Using Blockchain to support Data & Service Monetization

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Obaro Ogheneyoma Odiete

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
S7N 5C9
Canada

OR

College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 - 110 Science Place
Saskatoon SK S7N 5C9
Canada

Abstract

Two required features of a data monetization platform are query and retrieval of the metadata of the resources to be monetized. Centralized platforms rely on the maturity of traditional NoSQL database systems to support these features. These databases, for example,mm MongoDB allows for very efficient query and retrieval of data it stores. However, centralized platforms come with a bag of security and privacy concerns, making them not the ideal approach for a data monetization platform. On the other hand, most existing decentralized platforms are only partially decentralized. In this research, I developed Cowry, a platform for publishing metadata describing available resources (data or services), discovery of published metadata including fast search and filtering. My main contribution is a fully decentralized architecture that combines blockchain and traditional distributed database to gain additional features such as efficient query and retrieval of metadata stored on the blockchain.

Acknowledgements

I would like to express sincere thanks to my supervisors, Dr. Ralph Deters and Dr. Kevin Schneider for their guidance and funding throughout the duration of my Masters program. I would also like to appreciate all the members of my thesis committee: Dr. Chanchal Roy, Dr. Anh Dinh and Dr. Julita Vassileva for their thoughtful comments. In addition, I acknowledge my friends and colleagues in MADMUC Lab especially Kiemute Oyibo, Ifeoma Adaji and Tanvi Jain for all the interesting discussions. I also appreciate friends from outside of the lab including Richard Lomotey and Johnson Iyilade. Finally, I thank my parents and family for their patience and abiding love.

This thesis is dedicated to the Almighty God.

Contents

List of Tables

List of Figures

List of Abbreviations

AWS	Amazon Web Services
CAP	Consistency, Availability, Partition Tolerance
DAO	Decentralized Autonomous Organization
DPOS	Delegated Proof of Stake
ECDSA	Elliptic Curve Digital Signature Algorithm
IoT	Internet of Things
NoSQL	Not Only Structured Query Language
P2IRC	Plant Phenotyping and Imaging Research Centre
P2P	Peer-to-Peer
PBFT	Practical Byzantine Fault Tolerance
PoS	Proof-of-Stake
PoW	Proof-of-Work
S^2aaS	Sensing as a Service
SHA	Secure Hashing Algorithm
SQL	Structured Query Language
UTXO	Unspent Transaction Output

Chapter 1

Introduction

1.1 Background

Large amount of data is constantly generated around us every day. These data come from different sources such as Internet-connected everyday objects for example cooker, microwave, fridges exchanging data (a concept referred to as the Internet of Things [3, 72]), and humans interacting with each other via different social media. The data generated both by humans and machines fall into the category of Big Data because of its volume, variety, veracity, and velocity [14] as illustrated in Figure 1.1.

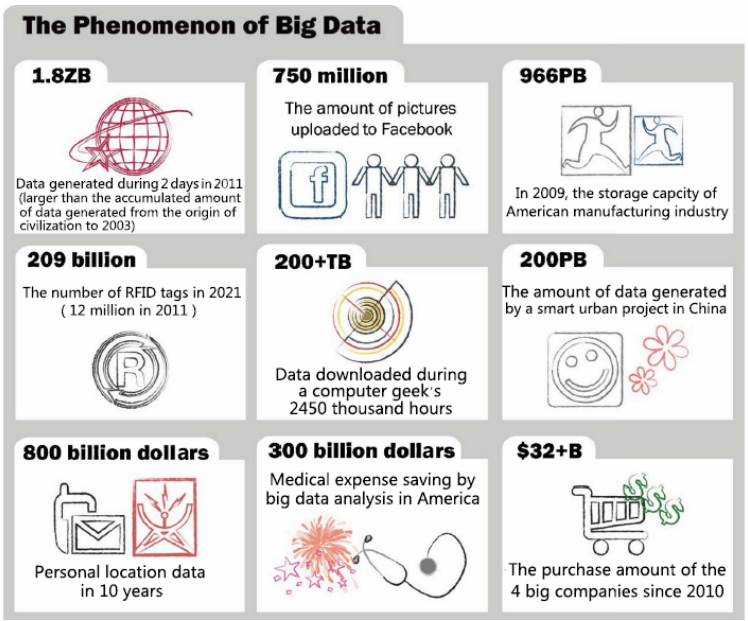

Figure 1.1: Continuously growing amount of data generated (adapted from [14])

These data hold a significant amount of knowledge that can be and is being used to build products and services that can improve human lives. For example, analyzing data from standard mobile phone logs, researchers could predict user's personality [25] which can be useful in personalization services, also researchers at Google used millions of images on the Internet to build an unsupervised learning model for training face detectors [53], a technology that can enhance security. In addition, researchers from the Plant Phenotyping

and Imaging Research Centre (P2IRC) are working on designing crops for global food security using plant genomic and crop phenotype data [68] etc. Besides research, consumer data would be useful to help retailers and manufacturers understand their customers better.

However, for the data to be useful (as mentioned above), thousands (even millions) of data points need to be aggregated from several sources or data owners. For example, for the analysis of phone logs, several participants need to provide their logs from phones, or in the P2IRC project, images and other data needs to be collected from several farmlands, and distributors need data from several different customers.

There are essentially three main participants involved in this data flow or exchange:

- **Data Owners:** The users (individuals or organizations) that [owns the devices that] generates the data.

- **Data Aggregators:** The users that aggregates and process the data for consumption either as is or as a service. For example, organizations interested in collecting data to build services on top, device manufacturers collecting data from sensors installed on their product.

- **Data Consumers:** The users that consumes the processed data. This includes users of services built on top of aggregated data, and users of individual data points.

Consider the illustration in Figure 1.2. The baby's parents are the data owners, the wearable device manufacturers act as data aggregators (use case 2) and the retail clothing store (use case 2) or the drug store (use case 1) are the data consumers.

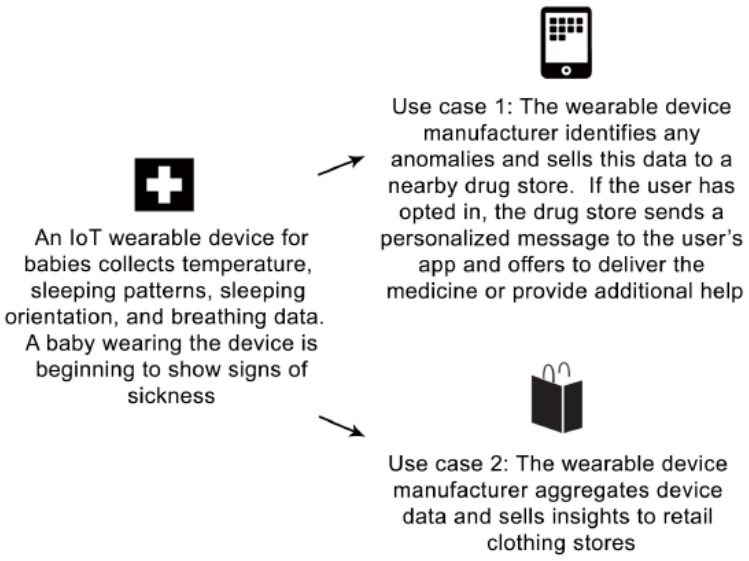

Figure 1.2: Data economy monetization example (taken from[70])

In recent years, there have been growing interest in building decentralized systems. Blockchain, has attracted a lot of attention across many different sectors for its ability to decentralize systems and remove the need for a middleman. Blockchain is a decentralized ledger replicated across participating nodes in a peer-to-peer (P2P) network. It is secured by strong cryptographic algorithms ensuring that no one node has full control over what is written into the ledger. The native integration of cryptocurrency has additional benefits that P2P transactions can be done cheaper, faster and in a secure privacy-preserving manner. The nature of blockchain positions it to be able to address problems in data management and there have been a lot of research to answers the daunting questions in the field.

1.2 Thesis Structure

The next chapter defines the problem area focused on in this thesis, and the research questions and objectives. Chapter 3, provides a general review of blockchain technology and existing works in blockchain-based approach to data management; It concludes with a highlight of questions not yet adequately answered in the literature. These questions would form the focus of the work in this thesis. Chapter 4 describes the proposed architecture. Chapter 5 covers details of the implementation of the architecture while Chapter 6 is dedicated to the experiments and evaluation of the proposed architecture to determine whether the research questions are addressed. The thesis is concluded in Chapter 7 with an outline of the contributions and future works.

Chapter 2

Problem Definition

The two main approaches for data monetization are:

1. **Centralized Model:** In this model, devices manufacturers collect and aggregates data from their devices, or service providers collects and aggregate data from using their platform e.g. Facebook, LinkedIn. For example, smart cars anonymously collect and send data about the car behavior [26]. Figure 2.1 illustrates the model.

Figure 2.1: Illustration of the centralized model

There a couple of challenges with this model [76]:

- Users gets little value in terms of monetary benefits from providing their data. (although they benefit from using the service or platform) [5].
- Users most times do not have control over what is collected about them or done with the data. A case in point is Facebook conducting experiments on user's data without permission [40].
- There are several privacy and security concerns as this data collected into huge data silos become targets by hackers [37] and government entities [44] that want to access or expose users' personal data.

This model characterizes most of the Internet today. This has prompted efforts into re-decentralizing the web. For example, Tim Berners-Lee, known as the creator of the world wide web envisions a better web where users have more control over their data - where it is stored and how it's accessed [35].

2. **Isolated Model:** In this model (Figure 2.2), different users have access to independent data silos or storage. There are also disadvantages to this approach including:

   - Limited value from the data, as the most value from data is gotten after aggregation with those from similar data owners. Imagine each individual storing (and maybe analyzing) their individual phone logs, there likely is very little new insights that can be drawn from it.
   - Little or no monetary value from the data.

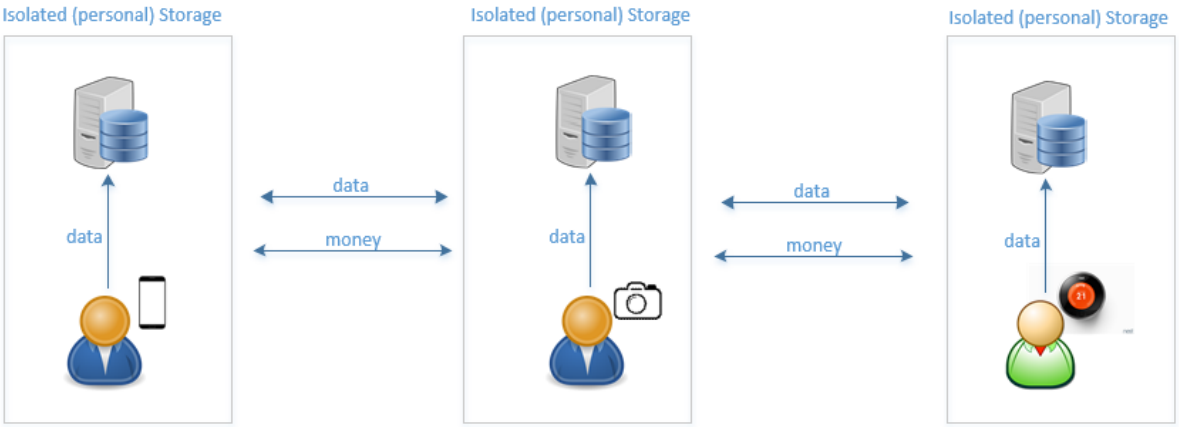

Figure 2.2: Illustration of the isolated model

To overcome the challenges of these two approaches, there is need for a hybrid model that leverages the best of both models:

- Data owners can selectively share data and profit from doing so.
- Data consumers can consume data individually and pay the data owners.
- Data aggregators can gather data from many sources and process it to produce new services.

For example, mobile applications such as Google Opinion Reward on Google Play, and Survey.com allows users to profit from answering questionnaire surveys selectively presented to them [76]. This model rewards data owners for providing their data.

2.1 Research Questions

Several works have already been done towards designs of systems for data monetization, that tries to achieve the above hybrid model. The literature review identified a couple of weaknesses in existing approaches, which would be investigated and addressed in this work. Based on the review, the focus in this thesis would be on answering the following research questions:

1. How can resources' (data and web services) metadata stored on the blockchain in a decentralized blockchain architecture be efficiently queried?
2. How can resources' (data and web services) metadata stored on the blockchain in a decentralized blockchain architecture be efficiently retrieved?

2.2 Research Objectives

The following goals would guide in answering the research question:

1. Conduct an in-depth literature review of blockchain technology and how it can be used to facilitate trusted data sharing among data owners.
2. Propose and design a decentralized architecture based on blockchain technology for publishing, discovery and exchange of data and web services for cryptocurrency.
3. Implement a proof-of-concept prototype of the proposed architecture.
4. Evaluate the performance and scalability of the architecture.

Chapter 3

Literature Review

To set the stage for the discussion of blockchain technology in later sections, this chapter begins with a detailed study of blockchain including its components, consensus mechanisms, features, applications, and challenges. It also discusses Bitcoin, the blockchain underlying the popular bitcoin cryptocurrency, MultiChain, a fork of Bitcoin for deploying private blockchains, BigChainDB, an attempt at leveraging the properties of traditional database along side those of blockchain and Hyperledger fabric, a scalable blockchain platform. Next, a detailed review of available literature on data monetization systems and related areas involving blockchain technology is provided. Finally, there is a brief discussion on restful web services, which is the methodology used in developing the Cowry platform.

3.1 Blockchain

Blockchain is not a new technology but an innovative marriage of ideas from well-established fields such as public key cryptography [61], distributed consensus [36] and peer-to-peer networking [60]. Blockchain is essentially a chain of blocks all of which are maintained on participating nodes, which do not fully trust each other, in a P2P network. Each block contains an ordered list of events (called transactions) mutually agreed upon by all nodes in the network. The "chain" results from each block referencing the cryptographic hash of the previous block; the first block, called the genesis block does not reference any block. A distributed consensus algorithm ensures the nodes agree on the block's content through a process called Mining.

3.1.1 Blockchain Ledger

Blockchain ledger is a distributed data structure comprising of "blocks" linked together to form a chain. It was introduced with Bitcoin to solve the fundamental problem of distributed digital currency - double spending [6] which is previously trivially solved using a central authority. Blockchain removes the need for a central trusted entity like a bank since all participants would have the full record of all transactions - according to Satoshi Nakamoto [66], "the only way to confirm the absence of a transaction is to be aware of all transactions". Figure 3.1 shows an illustration of a sequence of blocks forming a blockchain.

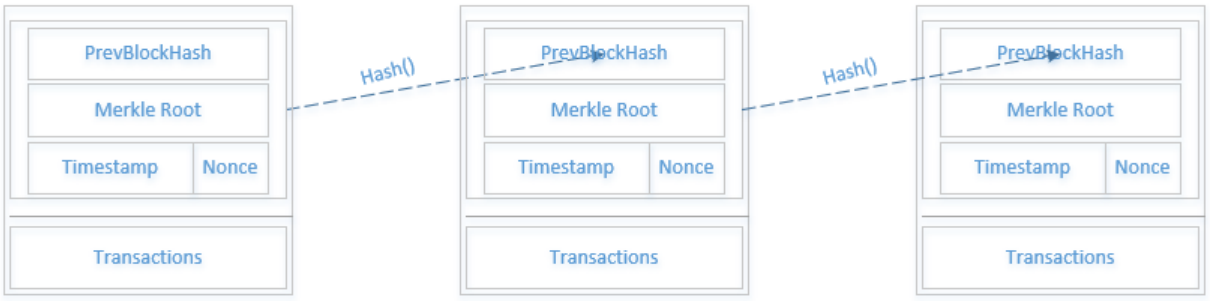

Figure 3.1: Sequence of blocks in a blockchain

Blocks

The structure of blockchain can be described as similar to a linked list with the nodes in the list representing the blocks. Each block has a header and a body. The cryptographic hash of its header identifies the block. The header contains a version number to indicate the rules used to verify the validity of the block, the hash of the previous block header (this is what "chains" the blocks together), the root of the Merkle tree [59], which is a hash of all the transactions in the block (see Figure 3.2), the current Unix timestamp, and a nonce [7]. The body of the block contains the number of transactions and a list of the transactions.

Transactions

Transactions are instructions that assigns ownership right for an amount of digital resource from the current owner signing the transaction to the new owner specified in the transaction. The transaction is signed by the private key of the sender and can be verified by the receiver using the sender's public key. The format of a transaction depends on the blockchain network; however, in general it includes the sender and recipient address, data payload and the amount. After transactions are created by participating nodes, they are propagated through the network in a P2P manner and verified by each node before propagating further. Figure 3.3 illustrates the digital signing of a transaction.

Transactions can also be used to store arbitrary data on the blockchain. This feature has been exploited in Bitcoin for different use-cases such as notarizing the existence of a document [50] or as a permanent decentralized data store [22, 32].

3.1.2 Blockchain Network

The blockchain network is a decentralized P2P network (Figure 3.4) where each user or participants interacts with the network via their node. To join the network, a compatible blockchain client is installed on the node. The network is decentralized because there is no central server and continues to function even if some nodes leave the network. Data from each node are validated by the receiving node and then forwarded to other

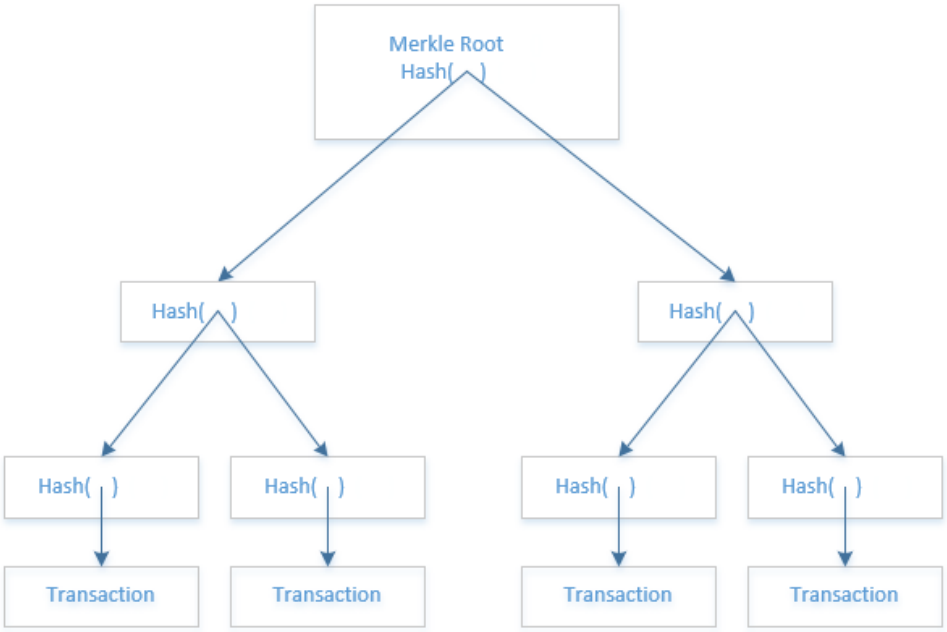

Figure 3.2: Merkle tree

nodes it is connected to thus a node can receive multiple copies of the same data.

3.1.3 Distributed Consensus

A consensus mechanism is required in every distributed system for all the nodes to agree on the state of the network. Different consensus algorithms have their advantages and limitations which overall determines many of the properties (for example throughput and latency) of the blockchain network.

The original consensus mechanism used with blockchain (from Bitcoin) is called Proof-of-Work (PoW) [9] discussed later. Some of the alternatives to PoW are:

- **Proof-of-Stake (PoS):** PoS [8, 49] is an energy efficient alternative to PoW where instead of measuring miners by computational power as in PoW, compares the amount of cryptocurrency a miner holds.
- **Delegated Proof-of-Stake (DPoS):** DPoS [52], similar to PoS, but for the difference in how the miners are selected.
- **Practical Byzantine Fault Tolerance (PBFT):** PBFT [13] gives a solution to the Byzantine Generals Problem [51]. It is one of the consensus algorithm available for use in Hyperledger Fabric [10] blockchain.

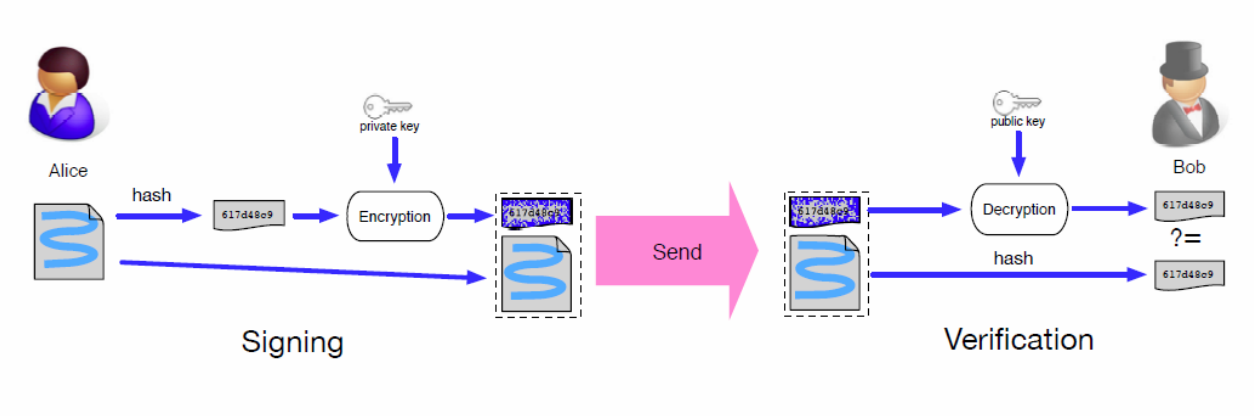

Figure 3.3: Digital signing of transaction (taken from [98])

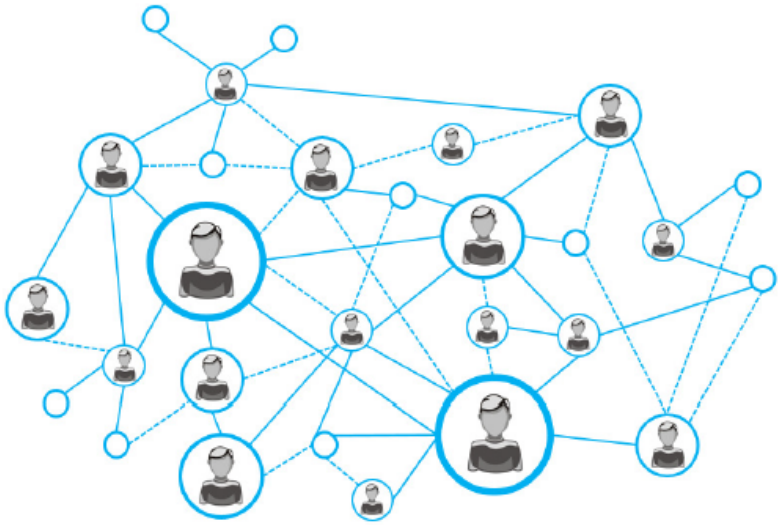

Figure 3.4: Decentralized network (taken from [98])

- **ScalableBFT:** This is an offshoot of Tangaroa protocol [23] which is an implementation of Raft consensus algorithm [69] with byzantine fault tolerance. It is the consensus protocol used in Kadena [56] blockchain.

Zheng et al. [98] gave a comparison of different approaches and more details can be found in [11, 15].

Proof-of-Work (PoW)

PoW [9] is Bitcoin's approach to consensus. It requires that each node solves a computationally difficult mathematical puzzle and grants the winning node the privilege to determine the next state of the network by appending its block to the chain. The following outlines how PoW is used in the Bitcoin blockchain:

1. Miners compute the hash value of the block header, each time changing the nonce (inserted in the block

header) until it is lower than or equal to a target (a 256-bit number).

2. The successful miners send its block to the other nodes.
3. The other nodes receive and verifies the proof of work by checking that the header's hash is correct.
4. The other nodes validate the transactions in the block before accepting it into their copy of the blockchain ledger.

The primary reason for PoW is to prevent Sybil attack [15, 93] which is possible in a public blockchain where anyone can join. In a Sybil attack [30], a single entity joins the distributed network using multiple identities. The underlying assumption of PoW is that owning the majority of the computational power is much more difficult than owning the majority of identities [93]. Without POW, a minority can easily have control over the state of the network.

Both Ethereum [94] and Bitcoin [66] uses the proof of work consensus protocol, however Bitcoin uses a variation of the protocol called "Nakamoto Consensus" [16], that uses SHA-256 cryptographic primitives [31] for its hashing function whereas Ethereum uses Ethash [33] which uses a SHA3 crytographic primitive for its hashing function. Nevertheless, there are other hashing algorithms that can be used for PoW such as Blake-256 [1] and Scrypt [74].

3.1.4 Cryptocurrency

Cryptocurrency is the first successful class of application built on top of the blockchain. It is a digital currency based on a P2P network and cryptographic tools [97]. The Bitcoin blockchain cryptocurrency, bitcoin is the first decentralized cryptocurrency and as of July 2017 has a capital of over 45 billion USD [20]. After bitcoin, many other cryptocurrencies have been developed. Figure 3.5 shows the percentage of total market capitalization of the top eight cryptocurrencies as of July 2017.

Cryptocurrency also serves as an incentive for the miners in a public blockchain. For example, in Bitcoin, the miners of the blocks are rewarded with freshly minted 12.5 bitcoins as of July 2017; this amount reduces by half every 210,000 blocks (approximately every 4 years) from its original value of 50 bitcoins in the beginning. Also, miners get to charge a small transaction fee in cryptocurrency for adding a transaction to the block.

3.1.5 Types of Blockchain

One of the ways blockchain have been categorized in literature is by which group of people can have access to the network. A public or permission-less blockchain places no restriction on who can connect provided their node is running a compatible client. A private or permissioned blockchain limits the participants to a set of white-listed users; some literature [98] defines a third type called consortium blockchain, which is a permissioned blockchain where the participants are part of a consortium rather than a single organization. Bitcoin and Ethereum are the most popular example of a public, permission-less blockchain. Examples of

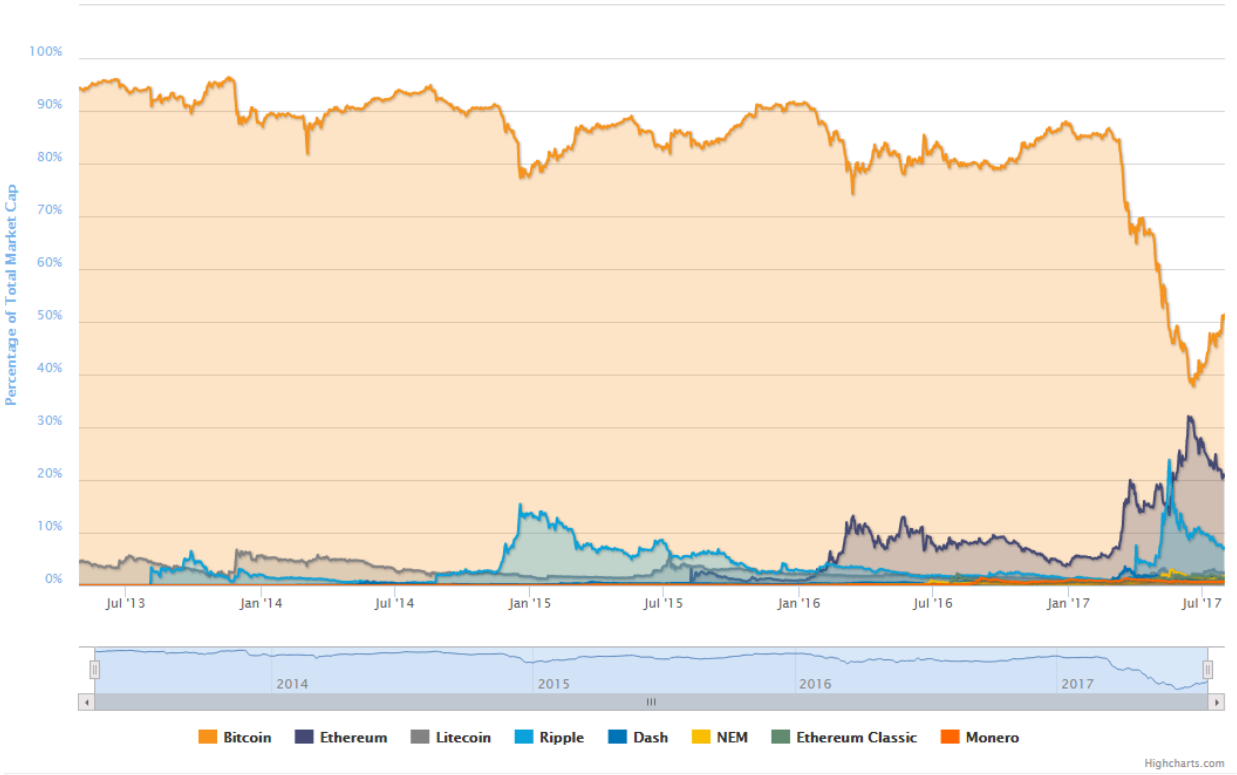

Figure 3.5: Cryptocurrency market capitalization (taken from [21])

platforms for deploying private or consortium blockchains are Eris [64], MultiChain [17], and Hyperledger fabric [55].

3.1.6 Blockchain as a Database

Many research [3, 78] have described the blockchain as a type of database - in particular as a distributed database - a shared ledger that consists of transactions. According to Dinh et al. [28], blockchain's approach of agreeing on the order of execution of transactions and replicating all the data on all nodes can be viewed as an example of a solution to the distributed transaction management problems in a database. Nevertheless, blockchains are quite different from traditional distributed database systems. For example, according to Tai et al. [90], while traditional distributed database may compromise consistency to gain availability (CAP theorem [38]), blockchain compromises scalability to gain trustless interactions; thus, in terms of performance, a blockchain will always be slower. Again, in term of the CAP theorem, blockchain gives up consistency for availability and partition tolerance and has been said to be "eventually consistent" [3] meaning consistent after the block is confirmed. Also, when compared to a traditional database, blockchain has a lower cost in terms of infrastructure and personnel [83].

Some other differences between blockchain and traditional databases were discussed in [41] and are summarized below:

- Blockchain enables non-trusting parties to share write-access to a database without the need for a central administrator. This is very significant because apart from introducing a whole new field of possible applications (for example P2P transactions as in Bitcoin), it also reduces the investment in infrastructure and personnel normally required to secure and maintain databases controlled by trusted authorities.

- One of blockchain features is transparency. Unfortunately, this comes at the expense of confidentiality. Traditional databases (SQL or NoSQL) provides confidentiality by restricting both read and write access to be managed centrally by an administrator.

- Blockchain database is more robust and fault tolerant than traditional database. It can be said that blockchain compromises performance for robustness because by replicating every transaction on all the nodes and densely connecting nodes together, the failure of some nodes does not affect the blockchain database. This is not the case with traditional databases, where to achieve close to the same level of robustness, expensive high-ends nodes and backups need to be put in place.

Nevertheless, despite their difference, both type of database has major roles to play and proper understanding of their characteristics is important to use the right kind for the right purpose and to combine them where necessary.

3.2 Bitcoin

Bitcoin [66] proposed in 2008 and implemented in 2009 was developed to address the problem of double spending in electronic payments. Traditional payment system suffered from many challenges: reversible transactions, high transaction cost, and lack of anonymity. All these challenges were due to the need to minimize fraud from double spending by using a trusted third party such as Banks. The idea of Bitcoin was to use cryptographic proof to replace trust in electronic payment. Blockchain is the core technology underlying Bitcoin. In fact, it is the first application of the blockchain technology and much of blockchain's popularity can be traced to it [28]. Many current blockchain implementations is derived from the one underlying Bitcoin [43].

The Bitcoin blockchain is a public (or permission-less) blockchain, meaning it has no restriction on who can connect and participate in the mining activities as long the node is running a compatible client. The current block size of the Bitcoin blockchain is 1Mb and it can handle about 7 transactions per seconds with a block formed every 10 minutes on average. As discussed before, Bitcoin uses PoW to achieve distributed consensus. It incentivizes its miners using transactions fees and mining rewards.

3.2.1 Transaction Model

Bitcoin transactions consists of a list of inputs and outputs. It uses the unspent transaction output (UTXO) model [27]. In this model, newly created (minted) bitcoins are UTXOs with the miners as owners having the right to spend it. The bitcoin is spent when they become the input of a transaction, and they create new UTXOs with new owners. Bitcoin uses a script to describe the output of a transaction. The script uses a stack-based programming language and is not Turing complete. This is illustrated in Figure 3.6.

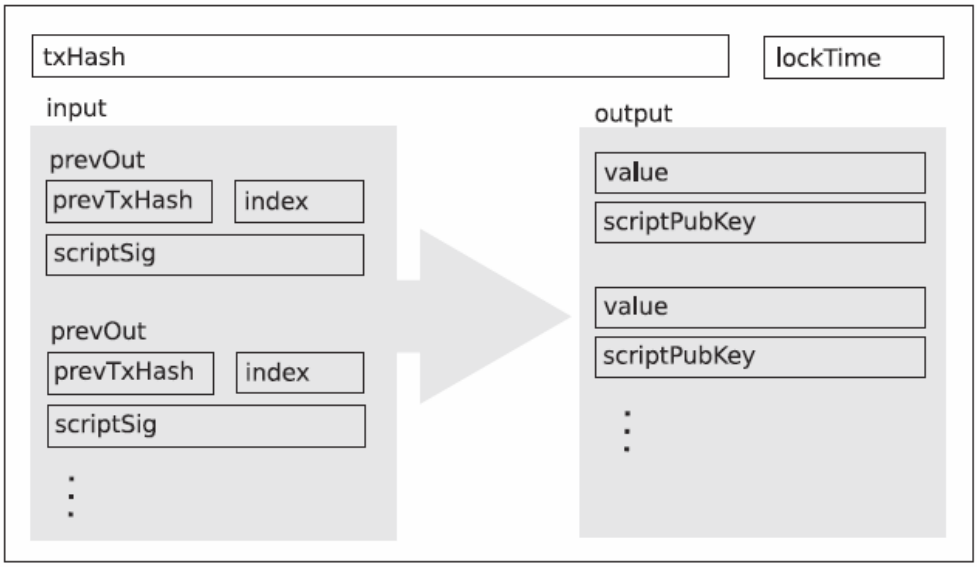

Figure 3.6: Sample bitcoin transaction showing output script (taken from [93])

3.2.2 Smart Contract

Bitcoin's script is not very flexible due to the limit of the scripting language. However, the need to build more complex scripts resting on the blockchain led to the development of new kinds of blockchain with Turing-complete scripting language. Smart Contract as they are called was originally proposed in early 1990s by Nick Szabo [89]. They are computer programs residing on the blockchain allowing interacting parties to technically enforce an agreement written with code that runs on every node on the blockchain network. It is analogous to a stored procedure in relational databases [65]. A number of high level programming languages such as Python and Solidity [82] can be used to write a smart contract. Since it resides on the blockchain, it has an address and is triggered by sending transaction to that address. The most popular example of smart contract styled blockchain is Ethereum.

3.3 MultiChain

MultiChain [17] is a platform for creating and deploying private blockchains. The motivation for the development of MultiChain was to solve some of the problems identified with the use of Bitcoin, from which it was derived, for institutional financial transactions. As a platform for private blockchain, it introduces features that ensures only permitted nodes can participate in the network activities including connecting, mining, and sending or receiving transactions. These permissions are configurable during the setup of the network and includes permissions such as anyone-can-connect, which indicates if anyone can connect to the network (as in a public blockchain) and anyone-can-mine, indicating if anyone can take turn to append blocks to the blockchain. The MultiChain permissions documentation [18] provides more details on these and other supported permissions. MultiChain differs from the Bitcoin in a number of ways including different consensus mechanism (MultiChain uses a scheme called Mining Diversity), direct support for third party assets and support for database-like feature via MultiChain Streams.

3.3.1 Mining Diversity

MultiChain uses a round-robin scheme to determine which permitted miner can append blocks to the chain. The idea is to limit the number of blocks that a single miner can append within a given window using a network parameter called mining diversity which can be set to a value between 0 and 1 (inclusive) during the blockchain setup. Nodes only attempt to mine if they have not mined any one of the $spacing - 1$ previous blocks where $spacing$ is calculated by $\#permitted\_miners * mining\,diversity$ else their blocks will be invalid.

3.3.2 Assets

Although a fork of Bitcoin core, MultiChain goes a step further by allowing arbitrary third party tokenized assets (virtual tokens representing real world assets) to be created and exchanged on the Blockchain. It enforces the same or higher level of cryptographic security for the transfer of these assets. One use of this feature is to create application-specific cryptocurrencies different from the native one provided by the platform.

3.3.3 Streams

Another feature MultiChain support is called Streams. A MultiChain Stream [42] is an append-only collection of items, implemented underneath as blockchain transactions. This abstraction allows the blockchain to be used for data retrieval and archival. Each item in a stream has 4 fields namely publisher(s), key, data, and timestamp. The key field allows data to be stored and retrieved like in a key-value database, the timestamp enables stored data to be retrieved in time order and the publisher field categorizes the items by their authors for retrieval. By this implementation, a Stream allows three types of databases on top the blockchain:

1. Key-value database or Document store

2. Time series database

3. Identity driven database

3.3.4 Oracle

Bitcoin and MultiChain supports two runtime parameters: blocknotify and walletnotify, that allows external scripts to run in response to some transaction activity on the blockchain [19]. This external script can be seen as an Oracle. An oracle is anything that is used to connect the blockchain to the off-chain world. In our use-case, the oracle is a script that is triggered when certain transaction activities occur on the blockchain. However, the use of oracle goes beyond this use case of responding to a transaction. It is also very important in retrieving input from the off-chain world. Oraclize [71] is an example of a platform that provides this service.

3.4 Other Blockchain Platforms

Some of the other platforms reviewed during to this research are: BigChainDB and Hyperledger fabric.

3.4.1 BigChainDB

BigChainDB [57] is an attempt to address some of the scalability problems of blockchain by implementing blockchain features on top of a traditional distributed database. The creators described it as "big-data database with blockchain characteristics". Typically, attempts at improving a blockchain platform, for example Bitcoin, either starts by building a new one from the scratch or by tweaking existing ones to gain database-like features and performance. BigChainDB takes the opposite approach by building blockchain features including decentralization, immutability, and native support for digital assets into existing big-data database. BigChainDB currently uses either MongoDB or RethinkDB as the database backend.

3.4.2 Hyperledger fabric

The Hyperledger fabric architecture is different from the traditional blockchain architecture described so far. In the traditional architecture, all nodes are required to execute every transaction, maintain a copy of the blockchain ledger and participate in the consensus process. However in Hyperledger fabric, the nodes are decoupled into two separate runtimes with three distinct roles: Endorser, Committer, and Consenter [55].

3.5 Characteristics, Applications & Challenges of Blockchain

3.5.1 Characteristics

One of the primary feature of the blockchain is its ability to allow direct interaction between non-trusting parties, removing the need for a trusted authority. This feature provides a number of additional benefits including lower costs (no need for expensive central servers and backups) and redundancy. Other characteristics of blockchain as discussed in the literature [15, 98] are:

- **Persistency**: Once the transaction has been recorded on the blockchain, it cannot be easily modified or falsified.
- **Anonymity**: The real identity of the user is not revealed during interactions on the blockchain as only a generated address is used for the transactions. Although, this is not a guarantee of perfect anonymity as a careful analyst can make connections between addresses and may be able to infer the user's real identity from such connections [58, 81].
- **Fault Tolerance**: The blockchain ledger is replicated across all nodes, hence providing redundancy even if any node fails or leaves the network.
- **Transparency**: Every participant of the network sees the same state of the transactions recorded on the blockchain.
- **Traceability**: Traceability is an extension of the persistency and transparency characteristics. Every transaction can be audited back till the first transaction.

3.5.2 Applications

The characteristics of the blockchain have attracted many people in industry and research to develop blockchain-based applications across different domains. The following discusses some of such applications categorized by their domains:

- **Healthcare:** The applications in healthcare are mostly centered around protecting access to medical data using the blockchain. In [2], Azaria et al. proposed and developed a prototype for a decentralized record management system for providing authentication, confidentiality, accountability and sharing of electronic medical records using blockchain technology. In the industry, blockchain-based record keeping service, Factom stores medical services data for example medical bills on the blockchain [88].
- **Education:** In education, blockchain has been used as a persistent record of intellectual achievements including records of education achievements & credits, ideas, and creative works [84]. Other institutions are also beginning to adopt blockchain in this way [29]. Also in [4], Bartling and Fecher organized many

proposed applications of blockchain around the stages of research from documentation of an idea to evaluation, publication and obtaining research funding. The authors [4] proposed that research data could be stored in a blockchain database after acquisition and could be made available initially to specific researchers and later to others. Also, Gipp et al. [39] proposed a system that uses the Bitcoin blockchain to create a tamper-proof timestamp for manuscripts submitted to a conference or journal. This publicly verifiable timestamp acts as proof of existence of the research manuscript at a point in time.

- **Finance:** Finance is the application area that seems to have received the most attention and a lot have been documented on how blockchain could be used in this domain [78, 45].

Applications in some other domains includes IoT [62], Security [85], Infrastructure [91], and Government [46].

3.5.3 Challenges

Despite the potentials of blockchain technology, it has some problems usually associated with the public blockchain. Two main challenges found in the literature are:

- **Scalability and Performance:** This is considered one of the main criticism for public blockchains in the literature especially by developers of private blockchain platforms. Bitcoin for example can only process an average of seven transactions per seconds. Other public blockchain platform have similar performance limits especially when compared to traditional databases. Also, blockchains does not scale well since each additional node still replicate and process the same transactions.

- **Data and User Privacy:** Since all the data on the blockchain are visible to all the participants, it does not support data privacy by default. This challenge is a major drawback especially in finance and legal use cases where data privacy is very crucial. Also, although information about the owner of a transaction is not revealed in the blockchain address, this does not guarantee complete user anonymity.

In addition, the immutability of smart contracts makes any error in coding it very dangerous as was brought to light by the DAO hack [86] on the Ethereum blockchain. With smart contract promising enforcing real-world contracts on the blockchain, there is the question of whether such contracts would also be legally binding by default. Other challenges discussed by Zheng et al. [98] include vulnerability to selfish miners, weakness of current consensus mechanisms and miners' centralization.

3.6 Data Monetization Solutions

Recall in Chapter 2, we identified a number of challenges with both paradigms of data monetization, seeking one that integrates the best features from both paradigm and hopefully discard the weaknesses. One model

that is directed towards achieving this integration is the Sensing-as-a-Service (S^2aaS) model, which promotes the exchange of sensor data between data owners and data consumers. According to Mišura and Žagar [63], the idea is for large number of users to have efficient access to data provided by sensors via the Internet. Some work has been done in this area [75, 77], however, we would focus on the literature concerning design and/or implementation of the model so as to remain within the design and implementation scope of this work.

Mišura and Žagar [63] described a model for a centralized data marketplace for IoT data. They envisioned a cloud service that allows sensor owners register their devices with relevant information about the sensor and the data it collects, and data consumers to query the system based on their requirement. They argued that such a marketplace differs from other general data marketplace such as Microsoft's Azure data market for a number of reasons:

- IoT data owners are small-time compared to large organizations that might want to sell data on a traditional data marketplace.
- The data consumers for IoT data would typically require data from multiple data owners.
- IoT data is usually required in real-time compared to the batch archive of traditional datasets, for example as implemented by Xu et al. [97].
- IoT data is usually paid for in advance and contains more sensitive information.

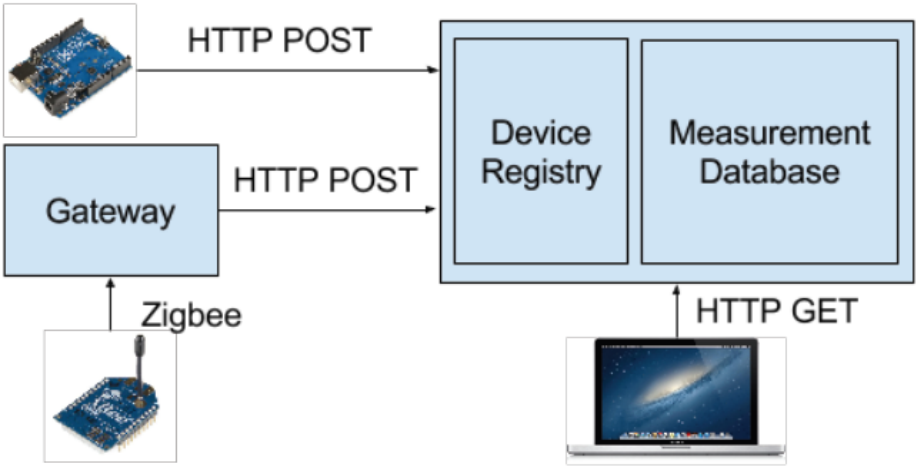

Figure 3.7: System architecture for centralized IoT data marketplace (taken from [63])

Their platform architecture addresses publishing and discovery using a centralized device registry (Figure 3.7). Data from the sensor owners are first stored in a central measurement database to provide efficient delivery to multiple consumers as well as caching. Both the sensor owners and consumers interact with the

platform using HTTP requests. They also reported on a system for ensuring data providers remain honest by evaluating the number of completed measurements divided by the number of agreed measurements. This measure is visible to the data consumers. Details of how the monetization would be implemented was not provided. Also, since their implementation is centralized, it introduces security and privacy issues.

Robert et al. [80] proposed a generic framework for data monetization also focusing on IoT data. The authors discussed considerations necessary for the design of a framework based on a P2P architecture. They outlined a number of key requirements that such a platform should have including enabling information publication and discovery, secure money transactions, encouraging competitive pricing, using open standard-based platform, open market, and incentive for data sharing. They also analyzed some existing state of the art platforms such as Placemeter [79] and Thingful [92] etc. and concluded that none of them meets all the requirements. While the authors identified the key features that would enable the integration of both paradigms mentioned in Chapter 2, what is lacking is a concrete design, implementation, and evaluation of such framework.

Noyen et al. [67] proposed Bitcoin as a protocol for S^2aaS. The authors identified three challenges in this space: sensor identification (uniquely identifying and authenticating sensor owners), sensor data provenance (tracing sensor data and securing from manipulations) and low-cost micropayment (incentives for sensor owners to share data). They also identified some characteristics of the Bitcoin blockchain protocol that made it suitable for S^2aaS applications including decentralization, pseudonymity, and low fees. A prototype of the idea was implemented by Wörner et al. [96].

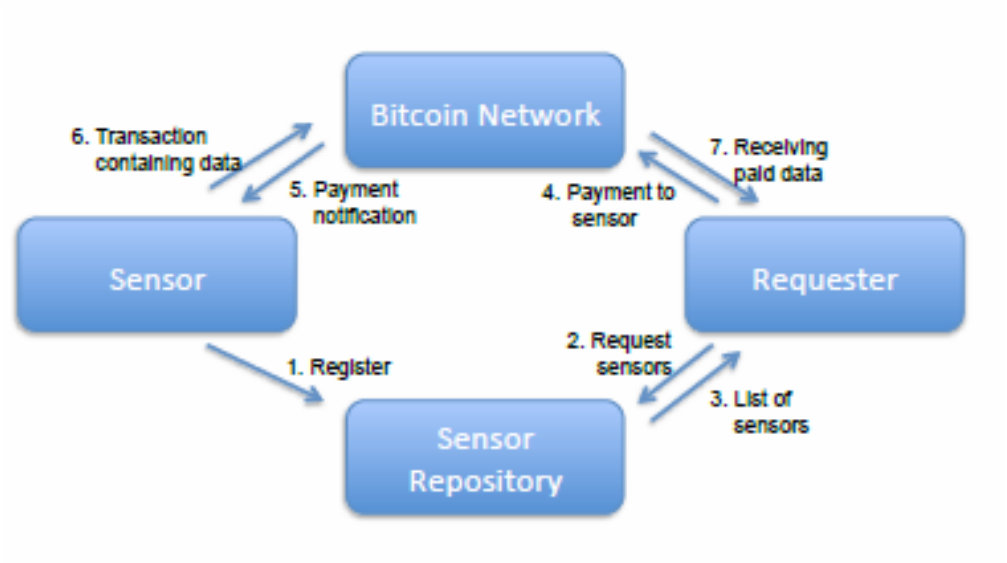

Figure 3.8: System architecture of exchanging data for cash with bitcoin (taken from [96])

The prototype consisted of three components (See Figure 3.8): a sensor client, sensor repository and a requester client. The sensor client was implemented as a web socket to know when a payment has been made to the sensor address and it responds by addressing and publishing a transaction to the blockchain

for the data consumer containing the requested data. The sensor repository was a centralized database with RESTful HTTP API and web interface allowing the sensor requester to search for desired sensor datasets. The authors identified a number of challenges with their implementation including that the sent data will be publicly readable by every participant on the blockchain and scaling issues as more data is exchanged and stored on the blockchain. The authors however did not show results of evaluating their platform.

Similarly, Dominic Wörner [95] developed a prototype for a decentralized market place for the exchange of data based on the 21 Bitcoin computer [87]. Their motivation was to "free" sensor's data which they argued is "trapped in application-specific environments" by providing financial incentives to share data. Their implementation also provided means of discovery using a centralized sensor registry based on a MongoDB database and they did not show any result of performance evaluation of their prototype. It is clear that many of the proposed model focused on S^2aaS business model. This needs to be generalized to all sorts of data or services. For example, a sensor may detect that an individual fell but not that the individual was mentally distracted or lighted headed from drinking, such additional information can come from surveys and questionnaires submitted via a web service.

Xu et al. [97] discussed a prototype of a platform for data monetization using smart contracts. They considered two scenarios: one where the data owner publishes their data to the platform and the data consumer browses for and select desired data set and the other, where the data consumers first pushes their jobs to the platform and the data owner can select jobs to provide data for. In both scenarios, the data owner is compensated for their data. The platform addresses the requirement of publishing and discovery using smart contracts, for example a dataset is registered by calling a dataset registry contract which stores description of the dataset along with a hash of the data; micro-payment infrastructures provided by the underlying blockchain cryptocurrency and provenance data is written for every event to the blockchain. The actual data is stored in an offchain storage platform due to the size. The authors also pointed out the need for a reputation and rating mechanism to ensure that the data owners remain honest especially in describing their dataset. The fact that the data is centrally stored introduces the possibility of surveillance and data breaches. In addition, a clear idea of how it would be implemented, including evaluation was not provided.

3.7 Other Related works

The section discusses the literature on the general area of data management on the blockchain including data access control, privacy, and provenance. The is important because blockchain-based approach taken in these application areas are very similar and the ideas are transferable.

3.7.1 Access Control

Researchers have sought to address data privacy and security problems using blockchain [73, 99]. One approach is by using the blockchain as an access control mechanism. Access control allows restrictions to

be placed on what can be done by a computer user or program. There are three traditional access control models: Mandatory Access Control (MAC), Role-based Access Control (RBAC) and Discretionary Access Control (DAC). Uurtsaikh Jamsrandorj [47] discussed these different access control models and then proposed a decentralized access control using blockchain. In [99], Zyskind et al. proposed a decentralized personal data management system to address the problem of organizations collecting and controlling huge amount of user's personal data. The authors argued that this model have resulted in many breaches in security and increase in surveillance. They developed a platform that makes use of the blockchain acting as an access control manager to data stored in an off-blockchain storage enabling the owner to grant or revoke access to the data at will. Figure 3.9 shows an overview of their platform.

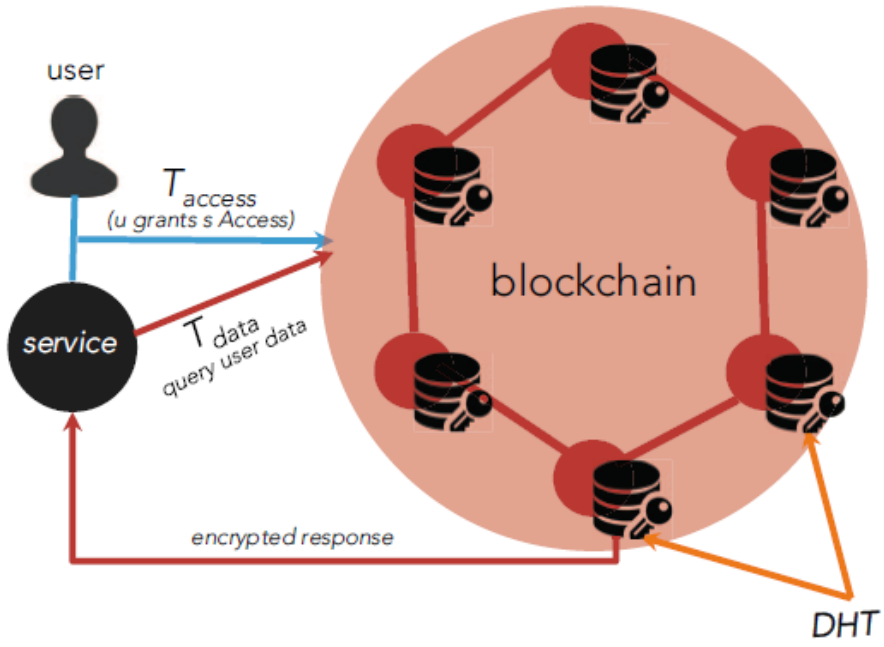

Figure 3.9: Overview of blockchain platform as an access control manager (taken from [99])

Essentially, the blockchain transactions carry instructions for storing, querying, and sharing the data. The data owner uses transactions to set permissions which are stored on the blockchain for accessing the data. The data owner or consumer uses transactions to query the data via the blockchain which first verifies if they have the appropriate permission to access the data. The authors affirm that the platform ensures the user remains owners of their data, has complete view of what data is collected and can determine who has access to their data. They also identified certain weaknesses of their platform such as inefficiency in data processing and that the platform does not prevent data consumers from retaining the data (and running other analysis) on it after the first query.

3.7.2 Provenance

Provenance is the record of the history of a data object from creation. It has several benefits including data auditability and is very relevant in many domains. Blockchain provides a very natural infrastructure for storing provenance data because of its features of persistency, transparency, and resistance to data manipulation. Liang et al. [54] proposed a decentralized and trusted architecture for storing provenance of cloud data objects on the blockchain. They argued that with the added features the blockchain provides, the provenance data would not be vulnerable to accidental or deliberate corruption which according to them is a problem with the current state-of-the-art cloud based provenance services.

Their platform, ProvChain (see Figure 3.10) supports provenance data collection, storage and validation while also guaranteeing the cloud user privacy and low overhead for the cloud storage application.

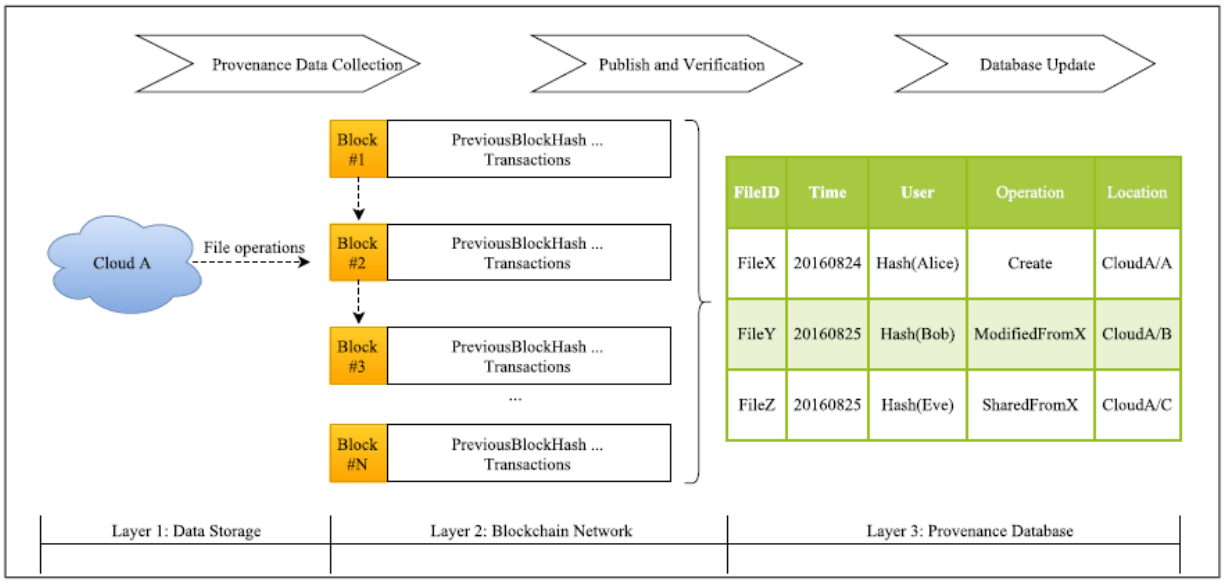

Figure 3.10: System architecture of ProvChain (taken from [54])

It works by monitoring subscribed user's operations and collecting provenance data, which is stored in a local provenance database, and the data hash published onto the blockchain network. The user can request the data in the provenance database be validated by a provenance validator. The validator queries the blockchain network for the details of the transaction holding the provenance data and stores that also in the provenance database, thus validating the provenance data. Only the hash of the user id is stored to protect the user's privacy.

3.8 RESTful Web Services

The Representational State Transfer (REST) is a set of design principles introduced in 2000 by Roy Fielding as part of his doctoral dissertation [34]. REST is not a strict standard but an architectural style. It describes the following six constraints [34]:

1. **Client-Server:** The principle behind this constraint is to separate the concerns of the clients from that of the server. For example the client does not have to know about the server's database and the server does not need to worry about the client's interface. This allows the clients and server to evolve independently and it improves scalability of the server and portability of the user interface.

2. **Stateless:** This constraint defines that the web-server should not be required to retain the state of the client application. Hence, every request should have all the information required to service the request. This helps increase the server scalability since it does not have to maintain states of each of its clients; it also simplifies implementation because it does not have to manage resource usage across requests. The Stateless constraint however causes repetitive data to be sent to the server every request, thus decreasing the network performance.

3. **Cache:** This defines that the server specifies if a response is cacheable or non-cacheable allowing the client to know if it can reuse the cached data in later requests or make a fresh request. This constraint improves the average latency over a number of interactions.

4. **Uniform Interface:** REST emphasizes a uniform interface between components of the network. This constraints establish a communication contract between the client and server thus decoupling the services from the actual implementation. Fielding [34] identified the following architectural constraints required to obtain a uniform interface:

    - **Identification of resources:** A resource is any web-based entity that can be uniquely identified and manipulated via the uniform interface. A resource is uniquely identifiable by a URI (Uniform Resource Identifier) which contains the name and address of the resource [24].
    - **Manipulation of resources through representations:** A resource is decoupled from their representation hence it can be represented in different ways depending on the component interacting with it. Resources are manipulated via their representations.
    - **Self-descriptive messages:** The request/response messages must contain the information required by the recipient to interpret it.
    - **Hypermedia as the engine of application state:** A resource representation can include URIs to enable navigation between resources.

5. **Layered System:** This principle allows multi-layered architecture such that servers can be installed in between the client and other components to provide services such as caching, security, and load balancing. Each layer is only aware of the immediate layer they are interacting with. The trade off here is that it introduces additional overhead and latency.

6. **Code on Demand:** This constraint allows the client to extend their functionality by downloading and executing codes such as applets or scripts from the server.

A RESTful web services is a framework built on the REST architectural principles. It uses the HTTP protocol for communication and the corresponding HTTP methods to interact with the resources:

- **GET:** This method is used to retrieve a representation of a resource.
- **HEAD:** This method is like GET however, only the response headers is retrieved.
- **POST:** This is used to create new resource.
- **PUT:** This is used to update a resource.
- **PATCH:** This updates parts of a resource. The request only contains the changes to the made to the resource.
- **DELETE:** This removes the specified resource.

3.9 Summary

This chapter set out to give a thorough background into blockchain technology before directing attention to existing designs and implementations of data monetization platforms in the literature. The review focused on academic publications only but due to the rapid development in the area, may be missing some work only published in blog posts and whitepapers. Table 3.1 gives a list of topics, papers and the key findings that would be relevant to this thesis.

Topics	Papers	Findings
Blockchain	[66, 98, 15, 93]	Blockchain technology can be used to replace the traditional middleman in centralized architectures, ensuring that no one has full control over the state of the network.
Blockchain and Database	[28, 90, 3, 83, 41]	Although blockchain and traditional database both store data, they do so in very different ways which results in different performance and features. While blockchain tends towards data security features, database tends towards scalability and performance features.
MultiChain	[17]	MultiChain is an easy to use private blockchain platform with support for database-like feature via MultiChain Streams.
BigChainDB	[57]	It is possible to use traditional NoSQL databases such as MongoDB to provide additional features that might not be best supported using only a blockchain.
Data Monetization	[97, 95, 96, 67, 80, 63]	Most existing work are either centralized or only partially decentralized.

Table 3.1: Summary of literature review

The review revealed that most existing work stores the resources' metadata using a centralized database. This approach relies on the robust support those databases provides, which makes it easier for the metadata to be queried and retrieved. However, it also introduces all the weaknesses of centralized systems.

In this thesis, the focus would be on answering these research questions:

1. How can resources' (data and web services) metadata stored on the blockchain in a decentralized blockchain architecture be efficiently queried?

2. How can resources' (data and web services) metadata stored on the blockchain in a decentralized blockchain architecture be efficiently retrieved?

Querying and retrieving resources' metadata stored on the blockchain is not trivial because blockchain's indexing is not designed by default to support such operations on arbitrary data. It is rather indexed for retrieving data by the blockchain transaction information, and block numbers. The result of this is a blockchain architecture that is fully decentralized and still allows efficient retrieval of the metadata stored on it.

Chapter 4

Architecture

The goal of this thesis is to propose, design, implement and evaluate an architecture on top of blockchain technology for publishing, discovery and exchange of data and web services for cryptocurrency. The proposed system is called Cowry; it is a decentralized data and services monetization platform built on top of a blockchain providing users ability to trade their data and services. This chapter will discuss the architectural design of Cowry.

4.1 Design Objectives

The literature review provided a few guidelines that helped guide the design of the architecture. The architecture will have the following features:

- **Publication & Discovery:** A RESTful API to publish available data, services, and jobs, and to query them using one or more parameters such as *location*, resource *type* (e.g. sensors) and *price* etc.
- **Decentralization:** Each node owner hosts their own data. The node represents a trusted entity and multiple users can host their data on a trusted node. The information about the data and services are stored on the blockchain leveraging benefits of decentralization including resistance to censorship.
- **Privacy and Secure Transactions:** Ensure privacy of the participants. All verification of user identity is done off-chain and only the hash is stored on the blockchain. All transactions are carried out through the blockchain's native support for cryptocurrency which provides pseudo-anonymity and secure transactions. All private information, for example data exchanged via the platform are securely encrypted before writing to the blockchain ledger.
- **Provenance:** Provide a history of every interactions and transaction. This is done by leveraging blockchain's feature of immutable storage. Every interaction is recorded immutably on the blockchain.
- **Fairness:** A mechanism to ensure fairness, by transparently penalizing attempts to defraud other participants. The buyer gives a rating for each seller's reputation after every transaction.

These objectives constitute majority of the standards defined in the literature for designing a decentralized data marketplace [80].

4.2 Design Considerations

In the design of the Cowry platform, we made certain design choices for the architecture including how cryptocurrencies, keys selection, reputation mechanism was implemented.

4.2.1 Cryptocurrency

The platform uses it own cryptocurrency independent of that of the underlying blockchain native currency. The platform digital coin is called cowrie (plural cowries)[1]. It is traded for exchange of digital resources. Using a currency different from the native currency of the blockchain platform further helps to make the solution independent of the underlying blockchain platform.

4.2.2 Hashing & Encryption

Hashing and encryption are very important elements in the architecture. Because of the open nature of the blockchain, everything published on it is visible to all network participants. To avoid this, sensitive information such as the resource shared was encrypted before publishing it. The AES (Advanced Encryption Standard) algorithm (symmetric encryption) with a key size of 256 bits and CBC (Cipher Block Chaining) encryption mode was used for encryption, while SHA256 was used for hashing.

4.2.3 Keys

Every account holder requires three different sets of keys:

- **Account key, A_k:** This is a public/private key pair [61] generated by the underlying blockchain and used to sign every transaction made by the account. The account holder address on the blockchain is derived from the public key.
- **Encryption key, E_k:** This is a 32-bit symmetric key that is provided by the buyer each time a resource is to be shared. It is used to encrypt the transaction details as well as the resource before writing it to the blockchain. The hash of this key is also required to retrieve the resource from the blockchain after the transaction is complete.
- **Sharing key, S_k:** This is a public/private key pair used to secure the transaction before it is completed. The encryption key, E_k is encrypted with the S_k public key of the data owner so it can use the private key to decrypt it. There are two reasons for choosing to use a different key-pair for sharing data and for authenticating transactions:
    1. The account keys are built from Elliptic Curve Digital Signature Algorithm (ECDSA) [48] in MultiChain and their primary purpose is for digital signatures.

[1] A long time ago, cowries were used as currency in Africa.

2. Since the Account key, A_k is tied to the user account, it is more secure to support new encryption keys for each data exchange instead of reusing the account key for all transactions.

4.2.4 Reputation

To ensure fairness of transactions both in terms of prices and quality of data and services, Cowry includes a reputation mechanism similar to that suggested by Mišura and Žagar [63]. A user's reputation is a derived from the weighted average of votes from other users:

$$reputation = \frac{\sum(cost_i * rating_i)}{\sum cost_i}$$

where $cost_i$ is the cost of the ith transaction by the data owner, $rating_i$ is the rating of the ith transaction by the consumer. The denominator normalizes the result. The data consumer (buyer) rate the data owner (seller) at the end of the transaction.

4.3 Cowry Architecture

In Cowry, the blockchain infrastructure replaces the centralized middleman (Figure 4.1) with a decentralized P2P network of nodes working together to record and validate every transaction.

Figure 4.1: Centralized model

In the centralized model (Figure 4.1), P2P exchange must go through a trusted third party. However, it also includes big firms acting as aggregators of user data and trading with it. The proposed solution serves as a middleware providing operations on top of the blockchain to utilize the infrastructure for exchanging data and services for cryptocurrency. Figure 4.2 shows a high-level overview of our proposed architecture.

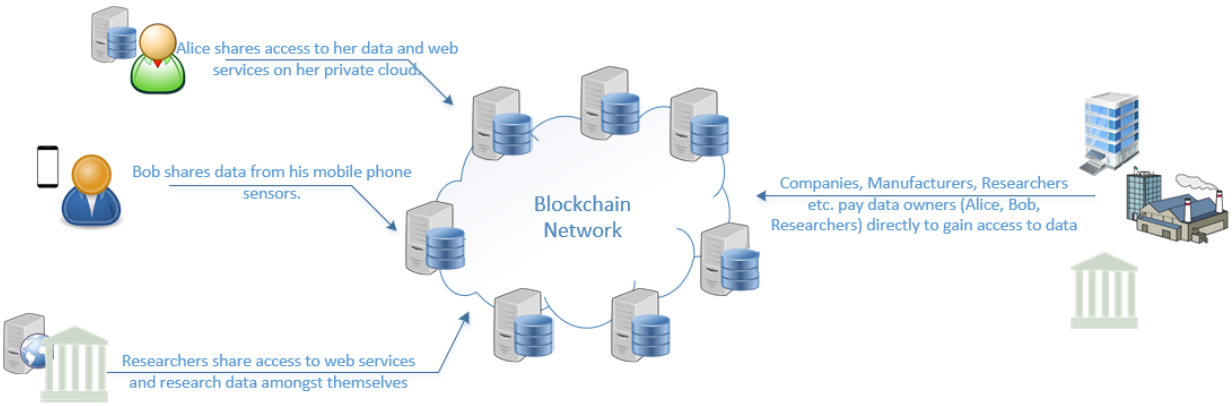

Figure 4.2: High-level overview of Cowry architecture

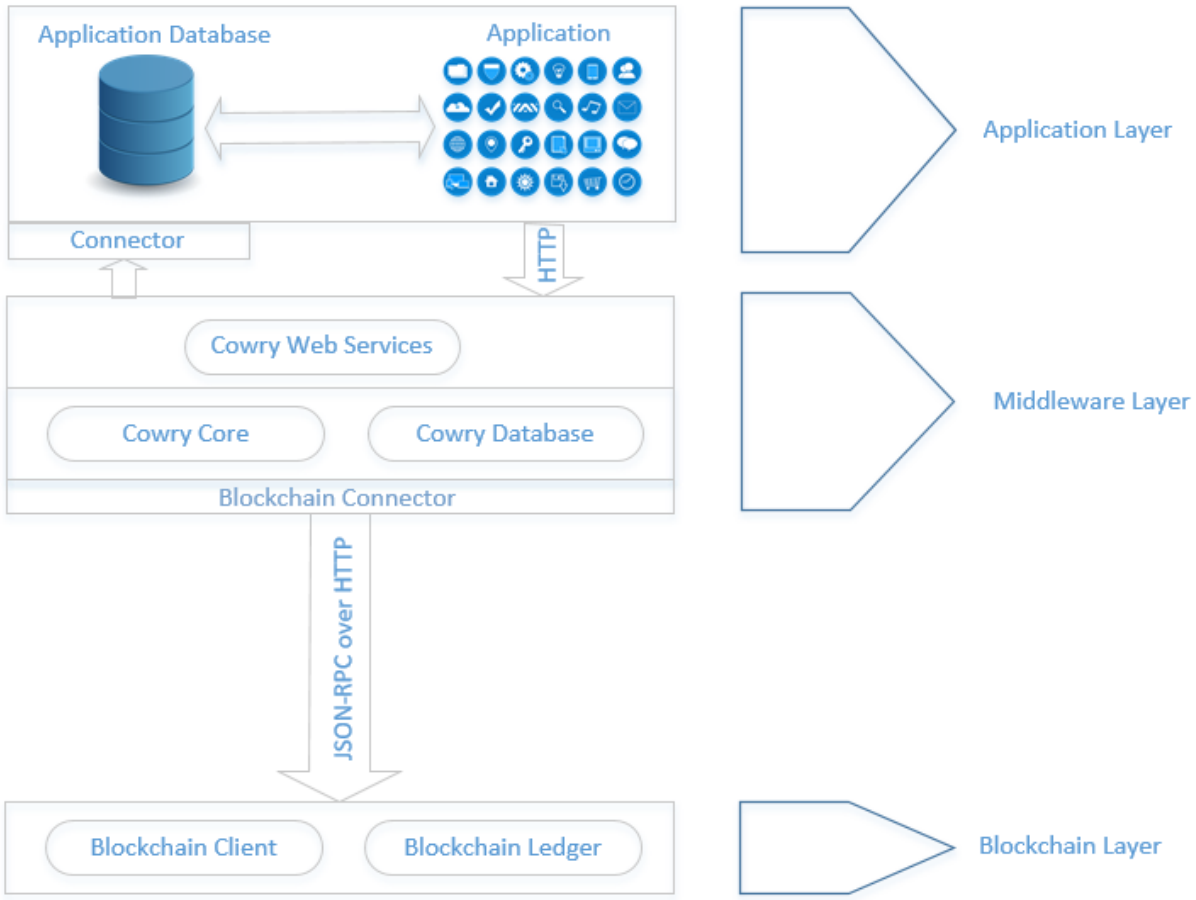

Figure 4.3: Architecture of Cowry nodes

The blockchain network (shown in Figure 4.2) consists of a number of Cowry nodes connected to each

other to form a decentralized P2P network. Internally, each Cowry node has three layers (shown in Figure 4.3): application layer, middleware layer and blockchain layer.

4.3.1 Application Layer

The application layer consists of application specific components and database. For example, this layer could contain the desktop application along with its internal database which hosts the data synchronized from a user's mobile phone. The application can connect to the Cowry core using RESTful web services to publish the data for trading.

4.3.2 Middleware Layer

The middleware layer consists of four components: Cowry Core, Blockchain Connector, Cowry Web service and Cowry Database.

- **Cowry Core:** The Cowry core defines a set of operations that enables the underlying blockchain to be used as a decentralized data and service marketplace. There are nine core operations provided by the Cowry middleware:
  - **Buy:** This allows the data consumer to request for a resource (data or service) published on the platform from the data owner. It requires the buyer's address as well as the resource unique identifier.
  - **Sell:** This allows the data owner to automatically respond to a buy request if it meets the price advertised. It is called automatically with the transaction data of the buy request.
  - **Rate:** This allows the data consumer to publish a rating between 0 and 1 for a transaction. It is called with details about the rating including the rated transaction id, the purchased resource id and the data consumer's comment.
  - **Search:** This allows a user to query the available data or jobs on the platform. The query is run on the node's local database and not on the blockchain.
  - **Sync:** This is used to synchronize the blockchain with the Cowry database. This is triggered automatically every time a new resource or job is published.
  - **Retrieve:** This is used to retrieved purchased resource from the blockchain. It is called with the symmetric key used for the transaction.
  - **View:** This allows a user to view the users, data, or jobs on the platform.
  - **Purge:** This purges the local database (Cowry Database) of expired entries. These entries are still available on the blockchain because of the immutability of the blockchain.

- **Register:** This allows a user to register an account, a dataset or job on the platform. It is called with a JSON object containing different fields of information that can be published without encryption (thus visible to all) on the blockchain.

- **Blockchain Connector:** The blockchain connector helps to achieve modularity and a bit of independence from the underlying blockchain infrastructure ensuring the possibility of implementing the same architecture for different blockchains simply by using different blockchain specific connector. Also, this decoupling ensures that majors changes to the blockchain implementation does not require changing Cowry core.

- **Cowry Web service:** The Cowry web service projects the operations of the Cowry core as services to be consumed via HTTP requests.

- **Cowry Database:** This component represents the unique part of this work. The Cowry database provides additional features for the platform that is not currently efficient using the underlying blockchain for example indexing for quick search and retrieval. The database caches some core data on the blockchain and index it for quick search. This is a trade-off between additional space (for the database) and performance improvement. A document style database was used because it natively supports the JSON format used by Cowry for storing the data on the blockchain.

4.3.3 Blockchain Layer

This layer contains the blockchain core client and local copy of the blockchain ledger. The blockchain ledger records all the transaction on the blockchain network. It was used to record all the interactions between participants, store the metadata of the participants and the metadata of the resource exchanged. As discussed in Chapter 3, all data on the ledger is replicated on every participating node in the network. The blockchain client used for our prototype is MultiChain.

We defined a number of components hosted on the blockchain ledger and interacting with the operations on Cowry core. Each component serves a single purpose. Figure 4.4 shows the interaction of the participants and components of the Cowry framework.

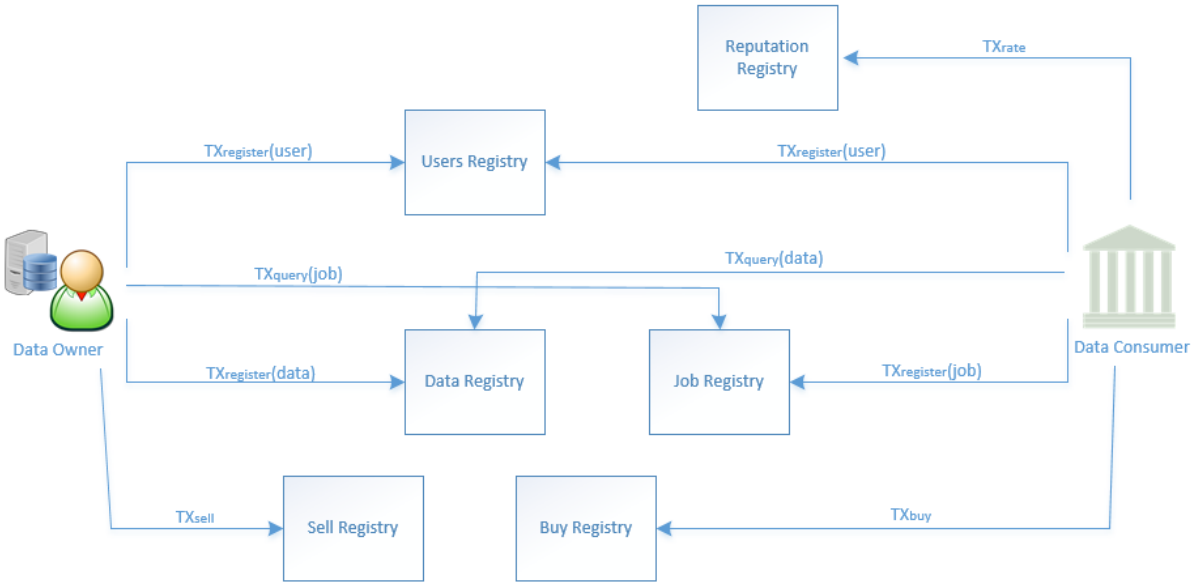

Figure 4.4: Blockchain ledger components and interaction

The transactions ($TX_{function}(argument)$) are blockchain transactions used for that function using the specified argument. For example, $TX_{register}(users)$ transaction represents a blockchain transaction that is sent to the user registry.

- **Data (or Service) Owner (DSO):** The DSO wants to grant access to data or services which its owns in exchange for micropayments in form of cryptocurrencies. The data could be in any form and is encoded into hexadecimal before writing to the blockchain. It can be the raw data requested or information on accessing it off-chain via a web service.
- **Data (or Service) Consumer (DSC):** The DSC is a user that wants to pay for access to certain resources. The DSC can also register a job describing the resource required and how it would be used.
- **Users Registry:** This registry maps a hash of the user's identifier (for example an ID used to register a user account) to an address on the blockchain. MultiChain Streams (discussed in Chapter 3) are used to store this mapping.
- **Data (or Service) Registry:** This registry is used to register a data or service on the blockchain. It stores relevant information about the data including the cost, the owner usage policies for example if the resource can be redistributed etc. The idea is by publishing this policy information on the blockchain, all parties involved in the transaction agree on the policy. However, violations can only be enforced off-chain using traditional legal approaches as currently used when such data rights are violated in

traditional systems. The data sent to the registry is represented as JSON. An example is shown in Figure 4.5.


```
{
   "id":"IoT21",
   "desc":"this is the default description",
   "hash":"d111ef112de42f76b5aafe30f6026c341219e8ed6705f45b0858a1677e0827a0",
   "license":"Open License",
   "location":"SK",
   "price":"11",
   "public_key":"-----BEGIN PUBLIC KEY-----
\nMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDEE1k0hAMmL9fBLgJSU+W2InAY\nqHkUBRp1+Z5yLMZqJd4NMJHMjdckXYyPNZ78yXt3Yw+Qdl
----END PUBLIC KEY-----",
   "size":"36",
   "type":"IoT",
   "validity":"2017-12-31 00:00:00"
}
```


Figure 4.5: Sample data example stored in the data registry

The *id*, *desc*, *price*, *public_key* are required fields, and they represent an identifier for the resource, a free-form description of the resource, the price, and the sellers public key respectively. The remaining fields are optional. The *hash* field is a hash of the actual resource, the *license* provides information about what and how the data can or cannot be used, the *location* field represents a location information for the dataset, the *size* represents the size of the data in bytes, *type* specifies the type of data and *validity* specifies an expiry date after which the metadata can be purged.

- **Job Registry:** This is like the Data registry. It is called by the data consumer to register a job requiring certain data set.
- **Buy Registry:** This is used to maintain a trace of every purchase request on the platform. It contains details of the request including the originator.
- **Sell Registry:** This maintains a record of every completed transaction along with the encrypted data transferred. The data consumer can also retrieve the purchased resource from this registry.
- **Reputation Registry:** This register maps an address on the blockchain to a rating for a transaction. The data consumer can view and filter by this rating. The rate can be viewed by every participant on the network.

4.4 Sequence Diagrams

Figure 4.6 shows the process of buying and selling on the Cowry platform. The buy transaction is initiated by a user interested in a resource published on the Cowry platform. The sell transaction is initiated automatically by the seller's node if the buy requests meets the predetermined specification (e.g. *price*).

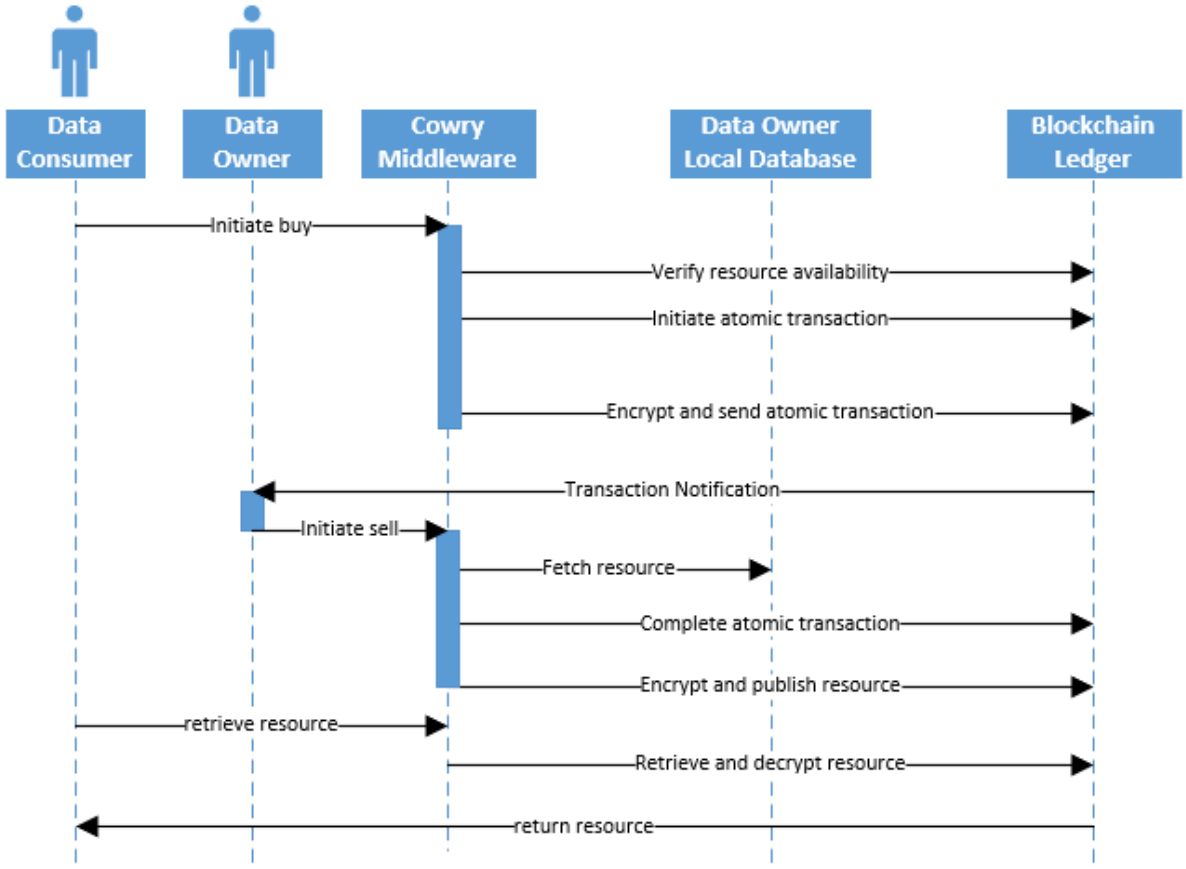

Figure 4.6: The process of buying and selling on Cowry platform

Figure 4.7 shows the Sync and Search process. Once the resource is uploaded and the metadata saved on the blockchain, it triggers an oracle on all the participating nodes which saves it to the Cowry database of that node. The metadata can then be queried with values for its different fields. For example, a user can query for dataset from a *location*, or a *type*. The query goes directly to the Cowry database instead of the blockchain.

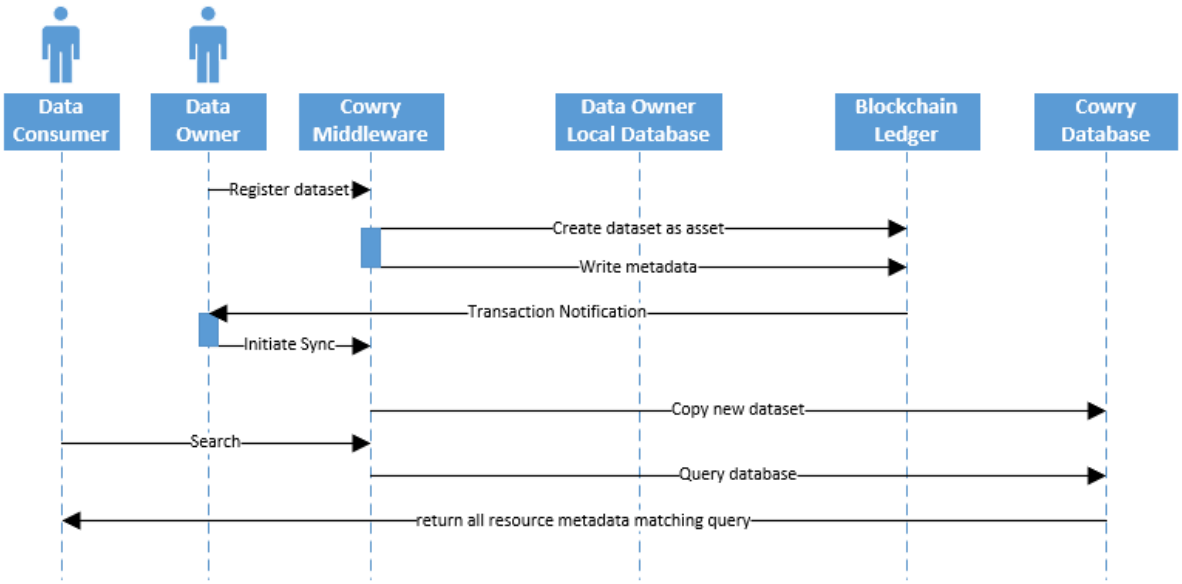

Figure 4.7: The process of syncing and searching on Cowry platform

4.5 Summary

This chapter discussed the design of the Cowry architecture. At the basic level, the design followed recommended practices in the literature for building data monetization platforms. Beyond that, the architecture incorporates traditional NoSQL database on every node to leverage their features. The next chapter would go into details of the implementation of the architecture.

Chapter 5

Implementation

A prototype of the architectural design in Chapter 4 was implemented. This chapter highlights the hardware and software technologies requirement as well as the implementation approach.

5.1 Software Requirement

Component	**Specification**
Operating System	Ubuntu 16.04 Desktop (64-bit)
Blockchain	MultiChain 1.0.2
Cowry Database	MongoDB v3.4.9
Application Database	MySQL 5.7.20
Server	Node v8.9.3 LTS

Table 5.1: Software requirements

Figure 5.1 shows the software components required for the implementation. The Cowry database is implemented using MongoDB. The choice of MongoDB was inspired by the BigChainDB blockchain platform described briefly in the literature review, and it also meets the design requirements for a document style NoSQL database supporting JSON, the format used in this work for storing data on the blockchain. For the application database, a MySQL database is used. A connector was implemented to interface the Cowry middleware to the MySQL database. Node.js was used to implement the web services. The MultiChain blockchain platform discussed in the literature was used as the core blockchain component.

5.2 Implementation

The implementation was done following the same layered approach of the architecture. This section discusses the setup of the blockchain and middleware layer. The implementation was done using the JavaScript programming language. Details on how to access the code is given in the Appendix.

5.2.1 Blockchain

The blockchain layer consists of the MultiChain client and the ledger. The client allows any participant to join and participate in a MultiChain private network. It takes care of many low-level operations such as connecting to the network, sending, and receiving transactions and maintaining the local copy of the blockchain ledger. During the setup of the first node in the MultiChain network, a configuration file needs to be customized to specify the parameters of the blockchain. A description of the different parameters can be found on the MultiChain developers site [19]. An example configuration file for the blockchain is shown in Figure 5.1.

```
# ==== MultiChain configuration file ====

# Created by multichain-util
# Protocol version: 10008

# This parameter set is VALID.
# To join network please run "multichaind cowryv1.1".

# The following parameters can only be edited if this file is a prototype of another configuration file.
# Please run "multichain-util clone cowryv1.1 <new-network-name>" to generate new network.


# Basic chain parameters

chain-protocol = multichain            # Chain protocol: multichain (permissions, native assets) or bitcoin
chain-description = MultiChain cowryv1.1 # Chain description, embedded in genesis block coinbase, max 256 chars.
root-stream-name = root                # Root stream name, blank means no root stream.
root-stream-open = true                # Allow anyone to publish in root stream
chain-is-testnet = false               # Content of the 'testnet' field of API responses, for compatibility.
target-block-time = 15                 # Target time between blocks (transaction confirmation delay), seconds. (2 - 86400)
maximum-block-size = 8388608           # Maximum block size in bytes. (1000 - 1000000000)

# Global permissions

anyone-can-connect = true              # Anyone can connect, i.e. a publicly readable blockchain.
anyone-can-send = true                 # Anyone can send, i.e. transaction signing not restricted by address.
anyone-can-receive = true              # Anyone can receive, i.e. transaction outputs not restricted by address.
anyone-can-receive-empty = true        # Anyone can receive empty output, i.e. without permission grants, asset transfers and zero native currency.
anyone-can-create = false              # Anyone can create new streams.
anyone-can-issue = true                # Anyone can issue new native assets.
anyone-can-mine = true                 # Anyone can mine blocks (confirm transactions).
anyone-can-activate = false            # Anyone can grant or revoke connect, send and receive permissions.
anyone-can-admin = false               # Anyone can grant or revoke all permissions.
support-miner-precheck = true          # Require special metadata output with cached scriptPubKey for input, to support advanced miner checks.
allow-p2sh-outputs = true              # Allow pay-to-scripthash (P2SH) scripts, often used for multisig.
allow-multisig-outputs = true          # Allow bare multisignature scripts, rarely used but still supported.
```

Figure 5.1: Snippet of Cowry blockchain platform configuration

5.2.2 Cowry Middleware

The implementation consists of seven modules: api.js, chain.js, routes.js, utils.js, cache.js, datastore.js, oracle.js, and two configuration files: blockchain.cfg, datastore.cfg.

- **api.js:** This module projects the core functions as web services.
- **chain.js:** This module manages the connection with the MultiChain blockchain. It uses configuration pre-set in the blockchain.cfg file.
- **routes.js:** This module implements the core functions of the Cowry middleware.
- **utils.js:** This provides utility functions for hashing, encryption, decryption, encoding and decoding.

- **cache.js:** This module manages the connection with the MongoDB database by providing high level function interface that are called by the Cowry core functions.
- **datastore.js:** This provides a high-level function interface for connecting to an MySQL database used in the application layer. It uses configurations set in the datastore.cfg file.
- **oracle.js:** This detects and initiate response to notifications from the blockchain. For example, a transaction uploading a new dataset triggers the sync operation to copy the metadata to the Cowry database.
- **datastore.cfg:** This file provides configuration options for the connection to the application database such as login credentials and database schema information.
- **blockchain.cfg:** This file provides configuration options for connecting to the blockchain such as blockchain host and JSON-RPC access credentials as well as initial setup options such as the name of currency, and initial amount to issue.

5.3 Deployment

To deploy the Cowry platform the first step is to setup the underlying blockchain infrastructure, including installing the MultiChain clients on the participating nodes. Assuming that has been done, the remaining steps are as follows:

1. On the first participating node:
   - Create the blockchain.

     $$multichain\text{-}util\ create\ [chain\text{-}name]$$

   - Configure the blockchain by customizing the params.dat file (automatically created in the previous step). For our experiments, we used the options shown in Figure 5.1. The portion not shown in the figure was not modified.
   - Configure the path to the oracle script using the $walletnotify$ runtime parameter.
   - Start the created blockchain.

     $$multichaind\ [chain\text{-}name]\ \text{-}daemon$$

   - Customize blockchain.cfg to allow the Cowry platform to connect to the blockchain.
   - Launch Cowry.

     $$node\ api.js$$

2. On the other participating nodes:

- Connect the nodes to the created blockchain network.

  multichaind [*chain-name*]@[*ip-address*]:[*port*] *-daemon*

- Customize blockchain.cfg to allow the Cowry platform to connect to the blockchain.
- Launch Cowry.

  node api.js

To use the platform, a user can upload data via the interface, the data is stored on the application database and the extracted metadata is written to the blockchain. To buy resource via the platform, the Cowry cryptocurrency is required. This work would not go into the details of purchase and distribution of the cryptocurrency.

5.4 Summary

This chapter described the implemented proof of concept prototype of the Cowry architecture. The next chapter discusses the evaluation of the prototype including how it accomplishes the goal of decentralized query and retrieval of resources stored on the blockchain.

Chapter 6

Evaluation

This chapter discusses the evaluation of the architecture proposed in Chapter 4, and implemented in Chapter 5, to determine whether it answers the research questions highlighted in Chapter 2. The evaluation of the architecture focuses on the two primary operations introduced in this research:

- **Sync:** This operation ensures that the Cowry database and the blockchain are synchronized.
- **Search:** This operation allows flexible query of the resources metadata stored on the blockchain (and cached on the Cowry database).

6.1 Performance Evaluation

Two group of experiments was conducted. The first set was carried out in a local network environment simulating a high-speed connection, and the other was conducted in the cloud using Amazon Web Services (AWS). In both environment, two metrics were used for determining the system performance: Throughput and Response Time.

- **Response Time:** This is the time between when the client sends the request and when it receives the response. It is measured in milliseconds (ms).
- **Throughput:** This is the maximum rate at which requests is handled by the platform. It is calculated as:

  $$number\,of\,requests\,/\,unit\,of\,time$$

  where the time is measured from the start of the first request to the end of the last requests. It is measured in requests/seconds.

The response time acts as a proxy for measuring the user experience as a slow application lead to poor user experience. The throughput is a proxy for the scalability of the platform by capturing the performance of the system as the number of requests increases. Apache JMeter was used for the measurements. It is an open source testing tool used for testing the performance of a variety of services, including web services.

The experiments would help determine the feasibility of building a fully decentralized data monetization platform based on the Cowry architecture.

6.1.1 Local Environment

Component	Detail
Blockchain Cluster	3 Nodes, each with specifications as follows: Ubuntu 16.04 Desktop (64-bit) Intel (R) Core (TM) i5-2400 CPU @ 3.10 GHz 16 GB RAM
Client	Windows 10 Education 64-bit OS Intel (R) Core (TM) i7-6700 CPU @ 3.40 GHz 3.41 GHz 32 GB RAM
Apache JMeter	3.1 r1770033

Table 6.1: Local environment setup requirements

The local environment experiment is setup as described in Table 6.1. Figure 6.1 shows the layout of the local environment. The Cowry nodes making up the blockchain cluster are connected within the University's local network. The machine acting as the client is also connected to the local network.

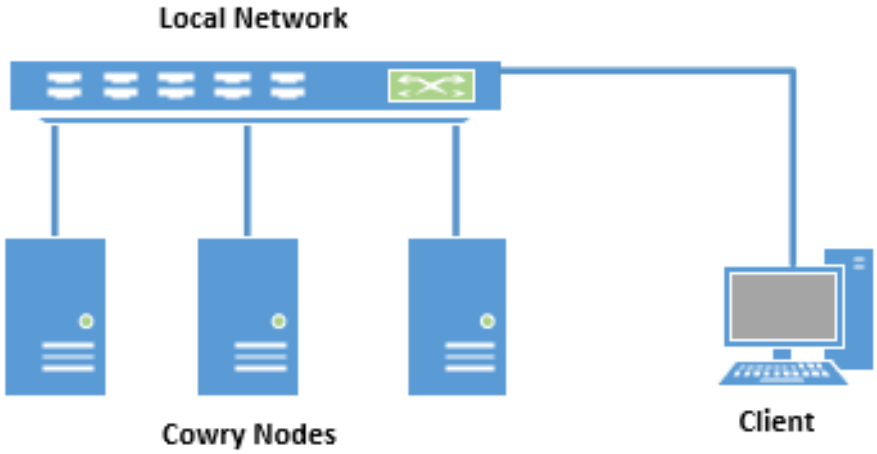

Figure 6.1: Local environment setup

In this environment, the following experiments was conducted:

1. Varying the number of users simultaneously accessing the platform.
2. Varying the delay between users' requests.

Experiment 1: Varying the number of users simultaneously accessing the platform

This experiment measures the average response time and throughput of the Sync and Search operation while varying the number of users making the requests. The number of users is 1, 2, 5, 10, 20, 50 and 100. The average request is taken over 10 iterations. The results are shown in Figure 6.2, 6.4, 6.6 and 6.7.

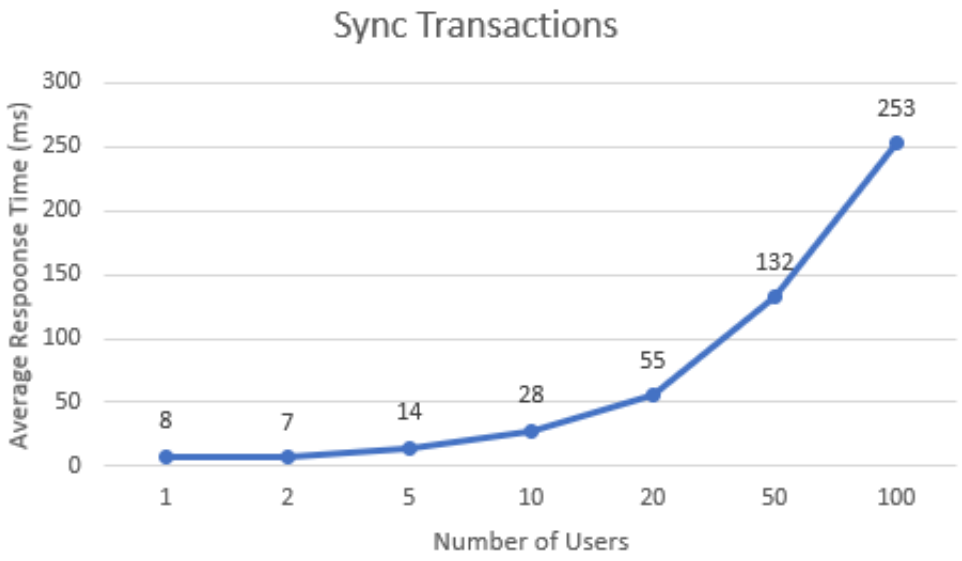

Figure 6.2: Average Response Time for Sync transactions

The result gives the average response time for the synchronization operation on a single user node at 8ms. This value rises to over 250ms for Cowry nodes supporting up to 100 users. The value indicates the time it takes the local Cowry database to synchronize with the blockchain as it is updated with data by up to 100 users. The implication is any search operation conducted from that node within that period before the Cowry database synchronizes would not give the current state of the blockchain. The standard deviation of the response time is shown in Figure 6.3. As expected, the standard deviation of the result rises as the number of users increases.

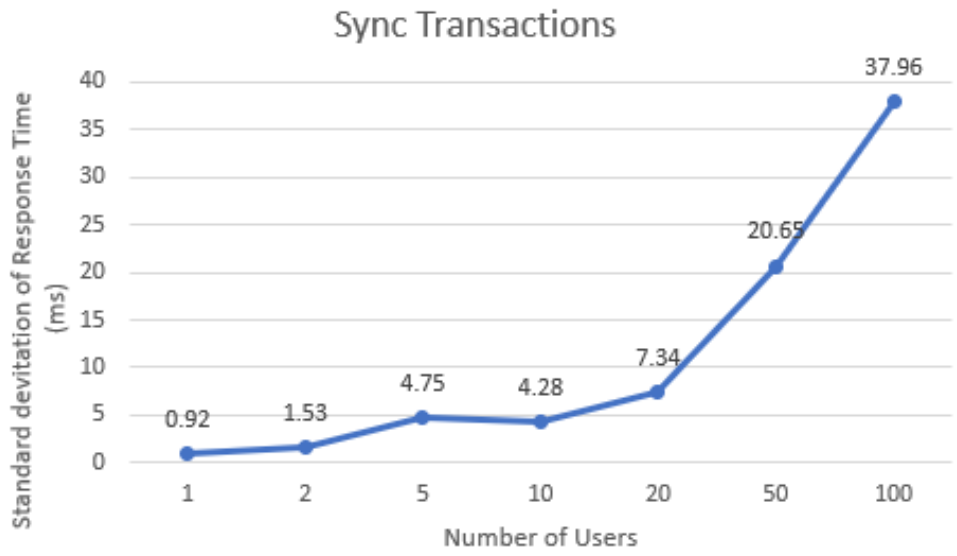

Figure 6.3: Standard Deviation of Response Time for Sync transactions

The search operation on a single user node has an average response time of 4ms. The search was conducted for dataset from a *location* and *type*. The first observation is that the search is only slightly faster than the synchronization of the Cowry database with the blockchain. For nodes supporting up to 100 users making simultaneous requests, the average response time rises to about 250ms. Nevertheless, the speed of the search also depends on the amount of data to be retrieved. More experiment using standard benchmarks would be required to effectively test the performance of the search operation. Figure 6.5 shows the standard deviation.

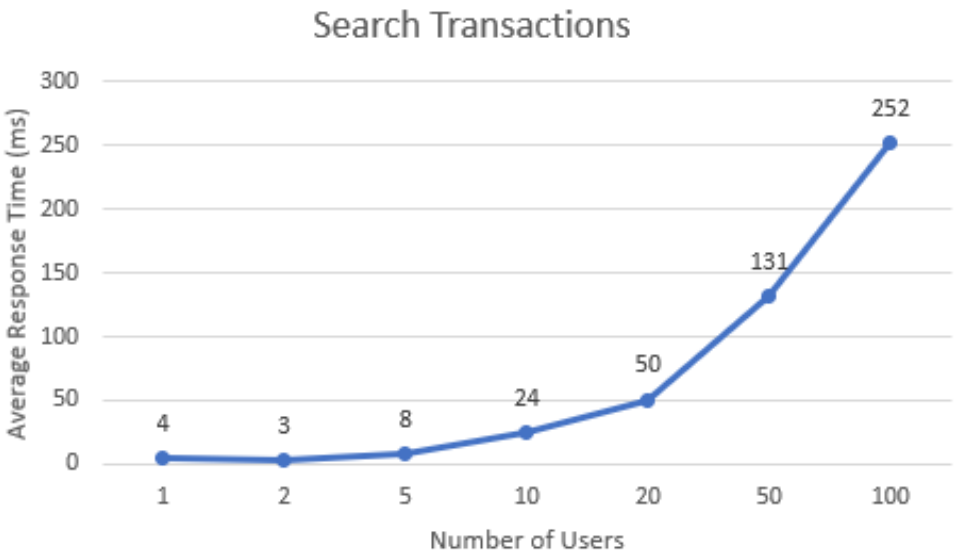

Figure 6.4: Average Response Time for Search transactions

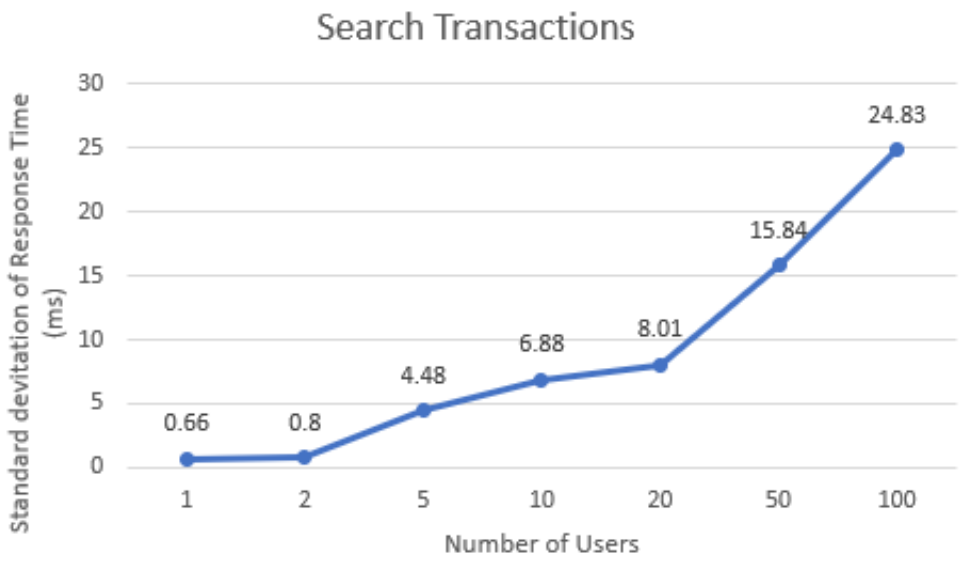

Figure 6.5: Standard Deviation of Response Time for Search transactions

The throughput for the sync transaction (Figure 6.6) of the node supporting only a single user rises from about 75 requests per seconds to almost 200 requests per seconds for nodes supporting up 100 users making simultaneous requests. This trend is very similar to that of the search transaction (Figure 6.7) which rises from about 79 requests per seconds to about 190 requests per seconds.

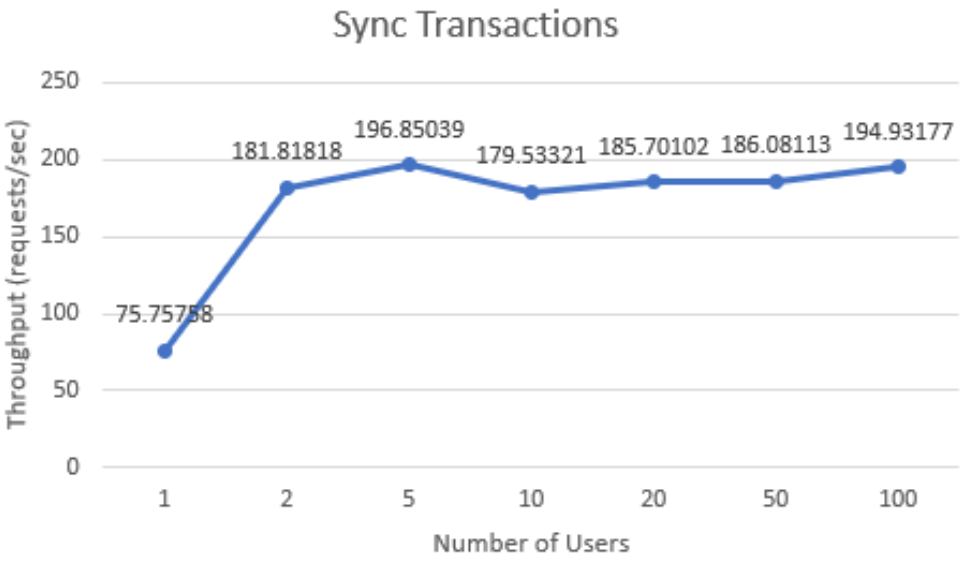

Figure 6.6: Throughput for Sync transactions

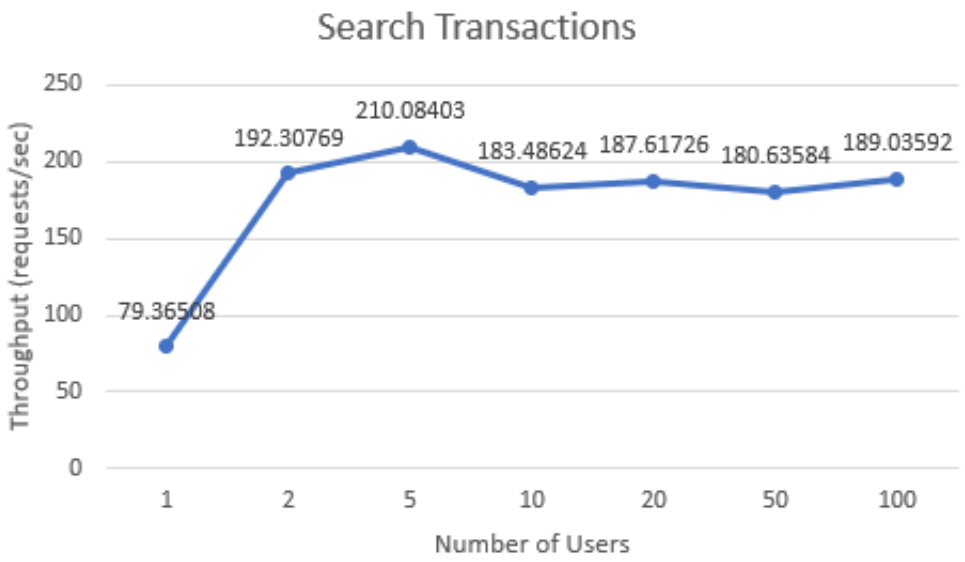

Figure 6.7: Throughput for Search transactions

Experiment 2: Varying the delay between users' requests

In this experiment, the delay between the user's requests was varied from 250ms to 1000ms (with steps of 250ms) for 100 users. The purpose of this experiment was to simulate real conditions, as typically the users accessing a node do not all connect at the same time. The latency and throughput of the Sync and Search operation is shown in Figure 6.8, 6.9, 6.12 and 6.13. Again, the average request is taken over 10 iterations. The standard deviation of the response time for both operations is shown in Figure 6.10 and 6.11.

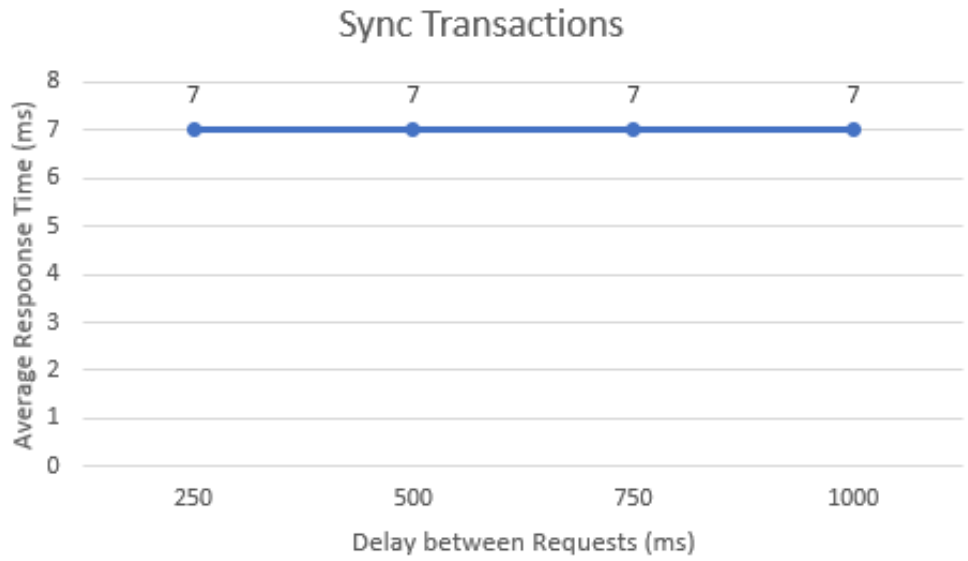

Figure 6.8: Average Response Time for Sync transactions - varying delay between 100 users' requests

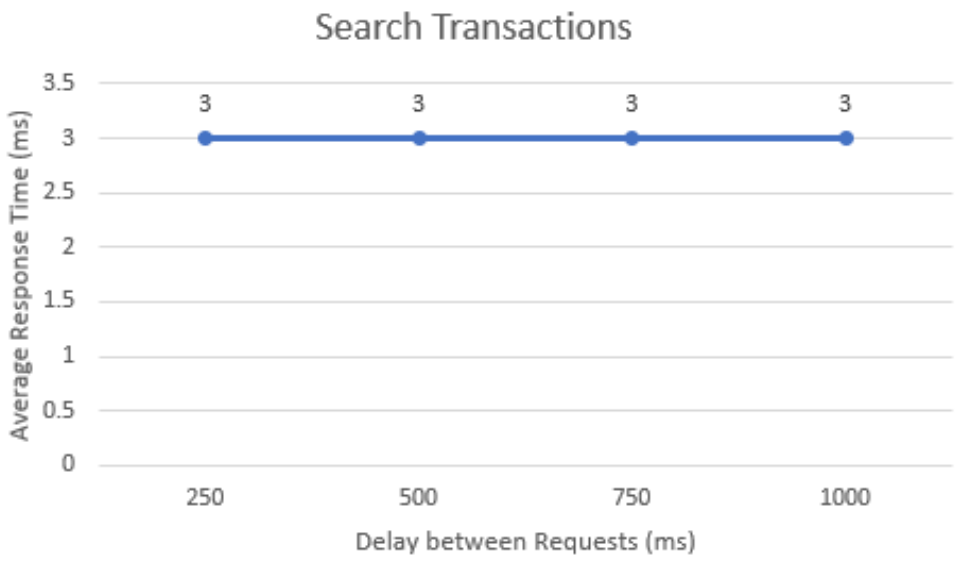

Figure 6.9: Average Response Time for Search transactions - varying delay between 100 users' requests

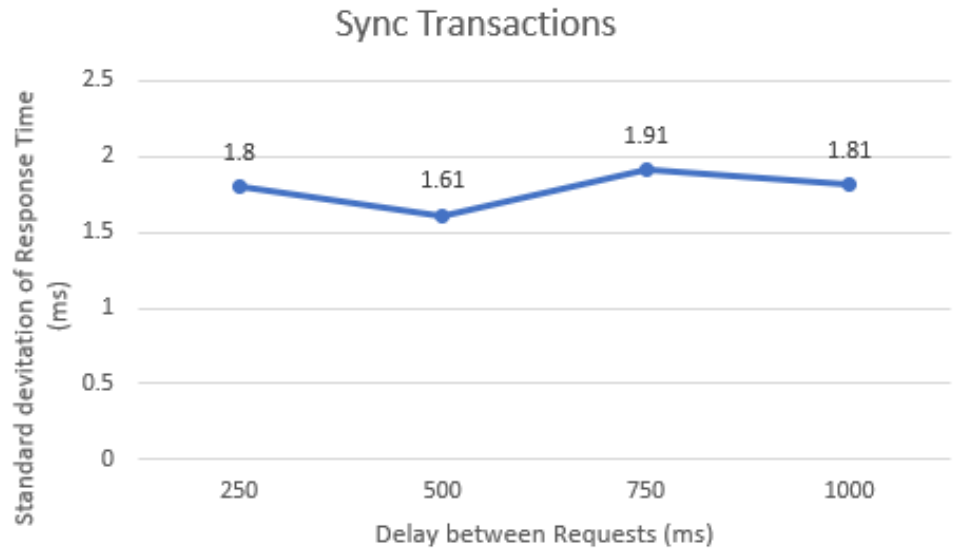

Figure 6.10: Standard Deviation of Response Time for Sync transactions - varying delay between 100 users' requests

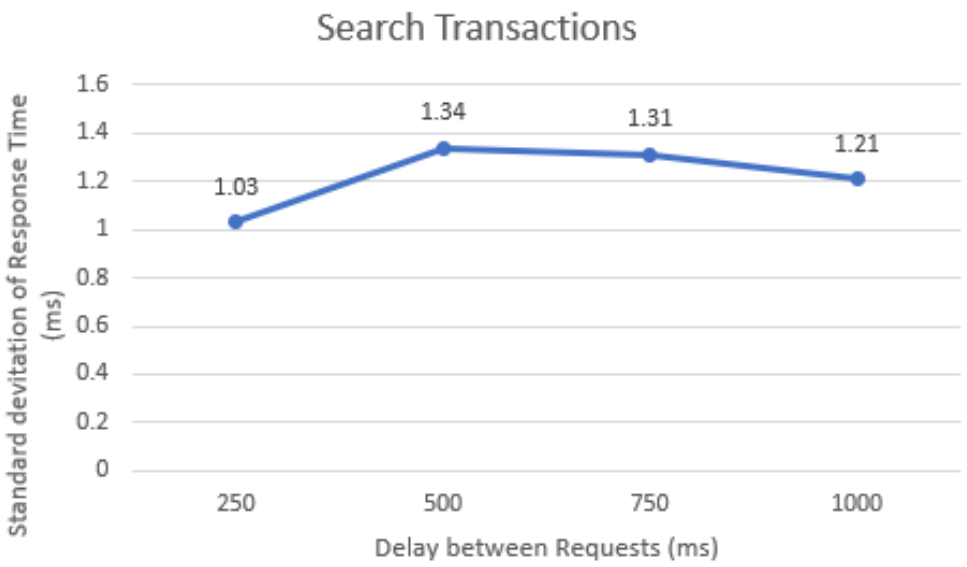

Figure 6.11: Standard Deviation of Response Time for Search transactions - varying delay between 100 users' requests

The average response time for the Sync and Search transaction on a node supporting 100 users making requests with delays from 250ms to 1000ms is constant. The result suggests that the delays does not impact on the performance. However, the throughput measurement shows a regular fall in the number of requests per seconds for both the sync and search transactions as the delay between requests is increase from 250ms to 1000ms. This is expected as the delays impacts on the total time used for computing the throughput.

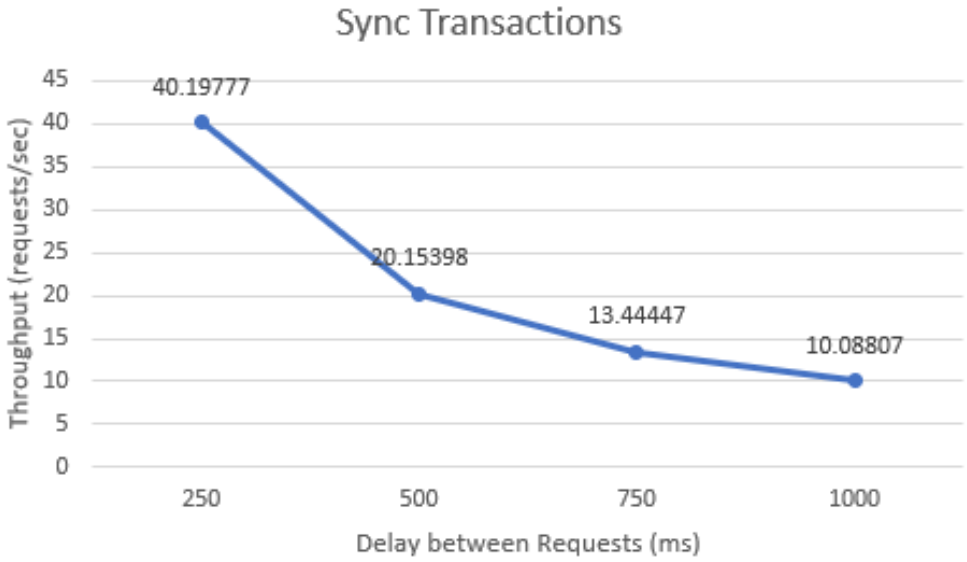

Figure 6.12: Throughput for Sync transactions - varying delay between 100 users' requests

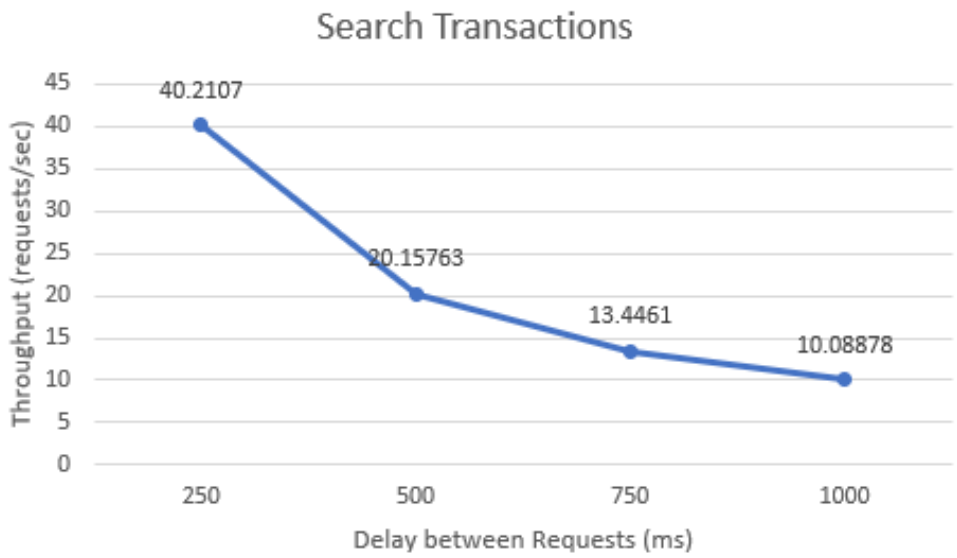

Figure 6.13: Throughput for Search transactions - varying delay between 100 users' requests

6.1.2 Cloud Environment

Practical deployment will usually be hosted in the cloud and accessed remotely. The cloud environment experiment is setup as described in Table 6.2.

Component	**Detail**
Blockchain Cluster	3 Instances, each with specifications as follows: Amazon Instance type t2.xLarge Ubuntu 16.04 LTS (64-bit) Intel Broadwell E5-2686v4 @ 2.3 GHz 16 GB RAM
Client	Windows 10 Education 64-bit OS Intel (R) Core (TM) i7-6700 CPU @ 3.40 GHz 3.41 GHz 32 GB RAM
Apache JMeter	3.1 r1770033

Table 6.2: Cloud environment setup requirements

Figure 6.14 shows the layout of the cloud environment. The Cowry nodes making up the blockchain cluster are connected within the AWS network. The machine acting as the client is on the local network and makes HTTP requests via the Internet.

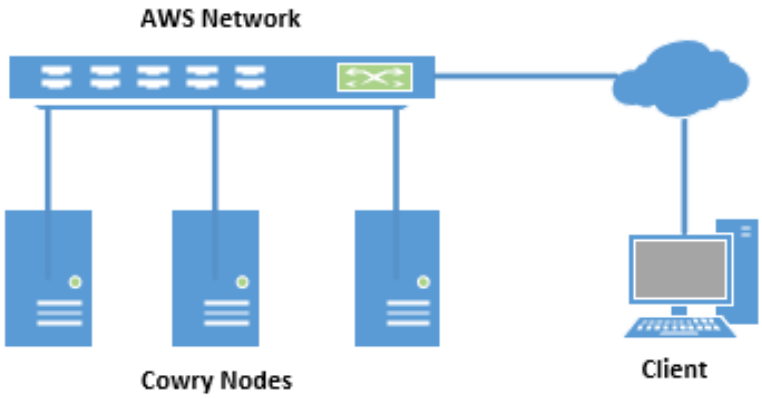

Figure 6.14: Cloud environment setup

In this environment, we repeated the same experiments:

3. Varying the number of users simultaneously accessing the platform.

4. Varying the delay between users' requests.

Experiment 3: Cloud: Varying the number of users simultaneously accessing the platform

This experiment measures the average response time and throughput of the Sync and Search operation, while varying the number of users making the requests. The number of users is 1, 2, 5, 10, 20, 50 and 100. The average request is taken over 10 iterations.

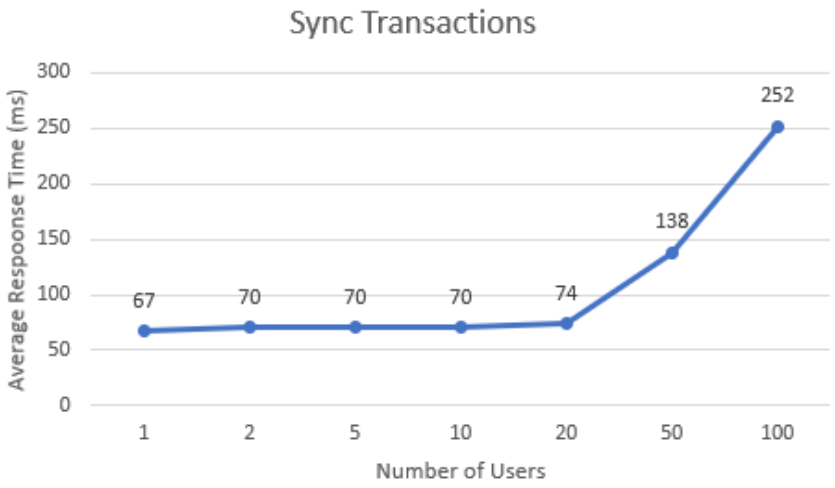

Figure 6.15: Average Response Time for Sync transactions on the cloud

Figure 6.15 shows the average response time for the synchronization operation on a single user node at 67ms. This value rises to over 250ms for Cowry nodes supporting up to 100 users. As with the local environment experiment, the value represents the time it takes the local Cowry database to synchronize with the blockchain as it is updated with data by up to 100 users. The standard deviation of the response time is shown in Figure 6.16.

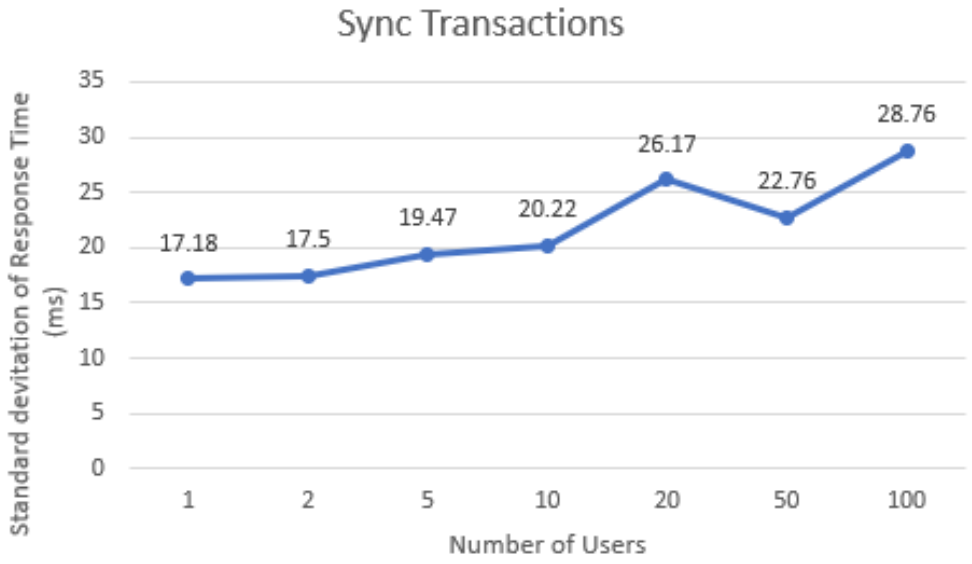

Figure 6.16: Standard Deviation of Response Time for Sync transactions on the cloud

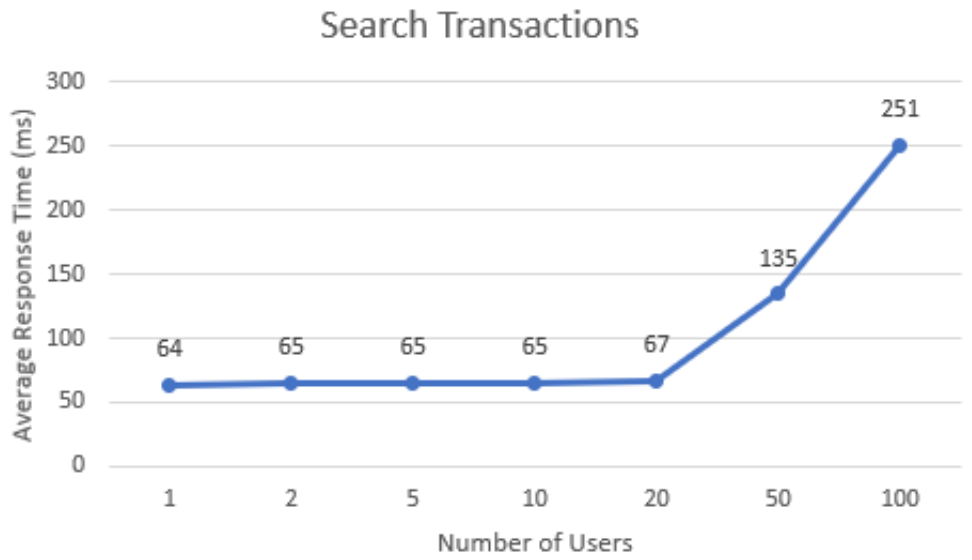

Figure 6.17: Average Response Time for Search transactions on the cloud

Figure 6.17 shows the search operation on a single user node has an average response time of 64ms. The

search was conducted for dataset from a *location* and of a *type*. The average response time for the sync and search transactions follows the same pattern. It is almost constant at about 67ms for the up to 20 users before rising sharply to 251ms for 100 users.

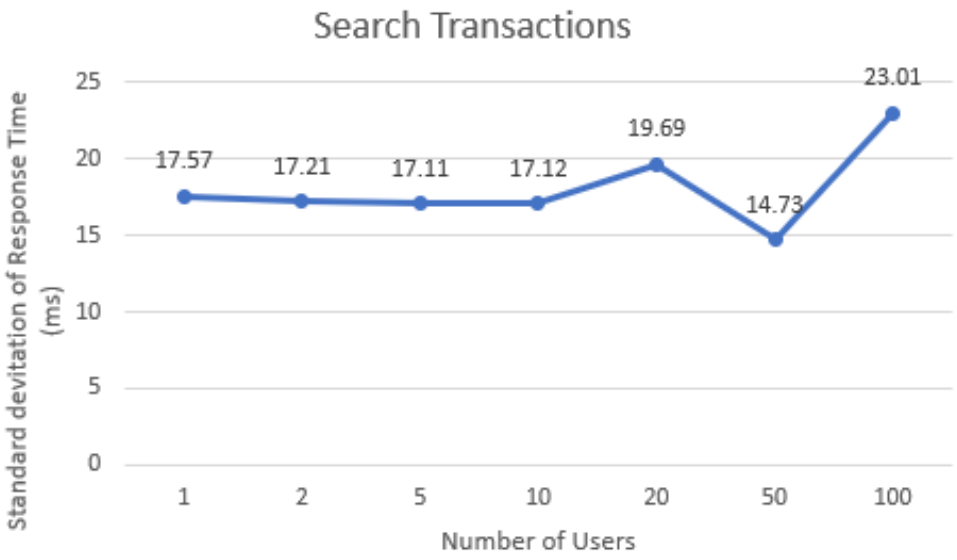

Figure 6.18: Standard Deviation of Response Time for Search transactions on the cloud

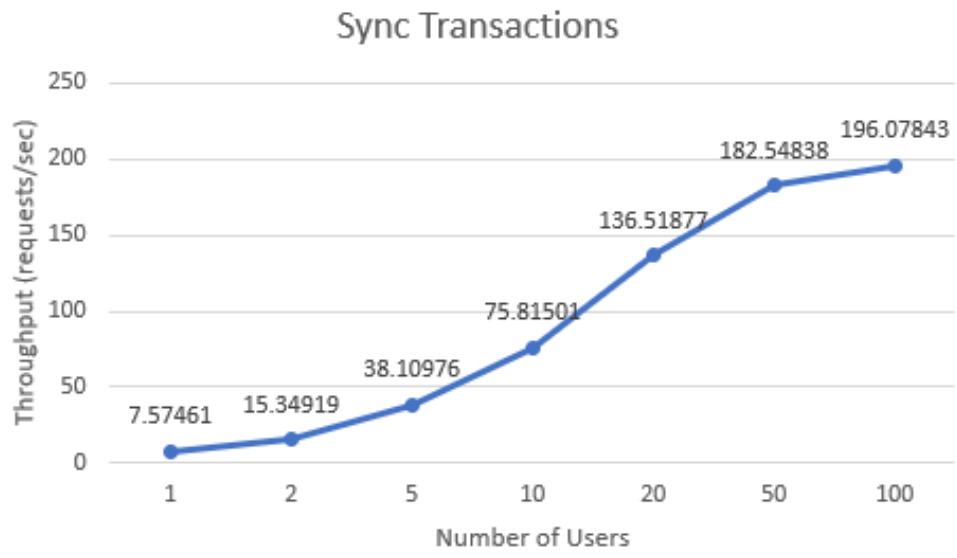

Figure 6.19: Throughput for Sync transactions on the cloud

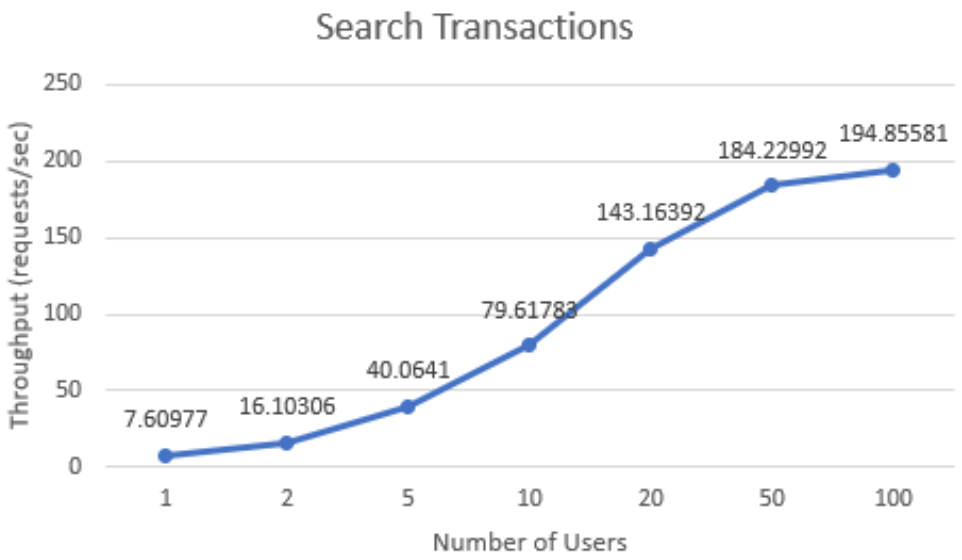

Figure 6.20: Throughput for Search transactions on the cloud

The throughput for the sync and search transaction (Figures 6.19, 6.20) for the node supporting only a single user rises from about 8 requests per seconds to almost 200 requests per seconds for nodes supporting up to 100 users making simultaneous requests. This result suggests that the platform scales well.

Experiment 4: Cloud: Varying the delay between users' requests

In this experiment, the delay between the users' requests was varied from 250ms to 1000ms (with steps of 250ms) for 100 users. Again, the average request is taken over 10 iterations. The standard deviation of the response time for both operations is shown in Figure 6.22 and 6.24.

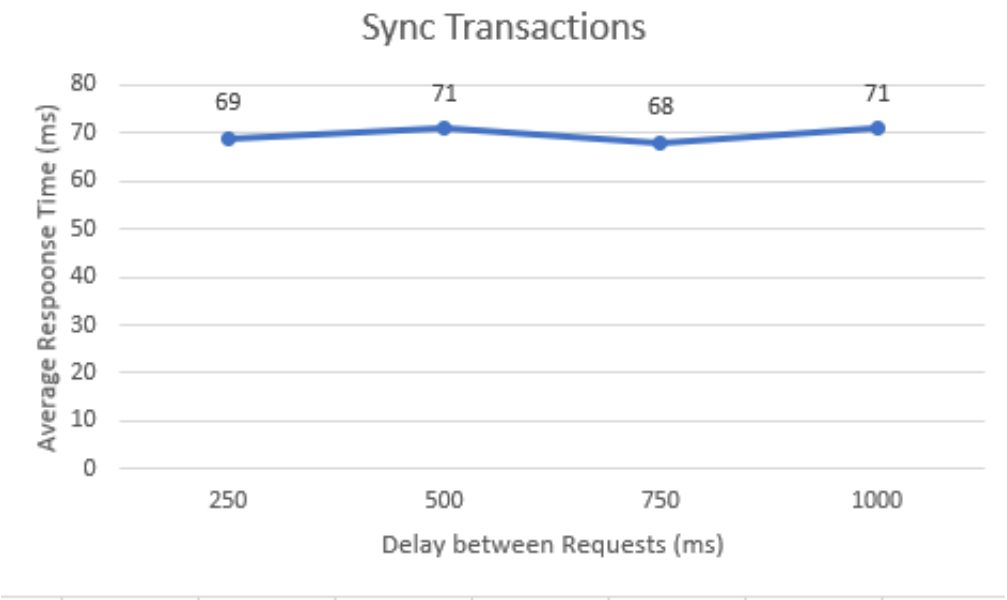

Figure 6.21: Average Response Time for Sync transactions on the cloud - varying delay between 100 users' requests

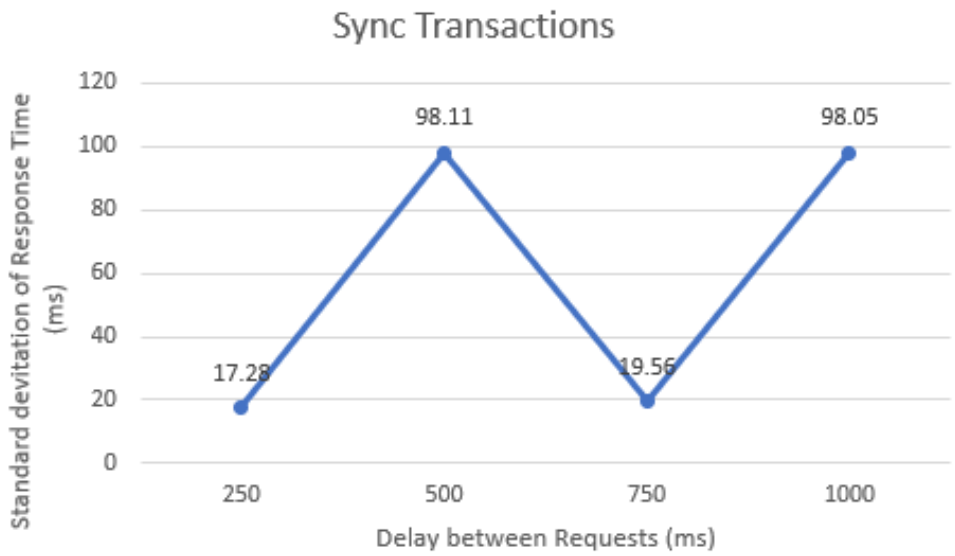

Figure 6.22: Standard Deviation of Response Time for Sync transactions on the cloud - varying delay between 100 users' requests

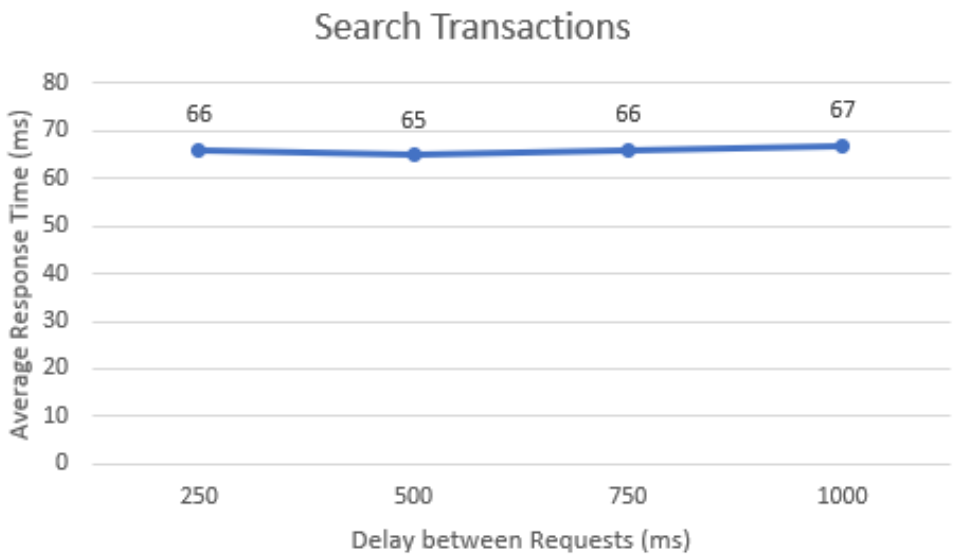

Figure 6.23: Average Response Time for Search transactions on the cloud - varying delay between 100 users' requests

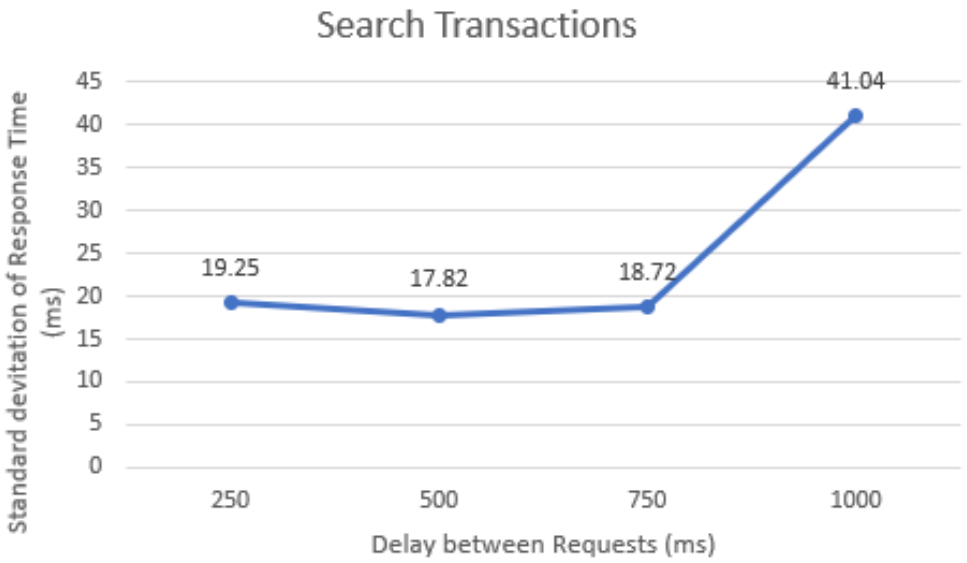

Figure 6.24: Standard Deviation of Response Time for Search transactions on the cloud - varying delay between 100 users' requests

Similar to the local experiment, the average response time for the Sync and Search transaction (Figures 6.21, 6.23) on a node supporting 100 users making requests with delays from 250ms to 1000ms is almost constant. This result suggests that the delays does not impact on the performance.

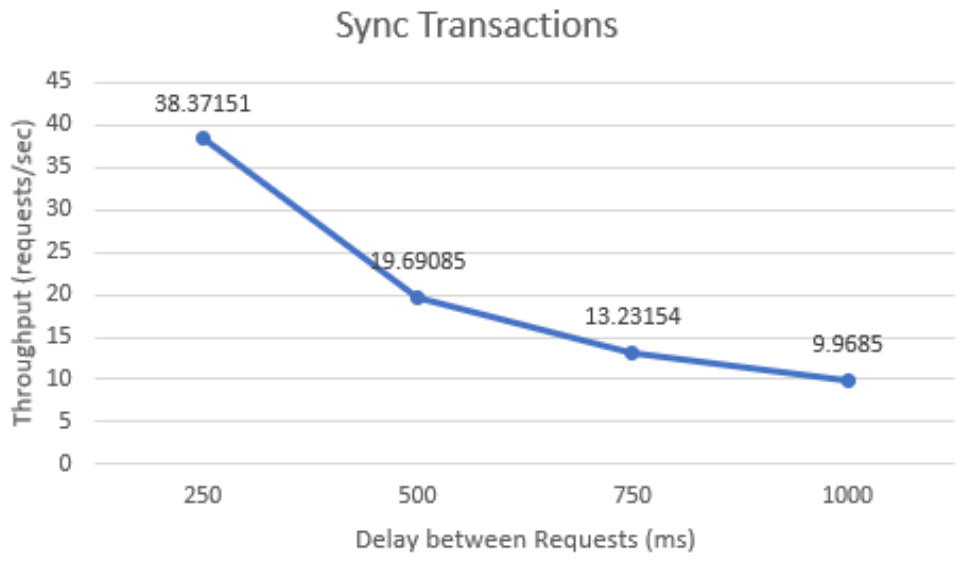

Figure 6.25: Throughput for Sync transactions on the cloud - varying delay between 100 users' requests

The throughput measurement (Figures 6.25, 6.26) shows a regular fall in the number of requests per seconds for both the sync and search transactions as the delay between requests is increase from 250ms to 1000ms. This is reasonable as the delays impacts on the total time used for computing the throughput.

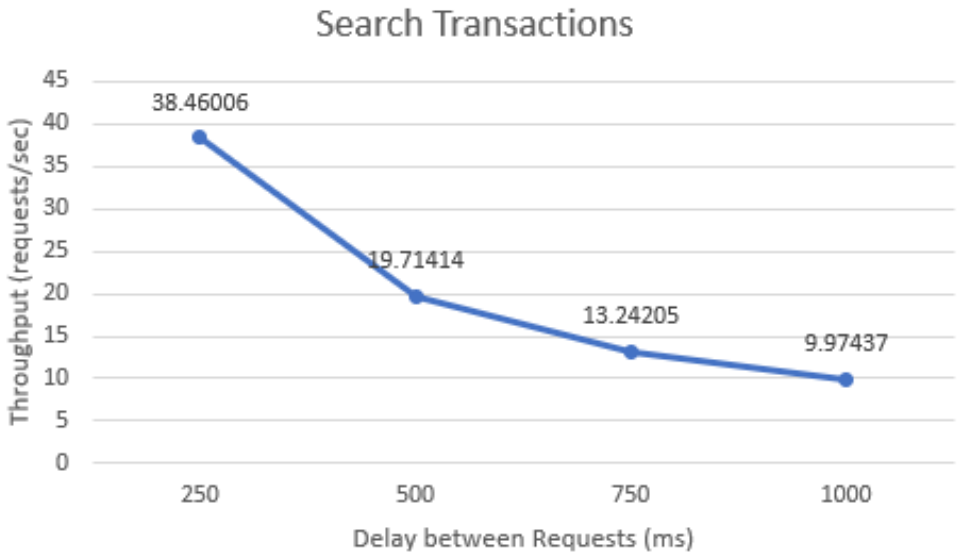

Figure 6.26: Throughput for Search transactions on the cloud - varying delay between 100 users' requests

6.2 Summary

From our experiments, it can be concluded that the blockchain synchronizes with the Cowry database, and the metadata can be searched at very low latency, indicating the scalability and efficiency of the architecture. By adopting a modular implementation, any more efficient NoSQL document-style database can be used instead of MongoDB and would still accomplish the same or better results. The experiments also indicate that database features of robust and flexible query and retrieval can be leveraged alongside the decentralization benefits of the blockchain. The framework implemented in this work handles this by fast local synchronization between the blockchain ledger and a traditional NoSQL database system. The downside to the architecture is the additional space required to store the synchronized data, nevertheless the use of an off-chain NoSQL database lends itself to existing tools for data mining and predictive analytics.

Chapter 7

Conclusion

Two required features of a data monetization platform are query and retrieval of the metadata of the resources to be monetized. Centralized platforms rely on the maturity of traditional NoSQL database systems to support these features. These databases for example MongoDB allows for very efficient query and retrieval of data it stores. However, centralized platforms come with a bag of security and privacy concerns, making them not the ideal approach for a data monetization platform. On the other hand, most existing decentralized platforms are only partially decentralized. They, for example, leverage blockchain technology for its support of cryptocurrency and micropayments and some of its other features but still default to storing the traded resources metadata on a centralized database.

In this work, I set out to determine how to efficiently query and retrieve metadata stored on the blockchain. I conducted an in-depth study of blockchain technology and reviewed the literature on blockchain-based data management. I then proposed an architecture for a fully decentralized platform that also leverages traditional database system to support efficient query and retrieval of resources' metadata. Cowry architecture was developed for publishing of metadata describing available resources (data or services), discovery of published resources including fast search and filtering, and exchange for cryptocurrency. I carried out several experiments to evaluate the efficiency and scalability of the architecture. The findings from the experiments includes:

- Varying the number of users simultaneously accessing the platform resulted in a low average response time which increased gradually as the number of users accessing the trusted node increased. The increase in the throughput suggests the solution is scalable.
- Varying the delays between users in attempt to mimic realistic conditions showed no difference in the response time; however, there is a decline in the throughput due to the delay introduced before running a request.

The results suggest that flexible search and retrieval of metadata on the blockchain can be done directly from the local NoSQL database with very low response time and high throughput.

7.1 Contributions

This research makes the following contributions:

- Literature review of blockchain technology and how it has been used to facilitate trusted data sharing among data owners.
- Prototype implementation of a fully decentralized architecture that combines blockchain and traditional distributed database to gain additional features such as flexible search and fast retrieval of metadata stored on the blockchain.

Table 7.1 shows how the Cowry platform would compare with centralized or many existing decentralized platforms

	Centralized Platforms	**Partially Decentralized Platform**	**Cowry Platform**
Fully Decentralized	No	No	Yes
Immutability	No	No	Yes
Rich Querying	Yes	Yes	Yes
Cryptocurrency	No	Yes	Yes
Low latency Retrieval	Yes	Yes	Yes

Table 7.1: Comparison of Cowry data monetization platform and other platforms

7.2 Limitations

This work is limited in a few ways:

- The experiments were not conducted using any standard benchmark datasets.
- Since dataset still must be written to the blockchain, it cannot support large datasets.
- The use of an additional database introduces a space overhead.

7.3 Future Works

The following additional work could be done to improve on the work done in this thesis:

- **Extend the Architecture**: Different blockchain platform have different architectures, some of which fits better for one use case than the other. A future work would be to explore how other blockchain

platforms can be combined with traditional database systems with the aim of leveraging the database matured features for building a data monetization platform. Some of the blockchain platforms to investigate include Ethereum and Hyperledger fabric, both of which supports smart contracts.

- **Improve on the reputation system**: Currently, there is still possibility of fraud with the reputation system where conniving participants work together to rate themselves. A related problem was also identified by Carboni [12], who used the bitcoin blockchain as a base for a decentralized and distributed feedback management system, and by Sharples and Domingue [84] who used the blockchain as a record of learning or intellectual effort. A future work would determine how to prevent such exploitation of the reputation system.

- **Explore Cryptocurrency & Market forces**: The current work is limited in that it does not go into details about the distribution of cryptocurrencies among the participant and how to tie the platform's cryptocurrency to fiat currencies.

- **Explore the effect of different hardware**: The current work did not show how increasing the number of nodes, memory and processor of the nodes would affect the performance and scalability. Experiments to determine this effect would be done in the future.

References

[1] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C-W Phan. Sha-3 proposal blake. *Submission to NIST*, 2008.

[2] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. Medrec: Using blockchain for medical data access and permission management. In *Open and Big Data (OBD), International Conference on*, pages 25–30. IEEE, 2016.

[3] Arshdeep Bahga and Vijay K Madisetti. Blockchain Platform for Industrial Internet of Things. *Journal of Software Engineering and Applications*, 9:533–546, 2016.

[4] Sönke Bartling and Benedikt Fecher. Could blockchain provide the technical fix to solve sciences reproducibility crisis? *Impact of Social Sciences Blog*, 2016.

[5] Ahmed Saleh Bataineh, Rabeb Mizouni, May El Barachi, and Jamal Bentahar. Monetizing personal data: A two-sided market approach. *Procedia Computer Science*, 83:472–479, 2016.

[6] Bitcoin Wiki. Double Spending. `https://en.bitcoin.it/wiki/Double-spending`. Accessed on March 28, 2017.

[7] Bitcoin Wiki. Nonce. `https://en.bitcoin.it/wiki/Nonce`. Accessed on July 27, 2017.

[8] Bitcoin Wiki. Proof of Stake. `https://en.bitcoin.it/wiki/Proof_of_Stake`. Accessed on July 28, 2017.

[9] Bitcoin Wiki. Proof of Work. `https://en.bitcoin.it/wiki/Proof_of_work`. Accessed on July 31, 2017.

[10] Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[11] Christian Cachin and Marko Vukolić. Blockchains consensus protocols in the wild. *arXiv preprint arXiv:1707.01873*, 2017.

[12] Davide Carboni. Feedback based reputation on top of the bitcoin blockchain. *arXiv preprint arXiv:1502.01504*, 2015.

[13] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[14] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209, 04 2014. Copyright - Springer Science+Business Media New York 2014; Last updated - 2014-08-30.

[15] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.

[16] Joseph Bonneau Andrew Miller Jeremy Clark, Arvind Narayanan Joshua A Kroll Edward, and W Felten. Research perspectives and challenges for bitcoin and cryptocurrencies.

[17] Coin Sciences. MultiChain - Open platform for blockchain applications. `https://www.multichain.com/`. Accessed on March 28, 2017.

[18] Coin Sciences. MultiChain permissions management. https://www.multichain.com/developers/permissions-management/. Accessed on August 10, 2017.

[19] Coin Sciences. MultiChain runtime parameters. https://www.multichain.com/developers/runtime-parameters/. Accessed on August 10, 2017.

[20] CoinMarketCap. CryptoCurrency Market Capitalizations. https://coinmarketcap.com/. Accessed on July 31, 2017.

[21] CoinMarketCap. CryptoCurrency Market Capitalizations - Percentage of Total Market Capitalization (Dominance). https://coinmarketcap.com/charts/. Accessed on July 31, 2017.

[22] CoinSpark. Upgrade your bitcoin with messaging and assets. http://coinspark.org/. Accessed on August 6, 2017.

[23] Christopher Copeland and Hongxia Zhong. Tangaroa: a byzantine fault tolerant raft, 2016.

[24] Brajesh De. Designing a restful api interface. In *API Management*, pages 29–57. Springer, 2017.

[25] Yves-Alexandre de Montjoye, Jordi Quoidbach, Florent Robic, and Alex Pentland. Predicting personality using novel mobile phone-based metrics. In *SBP*, pages 48–55. Springer, 2013.

[26] Matthew DeBord. Who owns connected car data? https://www.weforum.org/agenda/2015/09/who-owns-connected-car-data/. Accessed on May 30, 2017.

[27] J Dienelt. Understanding Ethereum. *New York, NY: CoinDesk*, 2016.

[28] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. Blockbench: A framework for analyzing private blockchains. *arXiv preprint arXiv:1703.04057*, 2017.

[29] Tim Dodd. University of Melbourne first in Australia to use blockchain for student records. https://goo.gl/acU5Bf. Accessed on August 7, 2017.

[30] John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[31] D Eastlake 3rd and Tony Hansen. US secure hash algorithms (SHA and SHA-based HMAC and HKDF). Technical report, 2011.

[32] EternityWall. Messages lasting forever. https://eternitywall.it/. Accessed on August 6, 2017.

[33] Ethereum Wiki. Ethash. https://github.com/ethereum/wiki/wiki/Ethash. Accessed on July 28, 2017.

[34] Roy T Fielding and Richard N Taylor. *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation, 2000.

[35] Klint Finley. Tim Berners-Lee, Inventor of the Web, plots a radical overhaul of his creation. https://www.wired.com/2017/04/tim-berners-lee-inventor-web-plots-radical-overhaul-creation/. Accessed on August 4, 2017.

[36] Michael J Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *International Conference on Fundamentals of Computation Theory*, pages 127–140. Springer, 1983.

[37] Thomas Fox-Brewster. Ashley Madison Breach Could Expose Privates Of 37 Million Cheaters. https://www.forbes.com/sites/thomasbrewster/2015/07/20/ashley-madison-attack/#85148765f48c. Accessed on June 30, 2017.

[38] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News*, 33(2):51–59, 2002.

[39] Bela Gipp, Corinna Breitinger, Norman Meuschke, and Joeran Beel. Cryptsubmit: Introducing securely timestamped manuscript submission and peer review feedback using the blockchain. 2017.

[40] Vindu Goel. Facebook Tinkers With Users Emotions in News Feed Experiment, Stirring Outcry. https://www.nytimes.com/2014/06/30/technology/facebook-tinkers-with-users-emotions-in-news-feed-experiment-stirring-outcry.html. Accessed on May 26, 2017.

[41] Gideon Greenspan. Blockchains vs centralized databases. http://www.multichain.com/blog/2016/03/blockchains-vs-centralized-databases/. Accessed on March 28, 2017.

[42] Gideon Greenspan. MultiChain data streams. http://www.multichain.com/developers/data-streams/. Accessed on March 28, 2017.

[43] Gideon Greenspan. MultiChain Private Blockchain - White Paper, 2015. Accessed on March 10, 2017.

[44] Glenn Greenwald and Ewen MacAskill. NSA Prism program taps in to user data of Apple, Google and others. https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data. Accessed on June 5, 2017.

[45] Ye Guo and Chen Liang. Blockchain application and outlook in the banking industry. *Financial Innovation*, 2(1):24, 2016.

[46] Heng Hou. The application of blockchain technology in e-government in china. In *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*, pages 1–4. IEEE, 2017.

[47] Uurtsaikh Jamsrandorj. Decentralized access control using blockchain. Master's thesis, 2017.

[48] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63, 2001.

[49] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

[50] Jeremy Kirk. Could the Bitcoin network be used as an ultrasecure notary service? https://goo.gl/ycbWYT. Accessed on August 1, 2017.

[51] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[52] Daniel Larimer. Delegated proof-of-stake (dpos). *Bitshare whitepaper*, 2014.

[53] Quoc V Le. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE, 2013.

[54] Xueping Liang, Sachin Shetty, Deepak Tosh, Charles Kamhoua, Kevin Kwiat, and Laurent Njilla. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *International Symposium on Cluster, Cloud and Grid Computing, IEEE/ACM*, 2017.

[55] Linux Foundation. Hyperledger Fabric. https://hyperledger.org/projects/fabric. Accessed on November 28, 2017.

[56] Will Martino. Kadena - The first scalable, high performance private blockchain. http://kadena.io/docs/Kadena-ConsensusWhitePaper-Aug2016.pdf. Accessed on August 6, 2017.

[57] Trent McConaghy, Rodolphe Marques, Andreas Müller, Dimitri De Jonghe, Troy McConaghy, Greg McMullen, Ryan Henderson, Sylvain Bellemare, and Alberto Granzotto. Bigchaindb: a scalable blockchain database. *white paper, BigChainDB*, 2016.

[58] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.

[59] Ralph Merkle. A digital signature based on a conventional encryption function. In *Advances in CryptologyCRYPTO87*, pages 369–378. Springer, 2006.

[60] Microsoft. Networking Basics: Peer-to-peer vs. server-based networks. `https://technet.microsoft.com/en-us/library/cc527483(v=ws.10).aspx`. Accessed on July 27, 2017.

[61] Microsoft. Understanding Public Key Cryptography. `https://technet.microsoft.com/en-us/library/aa998077(v=exchg.65).aspx`. Accessed on March 28, 2017.

[62] Paolo Missier, Shaimaa Bajoudah, Angelo Capossele, Andrea Gaglione, and Michele Nati. Mind my value: a decentralized infrastructure for fair and trusted iot data trading. 2017.

[63] Krešimir Mišura and Mario Žagar. Data marketplace for internet of things. In *Smart Systems and Technologies (SST), International Conference on*, pages 255–260. IEEE, 2016.

[64] Monax. Monax - The Ecosystem Application Platform. `https://monax.io/`. Accessed on March 28, 2017.

[65] MySQL 5.7 Reference Manual. Using Stored Routines (Procedures and Functions). `https://dev.mysql.com/doc/refman/5.7/en/stored-routines.html`. Accessed on December 18, 2017.

[66] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[67] Kay Noyen, Dirk Volland, Dominic Wörner, and Elgar Fleisch. When money learns to fly: Towards sensing as a service applications using bitcoin. *arXiv preprint arXiv:1409.5841*, 2014.

[68] U of S Plant Phenotyping and Imaging Research Centre. New U of S Plant Research Centre Launched to Design Crops for Global Food Security. `http://p2irc.usask.ca/articles/2016/new-u-of-s-plant-research-centre-launched-to-design-crops-for-global-food-security.php`. Accessed on June 29, 2017.

[69] Diego Ongaro and John K Ousterhout. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, pages 305–319, 2014.

[70] Albert Opher, Alex Chou, Andrew Onda, and Sounderrajan Krishna. The Rise of the Data Economy: Driving Value through Internet of Things Data Monetization - A Perspective for Chief Digital Officers and Chief Technology Officers. `https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=WWW12367USEN`. Accessed on May 30, 2017.

[71] Oraclize. Oraclize - blockchain oracle service, enabling data-rich smart contracts. `http://www.oraclize.it/`. Accessed on March 28, 2017.

[72] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and Communication Networks*, 9(18):5943–5964, 2016.

[73] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Towards a novel privacy-preserving access control model based on blockchain technology in iot. In *Europe and MENA Cooperation Advances in Information and Communication Technologies*, pages 523–533. Springer, 2017.

[74] Colin Percival. Tarsnap-the scrypt key derivation function and encryption utility.

[75] Charith Perera. *Sensing as a Service for Internet of Things: A Roadmap*. Lulu. com, 2017.

[76] Charith Perera. Sensing as a service (s2aas): Buying and selling iot data. *arXiv preprint arXiv:1702.02380*, 2017.

[77] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies*, 25(1):81–93, 2014.

[78] Gareth W Peters and Efstathios Panayi. Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money. In *Banking Beyond Banks and Money*, pages 239–278. Springer, 2016.

[79] Placemeter. Placemeter - Quantify the World. `https://www.placemeter.com/`. Accessed on June 1, 2017.

[80] Jérémy Robert, Sylvain Kubler, and Yves Le Traon. Micro-billing framework for iot: Research & technological foundations. In *Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on*, pages 301–308. IEEE, 2016.

[81] Tom Robinson. Bitcoin is not anonymous. `http://www.respublica.org.uk/disraeli-room-post/2015/03/24/bitcoin-is-not-anonymous/`. Accessed on August 1, 2017.

[82] Solidity Documentation. `https://solidity.readthedocs.io`. Accessed on August 11, 2017.

[83] Goldman Sachs. Blockchain putting theory into practice. *the-blockchain. com*, pages 25–32, 2016.

[84] Mike Sharples and John Domingue. The Blockchain and Kudos: A Distributed System for Educational Record, Reputation and Reward. In *European Conference on Technology Enhanced Learning*, pages 490–496. Springer, 2016.

[85] David Shrier, Weige Wu, and Alex Pentland. Blockchain & infrastructure (identity, data security). Technical report, Retrieved 27-11-16, from http://cdn. resources. getsmarter. ac/wp-content/uploads/2016/06/MIT_Blockain_Whitepaper_PartThree. pdf, 2016.

[86] David Siegel. Understanding The DAO Attack. `https://www.coindesk.com/understanding-dao-hack-journalists/`. Accessed on August 11, 2017.

[87] Balaji S. Srinivasan. The 21 Bitcoin Computer. `https://medium.com/@21/the-21-bitcoin-computer-1d28d652b57b`. Accessed on May 29, 2017.

[88] William Suberg. Factoms Latest Partnership Takes on US Healthcare. `https://cointelegraph.com/news/factoms-latest-partnership-takes-on-us-healthcare`. Accessed on August 7, 2017.

[89] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.

[90] Stefan Tai, Jacob Eberhardt, and Markus Klems. Not ACID , not BASE , but SALT A Transaction Processing Perspective on Blockchains. 2008.

[91] Don Tapscott and Alex Tapscott. How Blockchain Technology Can Reinvent The Power Grid. `http://fortune.com/2016/05/15/blockchain-reinvents-power-grid/`, 2016. Accessed on August 8, 2017.

[92] Thingful. Thingful - A Search Engine for the Internet of Things. `https://thingful.net/`. Accessed on June 1, 2017.

[93] Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123, 2015.

[94] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[95] Dominic Wörner. Design of a real-time data market based on the 21 bitcoin computer. In *Tackling Society's Grand Challenges with Design Science: 11th International Conference, DESRIST 2016, St. Johns, NL, Canada, May 23-25, 2016, Proceedings 11*, pages 228–232. Springer, 2016.

[96] Dominic Wörner and Thomas von Bomhard. When your sensor earns money: exchanging data for cash with bitcoin. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 295–298. ACM, 2014.

[97] Xiwei Xu, Cesare Pautasso, Liming Zhu, Vincent Gramoli, Alexander Ponomarev, An Binh Tran, and Shiping Chen. The blockchain as a software connector. In *Software Architecture (WICSA), 2016 13th Working IEEE/IFIP Conference on*, pages 182–191. IEEE, 2016.

[98] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, and Huaimin Wang. Blockchain Challenges and Opportunities: A Survey. 2016.

[99] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.