block. Once that happens, there's no way it can be reverted or considered defunct in the future, because Tendermint doesn't do forking.

6.6. INDEXING & QUERYING STRUCTURED DATA

Each node in a BigchainDB 2.0 network has its own local MongoDB database. That means that each node operator has access to the full power of MongoDB for indexing and querying the stored data (transactions, assets, metadata and blocks, all of which are JSON strings). Each node operator is free to decide how much of that power they expose to external users. One node operator might decide to index geospatial data and offer optimized geospatial queries via a REST API, whereas another node operator might decide to offer a GraphQL API. By default, BigchainDB 2.0 creates some MongoDB indexes and the BigchainDB HTTP API includes some endpoints for doing basic queries. However, as outlined in the previous paragraph, each node operator can add additional indexes and query APIs.

6.7. SYBIL TOLERANCE

Some blockchain networks (such as Bitcoin) allow anyone to add their node to the network. That brings the concern that someone could add so many nodes that they effectively control the network : a Sybil attack. Bitcoin makes Sybil attacks unlikely by making them prohibitively expensive. In a BigchainDB network, the governing organization behind the network controls the member list, so Sybil attacks are not an issue.

CHAPTER 7

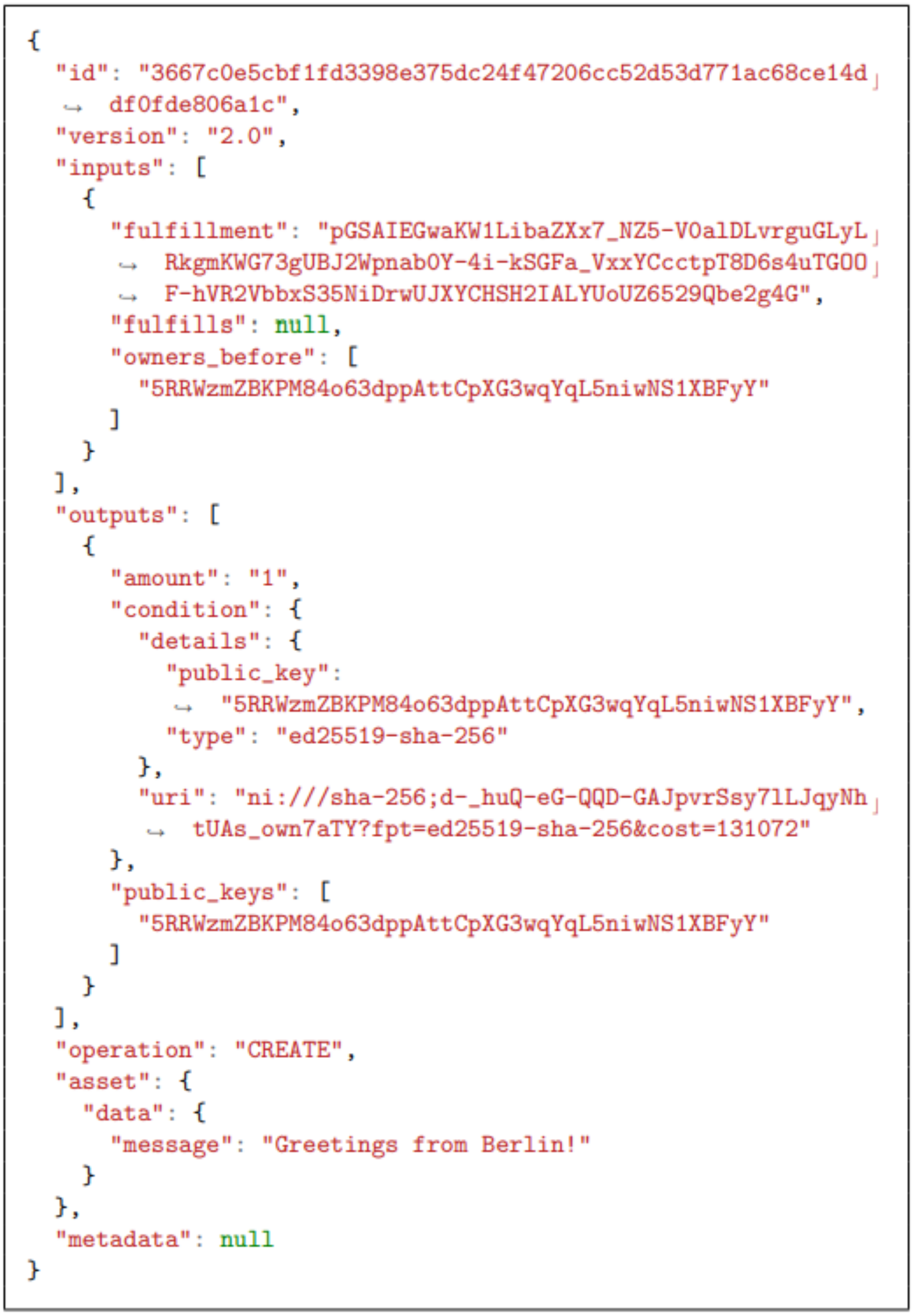
BIGCHAINDB TRANSACTION

A BigchainDB transaction is a JSON string that conforms to a BigchainDB Transactions Specification (Spec). There are two such specs: v1 and v2. If someone wants to construct a valid BigchainDB transaction, then they'll typically use a BigchainDB driver (software package). There's a list of BigchainDB drivers in the BigchainDB docs [16].

Once one has a transaction, they can send it to a BigchainDB network using the BigchainDB HTTP API [17]. More specifically, one would use one of the following endpoints, with the transaction in the body of the HTTP request:

- POST /api/v1/transactions
- POST /api/v1/transactions?mode=async
- POST /api/v1/transactions?mode=sync
- POST /api/v1/transactions?mode=commit

A BigchainDB driver could also be used to post a transaction. The HTTP request (containing the transaction) can be sent to any of the nodes in the BigchainDB network, or even more than one. After the HTTP request arrives successfully at the web server inside a BigchainDB node, it exposes a standard interface (Web Server Gateway Interface [WSGI]) which enables Python applications to talk to it. BigchainDB uses the Flask web application development framework to simplify working with WSGI. Flask is used to route the request to a Python method for handling that endpoint. That method checks the validity of the transaction. If it's not valid, then that's the end of the story for the transaction, the HTTP response status code is 400 (Error), and the response body gives some information about what was invalid.



```
{
  "id": "3667c0e5cbf1fd3398e375dc24f47206cc52d53d771ac68ce14d
   ↪ df0fde806a1c",
  "version": "2.0",
  "inputs": [
    {
      "fulfillment": "pGSAIEGwaKW1LibaZXx7_NZ5-V0alDLvrguGLyL
       ↪ RkgmKWG73gUBJ2Wpnab0Y-4i-kSGFa_VxxYCcctpT8D6s4uTGOO
       ↪ F-hVR2VbbxS35NiDrwUJXYCHSH2IALYUoUZ6529Qbe2g4G",
      "fulfills": null,
      "owners_before": [
        "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY"
      ]
    }
  ],
  "outputs": [
    {
      "amount": "1",
      "condition": {
        "details": {
          "public_key":
           ↪ "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY",
          "type": "ed25519-sha-256"
        },
        "uri": "ni:///sha-256;d-_huQ-eG-QQD-GAJpvrSsy7lLJqyNh
         ↪ tUAs_own7aTY?fpt=ed25519-sha-256&cost=131072"
      },
      "public_keys": [
        "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY"
      ]
    }
  ],
  "operation": "CREATE",
  "asset": {
    "data": {
      "message": "Greetings from Berlin!"
    }
  },
  "metadata": null
}
```

Fig 7 : A BigChainDB Transaction

If the transaction is valid, then it's converted to Base64 and put into a new JSON string with some other information (such as the mode). When a transaction is sent to a Tendermint node, it will run via CheckTx against the application. If it passes CheckTx, it will be included in the mempool, broadcast to other peers, and eventually included in a block. Since there are multiple phases to processing a transaction, we offer multiple endpoints to broadcast a transaction:

- /broadcast_tx_async
- /broadcast_tx_sync
- /broadcast_tx_commit

These correspond to no-processing, processing through the mempool, and processing through a block, respectively. That is, broadcast_tx async will return right away without waiting to hear if the transaction is even valid, while broadcast_tx sync will return with the result of running the transaction through CheckTx. Using broadcast_tx commit will wait until the transaction is committed in a block or until some timeout is reached, but will return right away if the transaction does not pass CheckTx. The benefit of using broadcast_tx commit is that the request returns after the transaction is committed (i.e. included in a block), but that can take on the order of a second. For a quick result, use broadcast_tx sync, but the transaction will not be committed until later, and by that point its effect on the state may change. If no mode was specified, then the default is async.

Every Tendermint instance has a local mempool (memory pool) of transactions which have passed initial validation, but haven't been included in a block yet. When Tendermint wants to determine if a transaction is valid, it sends the transaction to BigchainDB using a CheckTx request. It expects BigchainDB to implement CheckTx, and several other message types, all of which are explained in the ABCI Specification [18]. (ABCI stands for Application BlockChain Interface.)

When storing a transaction in a Blockchain network, BigchainDB removes the asset.data value (JSON string) and stores it in a separate collection of assets. It does that to facilitate the text search of assets. Similarly, the metadata value (JSON string) is also removed and stored in a separate collection. Tendermint writes the block to the blockchain after getting a response to the Commit message.

CHAPTER 6

CONCLUSION

The last decade was a decade of an intense interchanged and mutually influenced development of the database and blockchain technologies. A systematized study of those development trends was done. The CAP Theorem and DCS Theorem were discussed. BigchainDB is a blockchain database: it has both blockchain properties and database properties. That combination makes it useful for a wide variety of use cases, including supply chain, IP rights management, digital twins & IoT, identity, data governance and immutable audit trails. BigchainDB 2.0 is now production-ready for many use cases.

REFERENCES

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf,” 2009.

[2] M. J. M. Chowdhury, A. Colman, M. A. Kabir, J. Han, and P. Sarda, “Blockchain versus database: A critical analysis,” in 2018 IEEE TrustCom/BigDataSE, Aug 2018, pp. 1348–1353.

[3] E. A. Brewer, “Towards robust distributed systems,” in PODC, vol. 7. Portland, OR, 2000.

[4] S. Bergman, M. Asplund, and S. Nadjm-Tehrani, “Permissioned blockchains and distributed databases: A performance study,” Concurrency and Computation: Practice and Experience, p. e5227, 2018.

[5] Mayank Raikwar, Danilo Gligoroski, Goran Velinov, “Trends in Development of Databases and Blockchain”, Journal reference : 2020 Seventh International Conference on Software Defined Systems (SDS).

[6] S. Gilbert and N. Lynch, “Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services,” Acm Sigact News, vol. 33, no. 2, pp. 51–59, 2002.

[7] BigchainDB GmbH, “BigchainDB 2.0 The Blockchain Database”, May 2018.

[8] H. T. Kung and J. T. Robinson, “On optimistic methods for concurrency control,” ACM Trans. Data. Syst., vol. 6, no. 2, pp. 213–226, Jun. 1981.

[9] P. A. Bernstein and N. Goodman, “Multiversion concurrency control theory and algorithms,” ACM Trans. Data. Syst., vol. 8, no. 4, pp. 465–483, 1983.

[10] K. Zhang and H.-A. Jacobsen, “Towards dependable, scalable, and pervasive distributed ledgers with blockchains.” in ICDCS, 2018, pp. 1337–1346.

[11] J. Benet, “IPFS - content addressed, versioned, P2P file system,” arXiv, 2014. [Online]. Available: http://arxiv.org/abs/1407.3561.

[12] Jae Kwon. Tendermint: Consensus without Mining, fall 2014.

[13] Jae Kwon. Tendermint: Consensus without Mining, fall 2014.

[14] MongoDB. https://www.mongodb.com.

[15] BigchainDB Transactions Spec v2. https://github.com/bigchaindb/ BEPs/tree/master/13.

[16] BigchainDB Drivers & Tools.
https://docs.bigchaindb.com/projects/server/en/latest/drivers-clients/index.html

[17] BigchainDB HTTP API.
https://docs.bigchaindb.com/projects/server/en/latest/http-client-server-api.html

[18] Tendermint ABCI Specification. https://github.com/tendermint/ abci/blob/master/specification.rst