Centralization in the Decentralized Web: Challenges and Opportunities in IPFS Data Management

Ruizhe Shi
George Mason University
Fairfax, VA, USA
rshi3@gmu.edu

Ruizhi Cheng
George Mason University
Fairfax, VA, USA
rcheng4@gmu.edu

Yuqi Fu
University of Virginia
Charlottesville, VA, USA
jwx3px@virginia.edu

Bo Han
George Mason University
Fairfax, VA, USA
bohan@gmu.edu

Yue Cheng
University of Virginia
Charlottesville, VA, USA
mrz7dp@virginia.edu

Songqing Chen
George Mason University
Fairfax, VA, USA
sqchen@gmu.edu

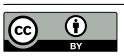
Abstract

The InterPlanetary File System (IPFS) is a pioneering effort for Web 3.0, well-known for its decentralized infrastructure. However, some recent studies have shown that IPFS exhibits a high degree of centralization and has integrated centralized components for improved performance. While this change contradicts the core decentralized ethos of IPFS and introduces risks of hurting the data replication level and thus availability, it also opens some opportunities for better data management and cost savings through deduplication.

To explore these challenges and opportunities, we start by collecting an extensive dataset of IPFS internal traffic spanning the last three years with 20+ billion messages. By analyzing this long-term trace, we obtain a more complete and accurate view of how the status of centralization evolves over an extended period. In particular, our study reveals that (1) IPFS shows a low replication level, with only 2.71% of data files replicated more than 5 times. While increasing replication enhances lookup performance and data availability, it adversely affects downloading throughput due to the overhead involved in managing peer connections, (2) there is a clear growing trend in centralization within IPFS in the last 3 years, with just 5% of peers now hosting over 80% of the content, significantly decreasing from 21.38% 3 years ago, which is largely driven by the increase of cloud nodes, (3) the default deduplication strategy of IPFS using Fixed-Size Chunking (FSC) is largely inefficient, especially with the default 256KB chunk size, showing near-zero duplication being detected. Although Content-Defined Chunking (CDC) with smaller chunks could save ~1.8 petabytes (PB) storage space, it could impact user performance negatively. We thus design and evaluate a new metadata format that optimizes deduplication without compromising performance.

CCS Concepts

• **Networks** → **Network measurement**; • **Information systems** → **Deduplication**.



WWW '25, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1274-6/25/04
https://doi.org/10.1145/3696410.3714627

Keywords

IPFS, Deduplication, Network Measurement

ACM Reference Format:
Ruizhe Shi, Ruizhi Cheng, Yuqi Fu, Bo Han, Yue Cheng, and Songqing Chen. 2025. Centralization in the Decentralized Web: Challenges and Opportunities in IPFS Data Management . In *Proceedings of the ACM Web Conference 2025 (WWW '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3696410.3714627

1 Introduction

From the underlying infrastructure support to the way how users interact with the web, there has been a clear trend toward centralization in the current web along with the wide adoption of cloud computing over the past two decades [14]. Although this undeniably provides convenient services for managing the web data and serving the users, it also raises growing concerns about data control, censorship, content moderation, and issues related to monetization and fairness.

In response to those concerns, recent years have witnessed a growing surge in the technology campaign referred to as "Web 3.0" or "Decentralized Web", such that no individual entity can control or censor the entire operation of the system. One of the leading efforts in this movement is the InterPlanetary File System (IPFS) [7]. IPFS uses a Peer-to-Peer (P2P) architecture, where identical data copies are distributed and shared among multiple peers in the system. Upon a request, a client can be served by any of available peers having a copy of the data. The adoption of IPFS is widespread, with over 250K active daily nodes and spanning 152 countries [33]. It also serves as the storage layer for various Decentralized Applications, such as Non-Fungible Token (NFT) [37].

Although IPFS is meant to be a decentralized system, some recent studies [6, 27] have looked into snapshots of the client accesses in IPFS and shown that IPFS exhibits a surprising degree of centralization when serving client requests, where a small group of cloud-based nodes serves the majority of the client accesses. The centralization in IPFS has indeed improved performance (as users are often served by faster cloud nodes) and enabled easier client access by integrating gateways to serve users not running the IPFS protocol directly [38]. However, such centralization has significant downsides, as previously noted. This raises questions about whether these issues stem from the limited number of copies of

each data file distributed within the system. Furthermore, it remains unclear whether those observations reflect temporary effects caused by dynamic system factors (*e.g.,* peer churn) and thus fail to fully represent the system's complete state.

On the other hand, regardless of the underlying reasons, from a practical standpoint, such centralization, if it exists, could present an opportunity for improved web data management and cost savings through deduplication [40], which is important for those cloud nodes. By default, IPFS employs deduplication locally to ensure fast publication performance and uses fixed-size chunking (FSC) [23] for data deduplication. However, it is still unclear how efficient this default deduplication strategy is in IPFS.

To explore answers to the above questions, we began our study by massively collecting the IPFS internal content exchange traces in the past 3 years, spanning from March 2021 to August 2024, with over 20 billion messages. This 3-year trace of internal IPFS traffic can help reveal the evolving of centralization changes, given the instant snapshots used in previous analysis [27]. By performing a systematic measurement and analysis on these traces, we aim to answer the following key questions: **RQ1:** How are the identical data copies distributed within the current IPFS network? Specifically, what is the level of data replication, and how does it affect user access patterns and performance? **RQ2:** How has the decentralization of IPFS evolved over the past three years, and what is the trend of the changes? Our measurement and analysis lead to the following findings.

- **[Replication]** IPFS exhibits a low replication level with 29.20% of Content Identifiers (CIDs) replicated more than once, and a mere 2.71% more than 5 times. Increasing replication in IPFS generally enhances lookup performance due to a higher possibility of being discovered. However, the increased replication negatively impacts downloading throughput after a certain level as a result of the extra overhead involved in consistently choosing (and switching to) better peers for downloading.
- **[Centralization]** There is a significantly growing trend of centralization in IPFS over the past 3 years. At the beginning of our measurement period, 21.38% of the peers hosted about 80% of the content. However, by the end of our measurement period, about 5% of the peers are responsible for hosting 80.55% of the content. This increasing centralization can be attributed to the growing adoption of cloud nodes, whose share has increased from 50.02% to 87.33% over the same period.

The observed centralization trends in IPFS, while seemingly at odds with its decentralized goal, present a unique opportunity to re-examine and improve storage efficiency, particularly the effectiveness of the default deduplication method, FSC. Centralization, in this context, can serve as a catalyst for optimizing deduplication in decentralized systems like IPFS. By leveraging the concentration of data and requests among a subset of nodes, we can design more efficient deduplication strategies that balance user performance with practical storage optimization. This leads us to explore **RQ3:** How efficient is IPFS's default deduplication mechanism (FSC), and how can centralization trends inform its improvement? Our analysis reveals the following key insights:

- **[Deduplication]** Currently, IPFS achieves negligible deduplication efficiency by using the default FSC method with the default 256KB chunk size, where the storage savings could be up to 1.8 PB using Content-Defined Chunking method with a smaller chunk size such as 4KB. We further show that a smaller chunk size could negatively impact user performance in terms of the deduplication speed and downloading throughput. To balance the user performance and deduplication efficiency, we propose a new metadata format by exploiting storage locality and evaluate its performance in a private IPFS network. The results show that, compared to the current design in IPFS, this new metadata format can improve the average file publication throughput by 2.24× and significantly reduce the IO overhead (by up to ∼ 50%) particularly for small chunk sizes.

The remainder of the paper is as follows. A brief background of IPFS is presented in Section 2. We present our methodology of data collection in Section 3, measurement analysis regarding replication in Section 4, centralization in Section 5, and deduplication in Section 6. We discuss the related work in Section 7 and make concluding remarks in Section 8.

2 Background

IPFS Overview. IPFS is a decentralized Web 3.0 system that builds on top of a P2P network and facilitates a set of protocols. It uses content-based addressing, where each file is split into multiple chunks and each chunk is represented by an immutable, hashed, and self-certifying CID. The physical addresses of CIDs are stored in the Distributed Hash Table (DHT) [22]. This enables a peer, identified by a unique PeerID, to query the DHT to locate and retrieve the content without relying on a centralized server. For users not participating in the IPFS network, Gateway services are provided for them to access IPFS content via HTTP.

DHT. DHT is a key component of the IPFS routing system, responsible for storing and retrieving content. It indexes two types of records: provider records, which map CIDs to the nodes that advertise and provide the content, and peer records, which map PeerIDs to the physical addresses of nodes (*e.g.,* IP addresses). The DHT allows the user to locate which node is serving the target content and advertise its own content without the need for a centralized server.

Content Publication and Deduplication. IPFS uses blocks for its operations, as each file is split into multiple chunks when added to the IPFS network. These chunks form a Merkle Directed Acyclic Graphs (DAG), where each node corresponds to a block. Specifically, raw data is stored in leaf blocks, while parent blocks contain metadata indicating how the original file is segmented and can be reconstructed. Similar to centralized deduplication systems, IPFS achieves deduplication at the chunking stage by ensuring that no identical chunks are stored. The default deduplication algorithm in IPFS is Fixed Size Chunking (FSC), set at a default size of 256 KB. During the process of chunking, the CID of each chunk is computed based on its content. The CID can be configured into two interchangeable formats: *version0* and *version1* [3]. Additionally, users can enhance deduplication efficiency by standardizing chunking algorithms or opting for variable-size methods like Rabin. Specifically, IPFS supports configurable algorithms such as Rabin Fingerprint [8] and Buzhash [9]. Different from centralized storage systems, which typically perform deduplication processes on a

central server and distribute the blocks afterwards, IPFS performs deduplication locally.

Content Retrieval and Bitswap. Content retrieval in IPFS involves two main stages: lookup and downloading. In the lookup stage, IPFS initially attempts to find which peers have the requested content. Before querying the DHT, IPFS uses the Bitswap [2], the file exchange protocol in IPFS, to send a *W*ANT-HAVE message to nearby peers, asking if they have the desired content. If no peers respond within a 1-second threshold, IPFS resorts to querying the DHT to locate the content. Once the content is located, the process enters the downloading stage, where Bitswap requests the actual data by sending a *W*ANT-BLOCK message to peers holding the required blocks. Bitswap then transfers the actual data blocks, completing the retrieval process.

3 Methodology

3.1 Data Collection

Bitswap Logs. We have gathered Bitswap traces from March 1st, 2021 to August 15th, 2024, utilizing a customized IPFS node implementation that supports unlimited peer connections, akin to the approach described in [5]. Our crawler captures all 1-hop Bitswap broadcast traffic and records it on disk. The data logged includes the timestamp, the sender's PeerID and network address, the type of request, the receiver's PeerID, and the targeted CID. Through this method, we have documented an average of approximately 21 million requests daily, amounting to a total of 1.8 billion unique CIDs.

DHT Logs. We also set up a modified version of a DHT server to collect the IPFS DHT *F*ind_Node traffic. We set up 2 virtual peer IDs and log all the incoming DHT requests to disks. Similar to the Bitswap messages, the content of the DHT *F*ind_Node requests includes the sender's PeerID and network address, the request type, and the target PeerID or CID depending on the type of request. This collection effort spanned from March 1st, 2021 to August 15th, 2024. From this data, we observed an average of approximately 1 million requests per day, encompassing a total of 120 million unique CIDs.

3.2 Ethical Consideration

This work is conducted under an IRB approval from our institution. Both the Bitswap and DHT traces contain IP addresses, yet our experiment does not attempt to map those IPs to any individuals or entities, as such analysis is not within the scope of our study. Furthermore, the IPs are anonymized and only mapped to country level for usage of geolocation mapping.

We also note that those Bitswap and DHT traces contain personal browsing and publication history. However, we do not attempt to track any personal usage and collect any personal information. Although we download the CIDs for the purpose of deduplication analysis, we do not attempt to perform any content analysis on them and only seek to understand the deduplication level of their host node. All the downloaded CIDs are deleted immediately once the deduplication analysis is done. Furthermore, we recognize that CID downloading could potentially introduce additional load on the IPFS network. However, we argue that this impact is minimal. The downloading process is spread over a three-week period, generating an average of 55 GB daily traffic, which is negligible compared to the estimated over 100 TB daily traffic on the IPFS network [33].

4 Data Replication

In this section, we start by examining the degree of replication within IPFS. Then we look at how different degrees of file replication across the network influence the efficiency and reliability of retrieving the content.

4.1 Degree of Replication

Methodology. Using the DHT logs and Bitswap logs, we have constructed a CID-provider mapping that retains only the CIDs representing a complete file or directory. This filtering process is facilitated by identifying the request pattern for CIDs in the logs, where a CID that corresponds to a complete file or directory is typically requested first. This allows us to bypass the need to download the CID to verify its contents.

Based on the CID versioning of IPFS, we further categorize the CIDs into two classes: *v*ersion0 and *v*ersion1 It is important to note that while different CID versions of a file can be inter-converted, they represent distinct objects as IPFS solely rely on CID to locate a file.

To verify replication accuracy, we send a *W*ANT-HAVE message to the peer who is assumed to be the provider. If the recipient fails to respond, we repeat this query every 4 hours for one week. If there is still no response after this period, we consider that the peer no longer provides this CID and is therefore not a valid provider. This method ensures that we are not only mapping CIDs to providers correctly but also verifying the presence and availability of the content they claim to host.

Result and Analysis. Figure 1 displays the cumulative distribution of CIDs categorized by *version0* and *version1*. In this study, we detect a total of 214 million CIDs, with 147 million belonging to *version0* and 67 million to *version1*. The red curve in the figure represents the cumulative distribution when all CIDs are converted to a single version, illustrating the overall trend.

The data reveals a low level of replication across the network. Specifically, 29.20% of CIDs are replicated more than once, and 2.71% of CIDs experience replication more than five times. This indicates that while the network is capable of hosting multiple copies of the same content, the actual practice of replication is not extensively utilized.

Interestingly, the analysis uncovers a significant amount of "replication wastage". We find that 18.24 million files have both *version0* and *version1* CIDs, which artificially inflates the count of unique content due to version differences.

> **Takeaway 1:** While IPFS is designed as a file-sharing system, its file replication level typically remains low with only about 30% of its files replicated more than once. Moreover, the issue is exacerbated by the fact that different CID versions can be generated for the same file, which inflates the perceived uniqueness of files within the system.

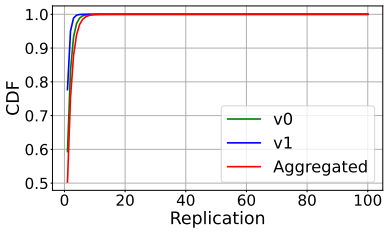


Figure 1: CID replication level in *version 0, version 1* and after aggregation.

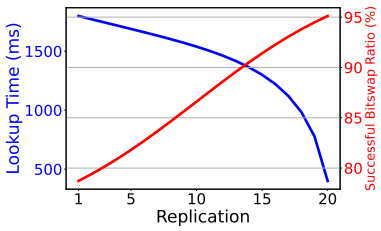


Figure 2: The performance of lookup in different replication levels.

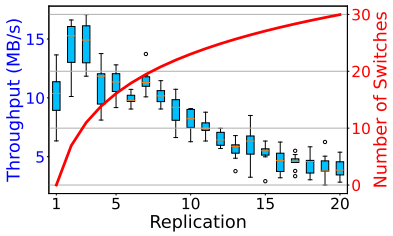


Figure 3: The performance of throughput in different replication levels.

4.2 Replication Impact on Performance

Methodology. To study how the replication level impacts the file retrieval performance in IPFS, we first set up an IPFS client (a `t2.medium` EC2 instance with 2 vCPUs and 4GB of memory) in central Europe. We let this IPFS client fetch 1,000 files, 10 MB each, in each replication level from 1 to 20. We record the timestamp to discover the provider (lookup time) and the timestamp to fetch the blocks (downloading time). We then calculate the throughput by dividing the downloading time instead of the entire retrieval time. It is also important to note that the provider can be discovered by Bitswap before launching the DHT as illustrated in Section 2. As such, we further examine their Bitswap success ratio as a function of replication level.

Results and Analysis. Figure 2 demonstrates the average lookup time (on the left *y-axis*) and the Bitswap success ratio (on the right *y-axis*) at each replication level. As can be seen in this figure, the replication level can drastically improve the lookup time performance. Specifically, the average lookup time decreases from 1817 ms at a replication level of 1 to 397 ms at a replication level of 20, while the Bitswap success ratio concurrently rises from 73.21% to 95.09%. This improvement is intuitive; as content is held by more peers, it becomes more readily detectable by Bitswap and DHT mechanisms, thereby reducing the time required for lookups.

Figure 3 shows the downloading throughput (on the left *y-axis*) at different replication levels. Different from the consistently improving trend observed in lookup time as the replication level increases, the downloading throughput initially improves but then decreases. Specifically, the average throughput peaks at 14.54 MB/s at the replication level of 2. Beyond this point, the throughput consistently decreases as the replication level increases.

To understand the underlying dynamics of this pattern, we further look into its downloading phase driven by Bitswap. We notice that there is a clear increasing trend of the request-peer switch as depicted in the blue curve in Figure 3. This "switch" refers to the action of changing the downloading peer to another peer during the file retrieval process. The switching is to get a better response time and balance the specific traffic from certain providers. However, while switching can optimize the download process under certain conditions, excessive switching can be detrimental. It involves consistently closing and reopening connections, as well as re-requesting the target block from a new source. This added complexity and overhead can significantly delay the entire downloading process, as each switch consumes time and potentially disrupts the steady flow of data transfer.

Takeaway 2: While increasing the replication level in IPFS improves lookup performance due to a higher Bitswap success ratio, it negatively impact the downloading throughput after a certain level because of the extra overhead involved in consistently choosing (and switching to) better peers to download from.

5 Centralization

Given the low replication level in IPFS found in the last section, we wonder how the content distribution changed within our 3-year dataset - potentially centralization as reported by [6, 27]. To this end, in this section, we aim to analyze the evolution of IPFS's centralization over a 3-year period by employing statistical measures such as the Gini Coefficient and Shannon entropy to quantify its centralization level.

5.1 Methodology

Entropy. Entropy quantifies the unpredictability or randomness in the output of an information source. For IPFS, the distribution and frequency of file access define the probability distribution. Files that are accessed more frequently demonstrate lower entropy, indicating reduced uncertainty and potential centralization in file access patterns. To quantify the centralization level as entropy, we have

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where n is the number of unique files/CIDs accessed while p_i is defined as follows:

$$p_i = \frac{\text{Accesses to file } i}{\text{Total accesses to all files}}$$

Gini Coefficient. The Gini coefficient is another useful metric, commonly utilized to measure inequality in distributions [36]. In the context of IPFS, it is used to quantify the inequality of storage distribution. The formula to calculate the Gini coefficient is as follows:

$$G = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|}{2n^2 \overline{x}},$$

where n is the number of nodes, x_i is the amount of data stored by node i, and $\overline{x}$ is the mean amount of data stored per node.

Data Synthesis. Using the Biswap and DHT logs, we analyze changes in the centralization level over time by examining data

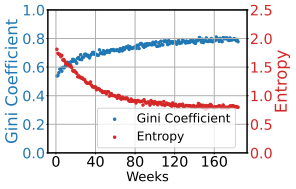


Figure 4: Gini Coefficient and entropy over the 3-year time period.

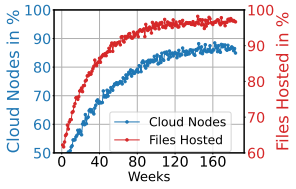


Figure 5: The share of cloud nodes and their host files over the 3-year time period.

weekly. Each week, we calculate the number of times a CID is requested and identify which peers own these CIDs. Table 1 lists the characteristics of this dataset. For this analysis, we make three key assumptions: (1) each CID represents a storage block of 256 KB, as this is the default block size of IPFS, (2) the peers hosting these CIDs remain consistently active in the network and continue to provide access to the CIDs without leaving the network, and (3) for those peers announcing multiple IP addresses as a result of peer churn, we count them as a single entity, *i.e.,* provider.

Table 1: Dataset characteristics used in the centralization level analysis.

CID	Requested By	Owned BY	Timestamp
c_0	$<...p_i...>$	$<...p_j...>$	$Week_k$

5.2 Results

Figure 4 depicts the changes in the Gini coefficient and entropy over the three-year period. Initially, in early 2021, the Gini Coefficient starts at 0.53, indicating a moderate level of storage inequality among peers. Over the course of this period, this coefficient increases significantly, reaching 0.78 by mid-2024. This rise signifies that storage inequality has become increasingly severe, with a small number of powerful peers accumulating a larger share of content. Specifically, by the last week of our measurement period, only 5% of the peers are responsible for hosting 80.55% of the content. In contrast, at the beginning of our measurement period, 21.38% of the peers hosted 80% of the content.

Similarly, the entropy of IPFS follows a comparable pattern to the Gini Coefficient, exhibiting a decreasing trend over the three years. The entropy metric measures the predictability of file access within the network; a decline in entropy suggests that access to popular contents has become more frequent and predictable. This decrease in entropy, along with the rise in the Gini Coefficient, points to a growing level of centralization in IPFS, highlighting a shift towards more centralized control within the IPFS network.

To understand why such centralization trends may be occurring, we are inspired by prior studies [6, 27], which suggest that the rise of centralized cloud nodes could be the reason. By identifying the IP addresses, we can distinguish cloud peers located in data centers and gateway nodes maintained for users without direct access to IPFS. Obtaining the IP addresses and PeerID of these gateway nodes is straightforward as they typically do not leave the network, making them easy to track.

We further analyze the share of storage these cloud nodes manage. Figure 5 depicts the percentage change of cloud nodes and the files they serve over the 3-year period. The data shows a clear increasing trend. For instance, at the start of our study period, cloud nodes comprised 50.02% of the peer set and hosted 52.32% of the files. By the end of the period, these figures dramatically increased to 87.33% of the peer set and 97.43% of the total files. This indicates an increasingly dominant role of cloud nodes within the IPFS network.

Our findings suggest a higher percentage of cloud node involvement than the results from previous studies [6], which estimated the percentage by crawling the DHT and building a network topology. Our results are derived directly from internal IPFS traffic data, which explains the greater share attributed to cloud nodes. More importantly, we observed a concerning trend towards centralization within our measurement period. This increased reliance on cloud nodes suggests a movement towards centralization within an ecosystem that fundamentally aims for decentralization.

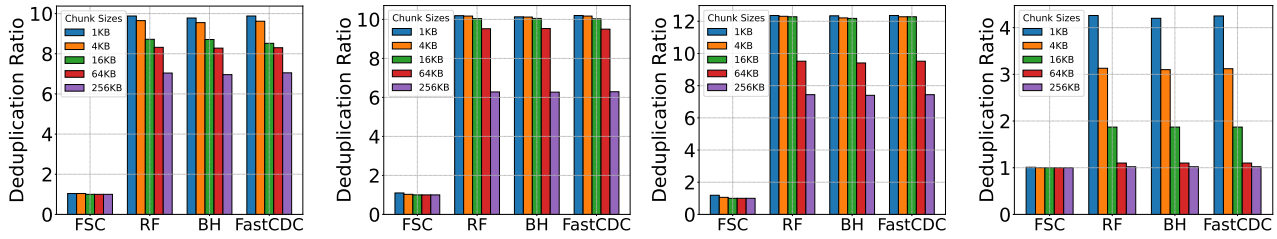
> **Takeaway 3:** Both the entropy and Gini Coefficient studies clearly show a significant centralization trend in IPFS over the 3-year dataset, suggesting that in practice, IPFS is on the opposite direction of its decentralized goal. As a result, file access in IPFS becomes more predictable and concentrated among fewer and powerful entities, further highlighting the centralization within IPFS.

6 Deduplication

In the last section, we show that centralization has kept growing quickly in the past three years in IPFS. While this is in the opposite direction of its original design goal and deserves more research, from a practical standpoint view, serving the majority of users from a small number of cloud-based nodes does improve the user experience and attract users. Furthermore, this may allow these cloud nodes to efficiently perform deduplication for storage savings. To assess this, we begin by examining the efficiency of the default deduplication algorithm employed by IPFS and compare it with other alternatives. We then show the trade-offs involved in incorporating those deduplication methods and propose and evaluate new solutions designed to balance these trade-offs.

6.1 Analysis of Deduplication Ratio

Motivation. Current IPFS employs a sequential and fixed-size chunking approach (FSC), where files are divided into multiple 256 KB chunks. However, FSC suffers from a boundary-shift problem that significantly lowers deduplication efficiency. For example, if one single bit is deleted at the beginning of a file, all current chunk cutpoints (*i.e.,* boundaries) declared by FSC will be shifted and no duplicate chunks will be detected. As such, in order to evaluate the deduplication technique employed by IPFS, we perform our deduplication ratio measurements on four traces, detailed in Table 2. (1) SNP: This trace is the "snapshot" of top 180 providers by sampling 1% of the CIDs in the Bitswap logs and DHT logs. (2) NFT: This is an NFT trace, scraped from the web of top-1000 NFT



(a) **SNP** (b) **NFT** (c) **MMA** (d) **TXT**

Figure 6: Comparison of different deduplication algorithms on different datasets.

digital assets of OpenSea [1] that have an IPFS url. (3) MMA: This is a multimedia trace, obtained from a previous public gateway dataset [10]. (4) TXT: This is a text trace. This category includes plain text and JSON files from the same dataset as the multimedia traces, reflecting previous findings that these file types are among the most prevalent in IPFS.

Table 2: Dataset characteristics used in the deduplication analysis.

Name	Size	# of Files	Description
SNP	18.2 TB	9.10 M	A snapshot of the top-180 providers.
NFT	6.7 TB	8.22 M	Top-1000 collections of OpenSea.
MMA	2.8 TB	1.37 M	Multimedia files collected from a public gateway dataset [10].
TXT	13.3 GB	4.01 M	Text and JSON files collected from a public gateway dataset [10].

Metrics. One common metric to evaluate deduplication efficiency is the deduplication ratio, which represents the ratio of the input dataset's size before and after deduplication. This metric reflects the ability of a deduplication technique to identify and eliminate duplicate data from the input. For instance, a deduplication ratio of 2 indicates that 50% of the input data is redundant and can be eliminated. Another important metric is deduplication speed, which measures the volume of data processed within a given time frame. This metric assesses the efficiency of the deduplication process in terms of throughput.

Methodology. To investigate the deduplication performance of the default FSC deduplication algorithm in IPFS, we utilize various chunk sizes ranging from 1KB to 1MB, incrementing each by powers of four. This approach allows us to understand how different chunk sizes impact the effectiveness of the FSC algorithm across the 4 datasets we employ. At the same time, we conduct controlled experiments to evaluate three alternative deduplication algorithms: Rabin Fingerprint (RF) [8], Buzhash (BH) [9], and FastCDC [41]. Each of these algorithms is configured to match the expected average chunk size used in the FSC approach. Our experiments are carried out using the original algorithms rather than the IPFS client to avoid other interferences. These tests are performed on an EC2 instance (`t2.xlarge`, 4 vCPUs, 16GB memory), applying each algorithm to the four datasets as specified.

Results. Figure 6 shows the deduplication ratio of different deduplication algorithms across different datasets. Notably, FSC shows a particularly low deduplication ratio, especially with the current default 256KB chunk size. For instance, FSC manages to eliminate only about 4% of duplicates in the snapshots of the top-180 providers. In stark contrast, content-based algorithms such as RF, BH, and FastCDC can reduce nearly 90% of duplicates in the same scenario, which can result in substantial storage savings of nearly 16 TB. These top-180 providers primarily use cloud services like AWS and Cloudflare. Given that our analysis sampled only 1% of the stored files, the total potential savings on these platforms could be around 1.8 PB. Considering the EC2 storage pricing of $0.08 per GB per month, the effective cost savings could exceed $100k per month. This significant financial impact highlights the economic benefits of optimizing deduplication strategies within IPFS, especially in cloud environments where storage costs are non-trivial.

> **Takeaway 4:** The FSC achieves about zero deduplication efficiency at the default chunk size in all datasets. In contrast, applying CDC methods can save up to 90% storage. This is especially worthwhile considering that most of the top providers of IPFS are located in cloud centers.

6.2 Trade-off of Deduplication Algorithms

Although Content-Defined Chunking (CDC) based deduplication algorithm can effectively eliminate duplicates and save storage space, it is at the expense of high *chunking* overhead, the process of splitting a file into chunks. To evaluate this overhead, we measure the deduplication speed in IPFS using a 100MB dummy file under different chunk sizes and deduplication algorithms. The measurement is performed on an EC2 instance (`t2.xlarge`). Note that we implement FastCDC in IPFS as it is not provided in the configurations and the chunking process only utilizes one thread. Table 3 presents the deduplication speed of different deduplication algorithms at different chunk sizes. As can be seen in this table, we observe that

(1) FSC outperforms the other 3 CDC based deduplication algorithms across all chunk sizes as it operates on a straightforward splitting of predetermined chunk size, (2) the deduplication speed becomes extremely slower when employing small chunk sizes like 1KB, despite these sizes offering better deduplication ratios.

Table 3: Deduplication speed (MB/s) by algorithm and chunk size in original IPFS.

Alg. / Chunk Size	1KB	4KB	16KB	64KB	256KB
FSC	0.12	0.37	1.92	7.91	30.3
RF	0.08	0.22	0.92	3.45	15.25
BH	0.08	0.28	1.12	4.05	16.32
FastCDC	0.09	0.32	1.57	5.21	21.55

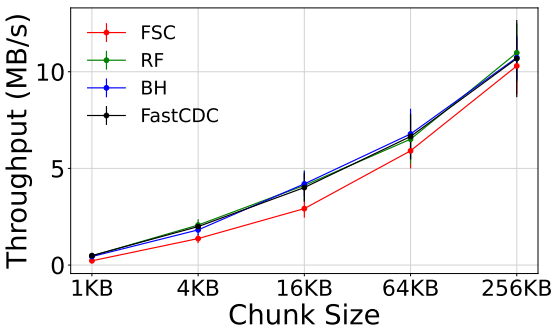


Figure 7: The downloading speed (MB/s) as a function of chunk size.

Another trade-off of applying the deduplication algorithm introduces is the downloading throughput. Previous works (*e.g.,* [27]) have shown that a small chunk size can be detrimental to the downloading throughput. Figure 7 further demonstrates this degradation with chunk size when downloading a 100MB file repeated 100 times. As shown in this figure, a 256KB chunk size offers a downloading throughput that is 50× larger than that of a 1KB chunk size when using the FastCDC. Although CDC can help improve the speed of data transfer as it can reduce the transferred data blocks, the improvement is minimal, improving by only 4.8% at a chunk size of 256KB compared to FSC. Nevertheless, the CDC method still experiences throughput degradation with decreasing chunk size. This significant difference underscores the need to balance the deduplication efficiency without compromising the overall performance of network data transfer and publication.

Takeaway 5: Typically, a smaller chunk size can facilitate a better deduplication ratio. However, such a small chunk size can negatively impact the user's performance: the deduplication speed and downloading throughput.

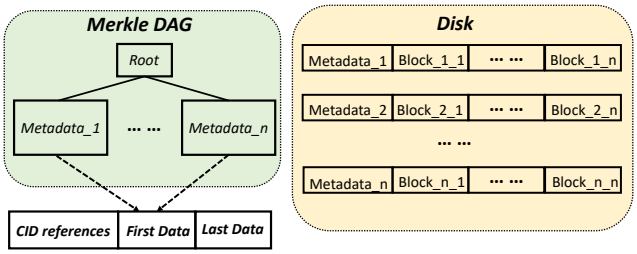


Figure 8: Overview of the metadata design.

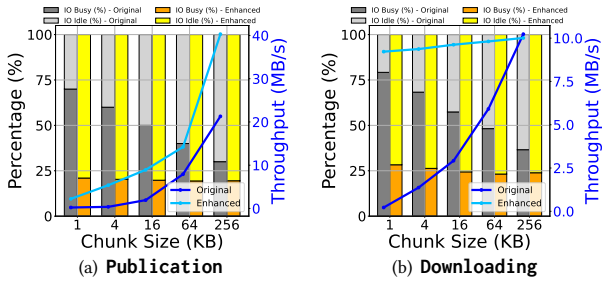
6.3 Metadata Based Deduplication

The relative poorer downloading and publication performance of IPFS is multidimensional not only because of the choice of the chunk size. There are other techniques like concurrent chunking or connections and employing dedicated nodes [38]to accelerate the performance of IPFS, which are orthogonal to the deduplication algorithms. Under the current framework of IPFS, we next discuss how to achieve a higher deduplication ratio and maintain a comparable downloading and publication performance. As such, we propose a new metadata based deduplication technique for content delivery network like IPFS.

Design of Metadata. The poor performance of IPFS downloading stems from its linear processing manner using a small chunk size, linearly retrieving and writing small chunks. To improve the downloading performance and maintain a decent deduplication ratio, we introduce a new Merkle DAG as metadata for each IPFS file. The overview of the new Merkle DAG is presented in Figure 8. The core ideology is as follows. (1) *Metadata format.* In the original IPFS, the internal nodes of the Merkle tree merely contain the children CIDs. In contrast, the new metadata design incorporates additional details: it specifies the range of bytes that the metadata represents. This is denoted as *"First Data"* and *"Last Data,"* indicating the relative positions of the metadata block. In addition, the new metadata references the CIDs based on their occurrence order and aggregates this with the count and size of occurrences in the form of *<CID, num, size>*, whereas the original IPFS system logs each CID sequentially without considering block locality. (2) *Storage locality.* Unlike the original IPFS, which stores blocks based on hash order to facilitate quick searches, our new design groups blocks that belong to the same file together and places the corresponding metadata adjacent to these blocks. For duplicated blocks, our system stores a pointer to the actual position. In other words, our new design organizes data file by file, as opposed to the original IPFS's method of storing data block by block.

In the original IPFS, files are fetched block by block. When using small chunk sizes, such as 1KB, the IPFS client must perform repetitive I/O operations to read and write these small blocks, which delays the entire downloading and publication process. Our proposed solution addresses this inefficiency by aggregating these small blocks into a larger block before any read/write operations on the disk, accelerating the downloading/publication process.

Performance Evaluation. In order to compare the enhanced IPFS with the new metadata design, we deploy the enhanced IPFS on a private IPFS cluster with 5 nodes (EC2 instance `t2.xlarge`, HDD



(a) **Publication** (b) **Downloading**

Figure 9: Performance comparison between the original IPFS and the enhanced IPFS.

storage). We first let the original IPFS and enhanced IPFS upload a 1GB identical dummy file, respectively. We repeat the uploading 100 times and reconfigure the IPFS client after every experiment. In both systems, the duplicate data is removed by FastCDC algorithm and we log the disk time and the publication throughput. Figure 9(a) shows the result of the IO-busy percentage and the publication throughput. As shown in this figure, the average publication throughput is 2.24× larger than the original IPFS. This is due to the fact that the enhanced IPFS can largely reduce the I/O time as a result of sequential access of the disk, where the average I/O-busy percentage is 19.82% compared to 50.07% of the original IPFS. Then, we let the original IPFS and enhanced IPFS clients fetch the uploaded 1GB file remotely and log the corresponding disk time and downloading throughput. Figure 9(b) shows the IO-busy percentage and the downloading throughput when retrieving the files. As shown in the figure, the enhanced IPFS achieves higher downloading throughput than the original IPFS, particularly for small chunk sizes. On the other hand, the downloading throughput of the enhanced IPFS is more resilient to the reduction in chunk size as a result of lower I/O overhead.

7 Related Work

Decentralized Storage Network. As a supplement to traditional cloud storage system, decentralized storage network (DSN) has been undergoing a fast development and widespread adoption as a result of the evolving blockchain technology. Besides IPFS, the most well-known DSNs are Storj [21], Sia [28], Filecoin [20] and Swarm [32], all of which aim to provide a censorship-resilient alternative to cloud storage. There are also prototypes of DSNs. For example, FileDAG [18] uses DAG-Rider as the consensus algorithm and builds a two-layer DAG-based blockchain ledger, facilitating flexible file indexing and multi-versioned file storage. As for the data management in DSNs, previous work have investigated load balancing [16, 30, 45], data security [17, 43, 44] and deduplication [42, 46, 47].

IPFS Measurement. As the most popular storage layer for Web3 applications, many prior studies have investigated various aspects of IPFS, ranging from measuring its performance [4, 26, 27, 33], analyzing its decentralization [6, 10, 27, 38], discussing its design and implementation [11, 19], and potential to support applications such as video streaming [39], among others. Although previous works [6, 10, 27, 38] have revealed the centralization trend of IPFS from various perspectives (*e.g.*, peer distribution, request origin), they are limited to snapshots of centralization at specific moments—essentially, the times at which the experiments are conducted. In contrast, our study aims to analyze the evolution of IPFS's centralization over a three-year period, presenting a more comprehensive view of centralization trends within IPFS rather than at isolated points. By quantifying trends in data replication and concentration over time, we provide a granular understanding of IPFS's drift toward centralization—a critical but previously unexamined aspect of its operational history.

IPFS Optimization. Recent advancements in IPFS have targeted two critical areas: (1) refining content discovery and routing mechanisms [13, 35, 38], (2) strengthening privacy guarantees [12]. The IPFS developers have improved Bitswap's content discovery capabilities, with studies showing that Bitswap now outperforms traditional DHT methods in efficiency [13]. Optimizations have also targeted the DHT, introducing specialized peers [38] and novel content publication strategies that leverage optimistic nodes to store peer records [34]. Furthermore, privacy enhancements have been proposed through a new Bitswap schema designed to reduce the traceability of Bitswap messages [12]. While prior work has overlooked storage redundancy, our study introduces novel deduplication strategies to address IPFS's growing inefficiencies in redundant data handling.

IPFS Security and Privacy. Given its decentralized nature, IPFS is naturally susceptible to Sybil attacks [15] and eclipse attacks [25, 29]. A recent study [31] uses its decentralized nature to conduct a content censorship attack at a low cost and proposes a technique to detect such attacks. Furthermore, it's worth noting that the utilization of Bitswap messages within IPFS has the potential to track individual usage patterns [5], thereby leading to privacy-related challenges.

8 Conclusion

IPFS exemplifies the decentralized ethos of Web 3.0 through its content-addressed architecture and peer-to-peer design. In this paper, we have conducted a measurement study based on a 3-year trace collected from IPFS internal traffic. Throughout our systematic and extensive analysis, we have three major findings: (1) persistently low data replication levels across nodes; (2) growing centralization with a small fraction of nodes disproportionately handling a majority of client requests; and (3) negligible deduplication efficiency with IPFS' default FSC deduplication strategy. Moreover, we have also proposed a new metadata design to accommodate CDC deduplication strategies to achieve a higher deduplication rate while balancing performance trade-offs. Our study reveals a paradox: while IPFS' operational reality contradicts its design promise, they may inadvertently enable opportunities such as optimized deduplication. We hope our work can provide new insights into the development and optimization of IPFS and Web 3.0 in the next stage.

Acknowledgment

We thank the anonymous reviewers for their valuable feedback. This work was partially supported by the National Science Foundation under CNS-2007153 and CNS-2322860. Some results presented in this paper were obtained using CloudBank [24], supported by the NSF award CNS-1925001.

References

[1] 2018. OpenSea, the largest NFT Marketplace. https://opensea.io/.
[2] 2019. IPFS Bitswap Website. https://docs.ipfs.tech/concepts/bitswap/.
[3] 2024. CID Version. https://docs.ipfs.tech/concepts/content-addressing/#cid-conversion.
[4] Omar Abdullah Lajam and Tarek Ahmed Helmy. 2021. Performance evaluation of ipfs in private networks. In *2021 4th International Conference on Data Storage and Data Engineering*. 77–84.
[5] Leonhard Balduf, Sebastian Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. 2022. Monitoring data requests in decentralized data storage systems: A case study of IPFS. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 658–668.
[6] Leonhard Balduf, Maciej Korczyński, Onur Ascigil, Navin V Keizer, George Pavlou, Björn Scheuermann, and Michał Król. 2023. The cloud strikes back: Investigating the decentralization of IPFS. In *Proceedings of the 2023 ACM on Internet Measurement Conference*. 391–405.
[7] Juan Benet. 2014. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561* (2014).
[8] Andrei Z Broder. 1993. Some applications of Rabin's fingerprinting method. In *Sequences II: Methods in Communication, Security, and Computer Science*. Springer, 143–152.
[9] Jonathan D Cohen. 1997. Recursive hashing functions for n-grams. *ACM Transactions on Information Systems (TOIS)* 15, 3 (1997), 291–320.
[10] Pedro Ákos Costa, João Leitão, and Yannis Psaras. 2022. Studying the workload of a fully decentralized Web3 system: IPFS. *arXiv preprint arXiv:2212.07375* (2022).
[11] Erik Daniel and Florian Tschorsch. 2022. IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks. *IEEE Communications Surveys & Tutorials* 24, 1 (2022), 31–52.
[12] Erik Daniel and Florian Tschorsch. 2024. Exploring the design space of privacy-enhanced content discovery for bitswap. *Computer Communications* 217 (2024), 12–24.
[13] Alfonso De la Rocha, David Dias, and Yiannis Psaras. 2021. Accelerating content routing with bitswap: A multi-path file transfer protocol in ipfs and filecoin. *San Francisco, CA, USA* (2021), 11.
[14] Trinh Viet Doan, Roland van Rijswijk-Deij, Oliver Hohlfeld, and Vaibhav Bajpai. 2022. An empirical view on consolidation of the web. *ACM Transactions on Internet Technology (TOIT)* 22, 3 (2022), 1–30.
[15] Peter Druschel, Frans Kaashoek, and Antony Rowstron. 2003. *Peer-to-Peer Systems: First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*. Vol. 2429. Springer.
[16] Yuefeng Du, Anxin Zhou, and Cong Wang. 2024. DWare: Cost-Efficient Decentralized Storage with Adaptive Middleware. *IEEE Transactions on Information Forensics and Security* (2024).
[17] Hechuan Guo, Minghui Xu, Jiahao Zhang, Chunchi Liu, Rajiv Ranjan, Dongxiao Yu, and Xiuzhen Cheng. 2024. BFT-DSN: A Byzantine Fault Tolerant Decentralized Storage Network. *IEEE Trans. Comput.* (2024).
[18] Hechuan Guo, Minghui Xu, Jiahao Zhang, Chunchi Liu, Dongxiao Yu, Schahram Dustdar, and Xiuzhen Cheng. 2023. FileDAG: A multi-version decentralized storage network built on DAG-based blockchain. *IEEE Trans. Comput.* 72, 11 (2023), 3191–3202.
[19] Huawei Huang, Jianru Lin, Baichuan Zheng, Zibin Zheng, and Jing Bian. 2020. When blockchain meets distributed file systems: An overview, challenges, and open issues. *IEEE Access* 8 (2020), 50574–50586.
[20] Protocol Labs. 2024. *Filecoin Documentation.* https://docs.filecoin.io
[21] Storj Labs. 2024. *Storj - Decentralized Cloud Storage.* https://www.storj.io
[22] Petar Maymounkov and David Mazières. 2002. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Peer-to-Peer Systems*, Peter Druschel, Frans Kaashoek, and Antony Rowstron (Eds.).
[23] Athicha Muthitacharoen, Benjie Chen, and David Mazieres. 2001. A low-bandwidth network file system. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*. 174–187.
[24] Michael Norman, Vince Kellen, Shava Smallen, Brian DeMeulle, Shawn Strande, Ed Lazowska, Naomi Alterman, Rob Fatland, Sarah Stone, Amanda Tan, et al. 2021. Cloudbank: Managed services to simplify cloud access for computer science research and education. In *Practice and Experience in Advanced Research Computing*. 1–4.
[25] Bernd Prünster, Alexander Marsalek, and Thomas Zefferer. 2022. Total Eclipse of the Heart - Disrupting the InterPlanetary File System. In *USENIX Security Symposium*.
[26] Jiajie Shen, Yi Li, Yangfan Zhou, and Xin Wang. 2019. Understanding I/O Performance of IPFS Storage: A Client's Perspective. In *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*.
[27] Ruizhe Shi, Ruizhi Cheng, Bo Han, Yue Cheng, and Songqing Chen. 2024. A Closer Look into IPFS: Accessibility, Content, and Performance. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 8, 2 (2024), 1–31.
[28] Sia. 2024. *Decentralized Storage for the Post-Cloud World.* https://sia.tech
[29] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. 2004. Defending against eclipse attacks on overlay networks. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*. 21–es.
[30] Victor J Sosa-Sosa, Alfredo Barron, Jose Luis Gonzalez-Compean, Jesus Carretero, and Ivan Lopez-Arevalo. 2020. Improving performance and capacity utilization in cloud storage for content delivery and sharing services. *IEEE Transactions on Cloud Computing* 10, 1 (2020), 439–450.
[31] Srivatsan Sridhar, Onur Ascigil, Navin Keizer, François Genon, Sébastien Pierre, Yiannis Psaras, Etienne Rivière, and Michał Król. 2023. Content Censorship in the InterPlanetary File System. *arXiv preprint arXiv:2307.12212* (2023).
[32] EthSwarm Team. 2024. *EthSwarm.* https://www.ethswarm.org
[33] Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. 2022. Design and evaluation of IPFS: a storage layer for the decentralized web. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 739–752.
[34] Dennis Trautwein, Yiluo Wei, Yiannis Psaras, Moritz Schubotz, Ignacio Castro, Bela Gipp, and Gareth Tyson. [n. d.]. IPFS in the Fast Lane: Accelerating Record Storage with Optimistic Provide.
[35] Dennis Trautwein, Yiluo Wei, Yiannis Psaras, Moritz Schubotz, Ignacio Castro, Bela Gipp, and Gareth Tyson. 2024. Ipfs in the fast lane: Accelerating record storage with optimistic provide. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 1920–1929.
[36] Rajesh Vasa, Markus Lumpe, Philip Branch, and Oscar Nierstrasz. 2009. Comparative analysis of evolving software systems using the Gini coefficient. In *2009 IEEE international conference on software maintenance*. IEEE, 179–188.
[37] Qin Wang, Rujia Li, Qi Wang, and Shiping Chen. 2021. Non-fungible token (NFT): Overview, evaluation, opportunities and challenges. *arXiv preprint arXiv:2105.07447* (2021).
[38] Yiluo Wei, Dennis Trautwein, Yiannis Psaras, Ignacio Castro, Will Scott, Aravindh Raman, and Gareth Tyson. 2024. The Eternal Tussle: Exploring the Role of Centralization in {IPFS}. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 441–454.
[39] Zhengyu Wu, ChengHao Ryan Yang, Santiago Vargas, and Aruna Balasubramanian. 2023. Is IPFS Ready for Decentralized Video Streaming?. In *Proceedings of the ACM Web Conference 2023*.
[40] Wen Xia, Hong Jiang, Dan Feng, Fred Douglis, Philip Shilane, Yu Hua, Min Fu, Yucheng Zhang, and Yukun Zhou. 2016. A comprehensive study of the past, present, and future of data deduplication. *Proc. IEEE* 104, 9 (2016), 1681–1710.
[41] Wen Xia, Yukun Zhou, Hong Jiang, Dan Feng, Yu Hua, Yuchong Hu, Qing Liu, and Yucheng Zhang. 2016. {FastCDC}: A fast and efficient {Content-Defined} chunking approach for data deduplication. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. 101–114.
[42] Yuanjian Xing, Zhenhua Li, and Yafei Dai. 2010. Peerdedupe: Insights into the peer-assisted sampling deduplication. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 1–10.
[43] Minghui Xu, Jiahao Zhang, Hechuan Guo, Xiuzhen Cheng, Dongxiao Yu, Qin Hu, Yijun Li, and Yipu Wu. 2024. FileDES: A Secure Scalable and Succinct Decentralized Encrypted Storage Network. *arXiv preprint arXiv:2403.14985* (2024).
[44] Siyi Yang, Ahmed Hareedy, Robert Calderbank, and Lara Dolecek. 2020. Topology-aware cooperative data protection in blockchain-based decentralized storage networks. In *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 622–627.
[45] Hao Yin, Zijian Zhang, Liehuang Zhu, Meng Li, Xiaojiang Du, Mohsen Guizani, and Bakh Khoussainov. 2020. A blockchain-based storage system with financial incentives for load-balancing. *IEEE Transactions on Network Science and Engineering* 8, 2 (2020), 1178–1188.
[46] Haoran Yuan, Xiaofeng Chen, Jianfeng Wang, Jiaming Yuan, Hongyang Yan, and Willy Susilo. 2020. Blockchain-based public auditing and secure deduplication with fair arbitration. *Information Sciences* 541 (2020), 409–425.
[47] Bo Zhang, Helei Cui, Yaxing Chen, Xiaoning Liu, Zhiwen Yu, and Bin Guo. 2022. Enabling secure deduplication in encrypted decentralized storage. In *International Conference on Network and System Security*. Springer, 459–475.