Sia Cloud Chain - Distributed Cloud Storage

https://siacloudchain.duckdns.org

Whitepaper modified by: SCC Development Team
Original document written by: Kenneth S Bell, Boris Nagaev, Pavel Dolgov

Abstract

Cloud data storage is currently highly centralized by a few large companies such as Google, Microsoft and Amazon. Organizations around the world have no choice but to store their critical data on those services since the cloud provides huge benefits over storage data locally. However, centralized services come with a risk of data breaches, network outages and government intrusion.

But it doesn't have to be this way. Several decentralized cloud services (Sia, ScPrime, Filecoin, Storj, etc) have emerged to challenge centralized services by building a decentralized cloud storage product on a fully distributed network. Sia Cloud Storage (SCC) joins those decentralized cloud services, but concentrates on serving organizations that must maintain "chain of custody" over its critical data.

Introduction

The Problem - Privacy, Security, Price

A handful of public cloud storage companies hold a decisive share of enterprise data, with Amazon's AWS IaaS product controlling over 49% of the market [R1]. Aggregating data in huge, purpose-built facilities is a reversion to the client-server model on a global scale with increased risk of damaging data breach, monopolistic pricing and questions of data ownership. Business managers need alternatives that return control to the customer.

R1: Nag, Syd. (2019) Market Share Analysis: IaaS and IUS, Worldwide, 2018. https://www.gartner.com/document/3947169?ref=solrAll&refval=225669178&qid=ee7cc875a489136e93c3a8

The Solution - Utilizing a Decentralized Cloud Provider

SCC is a decentralized mesh network of independent, but vetted (via background checks) storage providers compensated for dedicating storage space to create a globally connected "datacenter". Market-based incentives tied to distributed client access remove centralized control. The product weaves together strong encryption, data sharding, cryptographic signatures and a publicly auditable blockchain to ensure the highest security and durability of customer data in a self-organizing and highly cost-competitive environment.

Although SCC has its origins with ScPrime and Sia, it will not operate in the same space and will serve a different customer demographic. The primary focus of SCC is not to offer an almost unlimited amount of storage at inexpensive rates. SCC will operate an ecosystem consisting of a small group of vetted providers (small total system storage) offering services at rather expensive rates. The focus is on building a network of providers that support storage requirements for customers that rely on chain of custody and/or the use of extremely sensitive data. Therefore, each provider will need to pass a comprehensive background check in order to gain the ability to offer storage capacity. In the future, some customers may even require that providers acquire and maintain an active security clearance and/or pass a polygraph test. Providers must be willing to have computer systems (software and hardware) monitored and may be required to surrender computer, storage and network resources in order to support the sensitive operations of SCC's customer base. It is highly recommended that ordinary consumers and business organizations contract storage services from either ScPrime, Sia, Filecoin, or Storj.

Baseline Features

At its core, SCC is an open-source protocol with early feature abstraction. Leveraging open-source development is cost effective and provides high quality software as a starting point. The features that make SCC unique will require proprietary innovation, but the open-source code allows for meeting a tighter product deliverable timeline. The following features form the basis we build on:

Consensus

A distributed ledger of financial transactions between participants is publicly validated by computers executing Proof-of-Work (POW) algorithms in order to bundle "blocks" of transactions, colloquially referred to as mining. The result of a publicly verified transaction consensus is known as a blockchain [R2]. SCC has a block reward many times smaller than its peers. The minimal payout was adopted to encourage miners to use low power devices instead of power-hungry ASICs.

R2: Nakamoto, Satoshi. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System
https://bitcoin.org/bitcoin.pdf

Payment Channel / Smart Contracts

The project uses a state channel architecture and P2P networking for contract formation between storage clients and storage providers. Contracts stage data uploads via agreement on price, length of time and other criteria. Following a successful negotiation and data upload, providers submit regular proof that data is held as contracted. Publicly auditable transactions ensure valid proof results from ongoing contract payments while the inability to furnish proof results initially in non-payment and ultimately contract invalidation. Contracts include a root cryptographic hash composed of individual segments that are hashed into a Merkle tree. The root hash along with the data object size is used to verify storage proofs. Contracts specify duration, challenge frequency and payouts. Each client creates contracts with many providers based on desired price and performance characteristics. Finally, contracts include a fee deduction with the proceeds accruing to a secondary crypto token [R3]

R3 Champine, Luke. & Vorick, David. (2014) Sia: Simple Decentralized Storage
https://sia.tech/sia.pdf

Erasure Codes

Reed-Solomon erasure codes distribute data mathematically across a set of sectors, drives, nodes or computers for higher durability and efficiency over simple replication. Data is fragmented, expanded and encoded with redundant parity pieces, which are stored across a set of locations where statistically significant numbers may failover before original data is unrecoverable.

Encryption

The greatest challenge for cloud storage providers is privacy and security of client data. The average cost of an enterprise data breach is just under $1.5m in an environment companies have complete control over [R4]. Moving to the cloud involves relinquishing control and trusting another firm. The solution is an enforced end-to-end encryption before allowing data onto the network. Client-side encryption creates unique engineering challenges but is the basis for truly secure and private data that is protected from unauthorized access. As of this writing, most traditional cloud storage companies do not enforce end-to-end encryption.

R4: Mendoza, N.F. (2019) Data breaches now cost companies an average of $1.41 million. https://www.techrepublic.com/article/data-breaches-now-cost-companies-an-average-of-1-41-million/

Licensing/Compensation

The base code is freely and publicly usable on a non-restrictive MIT open-source license. SCC builds upon the open-source SCP source code. SCC will obviously not use any of SCP's proprietary code. The SCC product will be a mix of open source and its own proprietary code to support the product's unique focus. The entire license follows [R5].

R5: Champine, Luke. (2016) MIT Licensehttps://gitlab.com/NebulousLabs/Sia/blob/master/LICENSE

Business Model

Public cloud services and infrastructure are growing exponentially with compound annual growth rates (CAGR) of 20% forecast through 2022 and increasing beyond [R6]. Even with potential risk, enterprise is transitioning from onsite capital expense outlays and personnel to predictable operational expenses in the cloud. Competition is expected from incumbent providers and new technology, yet the opportunity is so large there will likely be many winners. The successful competitors will start by meeting basic requirements.

R6: Smith, Eileen. (2019) Worldwide Public Cloud Services Spending Will More Than Double by 2023. Retrieved from https://www.idc.com/getdoc.jsp?containerId=prUS45340719

- Compliance regimes for data access, retention and removal
- Service level agreements (SLAs) to guarantee performance
- Compatibility with de facto standards.

Competition

Amazon, Microsoft and Google account for an overwhelming percentage of the Enterprise cloud storage market with breathtaking scale across the planet. Individual AWS data centers house millions of servers running custom networking software and routing equipment in the single most profitable Amazon business segment and cost the company $500-600m to build. Though thousands of cloud storage brands exist, the majority simply rebrand and resell AWS capacity under a white label licensing. Competing with an entrenched, dominant incumbent requires superior privacy, security and performance delivered at a significantly lower price.

Market Opportunity

Predicted exponential growth begins with approximately 40 zettabytes (1 zettabyte equals a billion terabytes) of globally created data in 2018 leading to a fifteen-fold increase by 2030 [R7]. Falling costs and better security enable storage of data previously discarded or kept on less reliable and inexpensive media. One example are daily backup activities previously reserved for onsite tape mechanisms with media transported to secure offsite locations for preservation. While the current providers will spend on new installations, storage capacity must also come from new and novel directions if the need is to be met.

R7: Buss, Sebastian & Becker, Dennis & multiple others (2019) Digital Economy Compass
Retrieved from https://www.statista.com/study/52194/digital-economy-compass/

The SCC product provides complete (S3) compatibility, taking advantage of a well-developed ecosystem of 3rd party tools. The decentralized model allows organizations to save money, while still getting the benefits of encryption, redundancy and chain of custody guarantees.

Compliance

Data creators are increasingly required to adhere to a panoply of compliance regimes based on data protection, retention policies, privacy requirements, geographic region and more. A recent high-profile example is the European General Data Protection Regulation (GDPR) which gives a strict set of guidelines for enterprise use of personal data. The network must provide a transparent mechanism with auditability to meet compliance requirements.

SCC requires independent storage providers to forgo anonymity, allow system inventory and the use of performance monitoring tools. In this way SCC can comply with chain of custody standards. A software component called the "Relayer" has the ability to identify and list storage provider profiles for audit.

Service Level Agreements
SCC, as the operator of a decentralized network can create Service Level Agreements and compensate on performance not meeting a set standard. This is achieved with a thorough inventory of the provider network, using incentives to gain specific performance and a clear presentation of expectation when choosing a group of providers for any given contract-set.

Economics
At project startup, a new blockchain instantiated from a “genesis” block started emitting 30 SCC coins to miners solving the cryptographic riddle. The coin emission schedule will stay constant throughout the life of the project. A project development fee starting at 10% of the block reward and declining over a two-year period was set aside for developer costs. Based on general market conditions, this fund cannot suffice, and private investment is required to complete the product.

A “pre-mine” of coins created in the genesis block is meant to reward and incentivize potential supporters. Additionally, the pre-mine aimed to compensate longtime supporters of the SiaClassic blockchain. A breakdown follows:

- 126,000 for mining pool operators to assist with maturity payout (3600 SCC/day,7 days, 5 pools)
- 1 million to ScPrime for software development. Nebulous, Inc is acknowledged as the original developer of Sia. However, they have not been allocated any SCC coins based on the fact that they rejected ScPrime's coin offer. It is likely the same will happen with an offer from SCC.
- 9 million for initial project expenses (equipment, software, and supplies)
- 10 million for development team compensation (over 3 years)
- 100 million for airdrop to holders of the original SiaClassic coins generated from block 270K to 370K. Work is underway to recover a consensus file to enable those coins to transfer. Other means of verification are being evaluated as an alternative.
- 200 million for investors. It is expected that several rounds of investor engagement will happen. The 200 million SCC coins will be divided into smaller amounts and deposited into multiple wallets that will then be used to support the fund-raising operations.

The investment opportunities will be:

Number available	Number of coins	Price	Investment	Total Raised
2	40 million	$0.00025/coin	$10,000	$20,000
10	10 million	$0.0005/coin	$5,000	$50,000
20	1 million	$0.001/coin	$1,000	$20,000
Total = 32	Total = 200 million	-	-	Total = $90,000

Airdrop
SCC decided to conduct a 1:30 ratio airdrop for SiaClassic (former fork of SC) holders. The goal was to compensate individuals for investment into the failed SiaClassic project. Funds were created in the Genesis block to cover the period in time from block 270K to 370K of the SiaClassic blockchain. This time period would have generated about 3 billion SiaClassic coins. The airdrop will provide 1 Sia Cloud Chain coin for 30 SiaClassic coins. The SiaClassic wallet balance will have to be verified and a new Sia Cloud Chain address provided for the airdrop. As of today, SiaMining.com maintains a SiaClassic mining pool

and Tokenview.com maintains a block explorer for SiaClassic. The Sia Cloud Chain members have reached out to both organizations for a copy of the consensus file they maintain to operate the mining pool and block explorer. Those attempts were unsuccessful and the team is exploring other ways to verify wallet balances. Ultimately, any unclaimed airdrop funds will be distributed randomly to active holders of Sia Cloud Chain coins.

ScPrimefunds

The SCP project makes use of ScPrimefunds (SPF) as a means to provide equity in the network and earn a percentage of every transaction between clients and providers. Although 10K SPF were created on the genesis block, the SCC project does not utilize SPF in any capacity. The creation of those SPF was to keep the code in line with the original code. The developers of SCC fear that removing the SPF will introduce software bugs. All 10K SPF were placed into one address during the genesis block. This topic will be revisited at a future date and the SPF may be burned if the project will continue without them.

Supply and Coin Velocity

When thinking about whether a protocol's token can capture and sustain economic rent, what is relevant is whether the mining industry maintaining the protocol's blockchain is competitive, not the stickiness of users. The mining industry supporting any decentralized protocol must be a competitive market; otherwise, the protocol isn't decentralized [8].

R8: Pfeffer, John. (2017) An (Institutional) Investor's Take on Cryptoassets
https://s3.eu-west-2.amazonaws.com/john-pfeffer/An+Investor%27s+Take+on+Cryptoassets+v6.pdf

Products based on crypto-assets may suffer from a velocity problem limiting coin value and possibly leading to a lack of miners willing to spend electricity unprofitably. However, SCC mining software should be run on standard computers that will not consume much more electricity than the owners are already using for normal operations. SCC has about 320 million coins in circulation as of this writing. Miners claim a 30 SCC coin reward for each 12-minute block. The low amount of SCC introduced during mining helps to keep coins at a low velocity and ensures that they retain (or increase) their value. SCC coins can also be purchased on the open market instead of relying on mining. The project is currently setting up an atomic swap capability to allow the purchase and sale of SCC coins in a decentralized manner.

SCC will use a burning mechanism as a velocity sink if the coin velocity increases dramatically. However, this is not expected to happen as SCC has such a low emission rate. Alternative consensus mechanisms like Proof-of-Stake attempt to address velocity with large asset pools frozen to create artificial scarcity but the process has unintended consequences such as centralizing the coin supply.

With SCC, contract functions naturally slow coin velocity and create further scarcity with the additional benefit of blocking potential Sybil attacks, in which a provider(s) controlling a large percentage of a contract set could extort or damage customers. Allowance and collateral are features that act in concert to create an organic velocity sink as the network reaches scale. At contract formation, clients put coins into the contract in advance as an allowance to cover a predicted amount of storage space for a given period. As the contract progresses, amounts are deducted from the Allowance as payment until successful contract resolution or termination for non-performance. The Allowance is based on average storage pricing and occurs "under-the-hood" of the customer software.

Storage providers ensure contract performance by putting up collateral against the contract as insurance contracts are faithfully carried out. If a provider fails to successfully provide a storage proof, collateral funds are deducted from the contract pool and sent to an address specified for expenses associated with the operation of SCC. The financial penalty prevents providers from claiming to possess data they do not have or to gain control over any client's contract set. At the time of this writing, collateralization is set to approximately 1.5 to 2.5 times the cost of storage, though a 1:1 ratio likely serves the purpose and will become the future project recommendation.

Community Edition / 3rd Parties

To preserve decentralization, the project maintains an open-source Community Edition client for basic functionality. This version may not provide full distributed renter features but can be used by project supporters for basic archival storage or as a trial software.

Third party developers may also use the network as it is based on an open protocol. SCP can use incentives to prioritize customer traffic over 3rd parties or could engage them for shared revenue streams on the network.

Offensive/Illegal Material – Terms of Service

Decentralization is a response to infrastructure monopolized by entities who use the regulatory environment for profitable advantage and sometimes contrary to the spirit or intention of the rules. The public understands a need to prevent exploitation and the promotion of harmful illegal activities, but also recognize data companies partnering with the intelligence community or local police agencies to share data behind the scenes is an overreach.

In the ongoing evolution of decentralization, some proponents espouse absolute privacy and total freedoms. SCC recognizes the path to privacy and security requires a middle ground response. The product Terms of Service will include clear guidelines of what is legally permissible/acceptable on the SCC network and customer acceptance and adherence are required for continued use.

Some open-source cloud storage protocols are building tools to circumvent clear-headed rules, but may end up relegated to providing services for illegal content. The business use case for growing cloud storage can provide for privacy and security of personal data while still limiting harmful activities. The SCC product is intended to bridge the gap.

Implementation

Blockchain Specifics - Proof of Work

A key innovation of digital currency is cryptographic signature-based ownership which streamlines transfer of value at the cost of a potential double-spend problem where an asset sent to one party can then be sent again to a 2nd party. Traditional currencies rely on central banks to provide verification services to prevent double spending. Decentralized currencies use public verification, known as consensus on a blockchain. Entities known as miners "vote" on collections of transactions (a “block”) by guessing a discrete hash that is costly to produce through a randomized, low probability process requiring non-trivial trial and error on average before a valid proof is generated. A blockchain proof (as distinguished from a storage proof) must be accepted by a majority of participants and is appended to the previous block creating a "chain" of all transactions. The net cost of this proof-of-work consensus is the amount of electricity burned in solving the puzzle. The expense is large enough to ensure process validity transparently.

Successfully providing a valid proof (finding a block) generates a reward in coins used as currency on the SCC storage network. To compensate for increasing hardware speed and a varying number of miners, puzzle difficulty is adjusted over time by a moving average targeting a number of blocks with average block time set to a 10-minute average. If too many blocks are found in a period, difficulty increases to make it harder to discover a solution with the opposite occurring if too few blocks are found. Adjustments are banded at 1/3rd of and 3x the target block time with no individual block adjustment greater than 0.4%. Because storage contracts run for significant lengths of time, there is little value to faster block times. Block sizes are currently adequate for the amount of transaction volume, but solutions for scaling will need to be found in later years when the blockchain has experienced growth.

Transactions

Outputs represent an amount of coins with an identifier derived from transactions where output i in transaction t is defined as: H (t|| “output” ||i). H is a cryptographic hashing function and “output” is a string literal. Every input must come from a prior output, so an input is simply an output ID. Inputs and outputs are also paired with a set of spend conditions. Inputs contain the spend conditions themselves, while outputs contain their Merkle root hash [9].

R9: Champine, Luke. & Vorick, David. (2014) Sia: Simple Decentralized Storage
https://sia.tech/sia.pdf

Spend conditions must be met to unlock coins from a contract and include a time lock, a set of public keys and the number of signatures required. An output cannot be spent until the time lock has expired and enough of the specified keys have added their signature.

The spend conditions are hashed into a Merkle tree, using the time lock, the number of signatures required, and the public keys as leaves. The root hash of this tree is used as the address to which the coins are sent. In order to spend the coins, the spend conditions corresponding to the address hash must be provided. The use of a Merkle tree allows parties to selectively reveal information in the spend conditions. For example, the time lock can be revealed without revealing the number of public keys or the number of signatures required.

Cryptographic signatures are required for each transaction and paired with an input ID, a time lock, and a set of flags indicating which parts of the transaction have been signed. The input ID indicates which input the signature is being applied to. The time lock specifies when the signature becomes valid

Contracts

Contracts specify the length of time an object dataset is to be stored, likely defaulting to 30 days to coincide with customer billing cycles and also specify cost information not directly exposed to storage customers but used as a key input to formulate prices end-customers pay, including baseline storage and bandwidth costs. Contract formation fees charged to the client are slated for removal as an extraneous process with providers covering transaction fees as a cost of business. Storage proof provisioning and scheduling is also included in the contract.

Upon setup, the client software polls the network to get a list of all available providers. Based on desired attributes and requirements as set in the client, a group of storage providers are chosen for contract formation with more contracts created than are ultimately used to cover provider churn and availability. At present, contracts are limited to single instance use. A major feature of the SCC software is allowing multiple entities to access contract sets from multiple machines, key to a real public cloud storage competitor.

Storage Proofs

Storage proofs prevent a provider from faking or deduplicating data to store something less than the original agreement calls for. Providers submit proofs within an agreed upon window. Valid proofs result in a partial payment from the contract to the provider up to the contract resolution or renewal. If a valid proof is not provided, payment is instead sent to a [change to expense address] burn address. After a specified number of missed proofs, contracts are invalidated, and the segment is added to a calculation for repair.

Storage proofs do not have inputs or outputs; only a contract ID and proof data consisting of a segment of the original file and a list of hashes from the file's Merkle tree. Upon submission to the blockchain, proofs become publicly auditable. Storage proofs use a randomly selected segment and a random seed for challenge window Wi is given by: H(contract ID||H(Bi−1)) where Bi−1 is the block immediately prior to the beginning of Wi.

Relayer - Distributed Renter

The Relayer provides distributed client functionality. An application instance (or cluster of instances), the Relayer may reside internally at a customer site or run as an EC2 instance on AWS. The Relayer provides metadata efficiency using a database architecture for key pair and Posix attribute information. Metadata is indexed for searchability and easy recoverability with the database itself uploaded to the network for additional durability/redundancy [metadata database]. Additional features are included in Appendix A.

A distributed client allows multiple seats in an organization to access the same object datasets using a common set of contracts, key functionality to provide true file and object sharing, permissioning and other standard cloud storage features not currently possible in other decentralized storage projects.

In an example organizational setting, the HR department creates a bucket and uploads a project spreadsheet which the engineering department modifies. They might then attach new CAD drawings into the same bucket on the same contract set. Access control lists or other permissioning capabilities allow or deny read/modify/delete capability for objects and buckets, both inside and outside of the organization. The architecture also enables features like lifecycle rules and advanced versioning.

Wallet/Exchange

Distributed cloud storage has discernible benefits over traditional replicated clouds though inclusion of a native crypto currency is likely to create friction in adoption. Acquisition and custodianship of digital assets in a rapidly evolving space represents a new layer of risk for business. At the same time, there are a myriad of horror stories from companies involved in crypto-assets and people losing money to hackers and fraudsters. Though the Relayer and storage provider network operates on a native crypto coin, companies are not exposed to it in normal operation. Administrators have the ability to monitor coin transactions and wallet operations but have no custody responsibility or requirement to procure an ongoing coin budget.

Customer purchase and payment processes are similar to traditional cloud storage providers (credit card, purchase order through a web-based interface, etc). After receipt of payment, the software is authorized to purchase coins directly via API which are then locked into the contract set.

When more storage is required and contract renewals occur, additional coins are procured automatically at market prices. There is no excess of coins exposed in wallets. A multisignature wallet prevents customers from accessing Relayer features beyond what has been paid for and authorized by the application [10].

R:10 Multisignature
https://en.bitcoin.it/wiki/Multisignature

Erasure Codes

Redundancy and durability refer to thresholds before stored data is lost, damaged or unavailable. The most common redundancy schema is replication across multiple instances and ideally over a diverse geographic area. RAID configurations replicate datasets across drives in a single server, while multiple server installations may replicate data within a facility, but an event limiting access to the datacenter removes access to all copies. Expanding copies to other facilities is good but impacts cost and performance with at least two full copies required and increased bandwidth to upload and access them. For acceptable redundancy, replication must scale to 8x or higher and usually require copies over a wide geographical footprint.

Erasure codes provide higher durability with significant cost reduction through a mathematical process of data segmentation that includes copies of segments called parity pieces. The encoded segments are sent to separate storage locations with only a subset of pieces required to rebuild/download the entire dataset. Reed-Solomon codes use polynomial interpolation to create a scenario where n is the minimum number of pieces for a complete download, k is the number added parity pieces and m is the total number of pieces or erasure shares.

For example, splitting data into 2 pieces and adding 2 parity pieces create 4 pieces, which we can distribute to 4 separate locations. Of the 4, any two are able to provide a complete download and any two can fail with the data still fully available. In this scenario, the data is expanded 2x or the same as a single replication yet now two locations failing do not result in data loss. Increasing the number of pieces increases durability without impacting data expansion. Standard cloud companies with a small number of large facilities are limited in the number of pieces they can distribute and are forced to cover a large geographical area. They may use codes to limit replication across servers to mitigate risk of individual hardware failure, but usually within a single datacenter.

Contrast this with a network of thousands of hosts scattered throughout a customer region. Large code sets created locally with additional distant providers to protect against regional catastrophe create durability of 11 nines (99.999999999%) without extreme cost increase. SCC is coded to 10/30 where 30 pieces are uploaded and any 10 can provide full retrieval while the Relayer lets the customer decide on a variety of capabilities with code factor modification occurring automatically under the hood.

Replication has benefit for the small and individual file object repair and replace where erasure codes require an entire segment piece be modified and re-uploaded. Possible methods to address include smaller piece size, caching at the Relayer and possibly using replication for small files.

Durability Factors

Erasure codes create feature and performance flexibility using a dynamic contract set configuration in the Relayer client. Currently, a single client accesses a static set of contracts. A distributed contractor allows multiple client seats access to groups of contract sets, each specifying separate cost, performance, durability factors, access permissions and lifecycle rules. Some of these sets can have static properties with others dynamically changing across renewals. Some example use-cases follow.

Content Distribution Networks

Data produced and consumed at a single location may perform well with a localized aggregation of providers while a frequently accessed website with global traffic requires a larger contract set and provider sub-groupings located strategically to accommodate traffic surges. An erasure code factor with regional provider sub-groups strategically located looks a lot like an ad-hoc content distribution network (CDN) with costs far lower than traditional replication to the network edge.

Dynamic Durability

Different types of data with varied access profiles can also benefit. New and regularly produced data is often accessed heavily when first available then tapering over time with a long tail before reaching archival status. Higher code factors initially make more copies accessible over more providers ensuring low latency and high availability. As data ages, the number of providers in a contract set decay gracefully until a lifecycle rule (eg 90 days) specifies archival status with customer pricing changing along with the rules. In a traditional cloud company, the final step could incur additional egress fees while the last group of SCP providers do simple contract renewals.

Latency/Performance

Code factors impact latency and download performance. The core protocol envisions parallelism, but performance is not strong in practice. Latency and download performance may be enhanced through worker prioritization when file objects are requested. Using a 10/90 code factor as an example, any 10 pieces can create a full download. Tasking all 90 for the initial download, the fastest arriving pieces conclude with the rest discarded. Instead of grading providers with a score, the protocol should work through network congestion or provider saturation with no priority given to which provider ultimately sends the piece. There is higher cost to additional providers and higher bandwidth use with more discarded traffic. Customers pay a premium price for a tier of storage providing ultra-fast downloads and the lowest latency.

Custom Contract Sets

With the ability for multi-access contract sets and a network of varied providers, features are possible from custom contract sets based on desired provider qualities. An example is performance tiering where a target group of providers with fast downloads are included. The Relayer provides robust ability to find and select provider groups.

The Relayer can group full erasure coded copies inside the firewall and across corporate providers, including provider software on normal desktop clients. For large enterprises, it is possible to configure enough hosts internally on desktop equipment and potentially negating large storage server arrays. The benefits are higher durability and lower cost.

Most companies will prefer to continue using storage servers already configured. For these, the Relayer may communicate and use simple replication on internal data while seamlessly providing bucket and domain interfacing alongside external data. At minimum, with full S3 compatibility as outlined in the next section, companies can use one of many established management tools to combine internal data with SCP hosted data.

AWS S3 compatibility

Amazon's S3 object storage is a de facto standard in cloud storage. With hundreds of 3rd party applications and enterprise adoption/acceptance, S3 compatibility on distributed networks is a requirement. Applications should gracefully transition to the SCP network with as little as a URL change. At minimum, a user must be able to choose an arbitrary key, presented as a path to map data pieces to specific hosts including object hierarchies. Buckets are collections of objects included in these hierarchies with individual objects inside of a given Bucket also having a unique ID and path [11].

R11: (2019) Amazon S3 REST API Introduction
https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html

The base API must include the following for file object operations:

- Put - store in a given pathname
- Get - retrieve any given pathname
- List - listing of paths
- Delete - remove any given path

as well as Bucket operations:

- Create
- Delete
- List

Technical Proposals

The following are design proposals describing core architecture making up Relayer and provider functionality. This is the core innovation of the project and the foundation that allows for a true distributed cloud storage set of products.

Prepaid downloads

Creates a download budget within the confines of the initial contract parameters, including partial and full object downloads. When coupled with key-value addressing and Posix metadata identifying objects as “public” permissioned, this opens public file and object access for websites and other shared access. To facilitate this module, we add two Remote Procedure Calls (RPC) to the Relayer server.

1. TopUpToken (token [32] byte, resources) - creates/adds resources to the token specified in the argument. Funds are provided in a new contract revision with the Relayer client calling the RPC and paying the provider via the token for the budget allocation. Resources are of two types: number of bytes to download and number of sectors that need to be accessed. Both are used to calculate total cost of a download and upon using the token, available resources (number of sectors and bytes used by download) are decreased on the provider.

2. DownloadWithToken (token [32] byte, sector, offset, length) - clients download data internally or publicly shared links with resources drawn down from the token. Does not create new a contract revision.

Benefits

Multiple entities may download data objects from a contract simultaneously instead of each contract being associated with a single user or instance. Using tokens, one machine calls TopUpToken and then all instances call DownloadWithToken using the token. Most importantly, users do not require a full blockchain installation or access to the contract to use the token.

- Users upload data, create links, encode server names, tokens and initial sector ID (the sector will store the list of sector IDs with actual data), and give that link to another user with simple addressing.

- Exposing the DownloadWithToken RPC via websocket, all data on the contract is available to Web applications running in browsers with users able to download directly from the host network. A budget can be specified allowing for websites/applications to “pay” the host network using TopUpToken RPC with a small bit of JavaScript embedded in the site pages calling DownloadWithToken RPC via websocket to download the data.

Deferred Contract Updates

In the current base implementation, object uploading or modification in a distributed environment is limited to a single entity/action with contracts entering locked status as information is added or revised. Examples are changing allowance, prices, uploading, etc. Multiple concurrent object actions from diverse sources are not possible. The client(s) must wait for an action to conclude and the contract to be unlocked before proceeding. To allow for a distributed client, a deferred update architecture is introduced for contract modification.

As with Prepaid Downloads, the token is used and topped up as needed. A temporary key-value is introduced on the provider, outside of the contract. The Relayer references key-value pairs using a key from a new CopyFrom RPC. The renter module has full create, read, update and delete functionality (CRUD) of keys on host storage where there is an active and valid contract. Clients accessing the contracts/providers upload data until the token is depleted. A database structure (LevelDB) on the provider holds uploaded data key-pairs.

Description

Storage in the key-value store is charged based on amount used and time stored. The renter module performs a token “top-up” in putting a specified number of bytes/second on a new token. This budget is specified automatically in the Relayer configuration. Clients accessing the contract for object operations on the key-value store call the token. Storage providers track the remaining budget on the token, removing associated key-values when it reaches zero. Reading from the key-value store requires the token for prepaid downloads and is charged in the same way (downloaded X bytes => decrease token balance by X). The same model is used for listing keys and the unit of value is "byte" with pricing based on individual host prices. Token balance is decreased when a record is created or updated.

Each key in the key-value store is owned by the token and only this token is allowed to remove or update the record while any token can read it if it knows the key.

Storage Provider

The provider uses a local key-value store (e.g. leveldb) to store key-value pairs. Each value must include the ID of a parent token. A map from the parent token lists all associated keys, total used storage in bytes, last update timestamp and balance at that time. Expiration can be derived from this information with an In-memory priority queue to trigger removal of keys when the expiration is reached.

Client

New object data is immediately uploaded into the key-value store on providers and initially held locally as a temporary backup. When enough data is uploaded, a sector is created using CopyFrom and the data is removed from the key-value store and deleted from the local cache.

Micro-sectors

Tiny files are stored in one or multiple consecutive 16kB micro-sectors aligned inside normal 4MB sectors. These objects are put into key-value stores until the total size of an individual micro-sector reaches 4MB and a new sector is created.

High Throughput

Temporary storage allows large data uploads and parallelism in contract access. Data is uploaded from multiple machines and/or multiple locations and put into key-value storage. Clients send keys of uploaded data to the coordinator machine (Relayer) which handles contract updates with CopyFrom calls (sending keys as sources of data) and removes the keys from the key-value store.

Metadata Database

The foundation for a distributed client with multiple customers accessing the same contracts is to store object metadata in a database architecture. CockroachDB is chosen for SQL architecture with Pebble used for key-values. The database instance stores complex hierarchies like S3 buckets, objects, filesystems and blocks.

Relayer server instances run the following components (as code modules within a single binary, or as separate microservices):

- Contractor/Coordinator
- CockroachDB instance
- Pebble key-value storage
- File operations handler for Pebble
- Daemon with common list of contracts
- Lock service
- S3-compatible overlay

Components interact with multiple machine instances and platforms (completely scalable, any-to-any) with clients on mobile or web applications with standard authentication capability. Data originates with clients and is sent to the Relayer as a user request, which may include S3 objects, files or file operations (FUSE). Reed-Solomon erasure coding is applied to the data based on a redundancy factor chosen by the client [EC] and the resulting data shards are encrypted [encryption]. The Coordinator gets corresponding contracts and sectors for data and uploads to the storage provider network.

Keypair metadata is formed and combined with user specified metadata such as user permissions. Keypair metadata contains sector IDs, storage provider and encryption information. It is everything required to recover where a chunk of data is stored and how to read it back from the network of providers.

Metadata components may be calculated before data is uploaded to the provider network but becomes valid only after it is uploaded (sector IDs do not yet exist on the providers). In the case of FUSE object modifications, if a part of a file is changed, the metadata needs to be merged with existing metadata of the unmodified parts of the file. Otherwise, this metadata itself is a new full record.

Metadata is saved to a CockroachDB SQL database instance built on top of a single key-value store. CockroachDB is paired with Pebble (a LevelDB-inspired key-value store) with CockroachDB performing high-level SQL logic and transforming the information into a form Pebble can deal with. Pebble provides custom file handlers to implement the underlying file storage interface.

File handlers are a separate module which convert file operations requested by Pebble into a remote procedure call (RPC) to the storage network providers through the daemon (to upload or retrieve). File handlers also have a local file cache, to return frequently requested files quickly from local disk storage attached to the Relayer instead of retrieving from the storage network. This approach optimizes traffic and reduces latency. The Relayer includes caching for data to return frequent files from the disk.

A list of contracts must be common for all Relayer instances (in the case of a cluster), to ensure each contract is only used for writing by one process at a time. A Lock Service is used to prevent conflicting contract modification. Operations requiring specific contracts (pebble handler) must wait on the Lock Service similar to a mutual exclusion object (mutex).

Contract layout

Contracts reference two types of sectors: DB and File. DB sectors are divided into three categories:

- The first sector stores a list of all DB sectors (like MANIFEST file in databases). It also stores any tiny files produced by the DB and the data to inherit encryption keys from (see encryption section below).
- SSTable sectors These are unchangeable sectors representing database's SSTables (one sector for one SSTable). Sectors are 4MB size. They are created when the DB log file is sorted or when several sstables are merged. Their size is fixed.
- Sectors for append-only file (DB's LOG file) A log file (*.log) stores a sequence of recent updates in binary form. Each update is appended to the current log file. When the log file reaches a predetermined size (approximately 4MB by default), it is converted to a sorted table and a new log file is created for future updates. These sectors are used to store the LOG file. The core protocol allows sector modification, including partial which allows the Relayer to implement object append operations.

Encryption of DB sectors

A stream cipher approach is used to encrypt database sectors, using the key and utility data (Nonce) from the first sector. The first sector stores a set of nonces with each corresponding to one SSTable sector. The encryption key provided by the user and nonce is used to generate a 4MB-long key to encrypt an individual SSTable sector. This key is XOR'ed with the actual data to encrypt a sector. Nonces are simple sequential numbers, increasing each time the encryption happens in order to get new keys. A similar method is used to encrypt the first sector and special append-only sector.

Bootstrap

When the Relayer is first bootstrapped, a random seed is generated to create a new wallet and a new database instance without establishing contracts with the host network. The Contractor forms contracts from the storage provider database as based on administration decisions, excluding contracts already used by other Cockroach instances (from SQL tables). The contracts are saved to a table and the Relayer writes the ID of its Cockroach instance to the first sector of each contract and writes local database sectors to the hosts. The contract is then reserved for the given Cockroach instance globally.

Recovery

The Relayer is intended to keep a small amount of local data, primarily tiny objects and some frequently accessed items. Tiny files can be in-lined inside the database with frequently accessed objects cached. Recovering the Relayer instance includes the database, contracts and all key-value stores using the master password created when the Relayer is first instantiated. When a full recovery is initiated, the process locates contracts by querying the network and downloading all database IDs from the host network and repopulating the local database.

New RPCs

Erasure shares distributed over a large provider set create an issue for small object handling. In the present implementation, data shards are large enough (4MB) to allow strong durability without a large contract set. With files less than 4MB in size (e.g. jpg images), client software requires a method to identify the individual hash of a given shard and then a way to access specific sectors within the shard. Typically, the solution is to return the entire shard and incurring bandwidth and new charges if the file is modified and re-uploaded (saved).

New Remote Procedure Calls (RPC) mitigate this issue. 4MB sectors on storage providers are divided into 256 “micro-sectors”, each equal to 16KB. Small files may occupy from 1 to the 255 of these special sectors and allow for the creation of addressing to locate and access any given file object. This functionality creates an issue when some number of micro-sectors are released due to file object deletion or revision. A new process for sector defragmentation is required to reuse this space efficiently and do correct accounting on the host nodes.

One method is to wait for a new file object small enough to fit into the vacated space. A better solution is to set up a condition when the number of free micro-sectors is more than 50% of a full sector, defragment by waiting for two 50%-free sectors and copying the data to a new sector. The new RPCs to accommodate are as follows:

- HashMicrosectors([]struct{SectorID, microsectorSize int}) ([]struct{[]hashMicrosectors}) The RPC verifies individual micro-sectors while microsectorSize specifies a fixed micro-sector size without hardcoding a 16kB size. It takes a slice of the sector’s ID along with the micro-sector size then returns a slice of slices of the micro-sector hashes. microsectorSize is a power of 2 from 64 (crypto.SegmentSize) to 4M (modules.SectorSize). Returned hashes from the micro-sectors are the same hashes from a Merkle Tree that are on the level of given micro-sector size in the tree.

- CopyFrom(*ModWriteRequest) (LoopWriteResponse) ModWriteRequest = WriteRequest, with different Action struct. Our Action is the same except with Update/Append. Instead of data only, it's either data or LoopReadRequestSection, or key with (offset, length combination) in our key-value for Deferred Contract Updates. If CopyFrom receives a request to return the hashes of one or more micro-sectors, it returns them before reading the signature from renter. CopyFrom is used to copy sectors without having to download/reupload them.

Golang describing new RPC configuration:
https://play.golang.org/p/u1fYMyd2Ran

Network Development

Incentives

In a distributed sharing economy network, independent actors are marshaled to provide resources in exchange for financial incentives. They deploy assets already in service or bring new equipment online if earnings suggest profitability. Baseline prices are established via a market mechanism that allow providers to set the price of storage arbitrarily with competition for contracts guaranteeing an aggressive floor.

Incentives align goals of providers, customers and the network operator while improving price discovery, performance and capacity. Increased rewards begin on adherence to project guidelines.

The first of these binds storage pricing to a range in the face of potentially rapid and significant changes in underlying crypto-asset prices. This will be quickly ingested into the SCC protocol with hosts able to set auto-pricing in line with project recommendations directly in the provider software.

Incentive tiering also envisions provider self-identification (equipment inventory and personnel), performance stratification, capacity and geographic need as rewardable behavior. Network inventory, mapping and performance monitoring capability at SCC headquarters will lead providers to more lucrative contracts and guide them on how to capture maximum reward. Compliance regimes such as HIPAA could require custom firewalling or certifications beyond normal provisioning. The initial reward structure encourages providers to dedicate capacity, adhere to a pricing floor, and maintain a vetted status through recurring background checks.

Provider profitability is critical to project success and incentives will always be a component of provider acquisition and curation. With multiple decentralized storage projects in development and on the horizon, intense competition for available storage makes provider capture and retention dependent on profitability. Projects/products unable to provide lucrative returns may never achieve network effects required to compete on a serious level. At minimum, less competitive networks will incur higher churn rates as providers seek the highest yield on storage assets.

Churn/Object Repair
Networks experience data loss due to component failure, misconfiguration and abandonment. Cost of remediation is measured in equipment replacement, personnel cost, downtime opportunity cost and bandwidth required to restore lost durability. Centralized providers tend to be well-prepared for unplanned equipment failure and outages, maintaining spares and proactive equipment swap-outs though it is difficult to be prepared for a major facility or region outage.

Distributed P2P networks experience unplanned outages at a higher frequency with the same potential challenges and also losses from providers exercising freedom to put assets to use in some other capacity. This unplanned provider loss is referred to as "churn" with the protocol specifying a lost data threshold before a repair operation is called. Repairs on the distributed network are expensive operations, requiring full copies of data at the source client for re-encode and re-encryption before re-uploading to new providers.

Configurable durability is again the most viable solution with the baseline code factor set to ensure desired durability AND then padding to cover expected churn rates. As the network grows, it will be possible to set baseline durability just ahead of measured repair rates so that data seldom gets called for expensive repairs. The cost increase with higher durability should nearly always be less than the cost to repair for most data storage profiles. Even so, over longer periods of time contract sets will deteriorate, and data will need to be repaired to new providers. The Relayer should track these potential repair segments and warn data owners in advance so that risks can be mitigated.

Block and Network Metrics
Key to a robust network is metrics on as many surfaces as possible. For standard block exploration, the project uses a web-based Navigator combined with the SiaCloud explorer module. The project is working on using prime_exporter to send metrics to Prometheus.

Provider and Network Development
*** add Vetted providers ***
Network development and growth is a project priority. No truly decentralized P2P cloud storage network suitable for enterprise storage exists as of this writing. Valuable insight can be drawn from file sharing technologies like Napster, Gnutella and Bittorrent but key assumptions may prove inadequate over time and at scale. Providers around the globe face a variety of conditions that could make the product more or less viable including standard ISP capabilities, operational expenses and geopolitical concerns. In mainland China, ISPs are legally held responsible for activity on the network, a major challenge to decentralized storage projects where privacy is built-in and network contents are not easily identifiable.

Provider Software
The current provider software has been forked from the ScPrime software base. Although much effort has been put into developing the software, it is still immature and requires ongoing tweaking and monitoring by providers for consistent performance. It is the goal of the SiaCloud developers to contribute to the ScPrime project via gitlab.

Future versions will make use of containers extensively for a more consistent installation experience and lowered support burden. The software should allow for control of multiple provider installations assuming one per machine. And provider software requires better accounting capability to determine profitability and for tax reporting.

Latency / Bandwidth Constraints

A wide spectrum of provider performance is forecast for the network at scale with an unusually large percentage of home-based installations and weak connectivity. Consumer-grade ISP plans often provide asymmetric connections, limiting upload performance which translates to customer downloads. Limits on the amount of data transferred (caps) are also common leading to higher costs or periodic unavailability for individual providers. Finally, this group of providers are often less diligent in node upkeep, preferring a "set it and forget it" process. The benefit is lower cost storage with these "hobby hosts" making up a significant portion of competitive costing advantages over centralized services.

Though latency to any given provider is directly tied to customer location, the distribution of latencies over contract sets should converge in allowing customers to delegate levels of responsibility based on desired performance characteristics and available budget. A core service provided by the project is network analytics, segmentation, incentivizing providers and creating actionable intelligence that can be fed into the Relayer configuration.

Appendix A - Functional Requirements

Relayer

- Overview
    - Browser or app-based interface
    - REST API for custom interface – customers and 3rd party applications
    - Storage tiers; static and dynamic
    - User authorization/authentication
    - Logging, reporting capability
    - Public performance/availability data replication

- Installation
    - Cross-platform compatibility
    - Container orchestration – customer one touch installation

- Storage Network Interaction
    - Real-time network mapping, identification and monitoring across sia-based networks
    - Ongoing QoS audit of all storage nodes
    - Network reporting over specified periods
    - Ad-hoc storage node groupings for classes
    - Contract creation/renewals with Sia-based network hosts
    - Automatically form new contracts upon completion as required
    - Upload/Download/Replace files on storage nodes
    - Automatic instance backup of customer metadata and upload to network
    - Ongoing analytics (csv creation) for class analysis and lifecycle determination
    - Ongoing file audits
    - Repair operations on failed shard audits
    - Node whitelist/blacklist
    - Requires no customer access to crypto coin transactions
- Database
    - Store netmap results for a defined period
    - Store individual customer metadata, user info, credentials
    - Store bucket metadata
    - Cache frequently accessed files (if necessary/possible, needs evaluation)
    - Incremental backups are uploaded to the most durable node tier for replication

- Buckets/Objects
    - Key functionality = Create, Delete, List. Every bucket has a unique name/id
    - Top level namespace addressing/domain/bucketname
    - Bucket properties include owner, date created, date edited
    - Allow customer to add, delete, copy, replace, download objects (files)

    - User permissions (upload, delete, replace, download, view)
    - File Versioning at bucket or object level
    - CORS configuration (XML file) for all buckets
    - “tags” on buckets and/or objects
    - Lifecycle rules at bucket or object level to move from one node tier level to another

- Exchange
    - Customer ID account creation (if allowing 1 to 1 relationships)
    - Interact with other Relayer exchange clients
    - Interact with centralized Exchange APIs
    - Atomic swaps of sia-based currencies
    - Automatic wallet top ups based on use patters/predicted storage use
    - Wallet creation for Relayer customer on all sia-based networks
    - Provide transaction reporting, logging and balance information
    - Requester pays

Client

- Provide interface for bucket and file management
- Allow bucket create, rename, delete
- Allow upload, delete, replace, download, view capabilities
- Allow permissions settings per file or per bucket
- Allow Lifecycle settings per file or per bucket
- Provide accounting features
- Provide basic network monitoring features
- Use SAML for a Single Sign-on capability
- 2 Factor authentication

Network

- Host container software installs
- Best practices guidelines
- Storage node incentive program (storage used, capacity offered)
- Node/network performance dashboards
- Minimum configuration/Recommended configuration

References

Amazon S3 REST API Introduction
https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html

Buss, Sebastian & Becker, Dennis & multiple others (2019) Digital Economy Compass
Retrieved from https://www.statista.com/study/52194/digital-economy-compass/

Champine, Luke. (2016) MIT License
https://gitlab.com/NebulousLabs/Sia/blob/master/LICENSE

Champine, Luke. & Vorick, David. (2014) Sia: Simple Decentralized Storage
https://sia.tech/sia.pdf

Mendoza, N.F. (2019) Data breaches now cost companies an average of $1.41 million.
https://www.techrepublic.com/article/data-breaches-now-cost-companies-an-average-of-1-41-million/

Multisignature
https://en.bitcoin.it/wiki/Multisignature

Nag, Syd. (2019) Market Share Analysis: IaaS and IUS, Worldwide, 2018.
https://www.gartner.com/document/3947169?ref=solrAll&refval=225669178&qid=ee7cc875a489136e9
3c3a8

Nakamoto, Satoshi. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System
https://bitcoin.org/bitcoin.pdf

Pfeffer, John. (2017) An (Institutional) Investor’s Take on Cryptoassets
https://s3.eu-west-2.amazonaws.com/john-pfeffer/An+Investor%27s+Take+on+Cryptoassets+v6.pdf

Smith, Eileen. (2019) Worldwide Public Cloud Services Spending Will More Than Double by 2023.
https://www.idc.com/getdoc.jsp?containerId=prUS45340719