ANALYSIS OF BLOCKCHAIN-BASED STORAGE SYSTEMS

By

Phillipe S. Austria

Bachelor of Science (B.Sc.), Chemical Engineering
University of Washington
2010

A thesis submitted in partial fulfillment of
the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
August 2020

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

July 24, 2020

This thesis prepared by

Phillipe S. Austria

entitled

Analysis of Blockchain-Based Storage Systems

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Yoohwan Kim, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Dean

Ju-Yeon Jo, Ph.D.
Examination Committee Member

Wolfgang Bein, Ph.D.
Examination Committee Member

Sean Mulvenon, Ph.D.
Graduate College Faculty Representative

Abstract

Increasing storage needs has driven cloud storage to become a prevalent choice to store data for both business and personal use. Cloud service providers, such as Google, offer advantages over storing personal hard drives; however, customers lose privacy and require trust in the provider to act honestly with their data. This thesis explores an alternative to cloud storage using Blockchain technology. I focus on Sia, a blockchain-based storage platform that allows users to rent storage from other users.

In this study, I evaluate the security, performance and costs between the Sia and traditional cloud storage. I assessed security based on five categories: encryption, data access, durability, redundancy and availability. The performance and costs were evaluated through various storage tests using Sia and Google Drive. The results revealed Sia to have higher availability and less data access (more privacy). Google Drive displayed faster upload times, while Sia displayed faster download times. Additionally, the final cost of using Sia was comparable to cloud providers; however, was more than double of what was expected due to data redundancy costs. Further testing is needed to fully understand the complex costs structure of a blockchain-based storage platform.

Acknowledgements

I first wish to express my sincere appreciation to my academic adviser, Dr. Yoohwan Kim of the Computer Science department at the University of Nevada, Las Vegas. He has guided and supported me from the beginning of my master's journey. He has allowed this paper to be my own work, but also steered me to choose an interesting topic and asks important research questions.

Next, I would like to thank my committee members: Dr. Ju-Yeon Jo, Dr. Wolfgang Bein and Dr. Sean Mulvenon. I very much appreciate the time they have taken to assess and evaluate my thesis. Special thanks to Dr. Mulvenon, my former graduate assistantship manager, for the valuable advice and wisdom throughout my time at the Office of Research and Sponsored Projects.

Last but certainly not least, I wish to acknowledge the support and great love of my family, especially my mother. They kept me motivated and this work would not have been possible without their encouragement.

Phillipe S. Austria

University of Nevada, Las Vegas

August 2020

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Data today is being produced at an alarming rate. In 2018, an estimated 29 zettabytes were generated globally [1]. By 2025, International Data Corporation predicts data generation to grow to 175 zettabytes, a growth rate of 66%. At the consumer level, IDC estimated 36% of people's personal data is stored on a computer other than their own, often referred to as the cloud. That percentage is predicted to grow to 49% by 2025.

Cloud storage is a popular form of storage and has benefits over storing data on personal hardware such as lower cost savings, higher data integrity and more security protection [2]. According to [3], a few of the most top rated Cloud Service Providers (CSPs) with large market share include Google, Amazon and Dropbox. However, there are also disadvantages as to using CSPs.

The first disadvantage is privacy. CSPs are owned by a single entity, or in other words centralized. The consumer must have trust in the CSP to protect and ensure privacy of the data. Though data is normally encrypted before being stored on servers, CSPs have the ability to decrypt user data, which may concern consumers.

Second is cost. For small businesses and companies, using a CSP may prove to be less expensive; however, it may not be for a single user only wanting to store a relatively small amount of data. A user has the options of storing data on a personal storage device; however, they lose the security that CSPs are able to offer. Is there another alternative?

Blockchain is a technology that allows for decentralized, cryptographically secure and fault-tolerance systems. It was first announced in the Bitcoin white paper by S. Nakamoto revealing a new peer-to-peer currency [4]. The blockchain is similar to a ledger that records transactions between peers. Since the birth of Bitcoin, blockchain strength as a ledger has been leveraged for areas aside from financial technology.

Platforms such as Sia, Storj and Filecoin are using blockchain as an underpinning technology to for de-centralized file storage [5, 6, 7]. They advertise advantages of over centralized CSPs such as more privacy, better security and lower costs. One of the most unique features of these platforms is that they allow users to rent out personal hard drive space in return for money. Thus data is not kept by a single overseeing entity. The Blockchain, which everyone has a copy of, keeps a record of the location of stored data, payments and user transactions.

1.1 The Study

In this study, I evaluate the security, performance and costs of Blockchain-Based Storage (BBS) and compare findings to CSPs using traditional cloud storage (TCS). I specifically focus on the Sia platform as it currently the only production ready product. Other platforms are ether in beta or alpha stage at the time of writing this paper. Furthermore, more I use Google Drive as my comparison platform during the performance evaluation.

Security is a broad term. CSPs generally provide to customers how they will keep their data safe in a written contract called a Service Level Agreement (SLA). SLAs state how the provider will maintain storage metrics such as durability, redundancy, encryption, availability and data access (I define each of the terms in Section 2.1). These agreements however do not exists for BBS currently. The study aims to assess each metric for the Sia network using methods derived both research and industry.

For performance I a simple approach use a program to measure time it takes to upload and download various file sizes. Peer-to-peer (P2P) file storage systems, such BitTorrent, offer performance advantages [8]. In theory, because BBS leverages a P2P system, they in turn should exhibit certain performance gains from end user's point of view. This study hopes to observe the performance gains or losses.

In order to evaluate the cost, a test-bed was built to upload files and record cost related variables. Unlike CSPs which offer simple pricing models, i.e. $10 for 2 TB/Month, BBS pricing is more complex; there are several factors that that have an effect on the overall cost. Fortunately Sia offer API that users to track those factors. The source code is also available for readers to use [9].

Nevertheless, this study purpose is not advocate that one platform is better than the other rather to explore an new, exciting method for consumers to store data.

1.2 Related Works and Research Gaps

Because of the recent interest in using Blockchain for storage systems, little research has been done comparing BBS and TCS. The authors of [68] present a survey comparing more recent BBS platforms with TCS. The authors propose advantages of BBS such as: high security, availability, and privacy. Additionally, the survey claims BBS to use lower bandwidth and be more cost effective through a pay for usage model over monthly fees found in TCS. However being a survey, the study does not delve into technical details and provides little evidence to warrant claims. My study attempts to provide evidence to strengthen the claims in the survey.

Similar research proposes new BBS schemes for personal storage. The authors of [10] propose a BBS scheme of their own and provide a comparison to Filecoin, Sia and Storj. However the comparison does not include TCS in the comparison. Additionally, the study does not provide the methodology and how the results were produced and are challenged by the results of my study. For example, the results in [10] shows Sia to lack data confidentiality, while I concluded otherwise.

An abundance of research exists comparing CSPs to one another [11, 12, 13, 14]. The authors in [11] propose a methodology to understand and benchmark by testing 11 popular personal cloud storage services including Google Drive, Dropbox and Amazon. The study concluded there was no clear winner, each service had room for improvements. Moreover, factors such as application server performance, the server that prepares data for storage, and distance between data centers played a large role in overall performance of storage systems. Surprisingly, some services such as Dropbox were observed to perform better than the authors local distributed storage setup; demonstrating the engineering capabilities of company-backed services.

To date, no study has specifically looked at comparing BBS and TCS at a archecture level and through actual usage of a BBS platform. This study is the first, of my knowledge, to provide that comparison.

1.3 Motivations

Since the creation of Blockchain and Bitcoin in 2008 [4], development has focused on using the technology in the financial industry. According to a report released in 2018, over 5500 cryptocurrencies have been listed on Coinmarketcap [1]. This study displays a use case for Blockchain outside of of financial technology.

[1] https://coinmarketcap.com/

BBS systems are being studied for sensitive data that require access from multiple users, for example, medical data a patient, doctor and insurance company all need private access too. The authors in [15] proposed an alternative storage solution to conventional databases using a permissioned et blockchain architecture. The solution uses IPFS for storage and Tendermint Cosmos[2] as the blockchain ledger [16].

Another use case for BBS is to support the Internet of Things (IOT). The author in [17] evaluates storing IOT on the Hyper Ledger Framework, a blockchain developed by IBM[3]. Another study evaluates storing IOT data on the Ethereum blockchain [18]. The blockchain offers the benefits such as privacy and security; however, faces challenges such as latency and scalability.

[2]https://tendermint.com

[3]https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf

Chapter 2

Characteristics of File Storage Systems

2.1 Storage Security Definitions

2.1.1 Durability

Durability refers to the ability of a storage platform to provide long-term protection against disk failures, data degradation and other corruption [19]. The term is sometimes referred to as Data Integrity. CSPs such as Google, have thousands of machines that regularly experience disk failure which can cause data corruption or loss [20]. Providers must however keep their agreement with customers to ensure their data is intact.

Durability is generally expressed as an annual percentage rate, where 100% means no data loss. For example, if a CSP guarantees a durability of 99.999999999% (11 nines), means the customer could expect to lose one object every 659,000 years. The Durability can be calculated in two ways. The first is based off the probability of hardware failure:

$$Durability = 1 - \left(\frac{AFR}{\left(\frac{365}{MTTR}\right)} \right)^{n_{parity}} \tag{2.1}$$

where:

$AFR =$ is the Annualized Failure Rate

$MTTR =$ is the Mean Time to Repair in hours

$n_{parity} =$ number of parity nodes

The second durability equation is based on the Poisson Distribution [21]:

$$Durability = 1 - P(k) \tag{2.2}$$

$$P(k) = e^{\lambda} * \left(\frac{\lambda^k}{k!}\right) \tag{2.3}$$

$$\lambda = \frac{AFR * k}{\left(\frac{h_{year}}{MTTR}\right)} \tag{2.4}$$

where:

λ = average number of continuous events given a time interval

k = number of events (parity nodes)

e = exponential constant

h_{year} = hours in a year

Industry has two main methods to maintain durability: (1) use software algorithms to detect corruption and (2) use redundancy [21]. Erasure coding such as Reed-Solomon coding, is an approach to detect such correction and is widely used in storage media such as CDs, DVDs and RAID configurations [22]. Redundancy refers to the method of storing multiple copies of the data in multiple locations. When a file server becomes corrupt or crashes, another server can take its place. The next section discusses redundancy more in detail.

2.1.2 Redundancy

Data redundancy is a technology intended to prevent data loss and provide a real-time fail-safe against hard drive failure. The common notation used in SLAs is N+X, where N is the original copy and X represents the number of backups. Redundancy not only increases data integrity but also availability (described in the next section); however, having too many copies leads unwanted storage overhead. Two redundancy methods commonly used in file storage storage systems are replication and erasure encoding [22].

Replication writes multiple copies of a file to disks, normally in different physical locations. A replication factor determines how many copies of the object will be created. A commonly used factor is three – two copies and the original file are written to disk. Moreover, three times more storage is consumed during this process. A 100 MB file requires 300 MB of storage space.

Erasure Coding (EC) splits a file into smaller pieces, often called blocks or chunks. The blocks are encoded using erasure codes [23], resulting in the creation of special blocks called parity blocks. The parity blocks are used to reconstruct new blocks in the event the original data blocks are lost or become corrupt. The encoding process uses matrix algebra to create and is well described in a popular encoding scheme called Reed-Solomon [24].

EC uses a M of N scheme, where M blocks of N total blocks are needed to reconstruct the file. The number of parity blocks is (N-M). For example, 4 of 6 scheme partitions a file into four blocks and creates two parity blocks. Any four of the six blocks is needed to reconstruct the file. Furthermore, the scheme could handle the loss or corruption of any two blocks, and the original file could still be re-constructed.

There are advantages and disadvantages to using Replication and EC. A study in [25] concluded Replication required an order of magnitude more bandwidth, storage and disk seeks more than an EC system of the same size. In contrast, EC is more CPU intensive than Replication. Each new or modified block requires computation to encode and modify parity blocks, whereas in Replication, objects are simply copied and written to disk. Additionally, EC incurs higher latency because computing parity blocks take time.

The two redundancy methods are well studied and have been used industry for decades, providing system architects with the knowledge on how to choose the methods that best suit their file storage system. The authors in [22] concluded Replication is better suited for "warm" data, data accessed frequently, while EC is better suited or "cold" data, data infrequently accessed.

2.1.3 Availability

Availability quantifies the time a CSP guarantees that data and services are available to customers [26]. The value is generally given as a percentage per year. If availability in a SLA says 99.9%, that equates to approximately 8.77 hours of downtime per 365.25 days. Equation (2.5) shows one way to calculate availability using the meantime to repair (MTTR) and meantime to failure (MTTF) [27]. MTTR is the average of the amount of time needed to repair storage servers and MTTF is the average time between storage failures.

$$Availability = \frac{MTTF}{MTTF + MTTR} \tag{2.5}$$

where:

$$MTTF = \text{is the Mean Time to Failure}$$

$$MTTR = \text{is the Mean Time to Repair}$$

However, Equation (2.5) calculates the availability of a single machine or node system. A distributed storage systems uses multiple nodes, necessary and spares. Given a spare node (s), the probability of losing that node is (*1-a*), where a is availability of a single node. The probability of losing *s+1* nodes is $(1-a)^{s+1}$ [28].

To find the availability of multi-node system, I start with the number of ways *s+1* nodes can fail, f (Equation (2.6)). Next, f is used to find the probability of failure, F, for an n node system with s spares (Equation (2.7)). Lastly, the availability is one minus the probability of failure (Equation (2.8)).

$$f = \frac{n!}{(s+1)! * (n-s-1)!} \tag{2.6}$$

$$F = f * (1-a)^{s+1} \tag{2.7}$$

$$\text{availability} = 1 - F \tag{2.8}$$

where:

$$n = \text{total nodes}$$

$$s = \text{spare nodes}$$

$$a = \text{single node availability}$$

2.1.4 Encryption

Encryption in regards to data storage is a security method that transforms data such that the data becomes unreadable. Only those in possession of the encryption key can decrypt and access the data. The TCS platforms discussed in this study use the Advance Encryption Standard algorithm[1] to encrypt customer data [20, 29, 30]; AES is highly secure and performant [31].

[1] https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Data is encrypted by the application server (Figure 2.1). When users upload data to a TCS platform (1), a secure connection is normally provided and data is encrypted by the TCS/SSL protocol (2). Once data reaches the application server, it is unencrypted and re-encrypted again by the server before storage (3). By having the application server perform the encryption, allows the cloud provider to have possession of the encryption key. The CSP is now able to decrypt customer data.

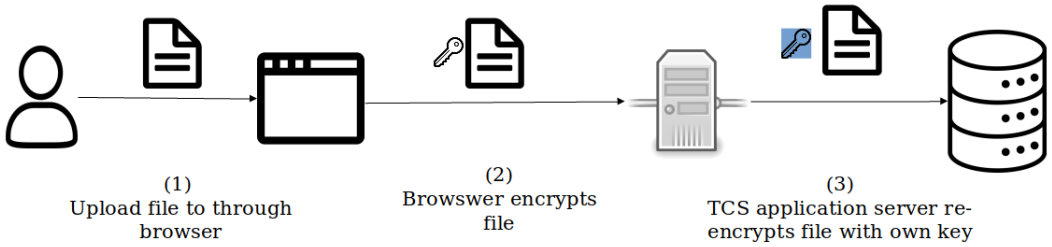


Figure 2.1: Traditional cloud storage encryption sequence.

Encryption prevents the data from being readable if it is stolen, although does not prevent the CSPs themselves from accessing and reading the data. Customers must trust their CSP to not violate privacy. I latter evaluate of risk of not encrypting data before uploading to the cloud in Security Evaluation (Chapter 4)

2.2 Distributed File Systems

In a traditional file system (TFS), data is organized by its physical location. Hard Drives, often called shared drives, are attached to file servers and shared amongst multiple clients. Clients then map the shared drives to its local network in order to see and access data on those drives (Figure 2.2). If a client needs to access a file on drive F, the address is `\\host-a\F\<file-name>`.

There are issues with TFS that make it difficult to use in large scale storage systems. One issue is that a TFS places the responsibility of being aware of shared drives on the clients. If a new file server or shared drive is added to the network, the client is unaware unless told about the new addition.

Duplication and version control pose another issue. Lets imagine two copies of a file server and all its shared drives exist, sever A and server B. The servers are in different physical location. If a client mapped to server A moves to the location of server B, his or her map is still pointing to server

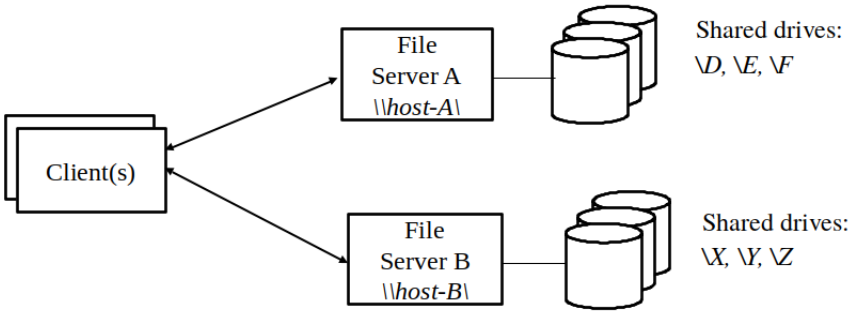


Figure 2.2: Traditional file system.

A, which is farther away. The client experiences high latency and adds more load the network. Moreover, if a duplicated shared drive has a file that is updated by one client, the other clients are unaware of that change. Managing locations and version control becomes complex as the system grows.

In a distributed files system (DFS), clients connect to a distributed file server and see shared drives as a single drive, as if it is attached to its local machine (Figure 2.3). A DFS uses a common naming system that arranges the shared drive in a unified logical view, allowing the data to be easily found. Client are no longer burdened with having to be aware aware of new file servers and shared drives

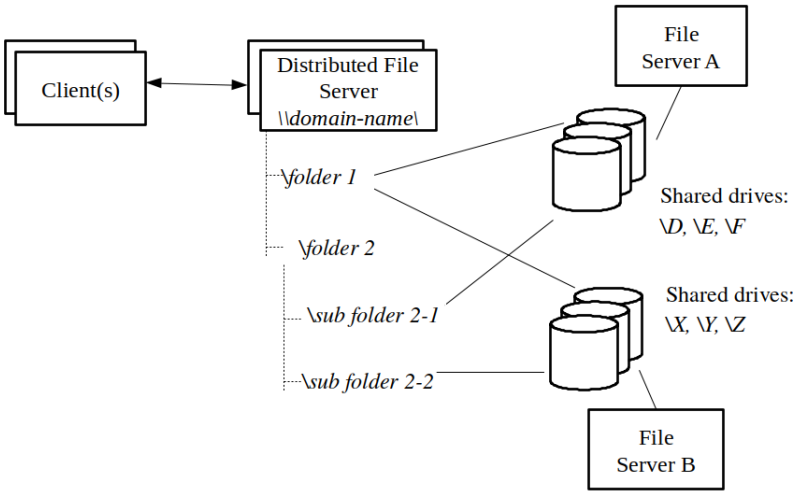


Figure 2.3: Distributed file system.

DFSs offers many benefits over TFSs. They handle multiple users with ease, present a consistent

view of shared drives, allow data access transparency and have more efficient load balancing [32]. Additionally, DFSs are more scalable and fault-tolerant, attributes needed by CSPs and their large scale services.

Figure 2.4 shows a DFS's general architecture. The application server partition files in smaller parts, called chunks, then store the chunks on different physical machines called chunk servers. Like a DFS, a uniform naming conversion is used to map the locations of chunks. Generally the location stored using a hierarchical tree structure, with the node of the tree representing the directory; however, this structure depends on the implementation of the platform.

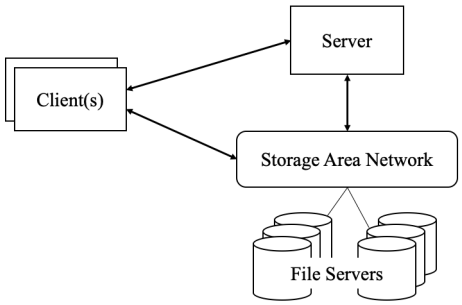


Figure 2.4: Distributed file system adapted for file storage.

2.3 Traditional Cloud Storage Systems

In this section, I discuss the architecture of three popular traditional cloud storage (TCS) services: Google Drive (GD), Dropbox and S3. Figure 2.1 shows the user base and current prices each file storage service charges. Amazon Web Services (AWS) does not release information about its S3 user base. Additionally, information on S3's architecture is also limited as AWS does not share technical details about its property service (discussed further in 2.3.3).

Table 2.1: Cloud storage services usage and prices.

	Google	Dropbox	AWS
Storage service	Google Drive	Dropbox	S3
Launch date	2012	2007	2006
User base	1 billion	12.7 million	n/a
Price	$9.99 (2 TB/Mo)	$9.99 (2 TB/Mo)	$4.00 - $23.00 (TB/Mo)

2.3.1 Google File System

GD is a file storage synchronization service created by Google in 2012[33]. While GD represents the application service, what the customer interacts with, data is actually stored using the Google File System (GFS) [20]. For this reason I focus on discussing the GFS architecture rather than GD. Before building GFS, the system architects had four assumptions:

1. Component failures were expected.
2. The system needed to handle large file sizes.
3. The system should handle many operations, majority appended new data.
4. The system should be flexible.

The GFS uses a master and slave architecture (Figure 2.5) [20]. The master stores metadata – information about the stored data itself – while the slaves, known as chunk servers, store user data. The GFS client represents applications that use GFS such as GD and Google Cloud Services. Clients do not read data through the master, instead they request the data location from the master server, then connect directly to the chunk servers. This helps to reduce load on the master.

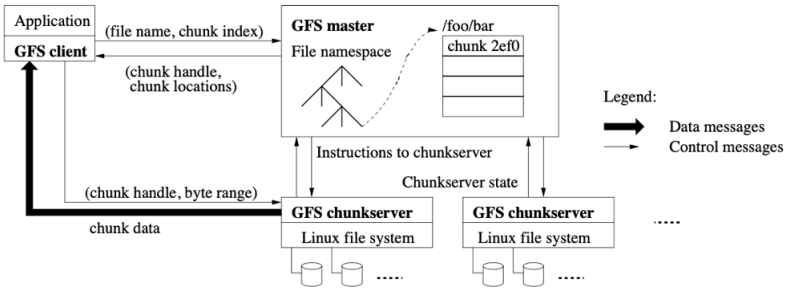


Figure 2.5: Architecture of the Google File System.

The GFS master stores three major types of metadata: the file and chunk namespaces, the mappings from files to chunks and location of chunk's replicas. Files are segmented into chunks; although Google does not disclose what kind of method or techniques the architects use for segmentation. Chunks size is a key design parameter and the architects choose 64 MB chunks, a relatively large chunk.

Chunk sizes play a role in performance. The architects observed larger chunk sizes have help to reduce load on both master and chunk servers by reducing client server interactions. Furthermore,

persistent TCP connections are maintained longer, reducing the amount of initiating connections. Larger chunk sizes however decrease performance with smaller sizes, specifically if the files are less than 64 MB. Small files must be padded with filler data before being stored.

The master servers main responsibility is storing the locations of the chunks. GFS initially had the master keep a persistent log of the chunks locations; however decided a simpler approach to have the master request chunk locations on startup. Chunk servers often leave and join the cluster, fail, restart, have their names changed, thus keeping masters in sync with chunk servers became a difficult tasks.

2.3.2 Dropbox

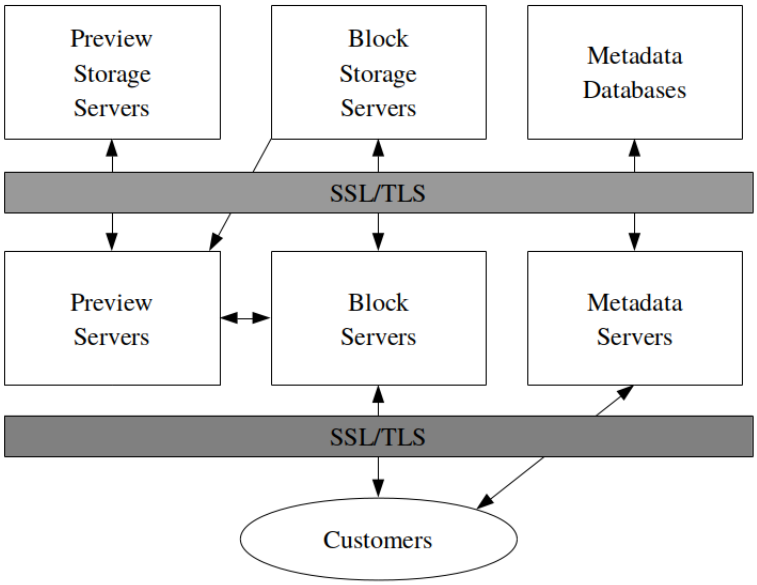


Figure 2.6: Dropbox architecture.

Dropbox's architecture consists of six components that work together to provide a performant and reliable file storage service (Figure 2.6). The following components represent servers and databases: block servers, Block Storage Servers, Metadata Server, Metadata Database, Preview Servers and Preview Storage Servers [29].

The Block Servers receive the data customers upload and partition the data into blocks. Blocks are synonymous to chunks in other platforms. Blocks are then encrypted and synchronized with its metadata; however, only new or modified blocks are synchronized. Block servers are also responsible for delivering files back to customers.

Block Storage Servers store the blocks after they have been encrypted and synced with its metadata. Each block is content addressable. Its address derives from the content's hash. This content addressable scheme allows for easy retrievability of individual blocks, rather than having to retrieve all the blocks that make up a file.

The Metadata Servers gather information about the customer and their stored data . Such data include the account owner name, email, file names, file types and file version. The metadata also serves as an index for the user account. The metadata itself is stored in the Metadata Database, which uses a MySql database. The database is replicated as performance and availability needs increase.

Drop allows customers to preview their stored data without having to download them. Preview Servers are responsible for retrieving the file's blocks from the block storage servers and producing a preview of the file. The Preview Server itself does not serve the preview to the customer, but sends it to the block server first. The block server then serves the preview to the customer. Files that are accessed frequently are cached in the Preview Storage Server allowing for increased file retrieval performance.

Figure 2.6 shows that data transferred between the servers and databases are encrypted using the SSL/TLS security protocol. Dropbox encrypts blocks using 128-bit and higher AES while files are in transit. Once the blocks reach the destination server, they are decrypted for usage. Blocks at rest are encrypted using 256-bit AES.

Despite the multiple components with Dropbox, the platform operates the same as general DFS. Files are broken down into smaller pieces and distributed across multiple servers. Additionally, the locations of the smaller pieces are stored in a central server, allowing the scattered data to be viewed in a unified, logical manner by clients.

2.3.3 AWS S3

Simple Storage Solution (S3)[2] is a distributed object storage solution offered by AWS, a subsidiary of Amazon. S3 was developed for a wide variety of applications beyond file storage such as: supporting websites, IOT, mobile applications and big data analysis. For archiving and long term file storage, AWS created S3 Glacier[3]. Although AWS does not publicly provide the details of S3's technical design, the company openly states on its product documentation[4] that S3 uses an distributed object

[2]https://aws.amazon.com/s3
[3]https://aws.amazon.com/glacier
[4]https://aws.amazon.com/what-is-cloud-object-storage

storage architecture.

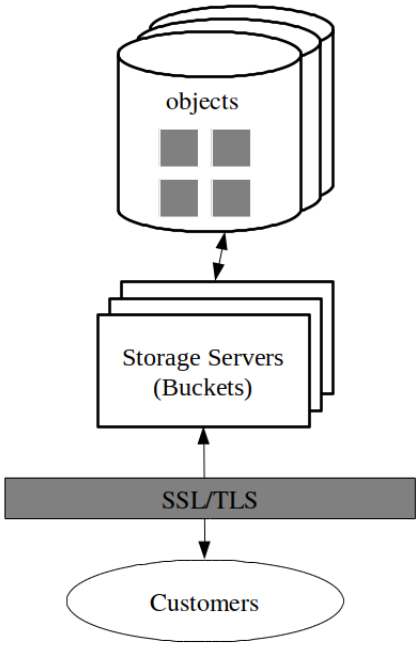


Figure 2.7: S3 Architecture.

The architecture of S3 is comprised of three entities (Figure 2.7):

- **Buckets**: are containers of objects and the root namespace for a group objects.
- **Objects**: the fundamental entities of S3. Each object contains the (stored) data itself, metadata about the object and a key.
- **Keys**: serve as unique identifiers of objects in a bucket.

Because S3 is web focused, objects are accessed through a web URL. Apart from being the root namespace, a bucket name serves as the root domain name in the URL. For example, a bucket named *pictures* would have a URL called *https://pictures.west.amazonaws.com/*, where west is the region the bucket is located physically. The key completes the URL, which includes the location within the bucket and the object name. Continuing with the example, a picture named *beach.jpg* in a folder called *2019* would have the URL *https://pictures.west.amazonaws.com/2019/beach.jpg*, where *2019/beach.jpg* serves as the key. A version number is added to a key when duplicates exist.

Though much of the underlying technology in S3 is proprietary, there are notable features in the documentation related to the scope of study. First, S3 uses replication for redundancy, which

is better suited for object storage than erasure coding (refer to Section 2.1.2). Second, S3 allows access control to buckets and objects to increase security. Lastly, objects can act as seeds in the BitTorrent network, allowing faster downloads for others; however, the feature comes with extra cost.

2.3.4 Summary

In summary, cloud service providers use various architectures and DFS implementations to provide fault tolerant and highly available file storage systems to its customers. Table 2.2 shows the level of service each provider agrees to uphold – the Service Level Agreement (SLA). Regardless of architecture, Google Drive, Amazon S3 and Dropbox claim to provide the same level of security.

Table 2.2: Service level agreement summary in traditional cloud systems.

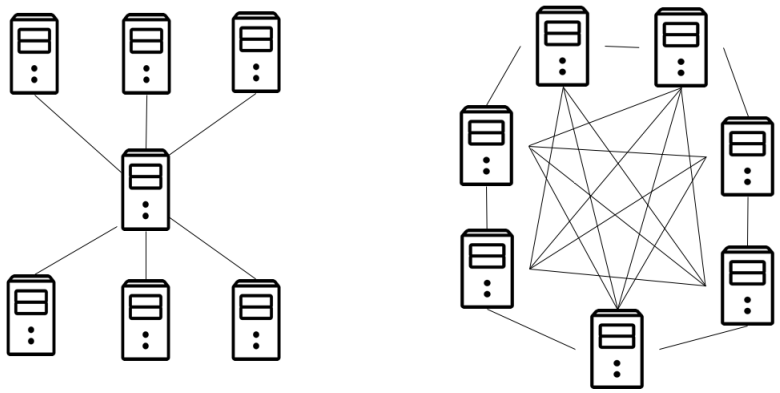
SLA items	Google Drive	Amazon S3	Dropbox
Encryption	AES	AES	AES
Redundancy Factor	N+2	N+2	N+2
Redundancy Method	replication	replication	replication
Durability (# of nines)	11	11	11
Availability (# of nines)	3	3	3

2.4 Peer To Peer Distributed File Storage

2.4.1 Peer To Peer Computing

The Peer To Peer (P2P) concept serves as one of the core methodologies in technologies being studied in this research. There is no exact definition for P2P, but can be thought of as a set of concepts and mechanisms for decentralized distributed computing and information exchange. Rather than multiple computers communicating with a single machine, known as a client-server model, computers find and interact with each other (Figure 2.8).

P2P offers advantageous over centralized systems such as having ability to share resources, scalability and fault tolerance [8]. In file storage and content delivery systems, P2P networks can improve the speed of delivery as more peers join the network [34]. There are however disadvantages as well. Retrieving and maintaining a global view of a P2P systems is more difficult. Additionally, performance can be compromised when when the system lacks peers. A study on the P2P file



(a) (b)

Figure 2.8: Client-Server architecture (a) and Peer To Peer system (b).

sharing system, BitTorrent, observed poor service availability when peer participation was low, which negatively affect download performance [35].

2.4.2 BitTorrent

BitTorrent (BT) is a communication protocol developed in 2001 for P2P file sharing. With over 45 million daily active users in 2016, BT is one of the most popular [36]. The protocol breaks files into pieces and distributes them to participating nodes (peers). Peers can download pieces in parallel rather than sequentially, resulting in faster download speeds. In this section, I discuss BT's architecture, file sharing sequence and notable features.

Architecture

A network using BT is composed of four main entities:

- **Peer** - a general term for nodes who upload and download pieces.
- **Seed** - a specific peer that has a file in its entirety and is sharing the file for other peers to download.
- **Leecher** - a peer downloading piece(s) of a file it needs from another peer.
- **Tracker** - a peer that keeps a log where pieces of a file are located and manages the file transfer process.

- **Torrent** - a file that contains metadata about a file being shared such as: file name, size, and hashes of the pieces.

Peers require a program to connect to other peers over the internet called a torrent client. The client also allows the creation and usage of torrent files. Currently there are many clients available; some popular ones are qBittorent, Vuze, uTorrent and BitTorrent [37]. Once peers are connected to each other, they can begin to share and download files.

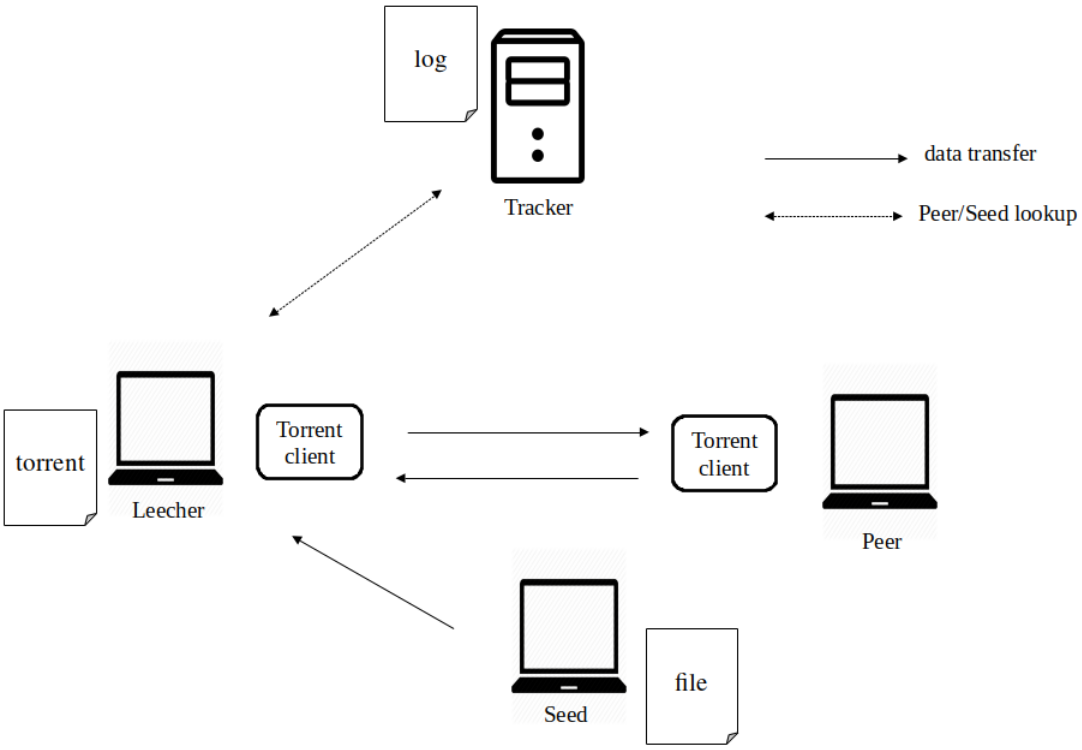


Figure 2.9: BitTorrent architecture.

File Sharing Sequence

The protocol for file sharing is outlined below:

1. Leecher obtains a torrent file and one seed must exist.
2. Client connects to a tracker.
3. The tracker directs the leecher to a seed and shares a log of all peers who have piece(s) of a desired file.

4. The leecher begins to download and pieces from seeds and other peers. The tracker helps to facilitate trading.

5. Once leecher gathers all pieces, it becomes a seed.

Trading pieces is an important step in the protocol, as it helps to distribute a file and increase its availability. BT uses a strategy derived from game theory, called tit-for-tat [38]. The algorithm is based on cooperation and reciprocity. When a leecher receives a needed piece from another seed or peer, the leecher has the opportunity to give a piece (from any file) in return. Peers start off by trusting each other, so if a Leecher does not have a piece to trade for one it needs, the peer/seed will still give the piece. Leechers can simply refuse to trade even if they have pieces; however, this situation is mitigated by the protocol's specific features.

Features

Peer Ranking and Choking

As peers trade more pieces, the higher their rank increases. Those who continually refuse to trade and only download pieces have their scores lowered. The ranking system is essentially a measure of participation. The higher the rank a leecher has, the more willing peers will be to give that leecher the piece(s) they need, increasing download speeds. Leeches with a lower rank have a higher chance of being refused a piece, referred to as choking. Choking increases download time since the leecher must look elsewhere for a needed piece. Leechers are chocked for a variable period of time, but are eventually unchoked. Although, they can be repeated choked as long as their peer ranking remains low.

Policy's

Within the protocols there are policies that dictate trading pieces and increase peer ranking. A few important policies are strict rarest first policy and random piece first [39]. Trackers helps find pieces that are harder to find, or pieces that have the least copies. The rare pieces are usually held by seed(s). Once a leecher gets a hold of the rare piece, the leecher can now trade off the piece as quickly as possible. This helps to spread the rarest piece and increases availability of the piece for the next leecher.

There are however times when the rarest first policy is overruled, such as when a leecher has no piece to trade. This policy helps to find the leecher any piece as soon as possible so that it can be

used to trade. Once the leecher has a piece, the rarest first policy takes over and the leecher starts to trade for rare pieces. There are several other policies used in the BT protocol, many of which are based on game theory to maintain a healthy trading economy.

Decentralized Trackers

Decentralized tracking was introduced in May 2005, removing the need for Trackers. Instead of the Tracker keeping a log of seeds and peers of a torrent, each peer now keeps a relatively small portion of the log using a DHT. Using a Kademlia-based algorithm, Mainline DHT, allows peers to calculate which parts of the log store are in local memory. Decentralized tracking mitigates the heavy traffic a Tracker incurs and distributes the burden amongst the Peer. Although, a study by [40] concluded DHTs do not have better performance than Trackers regarding availability. Trackers still exist in today's network to improve the speed of peer discovery.

2.5 Blockchain-Based File Storage Systems

2.5.1 Bitcoin Blockchain

The Blockchain is an immutable, cryptographically secure distributed ledger with byzantine fault tolerance [41]. Blockchain first introduced in the Bitcoin white paper by Nakamoto [4] and serves as a core data structure for consensus between participants. Since 2008 the Blockchain technology and the Bitcoin network has laid the foundation for many innovations, products and new research. Blockchain is becoming a core technology in creating de-centralized services. In this section, I provide a high level view of the Blockchain architecture, focusing on the Bitcoin protocol.

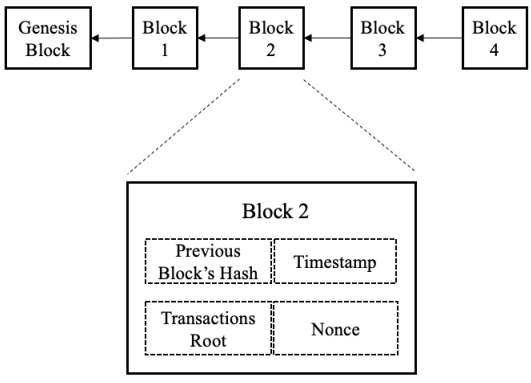


Figure 2.10: Block headers in a blockchain.

The Blockchain can be thought of as an append-only database in which every user has a copy of

the database. Similar to a tradition database, a Blockchain contains records, records that represent various types of data such as texts and files. Each record is recorded as a transaction, containing the record along with metadata. Furthermore, records are grouped by blocks which holds a limited amount of transactions – an amount decided upon when designing a Blockchain. Figure 2.10 shows a conceptual diagram of a Blockchain.

The amount of transactions is limited by the blocksize. In the Bitcoin network, the blocksize is 1 MB, thus each block can only hold 1 MB or less worth of transactions [4]. The blocksize also impacts network performance. Because blocks are downloaded in a peer-to-peer fashion from participating nodes, a blocksize limit ensures the network overhead will remain low and less time will be needed for peers to share block information [42]. The next sections describes how the mining process creates blocks.

Mining

The Bitcoin mining process is outlined below:

1. Miners gather transaction that have been broadcasted to the network.
2. Each transaction is validated and hashed using the sha256 algorithm.
3. A Merkle tree is formed from the individual hashes to form a single hash called the Merkle root
4. The miner uses the Merkle root, block variables, and a nonce to find a target hash. The miner continuously changes the nonce until the hash is found.
5. The first miner to find the target hash receives bitcoin as reward.

This consensus algorithm is referred to as Proof of Work (PoW), named after the work miners need to expel to find the target hash. Because PoW uses a large amount of energy in the form of electricity, the algorithm been proven to be an effective method to prevent nodes from hacking the network. Nodes would eventually be spending more on energy than the value they are trying to steal.

Transactions & Merkle Trees

Transactions are actions between nodes. In the case of Bitcoin, transactions represent the transfer to currency, Bitcoin, from one node to another. Transactions have the ability to represent many

actions and can been seen in various Blockchain platform. Each transaction is individually hashed and further hashed to created Merkle Tree, shown in Figure 2.11. The root hash is stored in the block header. Therefor any node that tries to change any transaction in the Merkle tree, will cause a large change in the root. The change would be caught by honest nodes and invalidate the block.

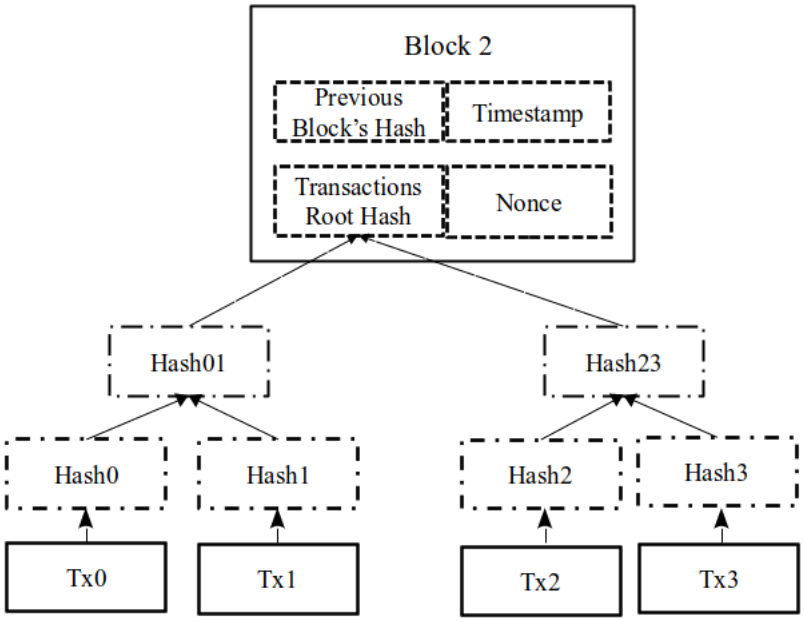


Figure 2.11: Blockchain transactions in a Merkle Tree.

Scalability Issues

Blockchains have two main scalability weaknesses: the transactions rate and the growing size of the Blockchain itself [42]. The transaction rate is generally described in transactions per second (TPS) and is effected by the block size and the block time – the average amount of time to mine a block. For comparison, Bitcoin can process about 7 TPS and Ethereum 15 TPS, while Visa can process 24,000 TPS [43]. BBS platforms that use blockchain, like Sia, inherit these limitations as well, which I discuss in the following section.

2.5.2 Sia

Sia is a Blockchain-based decentralized storage network, created by Nebulous Labs [5]. Through combining distributive file storage, P2P and Blockchain technology, Sia creates a marketplace to

allow users to rent out storage space in return for Siacoins (SC) without central authority.

Sia is composed of three types of nodes:

- **Renters**: users storing their data by paying (SC).
- **Hosts**: users that offer up personal storage for rent in return for SC.
- **Miners**: similar to miners in the Bitcoin network, miners in Sia miners validate transactions and participate in a mining process for the chance to win SC (refer to Section 2.5.1).

Every node has a copy of the blockchain stored locally. Each copy strengths the integrity of the system. Table 2.3 shows the specification of Sia's blockchain. It is similar to Bitcoin's in that both use PoW consensus algorithm and have a 10 minute block time. Other characteristic such as hashing algorithm, block size and transaction size are different.

Table 2.3: Sia blockchain specifications.

Item	Value	Unit
Consensus Algorithm	PoW	n/a
Block Time	10	minutes
Hashing Algorithm	Blake2B	n/a
Block Size	2	MB
Transaction Size	32*	KB

**32 KB is the max, average is 16 KB*

File Contracts

The file contract is type of transactions that allows the blockchain to record and manage where renters' data is being stored. In essence, a file contract is an agreement between the renter and host. A fee is charged by the host to the renter in order to create the contract. The fee covers bandwidth and computation resources for miners validating the contract. Important information in the contract include: storage window proof, conditions for successful contract fulfilment as well as failed contract fulfillment and the file Merkle root.

Sia creates a minimum of 30 file contracts, 50 is the default amount[5]. The default storage time is 3 months, after which the hosts has a small time window to provide proof – in the form

[5]Renter has the ability the choose the amount

of a hash – that they are in possession of a renter's data. This window is called the storage proof window. That window is typically 24 hours. If the host cannot provide the proof, they will forfeit the collateral and revenue. Formation of storage proofs are further discussed in Section 2.5.2.

Once the contract is created, the renter can begin storing data on the host. Renters upload data 4MB at a time, referred to as a slice. The file contract does not store the data itself, but only a hash of the slices using a Merkle tree data structure (refer to Figure 2.11). The next section discusses storing data more in detail.

Data Storage Process

In order to store data in a distributed and reliable manner, Sia store renters data using the following sequence:

1. Files are segmented into 40MB chunks.
2. Chunks are replicated by the redundancy factor.
3. Each chunk is further divided into 4MB slices.
4. Slices are encrypted using Threefish encryption scheme.
5. Slices are uploaded to hosts.
6. The file contract is updated.

Files are segmented using the Reed-Solomon erasure encoding (refer to Section 2.1.2). Each chuck is then replicated according to the redundancy factor. The default factor is three, but the renter can choose any factor. Although high redundancy factors will incur high costs.

Each chunk is then further segmented into 4 MB slices to prepare the data for encryption and upload to hosts. Sia chose to use the Threefish based on performance and security; however do not provide comparisons to other algorithms in their white paper. Because encryption is performed by the renter, slices are encrypted before uploading to hosts. Thus hosts do not have the ability decrypt the slices. Sia distributes the slices are distributed amongst 50 hosts or how many file contracts the renter has created.

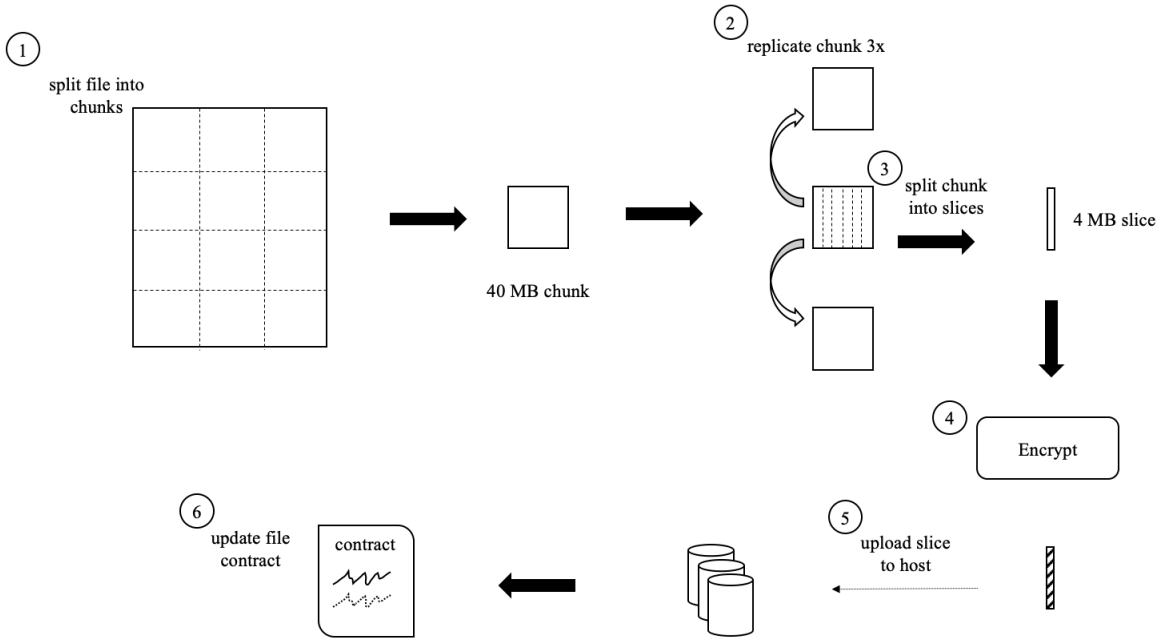


Figure 2.12: Sia data encoding and storage sequence.

Collateral

Sia requires hosts to put SC in a file contract alongside renters storage fees when a contract is formed, the amount is called the collateral. The collateral is a security measure in place to increase the trust in hosts and to promote honesty. If a host cannot fulfill a file contract, the collateral and the renter's payment is lost. Examples of not fulfilling a contract include having less that 97% uptime and not being able to produce a proper storage proof. The collateral is generally two to three times the storage fee, thus losing the collateral leads to significant profit loss and incentives hosts to act honestly.

Proof of Storage

After a contract expires, a hosts as a small window to prove that they are storing the data agreed upon in the file contract. The window is called the storage proof window and is normally 24 hours, however, the window is given in terms of block height, thus the window varies depending how fast miners mine and append blocks. During the storage proof window, a hash is chosen at random from the file contract and the hosts must then reproduce the hash by hashing the data that they themselves are hosting. If the hosts cannot produce the storage proof before the window closes,

the hosts forfeits the collateral and all revenue from storing the data. Revenue includes bandwidth fees.

The proof of storage concept's is implemented by other centralized platforms such as Filecoin and Storj. Though not impervious to cheating the system, such as faking a storage hash, the mechanism is a probabilistic and the chances of cheating decreases decrease as time moves. Furthermore, risking of losing collateral and revenue make the costs of cheating more expensive in the long run.

Redundancy and Repairing

Sia uses a 10 of 30 erasure encoding scheme for redundancy (refer to Section 2.1.2). As mentioned in Section 2.5.2, Sia creates 30 file contracts minimum where each contract holds one of the 30 erasure encoded block. Ten of the 30 hosts are needed to restore the renter's file(s). Additionally, the 10 of 30 scheme implies the renter's data is replicated 3 times.

The other 20 hosts are referred to as parity nodes and provide fault tolerance. If a one of non-parity hosts becomes unavailable, another is chosen by the Sia to provide the slices of data. Once 20% of parity hosts become unavailable, Sia begins a repair process. Unavailable hosts are replaced with new hosts and file contracts. Because of erasure encoding, data that was lost by unavailable nodes can be re-created by parity nodes and copied over to the new hosts.

Hosts Scoring

Similar to peer ranking in BitTorrent, hosts in Sia are given a score based on their participation. Higher scores allow hosts to receive more contracts than those with lower scores. Sia's scoring system values hosts reliability and fair pricing over performance. The following criteria impact a host's score:

- **Uptime**: hosts are penalized for having an uptime less than 98%.
- **Pricing**: lower fees increase the score; this includes all fees: bandwidth, contract and storage fees.
- **Collateral**: the more collateral a host puts in a contract, the higher the score becomes.
- **Version**: using a lower version decreases the score.
- **Storage Remaining**: hosts with less than 4TB of storage are penalized.
- **Age**: the longer a host has participated on Sia, the higher the score becomes.

Scalability

Because Sia Blockchain is the Proof of Work algorithm like Bitcoin and limited block size, it is inherently susceptible to scalability issues that Bitcoin faced: low transaction rate and the growing blockchain. In order to prevent the system from being overwhelmed, Sia only writes specific transactions to the Blockchain:

- Sia coin transfers
- Contract formation
- Allowance and collateral posting
- Storage proof
- Final contract revision

Majority of activity on Sia is comprised of updating the file contract. The file contract can be updated for the following reasons:

- Uploading new data to the hosts
- Deleting data from the hosts
- Downloading data

These transactions are done off chain and are not committed to the Blockchain. Off Chain transactions can be executed 100 per second (Figure 2.13[6]). Only the file revision of the contract is written to the Blockchain.

The growing size of the Blockchain is still a concern. Currently, the size of the Sia Blockchain is 18 GB, a growth rate of approximately 3 GB per year (Figure 2.14[7]).

2.5.3 Storj

Storj is a cloud storage solution that strives to provide a secure, private and decentralized system [6]. Like Sia, Storj allow users to rent out storage in return for cryptocurrency. Storj however does not use the blockchain for data management, but only to handle its cryptocurrency, STORJ.

[6]https://siastats.info/transactions
[7]https://siastats.info/blockchain_size

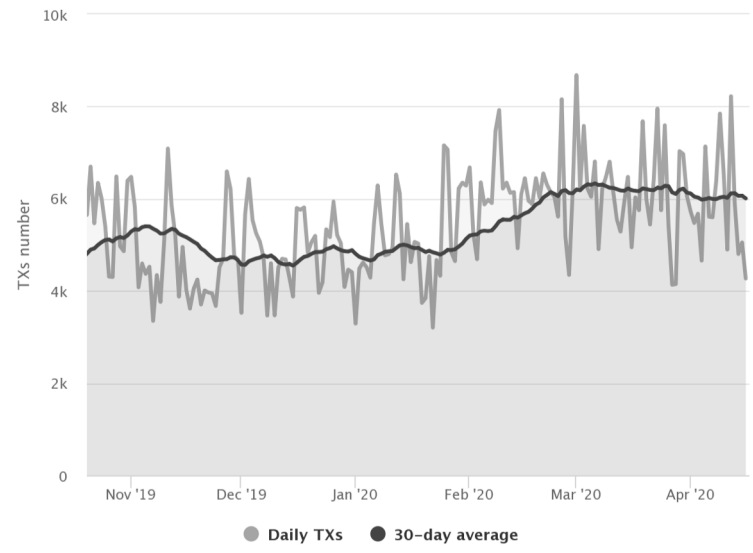


Figure 2.13: Sia network daily transactions.

Avoiding blockchain's scalability hurdles allowed the creators wanted to achieve a low latency and high throughput system.

Storj has three main system actors: storage nodes, satellites and uplinks (Figure 2.15). The creators refers to those particular actors as peer classes. Storage nodes store and return data. Additionally, storage nodes get paid in STORJ for storage and bandwidth usage. Satellites perform multiple duties such as: caching node addresses, storing metadata about stored files, maintaining node reputation and aggregate billing information. The last peer class, uplink, is a client application that encrypt data, erasure code data and communicate with other peer classes. Therefore a user must have an uplink installed in order to use Storj.

The data hierarchy shares a similar structure to Amazon S3 for compatibility reasons later discussed in the section. At top of the hierarchy are buckets, collections of paths. Each path contains a file's location and unique identifier. Files, called objects, are the base data type in Storj. They are unstructured, have no minimum or maximum limit and are composed of blocks of erasure coded data called segments.

Storj uses erasure coding (EC) during its storage process (Figure 2.16). EC allows the system to achieve high data redundancy and durability (refer to Section 2.1.2). After an uplink selects a file to upload (1), the file is partitioned into parts called segments (2). The segments are further partitioned into 4 MB slices called stripes (3). The stripes are then encrypted using AES and erasure encoded, creating shares (4, 5). Shares are aggregated into groups called pieces (7), which are uploaded to storage nodes (8). Lastly, satellites peers update their log to reflect the location of

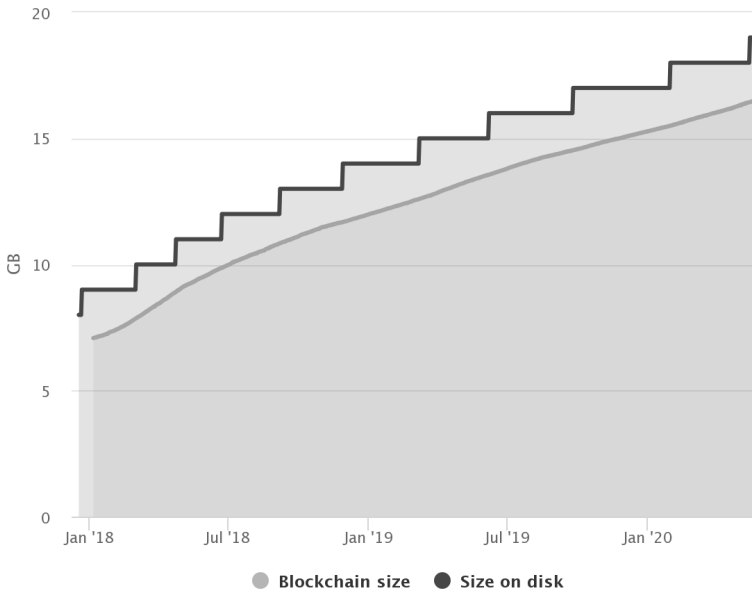


Figure 2.14: Sia blockchain size.

the pieces (9).

The creators proposed EC was better suited than replication for Storj's decentralized systems. Unlike Sia, which has a default 10 of 30 EC scheme, Storj changes based the load of the system. Schemes such as 20:40 and 40:120 are used. Higher parity is better for data redundancy, durability and availability. This however, requires higher bandwidth to maintain the parity data, especially during data recovery; all of which will cost the client more STORJ.

Storj does not have a native Blockchain, hence Ethereum and the ERC20 protocol is used to manage STORJ [44]. Satellites are responsible for recording billing information and controlling payment transfers to storage nodes' wallets on a monthly basis. Additionally, satellites are also paid for the duties they perform; however the details are not discussed in the white paper. Storj's decision to use ERC20 gives clients the ability to pay with various types of payments such as Ethereum and Bitcoin, which can then be exchanged for STORJ automatically and without client involvement.

Table 2.4 shows an comparison between Storj's and Sia's features. Though both platforms have similar features, the implementations are different. Their proof of storage algorithm is an example. In Storj the satellite erasure codes data segments, then asks storage nodes to present an identical result[8]. While in Sia, the file contract chooses a random hash from the data Merkle tree, then the hosts must reproduce the same hash. The features are well described in the white paper. What

[8]Referred to as Auditing in [6]

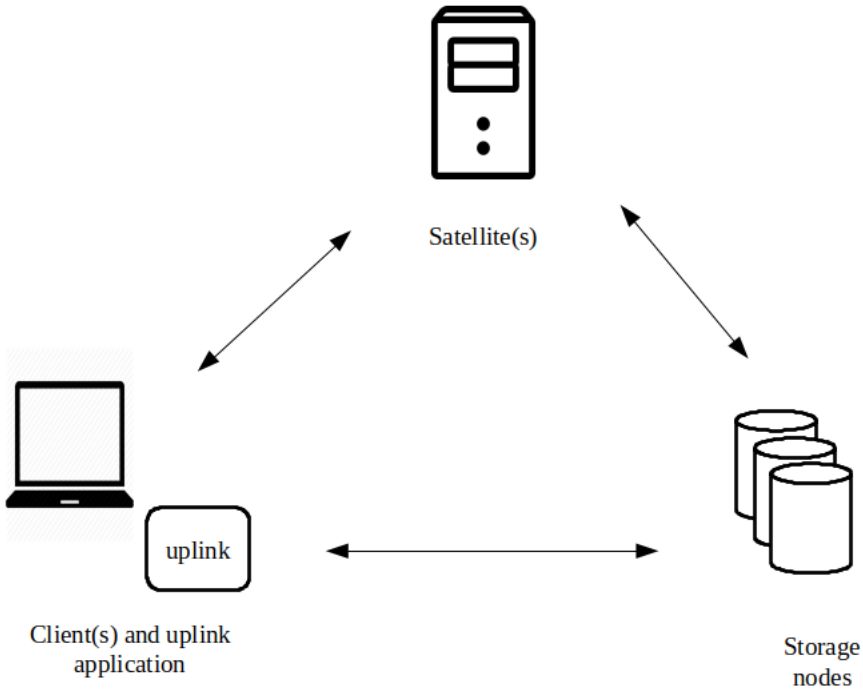


Figure 2.15: Storj architecture.

currently makes Storj unique is its compatibility with Amazon S3. It is the only platform, as I know, that natively integrates with a TCS.

Table 2.4: Architecture comparison between Storj and Sia

Feature	Storj	Sia
Blockchain	only for payment	file contracts and payments
Token	STORJ	SIACOIN
File storage management	satellites	file contracts
Redundancy method	erasure coding	erasure coding
Proof of Storage	yes	yes
Ranking System	Storage node reputation	Hosts ranking
Repair System	yes	yes

2.5.4 Filecoin

Filecoin is a public blockchain-based system developed by Protocol Labs in 2017 to work in conjunction with the company's distributed storage protocol, the Interplanetary File System (IPFS)

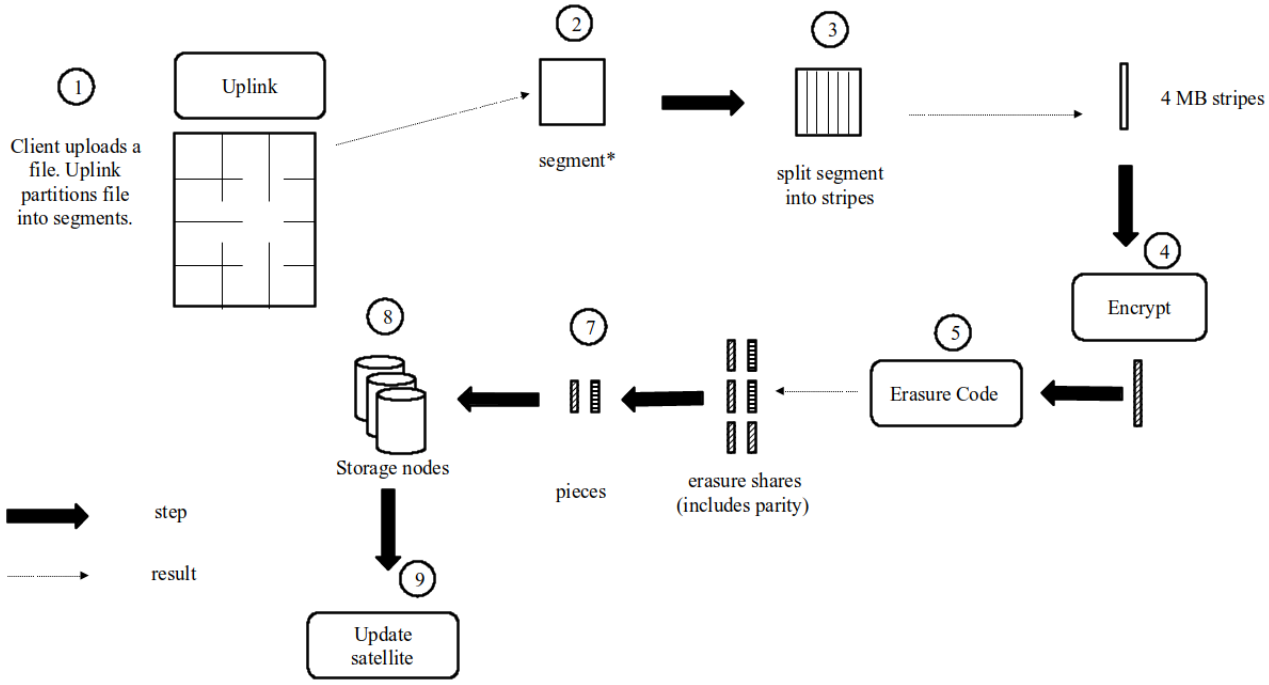


Figure 2.16: Storj data storage sequence.

[7]. IPFS allows users to store and share files similar to the BitTorrent protocol. However, nodes in IPFS do not receive any incentive to store and re-share files. The Filecoin blockchain was created to provide that incentive and allow users to be paid for storage services.

In this section, I first discuss IPFS's architecture and file-sharing protocol, the proceed to discuss Filecoin. Currently, Filecoin is in version 0.5.8, hence features and specification are still changing. For that reason, I give a high level view of its architecture and storage sequence. More details can be found in the Filecoin white paper [9].

IPFS Architecture

IPFS is a protocol, in a P2P network, for storing and accessing files, websites, applications and data [7]. The network is made up of participants, referred to as nodes and its neighbors called peers. Each node uses a client software to connect to other nodes, partition files into smaller pieces and verify those pieces are correct. I discuss pieces and how files are shared later in the section. The architecture embodies three concepts: content addressing, directed acyclic graphs (DAGs) and distributed hash tables (DHTs).

[9]https://filecoin.io/filecoin.pdf

Files in IPFS are identified by its content, rather than its location used in traditional file systems. IPFS uses a sha256 based hashing algorithm to create a file content identification (CID). Given the CID, a node can find and download the file. For example, if a user had a file named *my-file.txt* on the desktop, the location or URL of the file would be */home/desktop/my-file.txt*. In IPFS, if the result of *hash(my-file.txt)* is *Abcde*, the location of the file is simply */Abcde*.

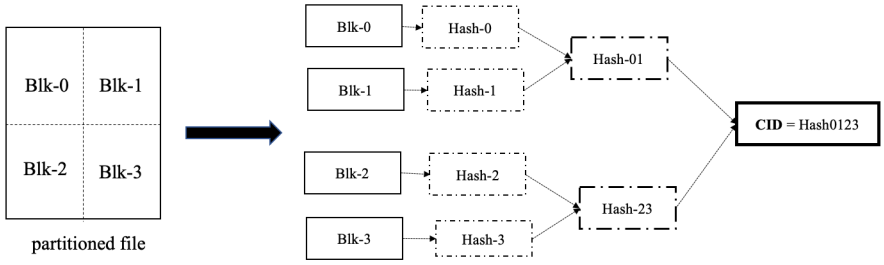


Figure 2.17: Creating a IPFS CID using a Merkle Tree.

Special Merkle trees, called Merkle DAGs[10], are used to maintain structure of folders and files (Figure 2.17). Files are partitioned into 256 KB pieces called blocks; files that are smaller than 256 KB are padded with filler bytes. Each block is hashed and combined to form the Merkle tree; the root of the tree is the CID. IPFS uses links to associate blocks of a file together and files within a folder (Figure 2.18).

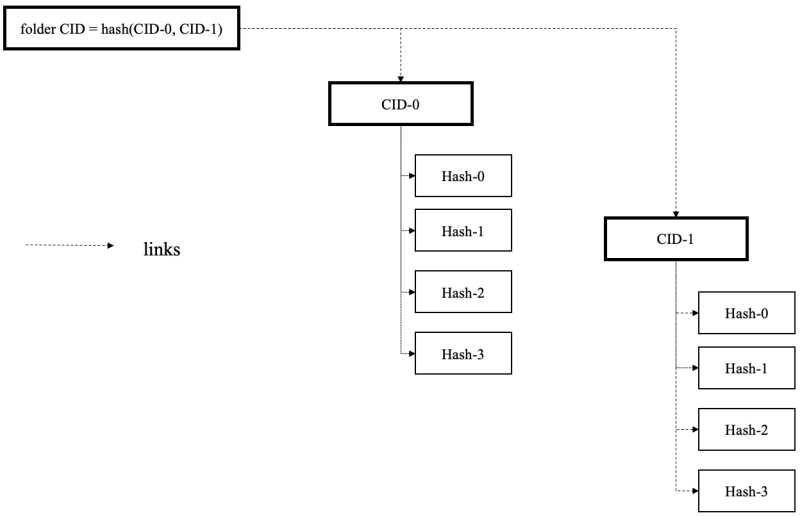


Figure 2.18: IPFS File structure with links.

DHTs help nodes find out which of their peers have the content they are searching for. Similar to BitTorrent's decentralized tracker, each node holds a piece of a large table (refer to Section 2.4.2).

[10]Merkle DAGs have no balance requirement like Merkle trees.

A table consists of keys and values, in which the keys are the file's CID and the values identify which node has the file. The identification is known as the PeerID. IPFS and uses a Kademlia based algorithm to decide which nodes store the key-pair value (refer to Section **??**). Ultimately, a node does not need to store the data itself, but can point in the direction to a peer that does.

File Sharing Protocol

The following steps outline how the IPFS protocol works. I assume node (A) has a CID to file and node (B) has that file, which are broken down into blocks.

1. (A) sends out a request to its peer list.
2. Peers look in its local DHT and attempt to find a matching CID.
3. Peers that find a match, create a sub-connection or session amongst themselves. Those that do not forward the request to a peer that closest match the CID.
4. Peers are continuously added to session when CID matches are found.
5. Session peers look at their local DHT to let (A) know who has the block(s), in this case (B).
6. (A) sends a request to (B) to see if it still has the block(s).
7. (B) responds 'yes' and starts to transfer the blocks to (A).

More than one node can have a desired file, in that case (A) would send a request to all nodes that potentially have the file. The peer that responds first will be the one that transfers the blocks. There is also a possibility that none of the peers have the file, either peer has left the network or the file was deleted.

IPFS lacks features that make it a viable decentralized storage solution. Nodes in IPFS do not have an incentive to share and store files. Recall in BitTorrent, sharing files increases a peer's rank. In the Sia and Storj platform, users are paid with cryptocurrency for leasing storage. Furthermore, IPFS does not guarantee a node is storing the correct file a client is look for either. To solve these issues and create a storage market place, Filecoin was developed.

Filecoin Architecture

The architecture is comprised of three types of nodes:

- Clients - users that pay have their data stored and retrieved.
- Storage Miner - store the full blockchain, book orders and store clients data.
- Retrieval Node - provide network resources to retrieve data from storage miners and deliver to clients.

Like the other BBS platforms discussed so far, each entity requires a client software or daemon to help nodes connect to each other and download the blockchain. Unlike Sia, where miners and storage nodes are separated, the storage nodes in IPFS are also the miners, hence the name storage miners (SM). SMs are both paid in a cryptocurrency, FIL, for storing clients files and have a chance to be rewarded during the mining process (refer to Section 2.5.1). Retrieval Nodes (RNs) do not participate in the mining and only receives FIL just for delivering data to clients.

The chances of mining the next block is dependent on a storage node's storage power. This feature is similar to Bitcoin's hashing power. The more storage a SM can prove to have, the higher its storage power increases.

Proof of Storage

IPFS uses two novel variations of Proof-of-Storage algorithms, Proof of Spacetime (PoSt) to help increase a SM's storage power and more importantly, to prove to clients their data is being properly stored [45].

The PoSt protocol allows a SM to prove to the network they have stored data for a period of time. The protocol additionally proves how much storage a SM actually has, which also affects storage power. Before a SM is eligible to mine the next block, they must produce a PoSt proof to the blockchain. Hence, PoSt is Filecoin's consensus algorithm.

PoR allows SMs to prove to clients they are storing data properly. The blockchain holds a record of block hashes and which SM is storing the block – I discuss how the hashes get there in the next section. During PoR, the blockchain selects a hash at random and presents a challenge to a SM to reproduce the hash. SMs who reproduce the hash get paid, those who do not and are punished, negatively affecting their storage power.

The details are PoR and PoSt can be found in the white paper [11] dedicated specifically to the two proofs.

[11] https://filecoin.io/proof-of-replication.pdf

Storage Sequence

File architecture models itself after a marketplace. I define the following terms to better understand the storage sequence:

- **Order** - a general request from a node. There are three types of orders: Bids, Asks and Deals.
- **Bid** - an order clients make to store data. Includes meta about the file, storage duration and a small amount of FIL to submit to the blockchain.
- **Ask** - an order SMs and RNs list with their asking price to to store and retrieve files.
- **Deal** - an order markets make after matching Bids with Asks.
- **Storage & Retrieval Market** - system within the blockchain that creates Deals.
- **Challenge** - a POS request from the blockchain to a SN.

The storage sequence is depicted in Figure 2.19 [7]. In this situation, I can assume that SMs and RNs have submitted Asks, letting the system know how much they charge for their services. Clients first submit a Bid to the blockchain, letting peers know they want to store data. Next the Storage Market begins to find a SN with a optimal asking price and sufficient storage to create a Deal with[12].

Once a Deal is created, the Client prepares a file by partitioning it into data blocks (not blockchain blocks), then sends the blocks to the SM for storage. The blockchain records the transaction, the SM's peer ID and hashes of the data blocks. Throughout the storage duration, the blockchain asks the SM to prove that it is still storing the data using PoR. Additionally, Clients pay SMs in installments until the storage duration is over.

When a Client wants to retrieve data, the client submits a bid to the blockchain; FIL is included with the bid to pay a RN. The Retrieval Market lists the bid and the first RN to to respond receives the Deal, allowing the client to receive the data as quickly as possible.

[12]In the current version of Filecoin, the Client actually manually selects which SN to store on.

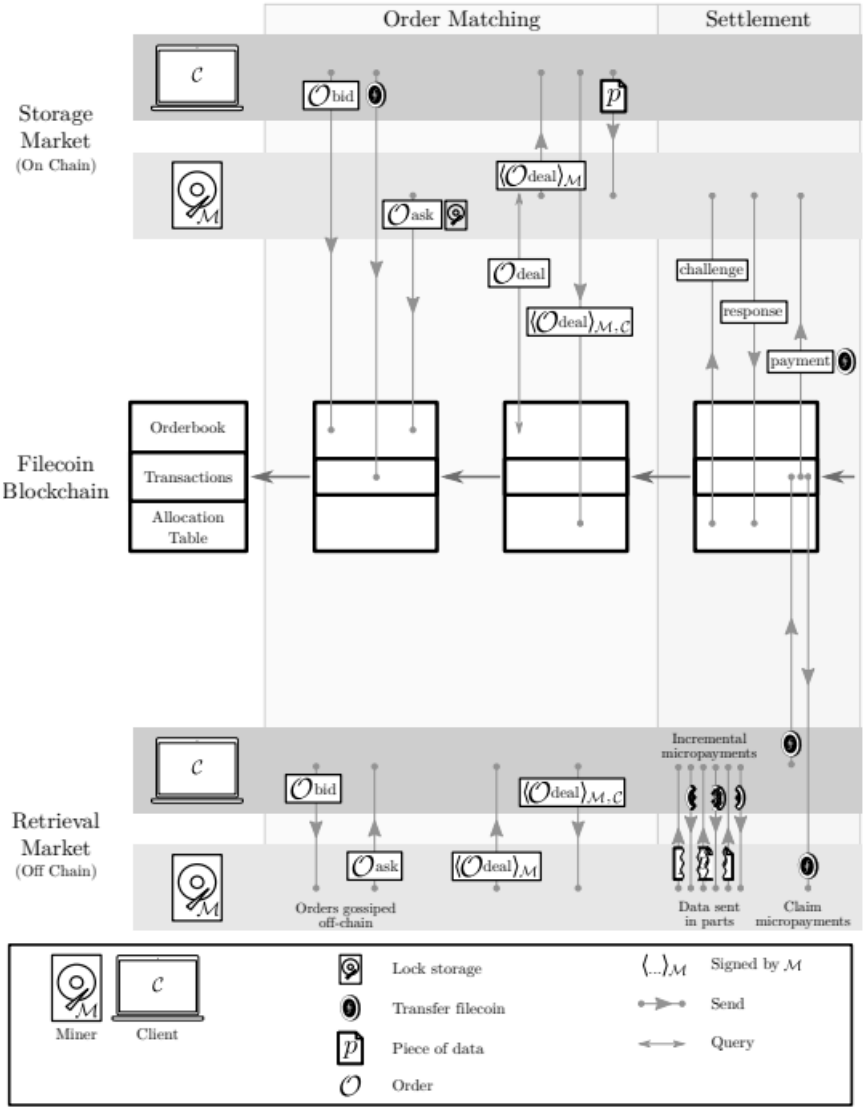


Figure 2.19: Filecoin data storage sequence.

Chapter 3

Performance and Costs Testing Methods

This chapter describes the testing environment and tests used to evaluate performance and costs. No tests were required to evaluate the security (Chapter 4). The chapter outlines installation, libraries, and how the data was gathered. Lastly, I provide the methodology on how the data was analyzed for evaluations (Chapters 5, 6 and 4).

3.1 Hardware

Performance and cost testing were done on a desktop machine. The specs of the machine are listed below:

- Operating system: Linux Ubuntu 18.04
- Processor speed: 2.3 GHz
- RAM: 16 GB
- Main hard drive: 256 GB solid state drive (SSD)
- Secondary hard drive: 2 TB hard disk drive (HDD)

The SSD was used for the performance test due to faster read and write speeds. The HDD was more suitable for the costs test, since that required a large amount of space and speed was not important.

3.2 Programming Languages

3.2.1 Node.js

Node.js[1] was the primary language used to code my tests. The language is an asynchronous event-driven JavaScript runtime designed to build scalable network applications. Node.js was chosen for two reasons, familiarity and availability of application programming interfaces (API), function calls that allow clients to communicate and control the application. Furthermore, the tests did not require high performance or memory safety, thus the convenience of dynamic language was valued over a static, typed language such as C++ or Java.

3.3 Testing Platforms

The performance and costs tests were based on Sia and Google Drive (GD). At the time of this study, Sia was the only Blockchain-based system that was open to the public and not in beta phase. While GD was not the only option to compare Sia as Additionally, GD offered 15 GB of free storage in which, while Dropbox and Amazon did not.

3.3.1 Sia

Installation

Sia offers to two methods for installation[2], the user interface (UI) or command line interface (CLI), see Figure 3.1. The UI provides a Graphical User Interface (GUI) to control the Sia Daemon (siad), while in the CLI the user must use terminal commands to control siad. Siad is the client application that allows users to interact with the Sia network. The UI was used for installation, while the CLI was used to run tests.

The UI gave an option to create a *new wallet* or *restore* a wallet from a previous seed (Figure 3.2). I created a new wallet for each test to ensure previous settings would not impact the next test. The only data copied from test to test was the blockchain data to avoid repeated, time-consuming downloads[3]. However, siad still needed to create the wallet and validate each block in the Blockchain which took several hours. This will vary depending on the specs of the machine's processor and hard drive.

[1] https://nodejs.org/en
[2] https://sia.tech/get-started
[3] The blockchain size was 18 GB.

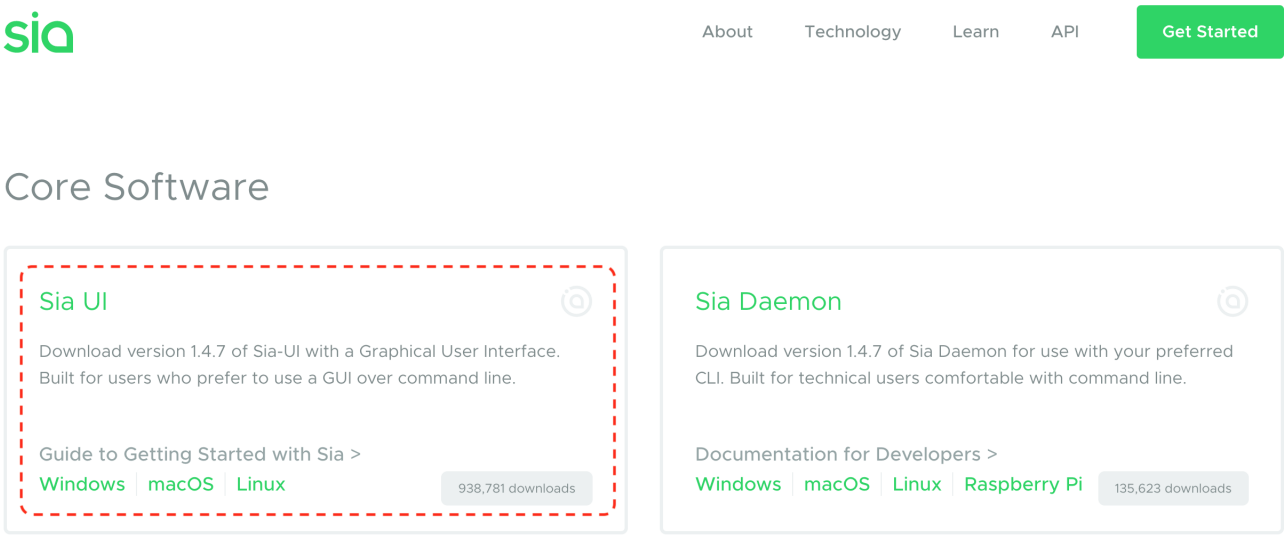


Figure 3.1: Sia installation options. UI selected for testing.

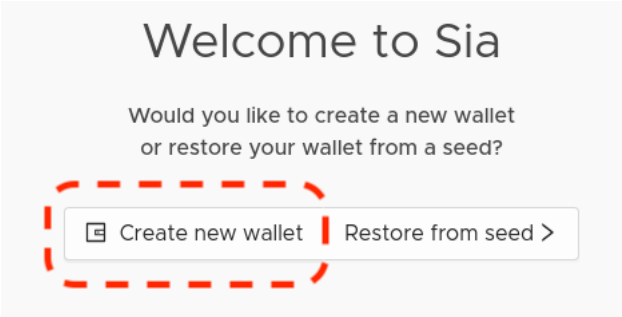


Figure 3.2: Sia wallet options. Chose new wallet for each test.

Purchasing Sia Coins

Sia Coins (SC) are the native cryptocurrency of Sia. Miners receive SC for mining and host nodes receive SC for successfully storing data (Sections 2.5.1 & 2.5.2). SC has a similar purpose as to bitcoin in the Bitcoin Blockchain and ether in Ethereum. Renters also need SC to rent storage.

SC could not be purchased with with USD, but must be traded for with other cryptocurrency, such as Bitcoin. The price of was SC was $0.0018 USD/SC at the time of this study. Bitcoin was purchased from Coinbase, an online fiat to cryptocurrency exchange. I traded Bitcoin for SC using Shapeshift[4] Before making the trade, A Sia wallet address must was generated (Figure 3.3). Then the addresses can be entered in the Shapeshift exchanger (Figure 3.4).

Sia additionally advertises other cryptocurrency exchange site such as Binance and Bittrex.

[4]https://classic.shapeshift.com/#/coins

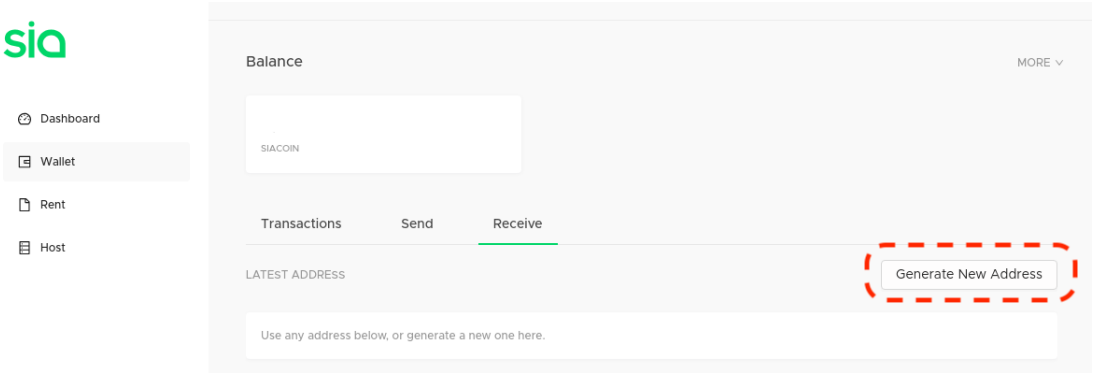


Figure 3.3: Generate Sia address.

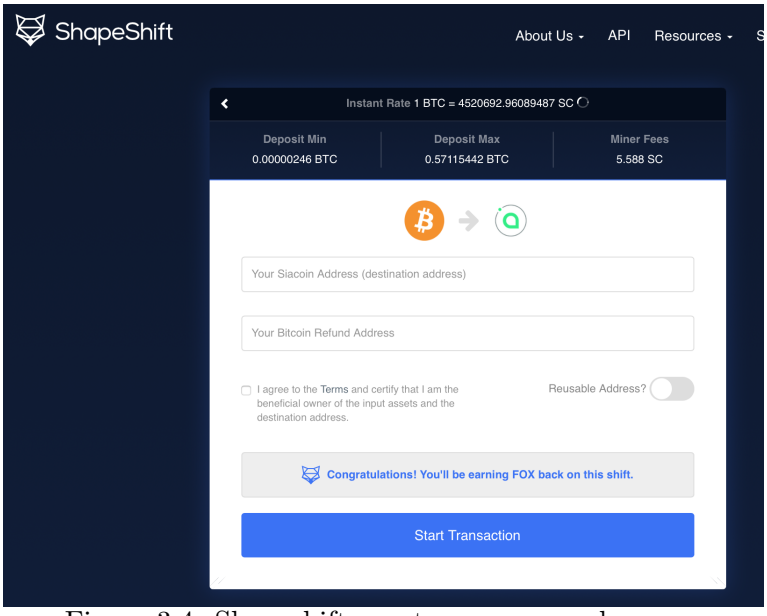


Figure 3.4: Shapeshift cryptocurrency exchanger.

Please refer to the Sia website for an up to date listing of exchanges that trade SC [5].

3.3.2 Google Drive

GD did not require software to download, however I needed to create a Google gmail account[6]. Once the account was created I were given access to the GD web application along with 15 free GB of file storage (Figure 3.5). Files could be uploaded or downloaded through the web application or through API calls which I describe in the next section.

[5]https://sia.tech/get-siacoin
[6]https://https://accounts.google.com/signup

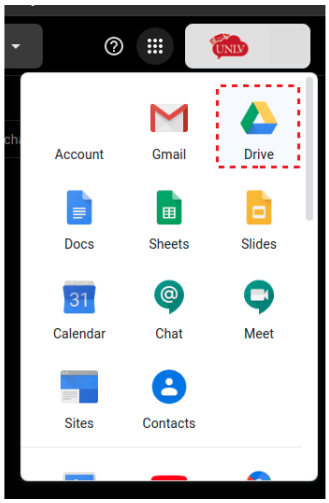


Figure 3.5: Access to Google Drive within Gmail account.

3.4 Application Programming Interface

Application Programming Interfaces (APIs) are commands that allow one application to communicate with another. The commands can be in the forms of functions, methods or http requests. Both Sia and GD provided APIs written in Node.js.

3.4.1 Sia API

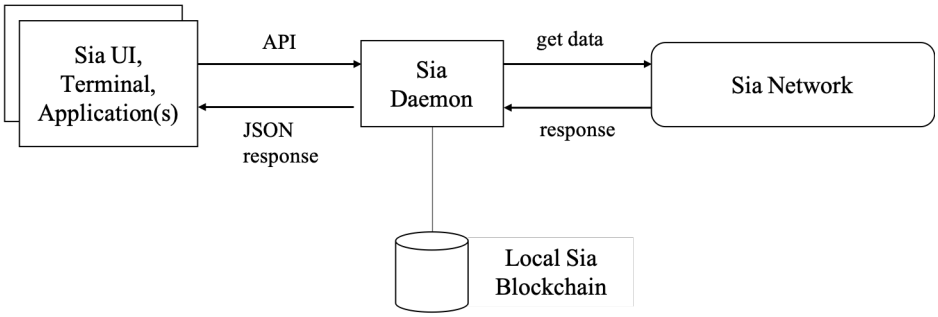


Figure 3.6: Diagram of Sia API communication.

Figure 3.6 shows how applications, the Sia daemon, blockchain and network communicate between each other. Sia's APIs are in the form of http requests and were used perform tasks such as uploading files, downloading files and collecting metrics data (Table 3.1). Once the request is made, siad responds with JavaScript Object Notation (JSON).

Table 3.1: Sia API to collect metrics from the daemon.

API	Request Type	Description
/gateway/bandwidth	GET	Returns the total upload and download bandwidth usage for the daemon.
/renter/contracts	GET	Returns the renters' contract.
/renter/files	GET	Returns the files uploaded to the Sia network belonging to the current wallet.
/files/download/siapath	GET	Downloads the file to the local file system.
/files/upload/siapath	POST	Uploads the file to Sia network.
/wallet	GET	Returns basic information about the wallet.

3.4.2 Google Drive API

Table 3.2 shows the API used to communicate with GD. Example code written in Node.js was given on GD's developer website[7]. Because GD is completely cloud based, the API commands are sent directly to the GD servers and not a local daemon like Sia (Figure 3.7). The response is also in JSON format.

Table 3.2: Google Drive API.

API	Description
file/	Uploads a file.
drive.files.list	Returns files and file ids stored on GD.
downloads/	Downloads a file given a file id.

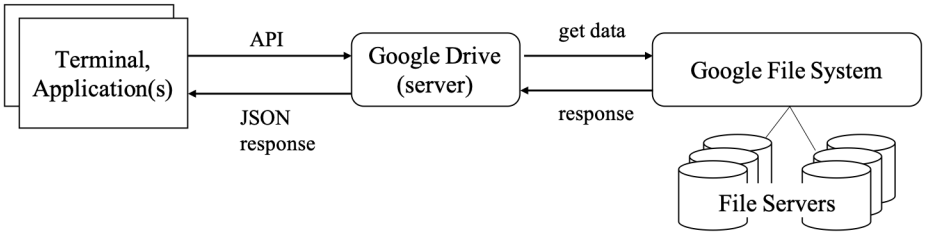


Figure 3.7: Diagram of Google Drive API communication.

[7]https://developers.google.com/drive/api/v3/manage-downloads#node.js

3.5 Performance Tests Description

The following section describes how performance tests were executed between Sia and GD. The tests consisted of timing two actions: uploading and download files. Three file sizes were used: 512 MB, 1 GB, and 10 GB. For each file size and action, I ran four trials, then calculated the average.

Table 3.3: Metrics recorded for performance tests.

Metric	Test Action	Description
available time	upload	Time before a file is ready to download (does not include file replication).
upload time	upload	Time it takes for a files to replicate and upload to all storage hosts.
download time	download	Time required to download a file.

Three metrics were measured: *available time*, *upload time* and *download time*. They are defined in 3.3. The *available* and *upload time* were measured in the upload tests and *download time* in the download test.

3.5.1 Upload Test

The upload test on Sia is outlined below:

1. Upload file with *file/upload/siapath* API.
2. Initiate measuring program and record the start time.
3. Call the */renter/files* API at a 1 second interval to check the available and upload progress.
4. Record the time when available and upload progress is complete.

Because the Sia API uploads files asynchronously, a measuring program was written to check the available and upload progress of the file (Figure 3.8). The measuring program called the upload API at an interval of one second (steps 2 and 3). The JSON response from the call contained the available and upload progress percentage of the file; values ranged from 0%-100%. The *available time* was recorded when the available progress reached 100% and *upload time* when the upload progress reached 100% (step 4).

The GD API uploads files synchronously. In other words, the API call does not return until the file is uploaded to the GD server and the file is available for download after. Hence the function

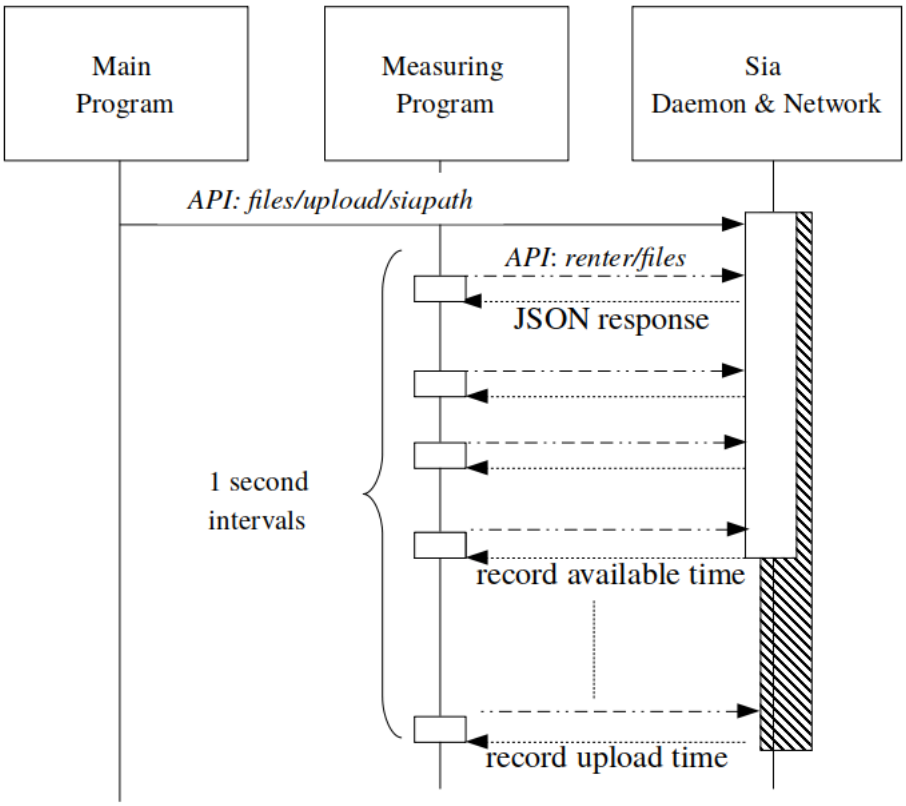


Figure 3.8: Sequence of Sia upload test program.

could simply be timed to obtain the *available time*. However, GD did not have an API to check if the file has been replicated. Thus I could not record an *upload time*.

3.5.2 Download Test

The Sia download API is synchronous. Unfortunate, I ran into issues when trying to use the API in Node.js. For that reason I used the Linux curl utility to call the API. To get the *download time*, curl has a *format* option that measures the http connection time. The connection automatically closes once the file was downloaded.

GD's download API is also synchronous. Similar to uploading files, the time for the call to finish executing was recorded as the *download time*

3.6 Costs Test Description

The Sia website advertises a storage estimate of $2/(TB*month). The purpose of the cost test was to observe of actual costs of storing data on Sia. The results were additionally compared to other CSP's prices. The test consisted of uploading 1 TB worth of files to the Sia network, while using a metrics program to collect wallet spending information. The test is outlined below:

1. Deposit and initial amount of SC into the wallet.
2. Begin file contract creation.
3. Start metrics program to collecting data at one minutes intervals.
4. At 50 contracts, start to upload files, five files at a time.
5. Tests ends when all files have been uploaded.

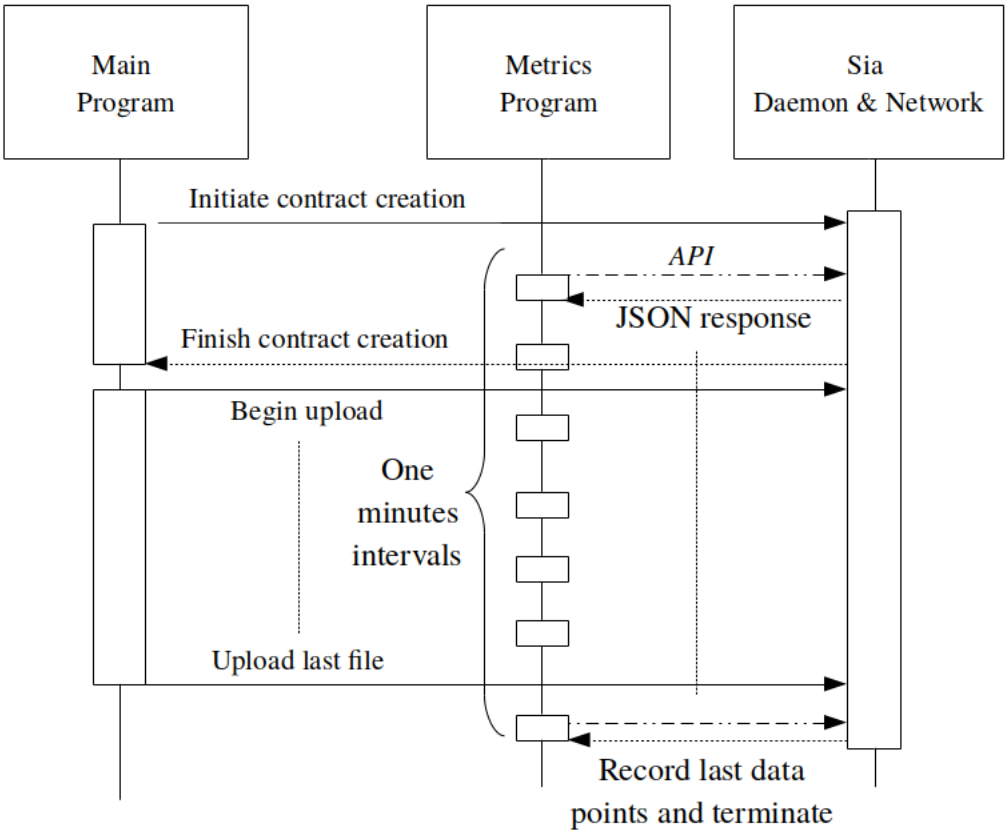


Figure 3.9: Sequence of Sia costs test program.

Before depositing money into the wallet (step 1), I used the renter calculator to calculate the estimate amount of sc to store the data[8]. The amount is referred to as the recommended allowance. The recommended allowance included fees to create contracts, upload the data and store the data for 3 months. The calculator recommended 2200 SC ($3.70); however 4000 SC was deposited for safety. The calculator input values are listed in table 3.4.

Table 3.4: Input for Sia renter calculator estimation.

Item	Value	Unit
Storage Amount to Rent	1	TB
Rental Duration	3	Months
Upload Bandwidth	1	TB
Storage Price	520	SC/TB
Contract Price	4	SC/TB
Upload Price	70	SC/TB
Number of Contracts	50	n/a
Contract Duration	3	Months

Table 3.5 show the conditions of the costs tests. The test files consisted of 100 binary files listed at 10 GB[9]. The exact size of each file was 9.7 GB, thus the total expected upload size was 977 GB.

Table 3.5: Sia costs test conditions.

Item	Value
# file contracts	50
# of files	100
size of each file	9.7 GB

Siad uploads files in an asynchronous manner. Preliminary testing revealed the upload rate diminishes as more files are attempting upload. Anymore than five files made slow upload progress. For this reason, the main program limited uploads to five files at a time. The limit was controlled using a queue data structure. Once a file was completely uploaded, it was removed from the queue and a new file entered the queue.

Table 3.1 list the API calls used to collect costs data. Because the API calls were http requests, real time data was difficult to collect. I choose a one-minute interval to call the API. Each call

[8]https://siasetup.info/tools/renting_calculator
[9]https://speed.hetzner.de/

responded with a JSON object that contained data about file contracts, uploaded files and wallet information (Table 3.6). Each would eventually generate 100+ data points. Thus, to minimize data collection, one-minute intervals sufficed.

Because the test focused on the cost of storing data, the files were not downloaded back to the local drive. Therefore downloading fees were set to zero in the calculator. Using the renter calculator, the cost to download was estimated to be 113 SC/TB ($0.20), which is minimal and has low impact on the final costs.

Table 3.6: Sia metrics collect for cost evaluation.

Category	Metric	Description
contracts	download spending	Amount of contract funds that have been spent on downloads.
	fees	Fees paid in order to form the file contract.
	renter funds	Remaining funds left for the renter to spend on uploads & downloads.
	size	Size of the file contract, which is typically equal to the number of bytes that have been uploaded to the host.
	total cost	Total cost to the wallet of forming the file contract. This includes both the fees and the funds allocated in the contract.
	upload spending	Amount of contract funds that have been spent on uploads.
files	file size	Size of the file in bytes.
	uploaded bytes	Total number of bytes successfully uploaded via current file contracts. This includes after file replication.
	upload progress	Percentage of the file uploaded, including redundancy.
wallet	confirmed siacoin balance	Number of siacoins, in hastings, available to the wallet as of the most recent block in the Blockchain. One hasting equals 10^{-24} Sia Coins.

3.7 Libraries

3.7.1 Sia JavaScript Bindings

The Sia JavaScript bindings made it easier to create Node.js applications by providing a simple wrapper around the API calls. The binding library was developed by Nebulous Labs, the same

company that makes Sia[10]. This was the only source of code that was not hand written.

3.7.2 Sia Metrics Library

The Sia Metrics Library was created to help gather information about files, contracts and wallet balance from the daemon. The library is comprised of functions that wrap the API calls (Table 3.1) in a http request. For example, a function in the Metrics Library called *getBalance()*; underneath, makes an http request using the */wallet* API as the URL address.

To use the library, I created driver program (labeled as Main Program in Figures 3.8 and 3.9) to import the library and execute its functions. The main program and library together collected the data. The metrics-library is available to view and use at [9].

3.7.3 Sia File Loader Library

The Sia File Loader library was created to stage and prepare files in the Sia costs test. Recall in Section 3.6, five files were uploaded at a time as part of the constraints of the Costs Test. Siad did not have a file limit setting when uploading files, thus this library was needed to ensure only five were were uploaded at a time, then continue until all files were uploaded. The library provides a queue data structure with operations to load and unload files.

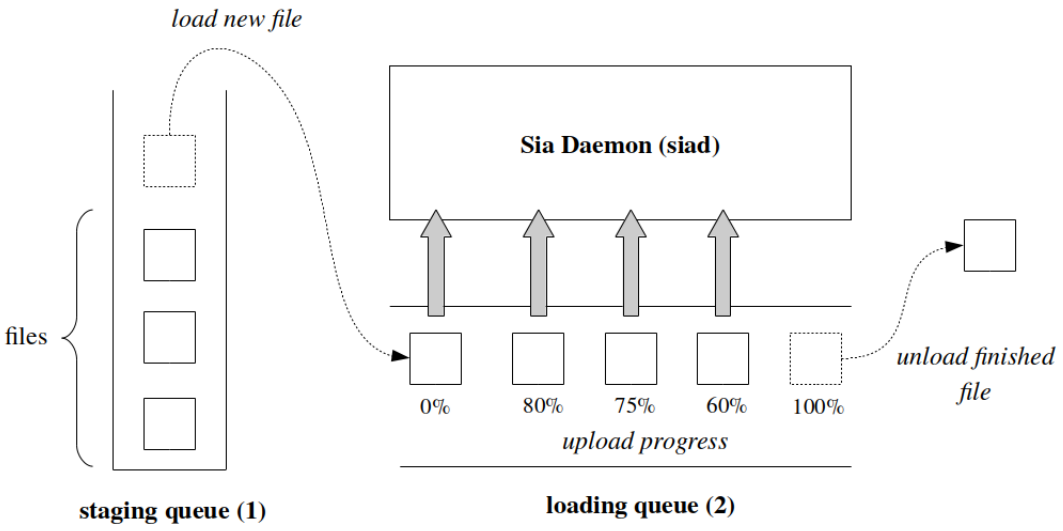


Figure 3.10: Driver program using the File Loader Library.

[10]https://github.com/NebulousLabs/Nodejs-Sia

Figure 3.10 shows conceptionally how the File Loader Library works within the main program. First, all the files were loaded into the staging queue (1). Files were then moved from (1) into the loading queue (2). Only five were allowed into (2) at a time. A file was removed from the (2) once it was completely uploaded (100% progress) to the Sia Network. The program ended once (1) and (2) were both empty.

3.8 Evaluation Methods

3.8.1 Security

No tests were ran for the security evaluation (Chapter 4).

3.8.2 Performance

The *available*, *upload* and *download* times were collected and used to calculate the average and standard deviation of all the trial runs. No other calculations were required for the performance evaluation.

3.8.3 Costs

The wallet API, */wallet*, returned the confirmed SC balance which was interpreted as the true amount of SC spent and remaining in the wallet. The author in [46] did the same. Spending discrepancies were found between wallet and file contract API. The discrepancies are discussed in Chapter 6.

There were three main costs to consider: contract creation fees, storage fees and upload fees. In the JSON response, the three were reported as *fees*, *storagespending* and *uploadspending* – using the */renter/contract* API. Download fees were not considered since I did download files when performing storage tests.

Additionally, a call to the */renter/contract* returned all 50 active contracts. Each contract displayed fees mentioned above. I totalled fees over all contracts to find the total storage cost.

Chapter 4

Security Evaluation

I start by evaluating the security between Sia and Traditional Cloud Systems (TCS). As discussed in Section 1.1, the security in this study encompasses five categories: encryption, data access, redundancy, durability and availability. Each category is addressed in the following sections. The performance and costs evaluations are discussed after in Chapters 5 and 6.

4.1 Encryption

Overall, the encryption schemes used Sia and CSPs were found to be comparable. Sia uses the Threefish to encrypt data before uploading to hosts [5], while TCS encrypt data with AES before storing on chunk servers [47, 29, 30].

There is little research comparing the two encryption schemes directly. Those that do conclude it is difficult to say one is better than the other and is a subjective matter [48]. Additionally, the authors conclude both schemes are secure, performant and at the same time vulnerable to various attacks. The authors in [31] rated AES at "excellent security" while Threefish at "secure". Their evaluation however concluded strength of an encryption algorithm depends on key management, number of keys and bits used in the key.

Key management is handled differently between blockchain-based systems (BBS) and TCS. TCS manage the customers private keys and have access to the data stored, while BBS place key management in the hands of the renters, thus hosts cannot read renters' data. The research in [49] suggests placing the keys in the hands of the renters is more secure and reduces data access.

Encryption strength is often described in terms of key length [50]. Keys are measured in bits. Longer longer keys provide stronger encryption at the expense of computation resources. Sia and

each of the TCS use a key bit length of 256. The number of keys used during the encryption process could not be found in the TCS documentation nor Sia's.

4.2 Data Access

Sia and the other BBS place key management in the hands of renters, otherwise known as client-side encrypting; data is encrypted before uploaded to hosts. Hosts are unable to view the encrypted data. In contrast, TCS encrypt the data server-side and have access to users' data. The customers must rely on trust in their CSP. Hence Sia provides stronger data access protection or in other words more privacy, than TCS.

Encrypting data before up uploading to the cloud not only limits data access, but also gives TCS less incentive to violate privacy and remain honest. The study in [49] assessed security of CSPs through equilibrium analysis. The authors use Nash Equilibrium, a form of game theory, to analyze various scenarios between customers and CPS. The authors concluded encrypting data before uploading to the cloud deters TCS from violating customer trust since the cost of having to decrypt the data generally outweighs the value of the data itself.

4.3 Redundancy

The redundancy factors are the same across Sia and TCS (Table 2.2 and 7.1). With each each platform offering a factor of N+2, what differs is the redundancy method. Replication and erasure coding are two methods used in achieving high redundancy and each have their advantages and disadvantages (Section 2.1.2).

The authors in [51] concluded erasure coding is advantageous for data that is unlikely to be accessed or for which read performance is not an issue and that replication is superior for data that is accessed frequently or for performance is important. The conclusion brings to light why the architects of each platform choose one redundancy method over the other.

GD and Amazon are not only file storage systems, but offer additional services which allow customers to access and edit data, increasing the amount of access counts. Dropbox and Sia are strictly file storage system. While Dropbox allows customers to preview data, the customers cannot edit while on the servers. Additionally, Dropbox in the past used Amazon S3 for storage, they have moved to their own in-house built storage system called Magic Pocket, which uses erasure encoding [52].

Because the redundancy factors are equal and one method is not considered better than the other, I evaluated Sia redundancy to be comparable to TCS.

4.4 Durability

Table 4.1: Durability calculated with Eq. 2.1

Item	Value	Unit
AFR	0.25	rate
n_{parity}	20	nodes
Durability	# of nines	
	94	

Table 4.2: Durability calculated with Eq. 2.2

Item	Value	Unit
AFR	0.25	rate
$MTTR$	0.67	hours
k	20	nodes
λ	$3.75E^{04}$	nodes
e	2.72	n/a
$P(k)$	$4.19E^{22}$	n/a
Durability	# of nines	
	15	

Recall durability refers to the ability of a storage platform to provide long-term protection against disk failures, data degradation and other corruption (Section 2.1.1). Table 4.1 and 4.2 show the durability based on the probability hardware failure and Poisson Distribution respectively. The durability calculated in using 2.1 resulted in 94 nines, which seemed improbable. I speculate equation was based on the assumption of using replication and not erasure encoding. Equation 2.2 resulted in a durability of 15 nines; a more reasonable amount. The 20 parity nodes are the main contributing factor to high durability.

Even though CSPs advertise a durability with nine 9's, it is difficult to evaluate if extra six 9's actually make a difference. Renters could expect to lose 0.000000000000001% of data a year with Sia and 0.000000001% a year with a TCS. Another way the view the loss, if customers were to store 10^6 objects with a TCS, the TCS could expect to lose 1 object every 10,000 years. With

Sia, if renters were to store the same number of objects, Sia could expect to lose 1 object every 100,000,000 years [53]. Does the extra 1,000 years make difference?

There is no industry standard for testing durability, let alone accurately calculating the metric. For this reason, I conclude Sia's durability to be comparable with other TCS.

4.5 Availability

Availability is how much time the storage provider guarantees that your data and services are available to you (Section 2.1.3). TCS guarantees an system wide availability of 99.9% according to their SLA. That is a down time of approximately 8.77 hours per year. In order to calculate the availability and compare against the SLA, the number of file chunk servers and backups would need to be known; neither which were not disclosed in documentation.

Given the SLA availability, I can however back calculate the number of server and backups using Equation 2.8. If a single chunk server is to assumed to have at least 95% availability , a TCS must have at least two chunk servers and two backups to achieve 99.9% system availability. Although, 95% is considered a conservative number; industries servers are observed to be higher [19], thus having one chunk server and two backups (three total) is enough to achieve 99.9% availability.

Table 4.3: Sia availability calculation.

Item	Value
total nodes (n)	30
spare nodes (s)	20
single node availability	97%
failure mode (f)	14307150
failure probablity (F)	1.4966E-25
Availability (1-F)	# of nines 25

Sia requires a single host availability of 97% – approximately 65 hours of down time during a three month contract. With 20 parity nodes, 21 hosts out of 30 would need to fail for a renter's data become unavailable. Equation 2.8 resulted in an availability with more than 25 nines (Table 4.3). Thus I evaluated Sia's availability to be higher than TCS.

Chapter 5

Performance Evaluation

In this chapter I evaluate the performance between Sia and Google Drive (GD). The performance tests consisted of uploading and downloading multiple size files (Section 3.5). I refer to the average times as just the available, upload and download times.

5.1 Results

Figure 5.1 shows the average available times between Sia and GD for various file sizes. The Sia/GD Ratio signifies the Sia times divided by the GD times. Recall from Table 3.3, the available time does not include replication time nor the time to upload replicated slices. Sia had slower available times for all file sizes – a minimum of 8 times slower than GD. Sia's time increased with the 10 GB file, resulting in an available time 12 times slower than GD.

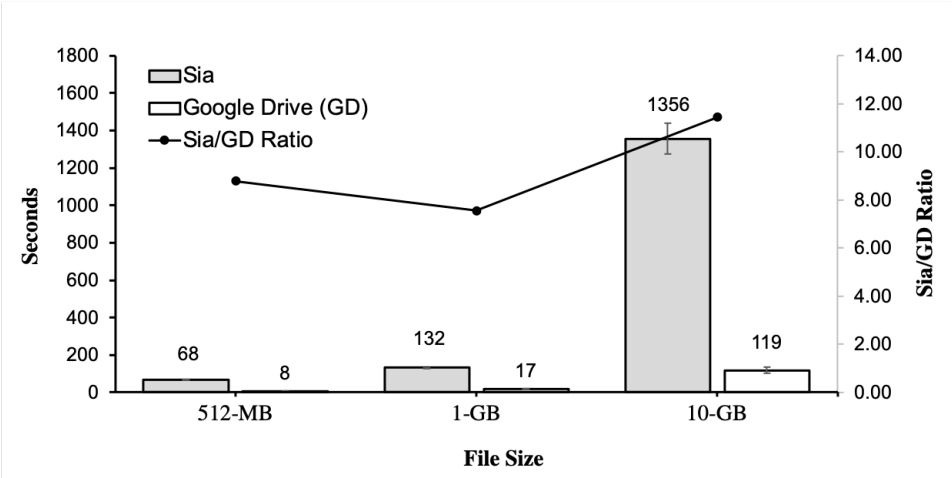


Figure 5.1: Available times and ratio between Sia and Google Drive.

The times and ratios from the download tests are shown in Figure 5.2. Sia had faster download

times for all file sizes. The GD/Sia ratio shows GD was a minimum of 1.5 times slower than Sia. GD's time increased with the 10 GB files, resulting in a download time three times slower than Sia.

During the download test, GD experienced more variation in times than the upload tests. Furthermore, the variation increased dramatically with the 10 GB files. The standard deviation reached +/- 163 seconds, approximately one-third the range. In contrast, Sia demonstrated consistent times with for all file sizes.

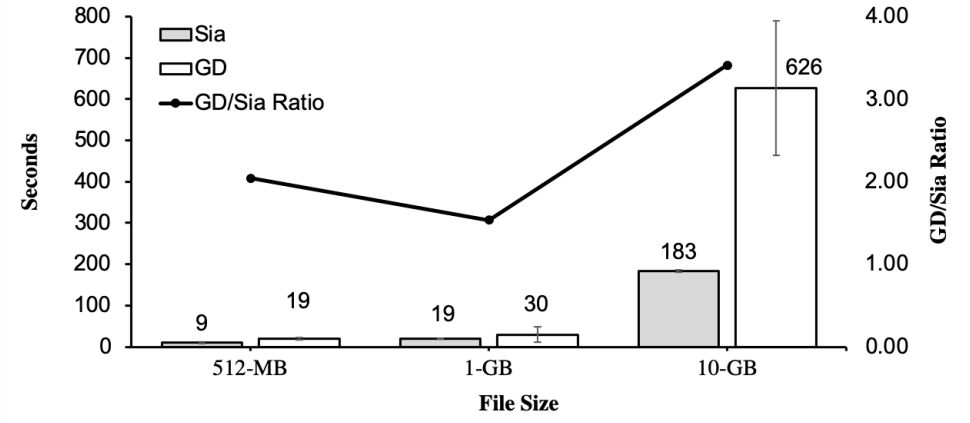


Figure 5.2: Download times and ratio between Sia and Google Drive.

Figure 5.3 shows the available and upload times for files stored on Sia. Recall the upload time is the time it takes for files to be replicated and stored on multiple Sia hosts (Table 3.3). All upload times were greater than available times. Additionally, the upload times displayed more variance.

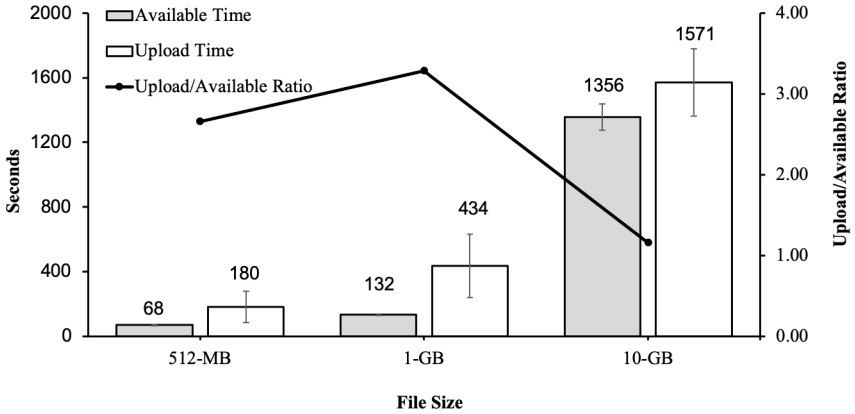


Figure 5.3: Available and upload times on Sia network.

5.2 Evaluation

5.2.1 Available Time

Longer available times with Sia were expected. Data is erasure encoded and encrypted before uploading to storage host. In contrast, GD replicates data after it has been uploaded to the server. Although, even if both platforms performed redundancy actions at the same step in the upload process, GD may still out perform Sia. Replication is simply faster than erasure coding. A study between the two methods show erasure encoding incurs more latency and is more CPU intensive than replication, which increases processing times [22].

Other factors that may have contributed to Sia's slower times include file contract updates and latency from communicating with multiple host servers. As data uploads to hosts, the Merkle tree in the file contract that represents the stored data must be updated. I speculate this needs to occur before the data is available for the renter to download. If so, extra time would be needed to find locate the file contract and update the Merkle tree, ultimately increasing availability time. The update however is committed to the Blockchain and does not require the 10 minute block confirmation.

GD requires less TCP connections when uploading a file as clients connects to one application server. In contrast, Sia requires minimum 10 connections, plus the 40 hosts for redundancy. Additionally, Sia hosts are not necessarily in the same geolocation. Figure 5.4 show the locations of Sia storage hosts. A renter's 50 contracts potentially include host on multiple continents. While GD file servers are grouped by data-center, and the center is generally chosen on the continent of the client[1].

However, according to host ranking criteria (Section 2.5.2), the geolocation does not factor into its hosts score; rather, host uptime and host age do. Suggesting Sia places more importance on hosts' reliability than performance gains through minimizing latency.

5.2.2 Download Time

The shorter download time on Sia highlights an advantage of P2P file storage systems. P2P systems are able to capitalize on the availability of another host if one is unavailable and bandwidth of every host [54]. During the download tests, Sia had 50 hosts to choose from. If a node was not available or slow to respond, siad could find another. Clients in GD rely on the availability of a single

[1]GD users do have the ability to choose regions.

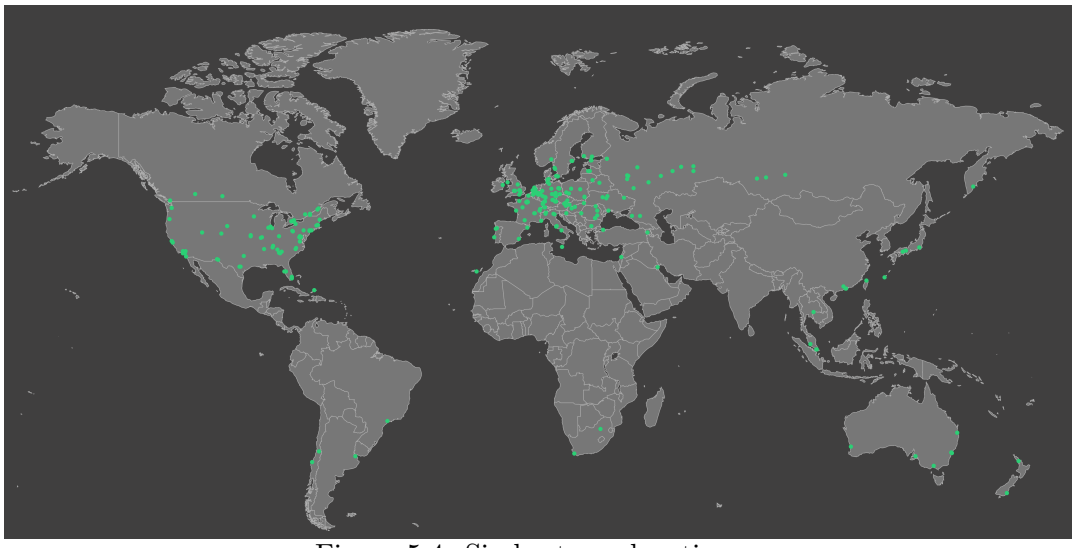


Figure 5.4: Sia hosts geolocations.

application server to retrieve the location of their data.

In regards to bandwidth, by having the file split amongst 10 hosts, each hosts has a smaller amount of data to deliver in parallel. Where as GD delivers a file from a single chunk server. The results in Figure 5.2 suggest the download performance gain would improve as file size increase beyond 10 GB.

5.2.3 Upload Time

Because Sia defaults to minimum three times redundancy, I expected the upload times to take at least three times longer than the available times. Figure 5.3 shows the 512 MB and 1 GB files took approximately three times longer than the average available times, meeting my expectations. However, the results for the for the 10 GB file were surprising. The upload time was only 15% longer than the available time. Given the available time of 1356 seconds, I expected an upload time of 4068 seconds (1356 * 3), but observed only 1571 seconds.

I speculate the erasure coding process occurs asynchronously and does not wait until the for the first 10 hosts to finish storage. Figure 5.3 shows the available times increased proportionally to the file size; the time doubled from 512 MB to 1 GB, then increased by an order of magnitude from 1 GB to 10 GB. However, the upload time did not follow this trend from 1 GB to 10 GB and only increased by a factor of 3.6 (1571 / 434). Thus it is possible Sia has a feature to optimize the storage process for large files through asynchronous uploading. More testing is needed to prove this claim.

Chapter 6

Costs Evaluations

6.1 Costs Test Results

Table 6.1 shows the summary for the 1 TB costs test. The Sia daemon (siad) stopped uploading data after 0.744 TB and did not continue to upload the remaining 0.233 TB, even with a remaining renter funds of 609 SC. Additionally, total contract spending amounted to 2747 SC; however, the entire 4000 SC allowance was spent. That leaves 1253 SC unaccounted for.

Table 6.1: Summary of balance and spending for storage test.

Item	Value	Unit
Starting wallet balance	4000	SC
Ending wallet balance	0	SC
Total contract spending	2747	SC
Remaining renter funds	609	SC
Attempted Storage Size	0.977	TB
Actual storage size	0.744	TB
Uploaded storage size	2.201	TB
Measured redundancy factor	2.956	n/a
Contracts formed	56	n/a

Figure 6.1 shows the *actual* and *uploaded storage size* while the files were being uploaded. The final *uploaded storage size* was 2.201 TB – the size after data is replicated. Given *actual storage size* of 0.744 TB, I measured redundancy factor of 2.956 (2.201 TB/ 0.744 TB). The redundancy was maintained at approximately 3x throughout the test.

Table 6.2 shows the estimate spending from the Sia Renter Calculator and the actual spending

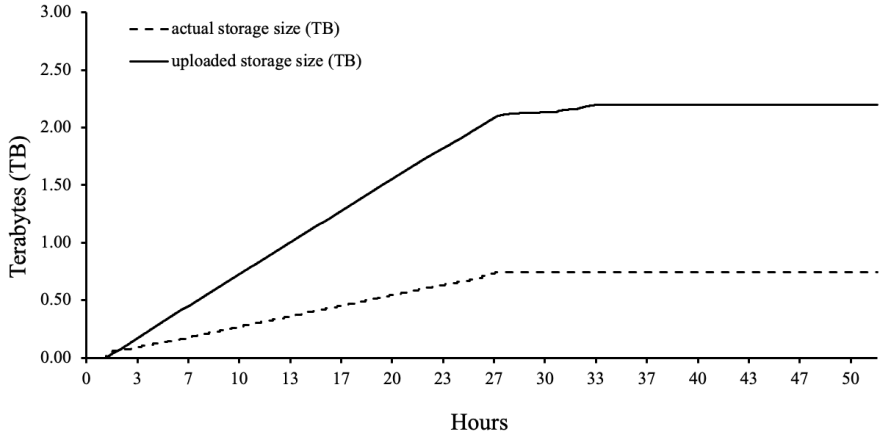


Figure 6.1: Actual and absolute storage size.

on contract, storage and upload fees. The estimates were based on the settings listed in Table 3.4. The actual contract and storage rates were higher than the estimates. While the actual upload rate was lower than what the calculator reported.

Table 6.2: Estimate and actual SC spend during storage test.

Fee Type	Spent(SC)	Rates		
		Estimate	Actual	Difference*
contract creation	274	4	5	-22%
storage (SC/TB*Mo)	1774	520	806	-55%
upload (SC/TB)	70	70	32	55%

* (estimate - actual)/estimate

Figure 6.2 shows the wallet balance and how SC were spent during the costs test. The data started to upload (and replicate) at the 1-hour mark, once 50 contracts were created. As more data continued to upload, contract spending increased. The wallet was depleted of funds at 18 hours; however, data continued to upload using the remaining renter funds. At the 27-hour mark, siad stopped uploading files even with a reaming renter balance of 609 SC.

Table 6.3 shows the final storage rate using the results of the costs test. The cost of other TCS platforms are also shows for comparison. Recall the final rate of Sia depends on the price paid for the SC, which was approximately $0.0018/SC.

6.2 Costs Test Evaluation

The costs test answers some of the question this study set out to explore, but also leaves us others:

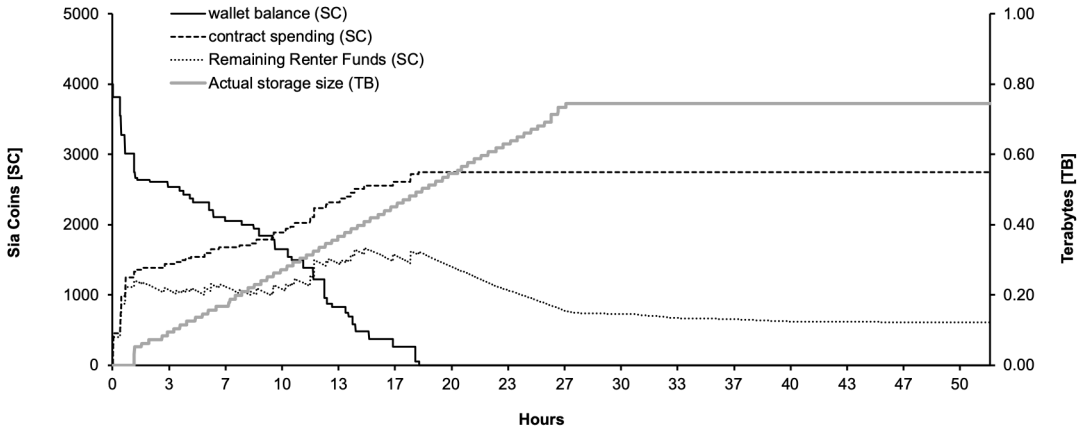


Figure 6.2: Balance and contract spending during storage test.

Table 6.3: Final storage rate of Sia compared with TCS rates.

Platform	Storage Cost (TB/Mo)
Sia	\$4.85*
Google Drive	\$5.00
AWS S3	\$23.00
AWS Glacier	\$4.00
Dropbox	\$5.00

* includes redundancy

- Why was the entire SC balance spent?
- Why did files stop uploading even when there was money remaining for renter funds?
- Why were more than 50 contracts created?

Why was the wallet balance spent?

Table 6.2 shows that estimated storage rate, 520 SC/Mo (\$0.94/Mo), was significantly lower than that actual rate, 806 SC/Mo (\$1.45/Mo). Additionally, the cost of redundancy was not taken into account. With a redundancy factor of 3x, brings the storage rate to 2418 SC/Mo (\$4.35/Mo). In order to store 1 TB of data, including contract and bandwidth fees, costs 2695 SC/Mo (\$4.85/Mo). Since default storage time is 3 months, approximately 7595 SC (\$13.68) was needed to successfully upload 1 TB.

Why were the 609 SC allocated to renter funds not spent?

In Figure 6.2, the confirmed SC balance reached zero near the 18-hour mark and the renter funds continued to decrease as more files uploaded. At the 27-hour mark, siad stopped uploading new files but still continued to replicate data chunks, during which renter funds were still being spent. This suggest once renter funds are locked into a contract, can only be used for redundancy, not for uploading new files. It is possible renter funds can be used to download files, but was not part of costs test. The author in [46] experienced a similar issue; his SC allowance was unexpectedly spent on new contracts even though 1200 SC remained in renter funds. The author was unable to explain the event. I too are unable to find a clear answer to this question and requires more investigation.

Why were more than 50 contracts created?

In total, 56 contracts were created even though I selected 50, the default value. Upon further investigation contracts were flagged with false for *goodforupload*. False means the contract cannot receive data according to the API page [1]. Six contracts became unresponsive by the end test, while 50 remained intact. Therefore, if the total amount active contracts drops below 50, siad creates new contracts until 50 is reached again.

[1] https://sia.tech/docs/#renter-contracts-get

Chapter 7

Discussions

In this study, I investigated the viability of BBS platforms by comparing Sia's security, performance and costs to TCS platforms. This chapter summarizes the results and evaluations from Chapters 4, 5 and 6. I then provide broader implications, limitations of the study and expand on potential future works.

The study began with reviewing pertinent information about DFS, blockchain and BBS architectures. Table 7.1 list architectural and security characteristics about Sia, Storj and Filecoin. TCS security metrics can found back in Table 2.2. The information was necessary for the security evaluation.

Table 7.1: Summary of blockchain-based storage architecture.

Characteristics	Sia	Storj	Filecoin
Cryptocurrency (Index)	siacoin (SC)	storj (STORJ)	filecoin (FIL)
Client side encryption	AES	Threefish	none
Redundancy Method	erasure coding	erasure coding	replication
Redundancy Factor	N+2	N+1/N+2	user chosen
Storage proof algorithm	proof of storage	proof of retrievability	proof of replication
Miner consensus algorithm	proof of work	proof of work*	proof of spacetime
Ranking system	yes	yes	yes

* uses Ethereum ER20 only to manage cryptocurrency

Figure 7.1 summarizes the security evaluation (Chapter 4) and reveals Sia's encryption, durability and redundancy to be comparable to TCS. Furthermore, Sia is able to provide a more private and available service through client-side encryption and P2P networking. Table 7.2 summarizes the

performance and cost evaluation (Chapters 5 and 6). Performance varied depending on whether files were being uploaded or downloaded. Sia experienced slower upload times than GD due to data redundancy and encryption occurring before upload. However, Sia achieved faster download times from capitalizing on the bandwidth of multiple hosts. Costs were similar to Google, Dropbox and Amazon Glacier (Table 6.1), but, higher than expected because of having to pay for data redundancy. Costs are also dependent on the value of SC.

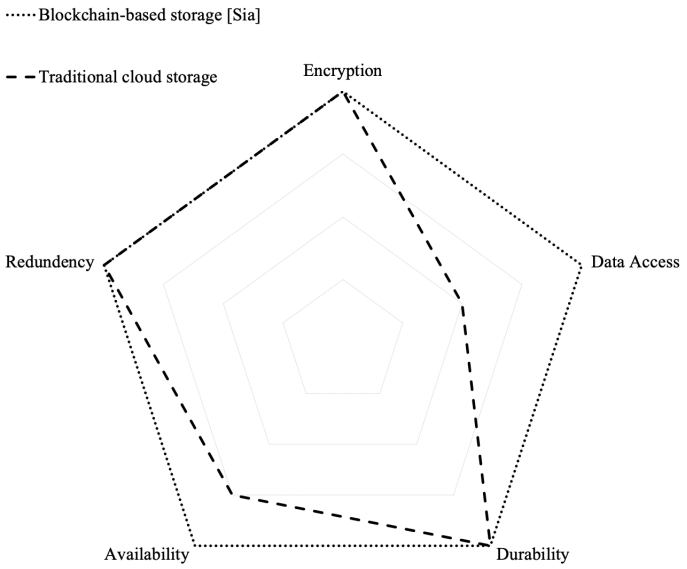


Figure 7.1: Security evaluation summary (outer radius is better).

Table 7.2: Performance and costs evaluation summary.

Evaluation	Test	Sia	Google Drive (GD)
Performance	upload	slower	faster
	download	faster	slower
Costs			
Estimate	$/(TB*Mo)	$2	
Actual	$/(TB*Mo)	$4.85[1]	$5

[1] Dependent on Siacoin price.

The combination of P2P systems and Blockchain technology allow for a file storage system that are just as durable, redundant and available as TCS. For those wanting a more private storage solution, where their data cannot be unencrypted by whom ever is storing it, BBS is an option assuming the platform offers client-side encryption. Each TCS provider discussed in this study are

able to encrypt their customers' data. Ultimately, any platform that does not do encrypt data before storing increases the likely hood for the holder to act dishonestly (Section 4.1).

Sia's $2/(TB*Mo) storage price initially intrigued us, given the current least expensive TCS plan is $5/(TB*Mo) (Table 6.3). The renter not only pays the original data storage fee, contract fee and bandwidth fee, but also for the redundancy process. With a 3x redundancy factor, uploading 1 TB of data costs 3 TB worth of data. This was the most unexpected result of the study since Sia did not explicitly state hidden cost on their product page; but perhaps is implied. Furthermore, the renter calculator did not account for redundancy, which resulted in not having enough SC in the wallet to upload all 1 TB worth of files. The evaluation revealed Sia's pricing model to be more complex than the simple model offered by TCS providers. In the end, this complexity may be the price of having a completely decentralized system where resources are provided by the users instead of central authority. Future studies of other BBS platforms may reveal the same.

Do I recommend BBS platforms? Based on the three evaluations, a BBS platform like Sia meets needs for personal storage, especially for those looking for a more private alternative. Although, there are reasons that may deter consumers from using the technology. The first reason is the pricing complexity. Renters will need to ensure they have enough balance in their wallet. If not, their data will either: not upload, not repair when hosts go down or unable to be downloaded. Another reason is the setup process (3.3.1). Setting up a BBS platform requires more steps than TCS. One must install the wallet software, download the blockchain, wait for the blockchain to sync, then finally obtain cryptocurrency. As opposed to signing up for an cloud account and entering credit card information. BBS services are however still relatively young compared to TCS, and the setup process may become easier in the future.

Can BBS survive and thrive against TCS? This is a difficult question to answer based off this study alone. There are two factors that may help answer the question and warrant further study. First is the level of participation of nodes in the system. Past studies have shown P2P systems such as BitTorrent suffer when peers participation level is low, and unlike BitTorrent, BBS have multiple types of nodes. BBS require miners to validate data and increase the integrity of the Blockchain, hosts to store data and most important renters to spend SC. A low level of participation of any type of node affects the security and performance of the entire system.

Another factor is the volatility of cryptocurrency value. The cost to store data on any BBS platforms depends on the value of their cryptocurrency. If the value of the cryptocurrency becomes too high, customers will likely look elsewhere to store their data. The value of the cryptocurrency

also impacts hosts. If the value drops too low, then hosts will not make profit for lending out storage and bandwidth.

7.1 Future Works

The costs evaluation (Section 6.2) left us with answered questions. Finding answers requires more testing and possibly a detailed review of the source code. Additionally, this study focused on costs as renter paying to storage. An interesting study would be to evaluate costs from the hosts and miners perspective. The ecosystem relies on all types of nodes participating and if either cannot be sustained financially, the platform as a whole suffers.

The scope of this research focused on comparing Sia to various TCS (Section 1.1). I recognize this would also be the study's limitation. Sia's security, performance and costs will be different from other BBS platforms such as Storj and Filecoin. Once Storj and Filecoin reach version 1.0, a similar study can be performed. It would be interesting to compare the results to Sia to which BBS is more performant and cost effective.

BBS storage has the potential to go beyond file storage. Technologies such as big data, deep learning and IOT produce large amounts of data and require storage. Future studies using blockchain as a solution would allow the potential to create innovative, decentralized computing services.

7.2 Conclusions

In this study, I explored an alternative to traditional cloud storage, Sia, a blockchain-based storage platform. The security, performance and costs were evaluated between of Sia and Google Drive. The evaluation revealed the encryption, redundancy and durability to be comparable. Furthermore, because of client-side encryption and erasure encoding scheme, Sia can provide more privacy and higher availability.

Regarding performance, Sia's upfront redundancy protocol and possible interactions with the blockchain yielded uploads times up to 12 times slower than Google Drive. In contrast, its download times were three times faster from utilizing peer-to-peer techniques. Lastly, Sia website advertised storing data would cost approximately \$2/(TB*Month); surprising, my storage test resulted in a cost of \$4.85/(TB*Month). Though higher, the cost is comparable to Google Drive's price and other platforms such as Dropbox who charge \$5/TB per month.

Appendix A

Acronyms

Amazon Web Services (AWS)

Blockchain-based Storage (BBS)

Cloud Service Providers (CSP)

Google File System (GFS)

Google Drive (GD)

Internet of Things (IOT)

Mean Time to Failure (MTTF)

Mean Time to Repair (MTTR)

Peer To Peer (P2P)

Proof of Replication (PoR)

Proof of Spacetime (PoSt)

Proof of Work (PoW)

Service Level Agreement (SLA)

Siacoin (SC)

Sia Daemon (siad)

Traditional File System (TFS)

Traditional Cloud Storage (TCS)

Bibliography

[1] I. D. Corporation, "The growth in connected iot devices is expected to generate 79.4zb of data in 2025, according to a new idc forcast." [online document], June 2019. https://www.idc.com/getdoc.jsp?containerId=prUS45213219.

[2] B. Shelby, "What is cloud storage and what are its advantages." [online], Aug. 2018. https://cloudstorageadvice.com/what-is-cloud-storage.

[3] J. Duffy and M. Muchmore, "The best cloud storage and file-sharing services for 2020." [online], June 2019. https://www.pcmag.com/picks/the-best-cloud-storage-and-file-sharing-services.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." [online document], Oct. 2008. https://bitcoin.org/bitcoin.pdf.

[5] D. Vorick and L. Champine, "Sia: Simple decentralized storage." [online document], Nov. 2014. https://sia.tech/sia.pdf.

[6] S. Labs, "Storj: A decentralized cloud storage network framework." [online document], Oct. 2018. https://github.com/storj/whitepaper.

[7] P. Labs, "Filecoin: A decentralized storage network." [online document], July 2017. https://filecoin.io/filecoin.pdf.

[8] A. Mauthe and D. Hutchison, "Peer-to-peer computing: Systems, concepts and characteristics," *PIK - Praxis der Informationsverarbeitung und Kommunikation*, vol. 26, April 2003.

[9] P. Austria, "Thesis source code," May 2020. https://gitlab.com/paustria/sia-loader.

[10] T. Zhu, C. LV, Z. Zeng, J. Wang, and B. Pei, "Blockchain-based decentralized storage scheme," *Journal of Physics: Conference Series*, vol. 1237, p. 042008, June 2019.

[11] E. Bocchi, I. Drago, and M. Mellia, "Personal cloud storage benchmarks and comparison," *IEEE Transactions on Cloud Computing*, vol. 99, pp. 1–1, April 2015.

[12] R. Gracia-Tinedo, M. Sánchez-Artigas, A. Moreno-Martínez, C. Gonzalez, and P. López, "Actively measuring personal cloud storage," p. To appear, June 2013.

[13] W. Hu, T. Yang, and J. Matthews, "The good, the bad and the ugly of consumer cloud storage," *Operating Systems Review*, vol. 44, pp. 110–115, Aug. 2010.

[14] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras, "Benchmarking personal cloud storage," pp. 205–212, Oct. 2013.

[15] R. Nygaard, H. Meling, and L. Jehl, "Distributed storage system based on permissioned blockchain," pp. 338–340, April 2018.

[16] Tendermint, "Cosmos whitepaper." [online document], 2016. https://cosmos.network/resources/whitepaper.

[17] A. David, "Managing iot data on hyperledger blockchain," Aug. 2019. M.S. thesis, Dept. of Comp. Sci., Univ. of Nevada, Las Vegas.

[18] V. Ramesh, "Storing iot data securely in a private ethereum blockchain," May 2019. M.S. thesis, Dept. of Comp. Sci., Univ. of Nevada, Las Vegas.

[19] W. T. Inc., "Wasabi extremely high durability protects mission-critical data." [online document], 2018. https://s3.wasabisys.com/wsbi-media/wp-content/uploads/2018/10/Wasabi_Durability_Tech_Brief.pdf.

[20] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," *ACM SIGOPS Operating Systems Review*, vol. 37, pp. 29–43, Dec. 2003.

[21] M. Wu, "Maybe you shouldn't be that concerned about data durability." [online], June 2010. https://blog.synology.com/data-durability.

[22] A. Shenoy, "The pros and cons of erasure coding & replication vs. raid in nexgen storage plarforms." [online], Sept. 2015. https://www.snia.org/educational-library/pros-and-cons-developing-erasure-coding-and-replication-instead-traditional-raid.

[23] J. Plank, "Erasure codes for storage systems." [online document], Dec. 2013. https://www.usenix.org/system/files/login/articles/10_plank-online.pdf.

[24] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, pp. 300–304, 06 1960.

[25] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," 04 2002.

[26] R. Bauer, "What's the diff: Durability vs availability." [online], June 2019. https://www.backblaze.com/blog/cloud-storage-durability-vs-availability/.

[27] E. Torres, G. Callou, and E. Andrade, "A hierarchical approach for availability and performance analysis of private cloud storage services," *Computing*, Jan. 2018.

[28] S. A. Inc. and W. Highleyman, "Calculating availability - redudant systems." [online document], Oct. 2006. http://www.availabilitydigest.com/public_articles/0101/calculating_availability.pdf.

[29] Dropbox, "Dropbox business security: A dropbox whitepaper." [online document], 2019. https://aem.dropbox.com/cms/content/dam/dropbox/www/en-us/business/solutions/solutions/white_paper/dfb_security_whitepaper.pdf.

[30] Amazon, "Protecting data using server-side encryption." [online], Jan. 2020. https://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html.

[31] R. Bhanot and R. Hans, "A review and comparative analysis of various encryption algorithms," *International Journal of Security and Its Applications*, vol. 9, pp. 289–306, 04 2015.

[32] E. Levy and A. Silberschatz, "Distributed file systems: Concepts and examples.," *ACM Comput. Surv.*, vol. 22, pp. 321–374, 12 1990.

[33] Google, "Introducing google drive...yes, really.." [online document], April 2012. http://googleblog.blogspot.in/2012/04/introducing-google-drive-yes-really.html.

[34] J. li, "On peer-to-peer (p2p) content delivery," *Peer-to-Peer Networking and Applications*, vol. 1, pp. 45–63, Mar. 2008.

[35] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A performance study of bittorrent-like peer-to-peer systems," *Selected Areas in Communications, IEEE Journal on*, vol. 25, pp. 155 – 169, Feb. 2007.

[36] C. Smith, "Interesting bittorrent statisics and facts (2020) — by the numbers." [online], Feb. 2020. https://expandedramblings.com/index.php/bittorrent-statistics-facts/.

[37] W. Gordon, "The best bittorrent clients for 2019." [online], Sept. 2019. https://www.pcmag.com/news/the-best-bittorrent-clients-for-2019.

[38] G. Neglia, G. Presti, H. Zhang, and D. Don, "A network formation game approach to study bittorrent tit-for-tat," in *Network Control and Optimization* (T. Chahed and B. Tuffin, eds.), pp. 13–22, Springer Berlin Heidelberg, Oct. 2007.

[39] B. Cohen, "Incentives build robustness in bittorrent," *Workshop on Economics of PeertoPeer systems*, vol. 6, June 2003.

[40] L. Ye, H. Zhang, W. Zhang, and J. Tan, "Measurement and analysis of bittorrent availability," *Parallel and Distributed Systems, International Conference on*, vol. 0, pp. 787–792, Jan. 2009.

[41] Q. Zhen, Y. Li, P. Chen, and X. Dong, "An innovative ipfs-based stroage model for blockchain," ieee, Dec. 2018.

[42] A. Chauhan, O. Malviya, M. Verma, and T. Mor, "Blockchain and scalability," ieee, July 2018.

[43] P. Magazine, "Crypto vs visa: transactions' speed compared." [online], Jan. 2020. https://payspacemagazine.com/cryptocurrency/crypto-vs-visa-transactions-speed-compared.

[44] S. Somin, G. Gordon, and Y. Altshuler, *Network Analysis of ERC20 Tokens Trading on Ethereum Blockchain: Proceedings of the Ninth International Conference on Complex Systems*, pp. 439–450. 07 2018.

[45] J. Benet, D. Dalrymple, and N. Greco, "Proof of replication." [online document], July 2017. https://filecoin.io/proof-of-replication.pdf.

[46] M. Lynch, "Sia load test wrap-up." [online], April 2018. https//blog.spaceduck.io/load-test-wrapup/.

[47] "Encryption at rest in google cloud platform." [online], April 2017. https://cloud.google.com/security/encryption-at-rest/default-encryption.

[48] Aim and Scope, "Comparitive anlaysis of aes, blowfish, twofish and threefish encryption algorithms," *The Journal of Analysis of Applied Mathematics*, vol. 10, 2017.

[49] Y. Wu, Y. Lyu, and Y. Shi, "Cloud storage security assessment through equilibrium analysis," *Tsinghua Science and Technology*, vol. 24, pp. 738–749, 12 2019.

[50] A. Mousa and H. Ahmad, "Evaluation of the rc4 algorithm for data encryption.," *International Journal of Computer Science and Applications*, 01 2006.

[51] J. Cook, R. Primmer, and A. Kwant, "Compare cost and performance of replication and erasure coding." [online document], July 2014. https://www.hitachi.com/rev/pdf/2014/r2014_july.pdf.

[52] J. Cowling, "Pocket watch: Verifying exabytes of data." [online], July 2016. https://dropbox.tech/infrastructure/pocket-watch.

[53] B. Wilson, "Backblaze durability is 99.999999999% - and why it doesn't matter." [online], July 2018. https://www.backblaze.com/blog/cloud-storage-durability.

[54] E. Biersack, P. Rodriguez, and P. Felber, "Performance analysis of peer-to-peer networks for file distribution," vol. 24, pp. 1–10, Springer, Jan. 2004.