Sia : Decentralized, Compensated, Self-Repairing Computer Storage

David Vorick
Rensselaer Polytechnic Institute

May 3, 2014

Abstract

Sia is a decentralized platform for computer storage acting as a marketplace for storage. On the supply side, host computers contribute storage in return for Sia's internal cryptocurrency - Siacoin. On the demand side, consumers use Siacoin to rent storage space. The price of storage is set by a market making algorithm that is sensitive to the volume of supply and demand. Files on Sia are distributed to a large number of randomly selected hosts. Erasure Coding is used to give the files redundancy, enabling files to be spread over hundreds of machines with highly reliable network redundancies as low as 1.2. Spreading the files over hundreds of hosts means that files can be downloaded in parallel from hundreds of places and achieve high throughput. As explored in the paper, Sia should be a cheaper and superior solution to cloud storage when compared against the existing centralized alternatives.

1 Introduction

Sia is a decentralized platform for storing data on the cloud. The storage on Sia is block level, like an unformatted hard drive. This makes Sia filesystem agnostic - you store information on Sia by setting and requesting arrays of bits. It is very easy to put a file system such as NTFS or ext4 on top of Sia. It is expected that the average user will not interact directly with Sia, but rather with an application such as a music player that streams directly from Sia. Storage on Sia is random-access, meaning that any set of bits can be requested without needing to download an entire file.

Each file is given to 128 machines that monitor the health of the file. If any of these machines goes offline, it is replaced by the network, keeping the file available and in good health. The file is broken into 128 pieces, of which M are redunant, where M is chosen by the uploader. Any M machines can go offline at the same time and the file will still be recoverable - this is because Reed-Solomon coding is used. If a machine goes offline or fails, it will be replaced within a few hours. Having a file split between 128 machines and being able to tolerate M faults every day means that even in scenarios of high paranoia M can be relatively small.

Reed-Solomon codes are maximum-distance separable. This means that redundancy is maximally efficient. For example, take a 100mb file that gets broken up into 128 pieces with M=28. You have 100 non-redundant pieces and 28 redundant pieces. Of these, ANY 28 pieces can be lost and the full file can still be recovered. Together, all 128 pieces consume 128mb of physical storage. The default value for M on Sia is 28.

Each file is stored on 128 machines, and the file can be downloaded in parallel from all of them. This allows Sia to obtain very high throughput for file downloads, even if each machine is only offering a little bandwidth. Additionally, for situations requiring reduced latency, only the closest $128 - M$ files need to be downloaded.

Sia is a decentralized platform for storing data on the Internet. With only a few kilobytes of overhead, any file can be stored on the Sia network. Sia supports anonymity, and guarantees uptime for the files. Files can be fetched with only a few hundred milliseconds of latency. Files are random-access, meaning a piece

of the file can be downloaded without fetching the whole file. All files are seeded by 192 peers, meaning files can be uploaded and downloaded at high speed. Sia minimizes file redundancy while maximizing speed and security, making Sia competitive with centralized services.

Files are stored by hosts in exchange for Siacoins. The price of storage is set algorithmically according to supply and demand. Siacoins are mined on the Sia network and are the only currency that can be used to purchase Sia storage. This constraint provides stability to the price of the siacoin - the price of the coin is tethered to the price of Sia storage. Siacoin is a superset of Bitcoin.

Files can be stored in two modes.

- Mutable - the contents of the file can be changed by anyone with permission. The original contents of the file are permanently lost after a change is made. Most files will be mutable.

- Immutable - once uploaded, the file can never be altered, and the size of the file is constant. At the end of the file, there is a pointer to another file, which can contain information about updates to the original file. This pointer can be null. Immutable files are appropriate for public and shared data and media - if you use a public file, you want to be sure that the uploader won't replace it with something else.

Files have a balance of siacoins, which is used to pay the hosts that store the file. Each block, a volume of siacoins is removed from the balance according to the current price of storage. When the balance runs out, the file is deleted. Anybody can increase the balance at any time by sending more siacoins to the file. There is no way to remove a file from the network aside from waiting until the balance depletes. There is no way to extract coins from the balance - once sent to a file, siacoins are locked into a single purpose. These constraints protect against censorship.

Sia can be used as a more efficient replacement for many of todays services. These services include Bittorrent, Bitcoin, and file lockers such as MediaFire. On fast connections, Sia could be used as a replacement for hard drives - computers would only need enough local storage to boot and connect to the internet.

One major advantage of Sia is that all shared files only need to be hosted once, yet no central service needs to be trusted or depended on. Another advantage of Sia is that you only need to rent as much storage as you are actively using - more storage can be rented on-the-fly.

2 Overview

Sia is built out of many small blockchains of 192 hosts each. These blockchains participate in a deterministic consensus algorithm instead of using a block mining algorithm such as proof-of-work. Blockchains are assembled randomly from hosts on the network - hosts cannot control which blockchain they are participating in. Each host must be contributing exactly 256GB to the network in order to be placed in a blockchain. A single machine operate as a group of hosts each contributing exactly 256GB, as long as the storage is unique. Blockchains will be honest as long as at least a simple majority of the hosts within the blockchain are honest.

The proof-of-storage algorithm makes use of the fact that data is redundant - each host is storing different Reed-Solomon coded pieces of the files uploaded to the network. By gathering file pieces from each host, the network can use the pieces to check for honesty and corruption simultaneously.

Each blockchain has a set of files which it tracks. As files are added and deleted, the network performs load-balancing between blockchains to keep the file distribution even.

Hosts wanting to join the network are put into a queue, where they wait until there is sufficient storage demand to justify creating a new blockchain. If there are many hosts joining the queue but few new files being added to the network, the price of storage will drop. If there are many files being uploaded and few hosts joining the queue, the price of storage will increase. The pricing algorithm tries to always keep hosts in a queue instead of putting files in a queue.

Blockchains are organized into a tree that manages the needs of the network. Parent blockchains have

aggregate information about their children - such as how much data is being stored by each child. Only the leaf blockchains actually store files - all the other blockchains are designed to be very lightweight. The root blockchain determines the network price, and also knows how many new hosts there are and how much free space there is on the network. When the root blockchain needs to do something such as create new blockchains, it delegates the work to its children. Those children delegate work to their children, until the action hits the leaf layer, where it is carried out.

The same happens in the reverse direction. When a leaf blockchain deletes a file, it tells its parent, who then updates its aggreagate resource. Parent blockchains are in charge of load-balancing their children, and they wait for their parents to load balane them.

Using a tree of blockchains allows each host to participate in a lightweight blockchain that stores only a fraction of the information on the network. Additionally, each host must only perform calculations on a fraction of the network.

The aggregate nature of the tree allows each host and wallet to verify that their segment of the network is honest, even though they cannot see every action occuring on the network.

Wallets take space on the network, and are charged the file price for existing on the network. Each block, money is removed from the wallet and when the funds run out, the wallet is deleted. As wallets only take up a few kilobytes, the amount charged is minimal.

When wallets send money to a different blockchain, the transaction propagates through the tree. Information is aggregated, meaning that the parent blockchain will only say how many coins have moved between children; it will not say which wallet the coins are from or which wallet the coins are for. This compression means that there needs to be a way to determine which wallets incoming siacoins are for, but it also means that the parent blockchains have a low maximum amount of work they have to perform per block.

To determine how aggregated transactions get divided, the sending blockchain messages the receiving blockchain, informing them that coins are incoming. The receiving blockchain then claims the coins from the parent.

Blockchains can look up the direct communcation address of other blockchains through a DHT. All leaf blockchains participate in the DHT. This allows blockchains to find eachother in log(n) time without sending high volumes of messages through their parents, and without needing to store the direct communication address of every blockchain in the network.

Any time that a host is caught by the network performing an illegal action, the host will be fined. The fine will be equal to or greater than the amount of damaged caused to the network. To make sure that hosts can be fined, they are required to have a balance equal to or greater than the maximum possible fine - a 'security deposit'. When a host leaves the network, the security deposit will be released.

The security measures in Sia ensure that an attacker cannot control which blockchain its hosts are participating; they are randomly distributed. Sia assumes that only simple majority of the hosts on the network are honest. If an attacker controls $\frac{n}{2}-1$ of the hosts on the network, the attacker has a 50% chance of controlling a particular blockchain, and will control roughly 50% of the blockchains on the network. Even controlling half of the hosts on the network, an attacker has a minimal chance of controlling more than 85% of the hosts in a particular blockchain.

In all interactions on Sia, it is assumed that blockchains are dishonest with 50% probability, and that all blockchains have at least 15% honest hosts. Sia also has tools to detect dishonest actions by hosts and blockchains, which result in fines and being kicked from the network.

3 Consensus Blockchains

Concensus blockchains are formed from 192 hosts all storing 256GB. Each blockchain hosts a set of files and wallets that are unique to that blockchain (each object only appears on a single blockchain). Blockchains have a 'state', which is a representation of the current status of the network. The state is updated in blocks, which are produced in a concensus algorithm.

Blocks are composed of 'heartbeats', which is an

update package from a host. Each block, every host must submit a heartbeat. Heartbeats contain storage proofs, keepalive information, and general network transactions and updates, and will be discussed in greater detail later.

The blockchain algorithm assumes that only a simple majority of hosts are honest. Each block, honest hosts must be guaranteed to have their heartbeat included. Furthermore, every honest host must be guaranteed to agree on the same block. This is achieved by the following algorithm:

1. All hosts send their heartbeats to all other hosts.
2. All hosts tell the other hosts which heartbeats they received, and sign that they received these heartbeats. In this step, hosts will fill in gaps in each other's list of heartbeats.
3. All hosts tell the other hosts which heartbeats they know were seen by at least $\frac{n}{2} + 1$ of the hosts on the network, and include the list of signatures proving that the hosts actually saw the heartbeats.

By the end of the third step, the honest hosts will be guaranteed to have all received the same set of heartbeats that have been seen by $\frac{n}{2} + 1$ of the hosts on the network. All hosts will therefore produce the same block. The process can then restart for the next block.

At this stage, a host can be indicted for giving two seperate updates to the network. For example, if a host gives a heartbeat to one host and a different heartbeat to another host, then the dishonest host is indicted and thrown from the swarm.

If a host is absent from the block, they are penalized for being inactive. After being inactive for 24 hours, the host is thrown from the swarm.

4 Concensus Algorithm Proof

Assumptions:

- 51% of the network is honest.
- All honest hosts can communicate freely. (no DOS attacks)
- The network is synchronized, meaning all hosts can complete each step within a known limited timeframe, and hosts can self-correct for drift

In step 1, every honest host will get heartbeats from every other honest host. If $\frac{n}{2}+1$ hosts are honest, every honest host is guaranteed to have their heartbeat seen by at least $\frac{n}{2} + 1$ hosts.

In step 2, every host tells each other host which heartbeats they have seen, and attatch a signature that they saw the heartbeat. This means that any heartbeat that was seen by any honest host will be seen by every other honest host.

In step 3, every host tells each other host which heartbeats they got which were seen by at least $\frac{n}{2}+1$ hosts. This is to guarantee that the heartbeat was seen by at least 1 honest host in the previous step. If it was seen by at least 1 honest host, then every honest host is guaranteed to have received it in this step. Therefore, every honest host is guaranteed to include every heartbeat that was seen by $\frac{n}{2}+1$ hosts in the second step.

5 Dishonest Blockchains

We assume that hosts are randomly assigned to blockchains. In a network where honest hosts only have a simple majority, an attacker can have the majority in many individual blockchains.

We can assume that all blockchains will have at least 15% honest hosts given that half of the hosts on the network are honest.

$$\sum_{i=163}^{192} \binom{192}{i} 0.5^i \, 0.5^{192-i} = 3.869 \times 10^{-24} \qquad (1)$$

If dishonest hosts have a majority in a blockchain, they can push honest hosts offline by preventing the honest heartbeats from getting the signatures required to be put in a block. This allows, in one block, or spread through many blocks, a dishonest blockchain to push off all of the honest hosts.

To protect against this type of attack, each time a host is dropped from the network, 3 additional hosts

are randomly selected to be dropped from the network. Assuming worst case, this means that for each honest host, the next three hosts that get dropped will be half honest and half dishonest, for a total expected drop of 2.5 honest hosts and 1.5 dishonest hosts, or an expected ratio of 60% honest hosts getting dropped. Assuming that the network is half honest, the individual blockchain will converge to 60% dishonest.

If individual blockchains converge to being 60% dishonest, with high probability no blockchain will have less than 15% honest hosts.

$$\sum_{i=163}^{192} \binom{192}{i} 0.6^i \, 0.4^{192-i} = 5.392 \times 10^{-14} \qquad (2)$$

This means that for all blockchains, we can assume that at least 15% hosts are honest.

6 File Storage Layout

Each blockchain has 192 and 256GB per host. Each 256GB block of data is split up into many smaller pieces, which are to be called slices. Slices have no minimum file size, but have a maximum file size of 1mb. The collection of slices on a single block of data is called a stack.

The slices in each stack correcspond to each other. For example, if the third slice on one host is 45kb, then the third slice on every single host is also 45kb. These inter-host sets of congruent slices are called rings.

Files are stored on multiple hosts using Reed-Solomon coding, which is maximum distance separable. First the user picks redundancy settings, which is what volume of the hosts need to retain their piece of the file in order for the file to be recovered. This is m of n recovery, where n equals 192. If m is 1, then the file will be duplicated on every single host, and can be recovered from only a single host. The redundancy is then 192, meaning that a 1GB file will take 192GB of storage on the Sia network. If m is $\frac{n}{2} + 1$, then the file can be recovered from any $\frac{n}{2} + 1$ hosts on the network, and the redundancy will be slightly less than 2. A 1GB file will therefore consume slightly less than 2GB of Sia storage.

Files have header data that includes the hash for each slice in the ring and the value of m. The value of m can be stored as a single byte. Each hash will be 32 bytes and there are 192pieces. The network assumes that under any circumstances, a blockchain will have at least 15%hosts, so the redundancy on the header data will be kept at $6\frac{2}{3}$. Altogether, this means that there is 40kb of overhead for a file.

Part of the heartbeat algorithm performs keepalives on hosts and verifies that they are storing their pieces as promised. If a host goes offline or corrupts their data, the network can use the Reed-Solomon coding to repair the file within minutes - no interaction from the user needed.

It is left to the user to pick optimal settings. When m is small, there is a higher chance that a malicious attacker can control enough of the network to knock the file offline.

7 Proof of Access

Each heartbeat, a host has to provide a proof that they have access to the file pieces that they are storing. Each block, a random 256 bytes segment of the storage stack is chosen to be included in the heartbeat. This segment will correspond to a portion of a file in a file ring. Each host will include their portion of the file in their heartbeat, which means that in the block there is enough information to fully reconstruct a small portion of the file. This will also reveal which hosts have the wrong data.

Only 256 bytes is requested each block because requesting more would greatly increase bandwidth requirements per block. This small amount is sufficient because it is randomly selected. If a host is missing a substantial portion of the file, it will only be a matter of time before some part of the missing portion is randomly selected. This does mean that tiny corrupted sections of file can go undetected for a long time, but even small corruptions will eventually be detected and are unlikely to prevent file recovery.

We can assume that a malicious host will receive all of the honest heartbeats before submitting their

own. This means that, given sufficient redundnacy, a malicious host will be able to rebuild their piece of the file every step from the other heartbeats, can cna perform proof of access without actually storing anything. To prevent that, this process is broken up across two stages.

In the first stage, hosts only submit the hash of the chosen segment prepended with their id. This forces hosts to lock in what their file piece is before they see any information revealed from the other hosts. In the second stage, hosts reveal what string produced the hash. If they reveal a string that does not hash to the string submitted in stage 1, the host is considered corrupt.

If a host submits a second stage string that does not match their first stage hash, they are thrown from the swarm.

If a host submits a second stage string that is discovered as corrupt during decoding, an appeal is started. The host must upload the entire slice that corresponds to the corrupted segment uploaded in the heartbeat. The other hosts upload their chunks of the header data for that slice, which will contain the hash of the slice that the host is supposed to have. If the slice matches the hash, and the segment uploaded matches the slice, then the piece is determined originally corrupt. The host is aquitted, and the file itself is fined to pay for the bandwidth used during the appeal process.

Otherwise, the host treated as not having submitted a heartbeat.

If, during the appeal process, the blockchain is discovered dishonest, or a section of hosts is discovered dishonest, then the blockchain enters a panic state.

8 Proof of Storage

Storage is considered valid on the network if it follows two constraints:

- It costs money to host files on the network.
- The storage is unique to the network.

The assumption for the first bullet point is that if a file costs money to upload, nobody will upload files that are not useful to them in some way. A host will not upload a fake file to himself or the network if the act of doing so forces the host to operate at a net loss. The second bullet point means that the host is not using Sia as it's source of storage, or that the host is not using a source that uses Sia as a source for storage.

The first point is protected by randomly collecting together swarms. Any swarm will be composed of random sets of nodes, and will not have more than 85% dishonest hosts. This means that any host attempting to use themselves as exclusive storage will have to host their files on at least 15% honest hosts. As long as the profit from mining on 85% of a fake file does not exceed the loss of paying for hosting on the 15%honest hosts, there will be no incentive to do fake mining in this manner. Therefore, a hard cap will be placed on the amount of coins that can be mined to satisfy this condition.

The second point is protected by the cost of downloads on the network. Downloading a file costs money, and if you are to complete the proof of access steps in the heartbeat, you need to download a fraction of the file each heartbeat. As long as the download system is constructed such that the cost of downloading repeatedly to perform heartbeat access proofs is greater than the reward from mining, this attack will not be a problem. The system for pricing downloads has currently not been created, but it will be created with this constraint in mind.

Game theory stuff here

9 The Problem of Delegation

These proofs have a weak vulnerability; malicious hosts can collaborate and remove redundancy from the file. Files have a redundancy so that they can be repaired if their hosts go offline. A group of collaborating hosts can remove the redundancy from their file and store a single nonredundant copy. When one of the group needs to do a proof on the file, the group builds the piece they need out of the pieces they have.

This is unhealthy for the network; if a single host goes offline, every host that is supposed to have redundant pieces of the file in the collaborating group

will lose the file. This will result in penalties for the hosts, but also results in a much higher liklihood of corruption of the file.

It is not certain that this can be avoided, though hopefully it can be discouraged.

10 False Redundancy

The network expects that hosts will go offline in a randomly distributed manner. There are penalties if hosts go offline simultaneously, and they are called 'false redundancy penalties.' There are two priciple ways that hosts would go down simultaneously. The first is if they are delegating files. The second is if all of the hosts are in the same geographic location, and some disaster like a power outage happens.

There may be other situations that cause hosts to go offline together. All such situations are viewed by the network as unacceptable and are penalized.

The severities of these penalties have not been decided. It has also not been determined how to measure false redundancy.

Pentalty stuff here

11 Entropy

Each block, a random file segment must be selected. This means that the blockchain needs an unpredictable way to produce entropy, otherwise a dishonest host could look ahead and only store the pieces that it knows will be "randomly" selected. Each block, every host will be required to produce a random string that is 32 bytes in length. There are no requirements on how this is to be done, but it is assumed that honest hosts will have a legitamate method for producing random strings.

Just like the proof of access, the random strings will have two stages. In the first stage, only the hash of the randomly generated string will be revealed. In the second stage, the actual random string will be revealed, and then all of the random strings will be appended deterministically and then hashed to produce the final random string for the block.

By determining entropy this way, we guarantee that if even a single host produces a random string, the final string will also be random.

This method of generating entropy is vulnerable to an attack: dishonest hosts can intentionally withdraw from the network to influence the outcome of the random number. The way to prevent this is to make sure that the penalty for missing a heartbeat is more expenive than favorably generating entropy for a potential 80% of dishonest hosts. There will be a penalty for missing heartbeats, and it will sufficiently satisfy this attack prevention mechanism.

Pentalty stuff here

12 Script Wallets

Wallets on Sia have a balance and a set of scripts. These wallets are responsible for every interaction between Sia and external entities. The scripts manage these interactions.

In addition to some general computational commands, scripts have system calls including

Send(amount)
: Send money from this wallet to another wallet on the network.
Will only send money to existing wallets.
Returns 'yes' if successful, 'no' if unsuccessful.

Rent(volume)
: Rent a volume of storage from the Sia network.
Returns a pointer to the file.

RentProtected(volume)
: Rent a volume of storage from Sia that is write-only.
Returns a pointer t othe file.

Release(filePointer)
: Release a file from the network.
You can only release files that you own.

Endorse(filePointer, amount)
: Sends a volume of money to a file, which will be used to pay for the file in the future.
If the file is not protected, the file will be copied

and the new file will be protected.
This amount can never be withdrawn from the file, and the file cannot be deleted until this amount is consumed.

Wallets consume file storage space on the network. Just like files are charged for consuming space, wallets are charged the same price.

13 Files

14 Updates

An update is a message to the network that needs to be processed. Messages can include money sends, file rents, or announcements of new hosts. Hosts are incentivised to include all updates into their heartbeats, because updates include small monetary compensation. The compensation comes from the entity submitting the update. This means that only entities with wallets on the local blockchain can submit updates.

15 Full Heartbeat

A full heartbeat will contain the following:

- Blockchain
- Parent Block
- Host Id
- File Proof Hash - Stage 1
- File Proof String - Stage 2
- Entropy Proof Hash - Stage 1
- Entropy Proof String - Stage 2
- Update List
- Host Signature

16 A Tree of Blockchains

Blockchains are organized into a tree shape, where 192 blockchains share a parent blockchain. The parent blockchains do not handle updates or store files, and are composed of a random set of hosts chosen from among their children. Parents of parents are also composed of random child hosts, but they are composed of a different set of hosts from the leafs - no host ever participates in more than a single leaf blockchain and a single parent blockchain.

The tree is how hosts can be confident that their siacoins are real and confirmed without needing to know every transaction on the network. Each parent keeps track of the aggregate balance of all the children.

17 Aggregate Transactions

18 Blockchain Certifications

19 Blookchain DHT

20 Economic Model

Siacoins are generated at a rate of twenty thousand per day. The number of siacoins being produced will increase by 5% per year, therefore exactly 1 year from the launch date, approximately twenty one thousand coins will be produced each day.

The goal of the Siacoin is to be a non-volatile currency. The permanent inflationary nature of the Siacoin makes it less attractive to speculation, which is what drives the volatility of Bitcoin. Additionally, the inflation means that over time the cruft of the network will be cleared out.

To fight high frequency trading, a network transaction fee of 1% will be instituted. All transactions using Siacoin will be charged 1%. This 1% is tossed from the network.

21 Conclusion

22 Appendix: Vocabulary

Segment
: An erasure coded piece of a physical file held by a single participant.

Sector
: A logical block of data stored on a quorum. This logical block is composed of a set of equal sized erasure coded segments, one segment held by each participant in the quorum housing the Sector.

Ring
: A physical block of erasure coded data. A ring is every segment for a particular file in a quorum. A ring is what the data looks like after it has been erasure coded; a sector is what the data looks like before it has been erasure coded.

Participant
: A network-connected computer offering a discreet and nonflexible volume of storage to the network.

Quorum
: A set of randomly chosen particpants working in consenses to monitor a fragment of the Sia network.

State
: The status of a quorum-monitored fragment of the Sia network. Every participant in the same quorum will have an identical state. Every quorum has a different state, as every quorum monitors a different fragment of the Sia network.

Block
: A set of updates to the State of a particular quorum.

Heartbeat
: A set of updates from a single participant in a quorum. A block is actually just a list of heartbeats, one from every participant in the quorum that the block acts upon.