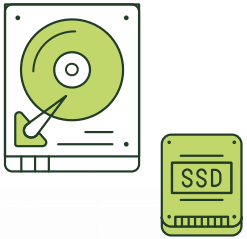
Proof of Space – Storage to secure the blockchain

Jonmichael Hands, VP Storage, Chia Network

Intro to cryptocurrency

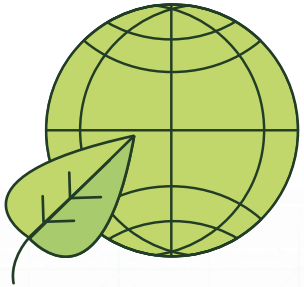
- **Inclusive and permissionless network:** Controlling your wealth is a human right
- **Censorship resistant:** difficult or impossible to block participants
- **Independent monetary policy:** trust math instead of people
- **Unstoppable applications:** A program developed for, and run on, a secure blockchain can never be changed or stopped
- **Global standards:** collaboration, fully open source, and available for free
- **Secure and Decentralized:** decentralization, consensus, and physical assets (capex) secure the network
- **Programmable Money:** Cryptocurrency should be easier to use than cash and harder to steal

Chia is a sustainable and secure blockchain



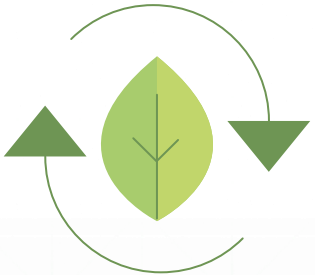
Leverages Storage

Proof of Space uses commodity storage which is vastly underutilized and power efficient



Decentralized and Inclusive

Low barrier to entry for farming has made Chia the most decentralized cryptocurrency in the world with hundreds of thousands of public nodes



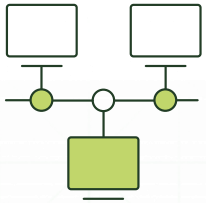
Incentivizes circular economy

Chia farming is optimized for used storage and will drive circular business models for storage devices, and reduce e-waste

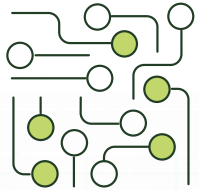
chia

Farming, not mining!

Chia farmers store cryptographic hashes in plot files on hard drive



Challenges come in from the network and a farmer checks plots for winning proofs



If farmer finds a winning proof, they broadcast that to the network to form a “block” on the blockchain

Farmer is rewarded Chia (XCH)

Currently 2 XCH per block

Farming is energy efficient, drives are mostly idle

chia

Proofs of Space and Time for the win!

Proofs of Space

- Farmers need to prove to the network that they are storing the data
- These proofs need to be easily and quickly verifiable
- The network needs to be resistant to attacks

Proofs of Time

- Verifiable Delay Function (VDF) that cannot be parallelized
- Easily verifiable that a real amount of time was spent with deterministic output
- Performs squaring computations within class groups of binary quadratic forms

chia

Weight of blockchain = space * VDF iterations

Magic math

Beyond Hellman's Time-Memory Trade-Offs with Applications to Proofs of Space

Hamza Abusalah[1], Joël Alwen[1], Bram Cohen[2], Danylo Khilko[3], Krzysztof Pietrzak[1], and Leonid Reyzin[4]

[1] Institute of Science and Technology Austria, habusalah|jalwen|pietrzak@ist.ac.at
[2] Chia Network, bram@chia.network
[3] ENS Paris, dkhilko@ukr.net
[4] Boston University, reyzin@cs.bu.edu

Abstract. Proofs of space (PoS) were suggested as more ecological and economical alternative to proofs of work, which are currently used in blockchain designs like Bitcoin. The existing PoS are based on rather sophisticated graph pebbling lower bounds. Much simpler and in several aspects more efficient schemes based on inverting random functions have been suggested, but they don't give meaningful security guarantees due to existing time-memory trade-offs.

In particular, Hellman showed that any *permutation* over a domain of size N can be inverted in time T by an algorithm that is given S bits of auxiliary information whenever $S \cdot T \approx N$ (e.g. $S = T \approx N^{1/2}$). For *functions* Hellman gives a weaker attack with $S^2 \cdot T \approx N^2$ (e.g., $S = T \approx N^{2/3}$). To prove lower bounds, one considers an adversary who has access to an oracle $f : [N] \to [N]$ and can make T oracle queries. The best known lower bound is $S \cdot T \in \Omega(N)$ and holds for random functions and permutations.

We construct functions that provably require more time and/or space to invert. Specifically, for any constant k we construct a function $[N] \to [N]$ that cannot be inverted unless $S^k \cdot T \in \Omega(N^k)$ (in particular, $S = T \approx N^{k/(k+1)}$). Our construction does not contradict Hellman's time-memory trade-off, because it cannot be efficiently evaluated in forward direction. However, its entire function table can be computed in time quasilinear in N, which is sufficient for the PoS application.

Our simplest construction is built from a random function oracle $g : [N] \times [N] \to [N]$ and a random permutation oracle $f : [N] \to [N]$ and is defined as $h(x) = g(x, x')$ where $f(x) = \pi(f(x'))$ with π being any involution without a fixed point, e.g. flipping all the bits. For this function we prove that any adversary who gets S bits of auxiliary information, makes at most T oracle queries, and inverts h on an ϵ fraction of outputs must satisfy $S^2 \cdot T \in \Omega(\epsilon^2 N^2)$.

1 Introduction

A proof of work (PoW), introduced by Dwork and Naor [DN93], is a proof system in which a prover $\mathcal{P}$ convinces a verifier $\mathcal{V}$ that he spent some computation with

Binary quadratic forms

Lipa Long
Chia Network

1 Introduction

Chia's underlying verifiable delay function (VDF) performs squaring computations within class groups of binary quadratic forms. This paper provides an introduction to binary quadratic forms and to their properties relevant to application in the Chia VDF, and an efficient algorithm for their composition is introduced.

A VDF is a sequential operation that takes a prescribed amount of time to compute (and which cannot be accelerated by parallelism) and which produces an accompanying proof by which the result may be quickly verified. The best known method for achieving a non-parallelizable sequential operation is repeated squaring in a group of unknown order. The unknown order requirement is due to the divisibility of the order of a finite group by the order of any element in the group; if the group order is known then the repeated squaring operation could be reduced modulo the order of the group, shortcutting the computation.

When using an RSA group in a VDF, the group is a multiplicative group $\mathbb{Z}/N$, where $N = pq$ such that p and q are primes, $p, q \neq 2$, and p and q are both unknown. $\mathbb{Z}/N$ is considered a group of unknown order because the difficulty of computing the group order is on par with the difficulty of computing the factors of N. To guarantee that the factors — and therefore the group order — are indeed unknown, a trusted setup may be used which ensures that the factors used to generate the group are not revealed.[1] However, a trusted setup requires that the party generating the trusted parameters destroys the keys once N is created. If such a party were malicious and otherwise fails the destroy the keys, the VDF's sequentiality requirement could be broken.

In contrast, using class groups of binary quadratic forms omits the trusted setup because the order of the class group of a negative prime discriminant d, where $|d| \equiv 3 \bmod 4$, is believed to be difficult to compute when $|d|$ is sufficiently large, making the order of the class group effectively unknown. Therefore, a suitable discriminant — and its associated class group — can be chosen without the need for a trusted setup, which is a major advantage for using class groups in applications requiring groups of unknown order.

The study of class groups is typically presented either in the context of binary quadratic forms or in the context of fractional ideals of algebraic number fields.[2] The ideal class group of an algebraic number field K is the quotient group J_K/P_K, where J_K is the group of fractional ideals of the ring of integers, O_K, of K, and P_K is its subgroup of principal fractional ideals. While many feel that this provides a more conceptually intuitive introduction to class groups, the representation of elements

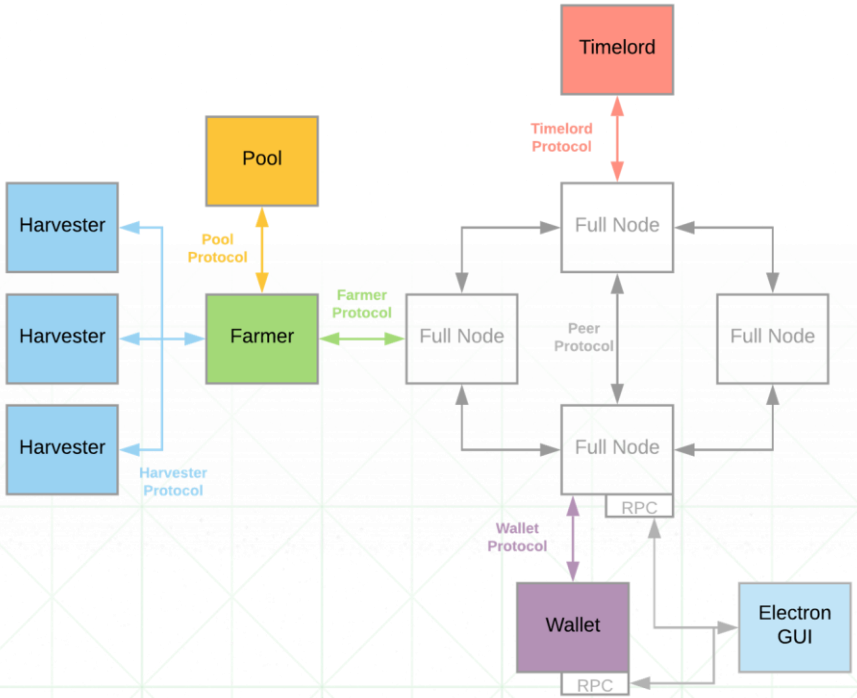
[1] Alternatively, RSA keys may be generated by multi-party protocols or by using an RSA modulus for which the prime factors are believed to be lost.
[2] See Appendix 7.3 for definitions of the following terms appearing in this paragraph: algebraic number field, quotient group, fractional ideal, ring of integers, and principal fractional ideal.

chia

Chia had "invent new math here" twice on roadmap

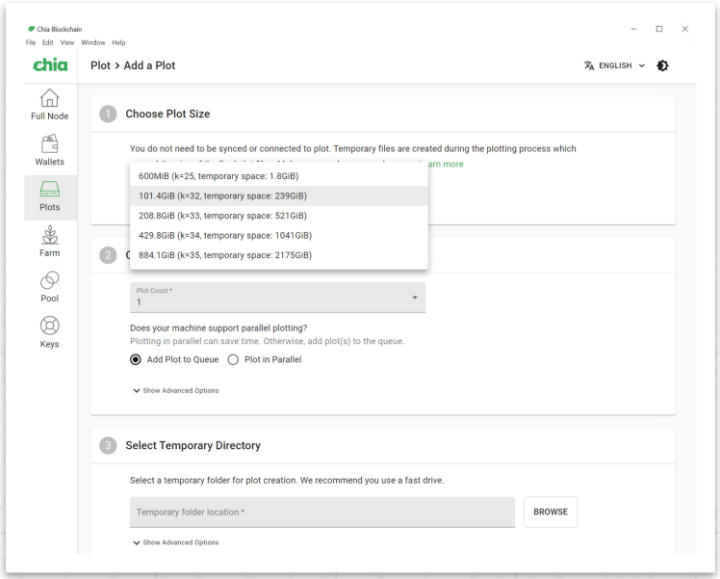
Chia Network Protocol

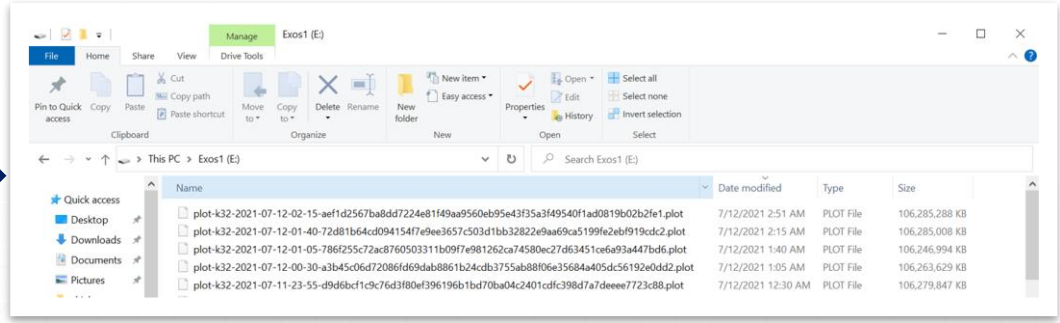
- **Plotting** - creating the proofs of space into a “plot file” requires upfront one-time use of compute resources, so that proofs can be quickly and easily verified later
- **Harvester** – lightweight machines controlled by the farmer that continuously check the plot files for proofs of space
- **Farming** - farmers compete to create blocks with proofs of space that meets the challenge requirements
- **Full node** - maintains an entire copy of the blockchain, propagates blocks, transactions, and proofs between peers

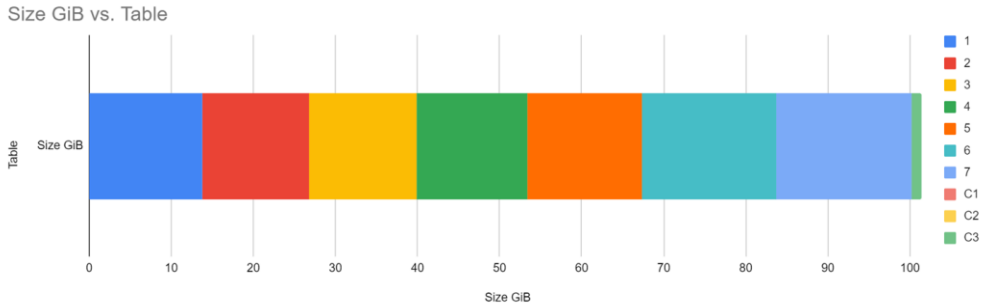


Files are created and then transferred to farming drive (HDD)

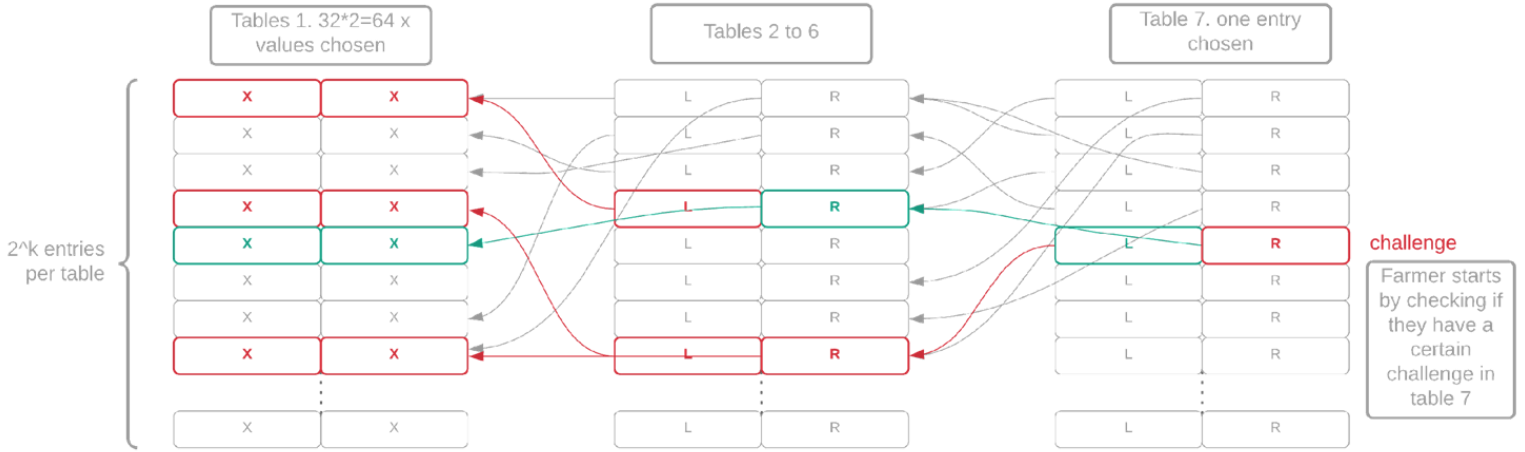
Use compute, memory and temporary storage to create plot files







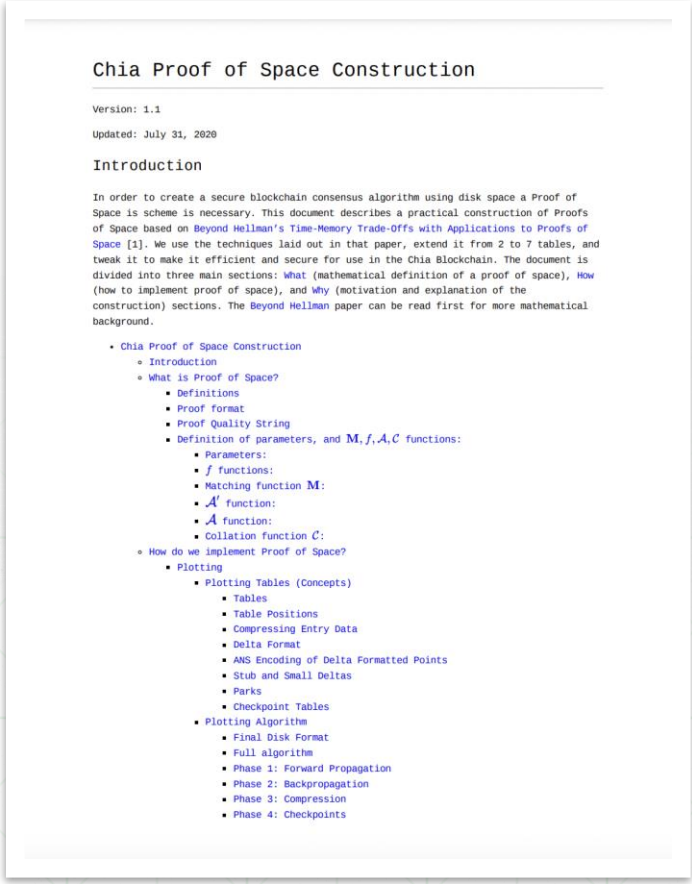
The Fix to Proofs of Space



- Proofs of space designed with 7 tables to resist Hellman attacks
- Need to be able to quickly retrieve proof (low disk io, cpu)
- Plot size determined by k parameter

Plotting is Write Intensive Workload

- Phases of plotting require a lot of sort on disk & algorithmic compression
- Each K=32 plot takes 1.43TB of disk writes
- Can be reduced by 72% with 110GB of DRAM
- Can be eliminated with 416GB DRAM with current high speed in-memory plotter



Chia Proof of Space Construction

Version: 1.1

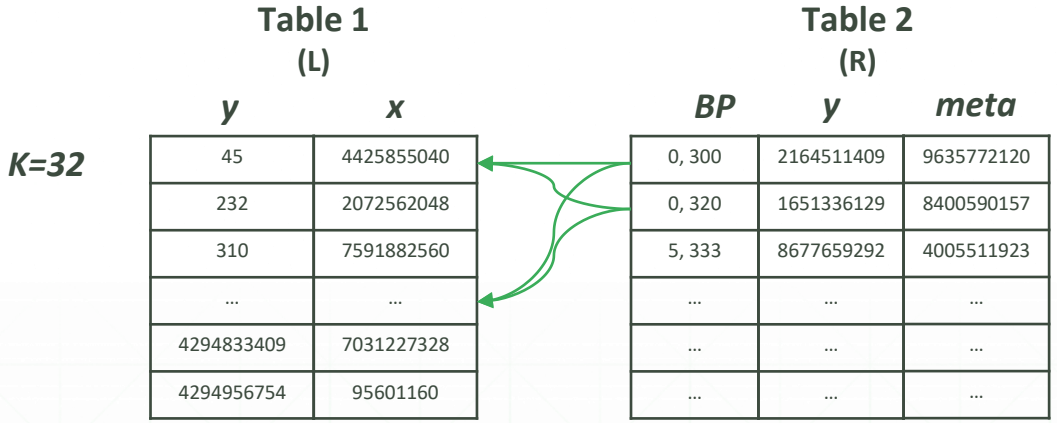
Updated: July 31, 2020

Introduction

In order to create a secure blockchain consensus algorithm using disk space a Proof of Space is scheme is necessary. This document describes a practical construction of Proofs of Space based on Beyond Hellman's Time-Memory Trade-Offs with Applications to Proofs of Space [1]. We use the techniques laid out in that paper, extend it from 2 to 7 tables, and tweak it to make it efficient and secure for use in the Chia Blockchain. The document is divided into three main sections: What (mathematical definition of a proof of space), How (how to implement proof of space), and Why (motivation and explanation of the construction) sections. The Beyond Hellman paper can be read first for more mathematical background.

Chia Plotting Phase 1

- The following steps are then performed for all tables in the plot:
  - Sort ***L*** entries on ***y***'s value.
  - Generate ***R*** table back pointers (***BP****s*) by matching pairs of ***y*** values on ***L***.
  - For each ***BP*** in ***R***, take the ***y*** from ***L*** pointed by first pointer and the ***metadata*** values pointed by both pointers of a ***BP*** entry.
    - For **Table 1**, the metadata is ***x***.
  - Hash each **y** and **metadata** pair using *BLAKE3* and extract a new ***y*** value and ***metadata*** value for table ***R*** from the hashed output.



Phase1: forward propagation, sorting, matching

Phase2: prune entries

phase3: compression

phase4: checkpointing

Current Chia plotters

- Chiapos (proof of space)
  - reference plotter
  - phase1 multi-thread support
  - any size k value
  - very slow
- MadMAx chia-plotter
  - pipelined / multi-thread
  - random io to disk
  - two temp directories
  - can use 110G of DRAM or SCM for temp2 to reduce disk writes
- Bladebit
  - fastest plotter
  - completely in-memory, requires 416GiB of DRAM
  - performance depends on cores, memory bandwidth
- Bladebit Disk
  - low min memory requirements
  - Cross-platform and OS compatibility
  - Sequential writes, reduce WAF
  - DRAM cache support
  - Takes full advantage of increased disk bandwidth from PCIe 4.0
  - Pipelined performance to max out CPU

chia

Plotting – All About Performance

Plotting hardware required

Server **Workstation** **Desktop**

High endurance NVMe

Create plots as fast as possible.
Requires use of hardware for duration of plotting (a few months)

chia

Or…

Use plotting as a service

Plotting Cloud

aws

akash

EQUINIX

Download plots from providers.
Requires high speed internet connection

How does farming work?

- Farmer receives **challenge** from VDF
- Farmer sends signage point to harvester
- Harvester applies **plot filter** to reduce the io required on disk (1/512)

```
plot filter bits = sha256(plot_id
+ challenge_hash + sp_hash)
```

- For plots that pass the filter, harvester performs **proof quality check**
- If quality meets required iterations (from difficulty) then proof of space is good
- Fetch entire **proof of space**

- Farmer has to prove to the pool how much capacity they have
- A **partial** is a full proof of space with additional metadata
- Goes through exact same process, but with a custom difficulty
- Pool adjust difficulty to target a certain amount of partials per day
- Pool estimates farming capacity with difficulty and good partials submitted

chia

Farming hardware required

JBOD **NAS** **Desktop** **DIY**

High capacity HDD

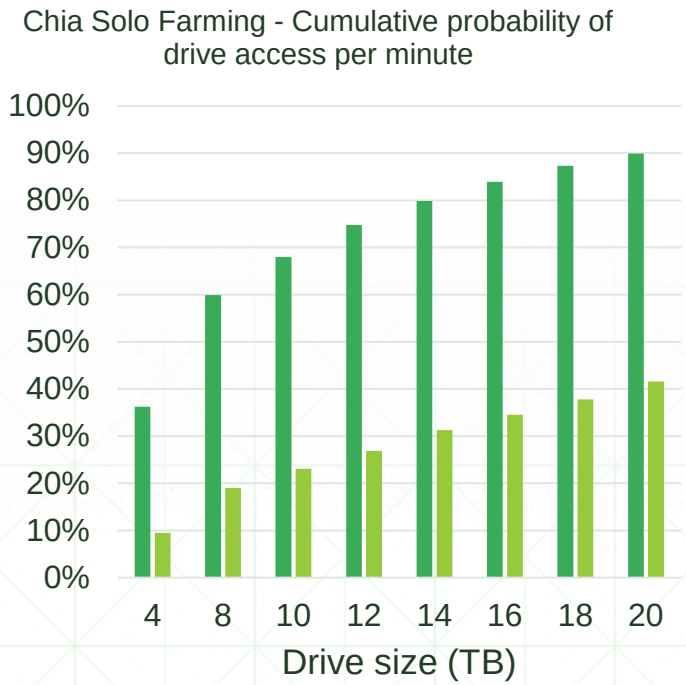
JBODs (Just a bunch of disk) houses many hard drives

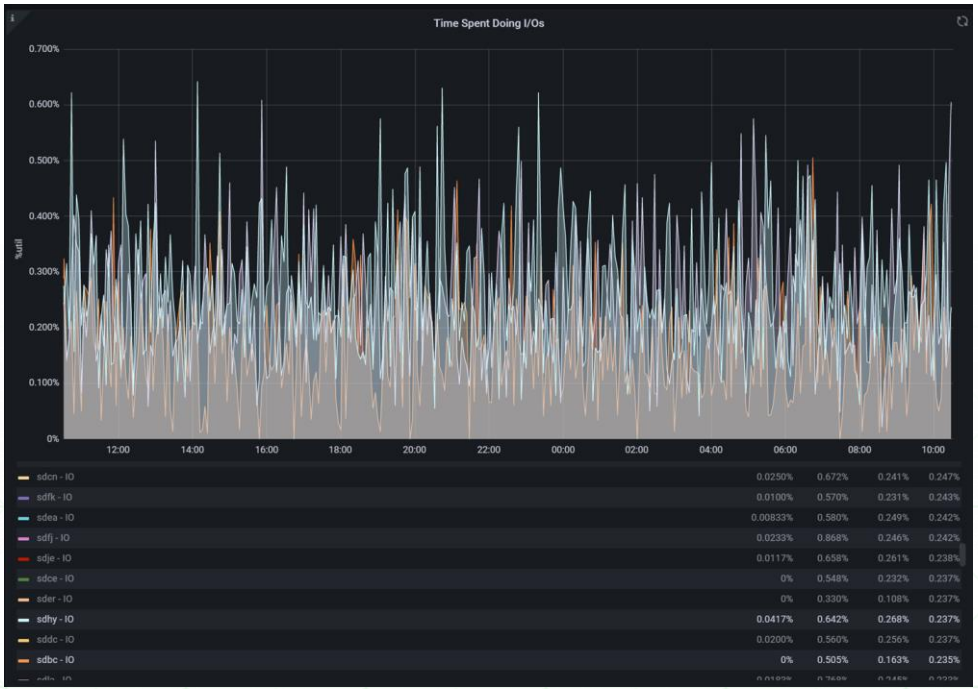
Store the most data in the smallest amount of space at lowest power

Use high-capacity HDDs for best power efficiency, or cheap used drives if rack space and power are more available

chia

Farming – accessed frequently but highly idle





Drives are idle 99.7% of the time

Farming Goals – Have the lowest TCO!

- Storage **density** - large number of drives per RU, and using large capacity drives
- Minimize **capex** – buying used, recertified, renewed drives, or leveraging underutilized storage
- Minimize **power consumption** of supporting infrastructure (farmer, harvester)
- Drive **power efficiency** - using efficient high-capacity drives, measured in W/TB
- Keep drives **cool** enough to not impact reliability
- Nice to have – minimize noise, hotswap, enclosure management- etc.

chia

Chia Farming TCO - Power Consumption

Power of farmer and harvesters	JBODs, NAS, or enclosures	Drives	Networking	Rack
CPU, DRAM, boot drive, HBA	Includes expanders, fans, PSUs	Power active = random read at 1% Power Idle = Idle A at 99%	Routers and switches	Cooling Calculate PUE

In an optimal farming setup the majority of power from drives!

Chia Farming TCO Rack scale: modern vs used

90 bay with 18TBs

45 bay with used