MystenLabs

Walrus

An Efficient Decentralized Storage Network

October | 2024

Goals

- Decentralized BFT storage for large blobs
- Efficient writes & reads
- Verifiable availability
- Low storage overhead instead of full replication
- Storage committee can change over time based on stake distribution
- Incentives for continued storage

Non-Objectives

- Not a CDN
- Not a Database
- No re-implementation of consensus or execution → rely on smart contracts
- Walrus itself does not provide distributed encryption/decryption to support a full private storage ecosystem (but it is compatible).

Use Cases

- Storage of media for NFT or dapps
- Decentralized Websites → Walrus Sites
- Data sets, models weights, proofs of correct training for AI models
- Storage of archives (blockchain history, websites, papers)
- Support availability for L2s (on ETH and others)
- Support subscription models for media

Sui vs Walrus Storage

Sui today:

- Full copy on each Validator
- 105x storage cost
- Relatively small objects

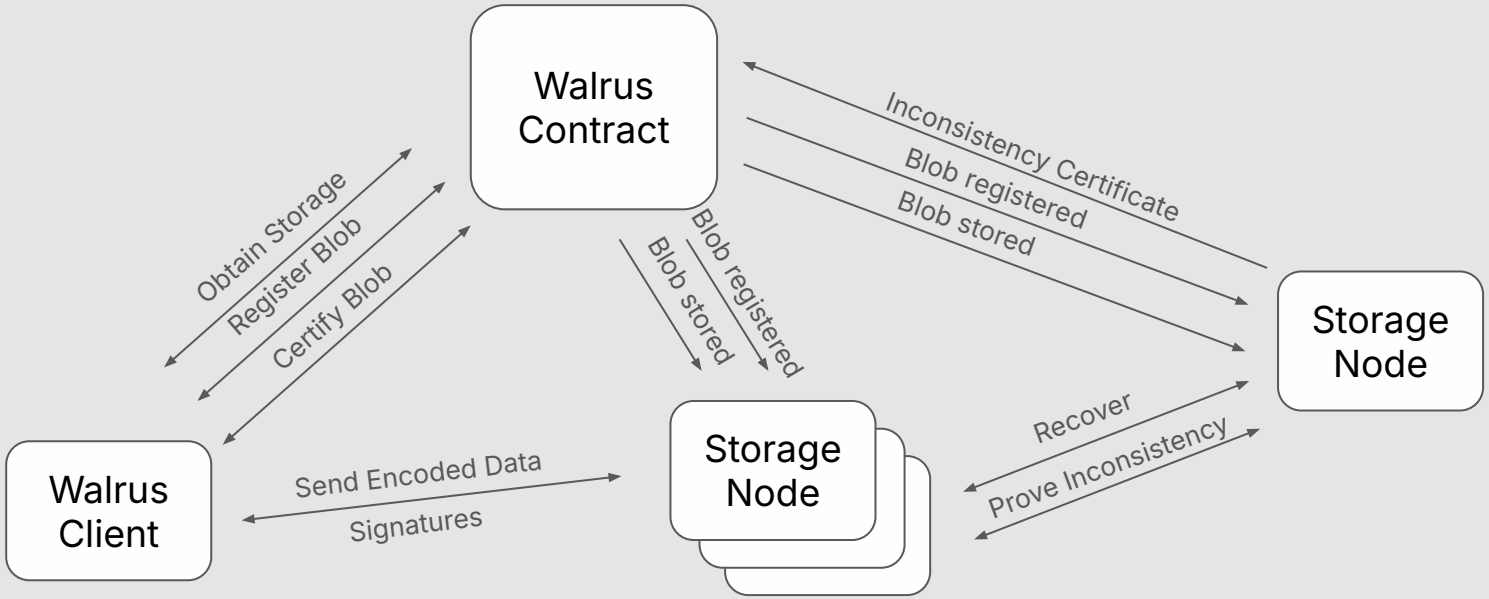
Walrus:

- Erasure coded and distributed
- 4.5x storage cost
- Large blobs (MBs to GBs)

Architecture

- Clients
  - Store & read blobs
  - Caches (optional): store full blobs and make them available over web2
  - Publishers (optional): store blobs on behalf of users
- Storage Nodes
  - Store one or multiple shards (based on stake)
  - Assume that ⅔ of shards are stored by correct nodes
- Smart contracts
  - Used for coordination

Storing a Blob



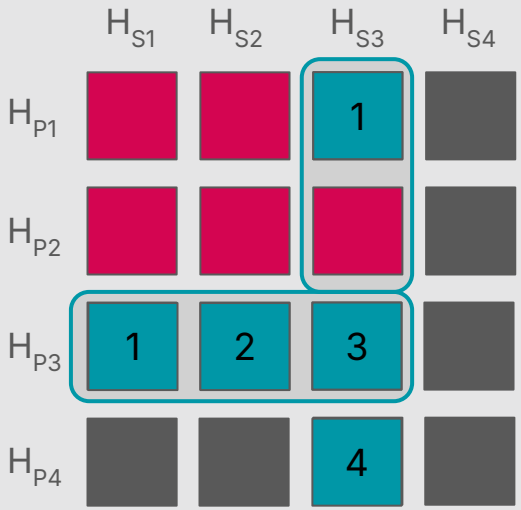
Reading a Blob

- Wait for f+1 responses
- Decode Blob
- Re-encode Blob & recompute Blob ID
  - If inconsistent, read “None”

Encoding - Goals

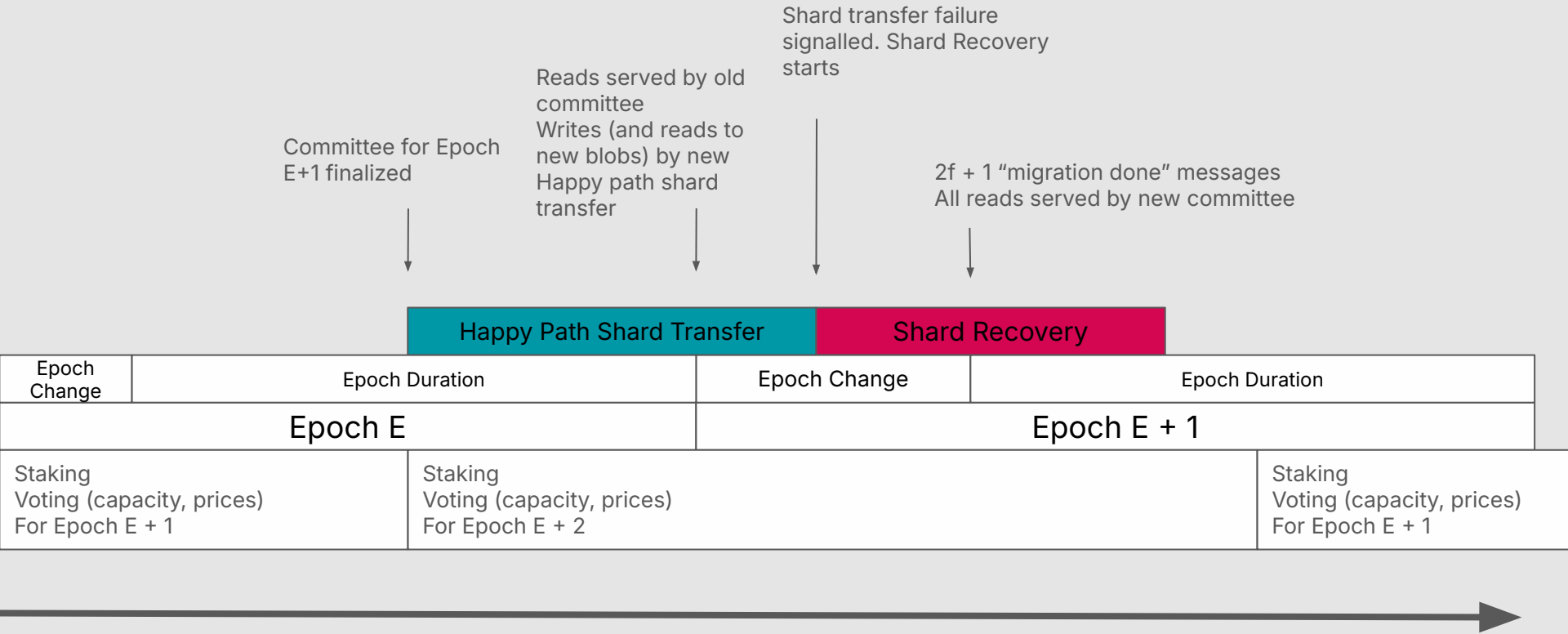
- Low overhead
- Efficient Encoding/Decoding
- Efficient Recovery
    - Nodes recovering their shards should not need to reconstruct everything
- Consistency checks
    - Everyone either reads the originally encoded blob or None

Encoding & Recovery (Red Stuff)



- Encoding uses linear erasure code (RaptorQ)
- Sliver root hashes (H_{P1}, H_{P2} etc.) stored on all nodes
  - Allows authentication of symbols (Merkle tree)
  - Blob ID is root of Merkle tree over sliver roots
- 2f + 1 symbols of primary sliver f + 1 of secondary sliver stored per shard
- Recovery with (f + 1) + (2f + 1) authenticated symbols + metadata
  - Inconsistency proof consists of recovery symbols
- Based on crash-tolerant Twin-Code framework (Rashmi et al., 2011)

Epoch Change



Storage Attestation

- Goal: Provide incentives for correct nodes
  - Nodes should receive payments if they provide evidence that they are storing the correct data
  - Avoid expensive proofs/protocols like filecoin
  - Make it hard to “outsource” challenges
- Practical Solution:
  - Storage nodes issue random challenges to each other in every epoch
  - Each challenge consists of multiple sequential subchallenges, which determine a specific symbol that needs to be provided (and authenticated)
  - Storage nodes measure response time and attest (on-chain) if they received correct responses quickly enough
  - If enough (threshold tbd, between f+1 and 2f+1) nodes attest correctness of a shard, the shard gets paid

Walrus Testnet Launched Today!

Try it yourself:
https://docs.walrus.site

Backup

Web3 storage L1s side-by-side

	Filecoin	Arweave	Sia	Storj	Walrus
Network	Independent L1	Independent L1	Independent L1	Ethereum	Sui + storage node network
Committee Composition	Proof-of-Spacetime	Proof-of-Access	Proof-of-Work	Proof-of-Stake	Proof-of-Stake
Proof-of-Storage	Proof-of-Replication	Proof-of-Access	Merkle Proofs	Merkle Proofs	Sui blob certification
Durability Mechanisms	Replication by Choice	Replication by Default	Erasure Coding by Default	Erasure Coding on some node	Erasure coding over all nodes
Privacy Mechanism	Encrypted by Choice	Encrypted by Choice	Encrypted by Default	Encrypted by Choice	Encrypted by Choice
Smart Contract Capability	Soon via FVM	Yes via SmartWeave	Yes via File Contracts	No	Yes via Sui
Pricing Model	Market-Based	Model-Based	Market-Based	Fixed, set by Storj Labs	Market-based
Decentrality	Fully	Fully	Fully	Partially	Fully
Sector Size	32 GB	NA	40 MB	64 MB	200 KiB to 1 GB
Latency	Performance is slower than a floppy disk as saving a 1MB file takes about 5 to 10 minutes.	Retrieval times may not be as fast as Filecoin's optimized solutions. Also, Arweave's block time is appx 2 minutes.	Fast speed for upload and download (200 ms per TTFB, 100 Gbps network wide). However, retrieving files may take longer compared to centralized solutions.	Fast retrieval for streaming and large files (24 Gbps); however, have low upload speed.	Fast writes and fast reads distributed across fastest of ⅔ to ⅓ of distributed nodes. Compatible with caches and CDNs.
Data Guarantee	Every 24 hours, repair automatically	Spot check on random samples	Check upon retrieval or contract expiration	Every 24 hours by satellite, and repair automatically	Spot checks between nodes, and tolerates ⅓ - ⅔ node failures. Every epoch up to ⅓ heal.

Source: Messari (Updated as of January 2023)