A Closer Look into IPFS: Accessibility, Content, and Performance

RUIZHE SHI, George Mason University, USA
RUIZHI CHENG, George Mason University, USA
BO HAN, George Mason University, USA
YUE CHENG, University of Virginia, USA
SONGQING CHEN, George Mason University, USA

The InterPlanetary File System (IPFS) has recently gained considerable attention. While prior research has focused on understanding its performance characterization and application support, it remains unclear: (1) what kind of files/content are stored in IPFS, (2) who are providing these files, (3) are these files always accessible, and (4) what affects the file access performance.

To answer these questions, in this paper, we perform measurement and analysis on over 4 million files associated with CIDs (content IDs) that appeared in publicly available IPFS datasets. Our results reveal the following key findings: (1) Mixed file accessibility: while IPFS is not designed for a permanent storage, accessing a non-trivial portion of files, such as those of NFTs and video streams, often requires multiple retrieval attempts, potentially blocking NFT transactions and negatively affecting the user experience. (2) Dominance of NFT (non-fungible token) and video files: about 50% of stored files are NFT-related, followed by a large portion of video files, among which about half are pirated movies and adult content. (3) Centralization of content providers: a small number of peers (top-50), mostly cloud nodes hosted by tech companies, serve a large portion (95%) of files, deviating from IPFS's intended design goal. (4) High variation of downloading throughput and lookup time: large file retrievals experience lower average throughput due to more overhead for resolving file chunk CIDs, and looking up files hosted by non-cloud nodes takes longer. We hope that our findings can offer valuable insights for (1) IPFS application developers to take into consideration these characteristics when building applications on top of IPFS, and (2) IPFS system developers to improve IPFS and similar systems to be developed for Web3.

CCS Concepts: • **Networks** → *Network measurement*; *Network performance analysis*; • **Computer systems organization** → *Peer-to-peer architectures.*

Additional Key Words and Phrases: peer-to-peer network, IPFS, network measurement

ACM Reference Format:
Ruizhe Shi, Ruizhi Cheng, Bo Han, Yue Cheng, and Songqing Chen. 2024. A Closer Look into IPFS: Accessibility, Content, and Performance. *Proc. ACM Meas. Anal. Comput. Syst.* 8, 2, Article 20 (June 2024), 31 pages. https://doi.org/10.1145/3656015

1 INTRODUCTION

The traditional web architecture has been utilizing a client-server model by dividing network nodes into servers and clients [45]. This structure inherently leads to a situation where a substantial

Authors' addresses: Ruizhe Shi, George Mason University, Fairfax, USA, rshi3@gmu.edu; Ruizhi Cheng, George Mason University, Fairfax, USA, rcheng4@gmu.edu; Bo Han, George Mason University, Fairfax, USA, bohan@gmu.edu; Yue Cheng, University of Virginia, Charlottesville, USA, mrz7dp@virginia.edu; Songqing Chen, George Mason University, Fairfax, USA, sqchen@gmu.edu.

ACM 2476-1249/2024/6-ART20
https://doi.org/10.1145/3656015

population of clients becomes heavily reliant on a single, or a few dominant server(s) for their data needs. This centralization is often necessitated by the need for efficient data management and transmission and providing > 99% uptime [20], which, however, raises concerns that the data and content are often centralized in a small group of companies, such as those tech giants. As such, Web 3.0 (also known as Web3) [50] envisions that the next-generation web should incorporate a decentralized structure, exemplified by the operation of cryptocurrencies such as Bitcoin [59], Ethereum [33] on peer-to-peer (P2P) networks [32], and Metaverse [35, 36], which have frequently reaped the media headlines in the past few years.

The InterPlanetary File System (IPFS) [25], by integrating content-based addressing [34, 55, 65] and benefiting from a P2P structure, is an early effort towards this vision. By harnessing the power of distributed P2P networks, IPFS aims to replace traditional web protocols, such as Hypertext Transfer Protocol (HTTP), in the next generation of the Internet and enables a more resilient and decentralized alternative to how information is accessed and shared online [25]. Since its first release in 2015, IPFS is gaining increasing popularity in the field of Web3 technologies [10]. By default, a client is required to install an IPFS client software in order to participate in content storage sharing and retrieval. To facilitate accesses from clients without using IPFS software, but rather the traditional web protocols (*i.e.,* HTTP/HTTPS), IPFS also supports gateways, which act as a proxy so that traditional clients can still access content stored in the IPFS network [7].

The increasing popularity of IPFS has attracted a number of studies that examined its performance [27, 68, 72], decentralization [31], content duplication [60], as well as its design and implementation, deployment experience [30, 72], and potential to support applications such as video streaming [75], among others. While these studies [30, 31, 68, 72] provided insights into various aspects of IPFS, including its structure, protocol, user participation, and node distribution, there remains a gap in understanding the actual content stored in IPFS. Furthermore, prior studies [27, 68, 72] have limitations in their performance analysis, relying solely on dummy files and private clients for experiments, and mostly neglecting large files (over 16MB). This motivates our study, focusing on the analysis of IPFS public gateway and network traces. As such, our study aims to evaluate the performance of IPFS and address the following questions regarding content and its providers:

Q1: What kind of files are stored in IPFS? Answering this question will provide us with a better understanding of whether the content shared in IPFS aligns well with our current web (*e.g.,* Web 2.0). This can also shed light on the potential of IPFS replacing the existing web. On the other hand, as a storage system, data availability/accessibility is also important. Traditionally, web content hosted by reliable services, such as public clouds, typically ensures 24/7 accessibility with a > 99.9% monthly uptime SLA (service-level agreement) [20]. This leads to our second question: ***Q2: Are data stored in IPFS always accessible?*** While hosting content across peers can minimize content centralization, it is also desirable to maintain comparable accessibility in a decentralized fashion. It is worth noting that IPFS does not aim to provide permanent file storage. But as a storage system, it is crucial to enable persistent and highly available storage for certain files such as non-fungible tokens (NFTs) files and video streaming files in order to support the transactions or video streaming applications. Lastly, ***Q3: Who are the content providers?*** A common concern of traditional web services is that the content is predominantly served by or through a few large organizations, such as tech giants. IPFS aims to counteract this centralized trend by promoting decentralization. Therefore, an analysis of content providers can provide insights into the extent to which IPFS achieves its design goal.

Seeking answers to the above research questions is not straightforward because IPFS is a large-scale P2P network. As such, it is extremely challenging, if not impossible, to obtain a global view of the system in order to answer these questions. In this paper, we start with a list of content IDs

(CIDs) extracted from a publicly-available two-week-long gateway log [39] and retrieved 4 million files corresponding to those CIDs. During file retrieval, we also instrument our clients to timestamp the retrieval phases including lookup and downloading. Analysis of these data can shed light on ***Q4: How is the file retrieval performance?*** To further supplement our study, we employ the same process using a network trace obtained from a prior study [30]. By analyzing the downloaded files and the information obtained during the retrieval process, we set to answer the above questions. Furthermore, our collected information enables us to evaluate the file access performance on a large scale. The highlights of our findings are:

(1) Our analysis on file accessibility in IPFS shows mixed results: while the majority of the files corresponding to the 4 million+ CIDs extracted from a gateway log (1-year old) can still be accessed, only about half can be downloaded right away and it takes multiple attempts in 6 days to download the other half. Furthermore, a small portion of the files remains inaccessible after repeated attempts for a week. For NFT files, about 20% of corresponding files are not instantly accessible, which can block the business transactions of NFTs. We repeat the experiments on CIDs that are six-month old, one-month old, and zero-day old (crawling the DHT and downloading the found CIDs instantly, detailed in Section 4.1). Our results show that no matter how "young" the data is, a portion of the CIDs are not instantly available and need multiple attempts to be retrieved.
(2) Upon examining the files that we successfully retrieved, we found that currently IPFS has been primarily utilized by NFT and video applications: about 50% of files stored in IPFS are NFT related, followed by a large portion (*e.g.,* 33%+ in terms of total file size) of video files, among which about half are used by services serving adult content or pirated movies.
(3) By looking into which peers have provided the content during our file retrievals, we found that the content providers are highly centralized. For example, 95% of retrieved files are from top-50 providers, and many of these providers are located in datacenters, serving as either storage nodes or cloud nodes for data such as NFTs.
(4) The file access performance shows high variation. Specifically, the average retrieval time of small files is dominated by their lookup time, which is about 4× of that of large files on average, while the file retrieval throughput of larger files sees a decreasing trend on average. We also find that the lookup time is highly dependent on the content providers: as a higher ratio of small files are stored on non-cloud nodes that are more likely behind NAT, it takes longer to identify these providers.

We hope our analysis can provide valuable insights to (1) IPFS application developers that they need to take into consideration these IPFS characteristics when building their applications, and (2) IPFS system developers on how to improve the IPFS in the future. As such, we also explore the impact of different chunk sizes on retrieval throughput. The result shows that a dynamic chunk size can help accelerate the IPFS retrieval performance.

The remainder of the paper is as follows. We provide background information about IPFS in Section 2, followed by a discussion of related work. We present our measurement methodology in Section 3. Section 4 and Section 5 present the IPFS file analysis, and our performance measurement results and potential improvements, respectively. We discuss our findings further in Section 6. We make concluding remarks in Section 7.

2 BACKGROUND AND RELATED WORK

In this section, we first provide a brief overview of the components of IPFS and elaborate on how files are published and retrieved on IPFS as illustrated in Figure 1. Then, we survey recent research related to IPFS and distinguish our work from these previous studies.

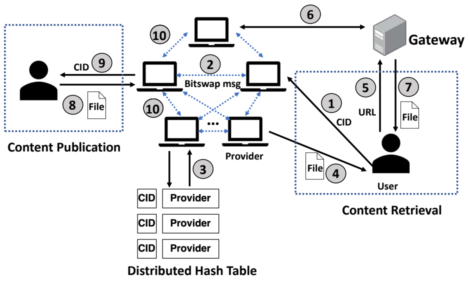


Fig. 1. An overview of the IPFS content publication and retrieval process.

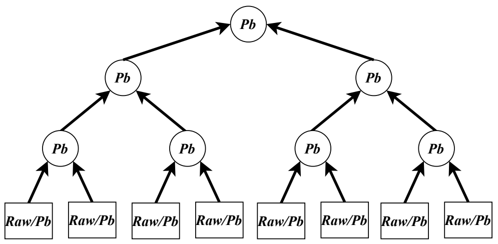


Fig. 2. An illustration of a file's Merkel Hash tree. Pb: DagProtobuf. Raw: raw data.

2.1 A Primer on IPFS

Content Object. Aiming to underpin Web3, IPFS is designed and implemented as a decentralized file system that relies on efficient content addressing [34, 55, 65]. In IPFS, every piece of data is assigned a unique hash, known as the Content Identifier (CID). When a file is uploaded to IPFS, it is partitioned into fixed-sized units termed "chunks", with a default size of 256KB. Each chunk is then assigned a distinct hash computed from its content, which functions as the chunk's identifier (*i.e.,* the CID). These chunks are then encoded and organized in a tree structure called a Merkle Directed Acyclic Graph (DAG) [9]. As shown in Figure 2, each leaf node represents either a file or a file chunk. Each encoded file chunk contains its data and links to related objects. When a CID is looked up in the network, the local client retrieves the object links related to the given CID, thus getting all its DAG file chunks. This feature renders IPFS a self-certifying system [58], in which recipients can verify the integrity of file content by comparing the data's hash with its corresponding CID. This process ensures that any alterations to the file content are promptly detected.

The encoding of a chunk can be inferred from its CID through a mapping mechanism known as Multicodec [5]. Multicodec plays a crucial role in IPFS as it helps interpret the data after it is fetched, with key representations including: (1) DagProtobuf (dag-pb), a common way to encode the data; typically, these correspond to files and directories within IPFS; (2) Raw, representing segments of data or leaf nodes of a file; (3) DagCBOR (dag-cbor) and DagJSON (dag-json), which are alternative encoding mechanisms for DagProtobuf. They encode a generalized data model for hash-linked data and can be easily converted to a `JSON` file.

Figure 2 shows an example of a DAG. The leaf node of the DAG can be either a raw file or a protobuf, containing the original content of a file. Parent nodes are encoded in protobuf type. They are generated when the file is chunked, containing information about how the chunks of a file are organized.

Bitswap. Bitswap [4] is a simple file chunk exchange protocol, similar to BitTorrent [41, 48]. When requesting a file chunk via Bitswap, a peer broadcasts content requests to its neighbors using a `WANT-HAVE` message. Recipient peers respond with a `HAVE` message if they have the requested chunks. Upon receiving the `HAVE` message, the requesting peer sends a `WANT-BLOCK` message and the recipient responds with the target chunk.

IPFS Peers and DHT. Similar to CID, every node or peer[1] participating in the network possesses a unique PeerID, generated through a combination of public key cryptography and hash functions. The node owning specific data is regarded as the provider of the data. When a node aims to publish data to the network, it must signal other nodes of its provider status. To facilitate this, IPFS employs Kademlia [57], a type of Distributed Hash Table (DHT) [53, 66, 70, 77] to store the provider-content record, which maps a CID to corresponding PeerIDs. To publish these records, the provider will

[1]In this paper, we use peer and node interchangeably.

send them to its k closest peers [72] where k can be set to, for example, 20. DHT also stores the mapping of PeerID and the actual peer address, denoted as Multiaddress.

Content Retrieval. As shown in Figure 1, IPFS content retrieval via the network follows: ① *Request content via a CID.* ② *and* ③ *Look up provider record.* ④ *Connect to the provider and download the content.* Content retrieval via the gateway follows: ⑤ *Access the gateway via HTTP.* ⑥ *Search the network if the content is not in the local cache.* ⑦ *Return the data if the content is found.* Content publication follows: ⑧ *Add the content to the local IPFS repository.* ⑨ *The node partitions content into chunks and each chunk is hashed.* ⑩ *Advertise the record to neighbors.* To access a file chunk, a user starts by querying its CID. The retrieval process comprises two stages: lookup and downloading. The lookup is to identify the content provider. It can be done by Bitswap discovery or DHT walk. During the lookup phase, the requesting node initiates a content request and sends `WANT-HAVE` messages to all of its neighbors using the Bitswap protocol. Once some peer responds with a `HAVE` message, the provider is resolved and the lookup is done. If no peer responds in this stage within a time interval (configured as provSearchDelay in the source code and has a default value of 1 second), the requesting node resorts to querying the DHT until it identifies the provider of the requested content. In the worst-case scenario, the entire network may be searched. If the lookup is successful, the requesting node can establish a connection with the content provider to start the downloading phase based on the Multiaddress information of the peer (See Figure 1). The downloading is done using Bitswap, where the requesting node sends a `WANT-BLOCK` message to the provider. It is worth noting that a file consisting of multiple chunks will require the lookup and downloading for each chunk recursively.

Gateway Access. An alternative method to access the content in IPFS is through an IPFS gateway [7]. The purpose of IPFS gateways is to enable users who do not have IPFS software installed to access the data via traditional HTTP protocol. In this method, the CID of the requested content is embedded in the HTTP URL. Therefore, an IPFS gateway serves as a proxy between traditional web access and IPFS. There are a number of gateways deployed in the IPFS network from time to time, by Protocol Labs or third parties. They cache the content using the Least Recently Used (LRU) cache replacement policy and some also serve as the permanent storage for Web3 content [16].

2.2 Related Work

As Web3 continues to evolve, IPFS has recently attracted research from various perspectives. Trautwein *et al.* [72] provided a comprehensive overview of the design and implementation of IPFS, as well as its deployment in practice. They also presented evaluations of the IPFS performance in terms of content retrieval and publication, providing valuable insights into the functionality and capabilities of this decentralized storage solution. Costa *et al.* [39] mainly focused on the analysis of IPFS gateways. They collected access traces from one of the most popular IPFS gateways located in North America for a period of two weeks. They analyzed the provider content in terms of geographical distribution and the request frequency on a daily basis. Balduf *et al.* [31] presented a comprehensive analysis of the traffic and content provider records of IPFS and evaluated the decentralization of the IPFS ecosystem. Their work indicated that current IPFS highly relies on cloud infrastructure and has a significant centralization of contents in the cloud nodes.

To understand the I/O performance of IPFS, Shen *et al.* [68] conducted measurements with different sizes of I/O requests from different locations via Amazon Web Services (AWS) and showed that both the lookup and downloading performance can vary significantly in the entire file retrieval process. Lajam *et al.* [27] compared reading and writing operations with traditional network protocols such as HTTP and FTP and showed that IPFS has a higher latency for files up to 64 MB. Research has also been conducted in network size measurement [30, 40, 51], showing that the

IPFS network size is in the range from 10k to 60k. Moreover, Ascigil *et al.* [29] found that IPFS has performance constraints when it comes to delivering video-on-demand content.

In addition to IPFS performance analysis, some studies investigated the feasibility of supporting various applications on IPFS. For example, Dtube [14] is an online video streaming application built on top of IPFS. Wu *et al.* [75] conducted a measurement study on IPFS to investigate its feasibility of supporting video streaming. Furthermore, Doan *et al.* [43] have shown that Dtube actually retrieves contents from a centralized server. Aderemi *et al.* [28] conducted a qualitative and quantitative study to evaluate if IPFS is a good choice for RPKI (Resource Public Key Infrastructure) repository distribution and identified places for improvement. Filecoin [2], a decentralized storage network, integrates with IPFS to create a system where data can be redundantly stored across multiple independent storage providers. This approach introduces an economic incentive model for long-term data preservation in a decentralized environment. In addition, IPFS has also been used for various applications including storage [78], secure communication [67], and social network [76], where it serves as a backend.

There are rising concerns related to security and privacy with IPFS. Li *et al.* [56] proposed a three-tier storage framework atop blockchain and IPFS to allow users to gain control of their personal data. The use of Bitswap messages within IPFS can potentially lead to tracking of personal usage [30], which poses privacy risks. As a decentralized network, IPFS is inherently vulnerable to Sybil attacks [44] and eclipse attacks [62, 69].

Lastly, in addition to IPFS, other decentralized web technologies have also been studied, including server-based federated services such as Mastodon [64], Pleroma [49], Diaspora [47], and NextCloud [12].

Different from prior studies, this work investigates what kind of content is stored on IPFS, who are providing such content, as well as file availability and access performance by actually retrieving a large amount of files from IPFS based on real user accesses.

3 METHODOLOGY AND DATASETS

In this section, we present the methodology for studying IPFS. Due to the decentralized nature of IPFS, it is challenging, if not possible at all, to monitor the entire IPFS network and obtain records and activities of the network.

3.1 Measurement Methodology

Measurement Setup. We first extracted the CIDs from a publicly available gateway log that was used in a prior study [39]. Based on the CIDs, we instrumented our IPFS clients to look up and download their corresponding files via the IPFS network. During this process, our clients timestamped different activities so that at the end we had a complete time record for the access of each CID. We followed the same approach to processing the network trace used in another prior study [30].

Our 22 content retrieval clients were running on AWS instances and all the nodes were located in the same AWS region, the Paris region. Our experiments were conducted in this region by default if not otherwise explicitly mentioned. They are `t2.2xlarge` instances with 32 GB memory and 8 vCPU each. The IPFS software version is 0.21.0, which is the latest as of April 15th, 2023. All of our content retrievals happened from the middle of April to early May 2023.

Content Retrieval. CID-based content retrieval from IPFS generally consists of two steps: (1) lookup (resolve to get the provider record) and (2) download (directly download files from the provider), as illustrated in Section 2.2. Since we instrumented the source code of our clients to obtain the time duration for lookup and downloading time, we can infer the downloading throughput and lookup time, which we will analyze later.

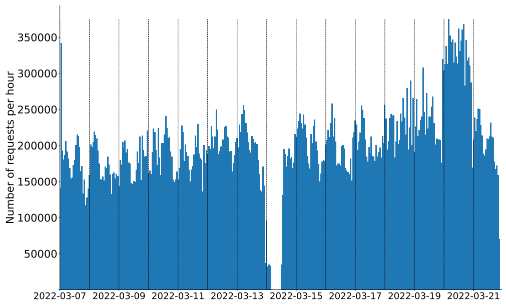


Fig. 3. The hourly request rate of Gateway-22 over a span of 7 days.

3.2 Ethical Considerations

Our initial dataset only contained anonymized information, such as CIDs and peerIDs. In our measurement, we also collected IP address information for certain peers during routing, as well as for the providers involved in the retrieval of files. However, we only used such information for geolocation analysis by mapping the IP address to the city, and did not trace the IP. We will not share the data with anyone without anonymization. While we analyze the content of downloaded files, we do not have any direct information about who has ever accessed the same content, and thus there are no privacy concerns.

Since we actually downloaded the files, the file retrieval process has introduced some traffic, which could impact the performance of IPFS. For this consideration, we performed the retrieval using up to 22 simultaneous instances, and over 90% of the retrievals were completed within 30s. While our downloading may introduce some disturbance to other IPFS applications or users during the same time, we believe its impact is minimal because our week-long collection requests are about 0.4% of the weekly request number of IPFS (over 1 billion) and our traffic is less than 0.1% of the daily traffic in IPFS (over 100 TB) [72]. Some of the downloaded files may involve illegal content. We never shared or will share them or the CIDs with any other entity. We also obtained the IRB approval for our study.

Table 1. Decoded CIDs from the two datasets.

Type of codec	Gateway-22	Network-22
raw	265,466 (7.39%)	251,412,462 (40.03%)
dag-pb	4,012,243 (92.30%)	365,207,766 (58.15%)
others	28,101 (0.31%)	11,420,640 (1.82%)

3.3 Data Overview

Gateway Dataset (Gateway-22). The original gateway log includes information about requests from a gateway managed by Protocol Labs [26]. The log covers a period of 2 weeks, starting from 2022-03-07 UTC 00:00:52+00:00 to 2022-03-21 UTC 22:23:37+00:00. Among the gateway requests, we excluded those where the request has no valid CIDs or the HTTP response status code is not 200 (`200:success`) or the request is not a `GET` request. Additionally, we removed the `POST` requests, as they were rejected by the gateway. Following this pre-processing, our Gateway-22 dataset contains 84, 527, 938 requests. Each request corresponds to a tuple including details such as the timestamp, requester's city-level location, HTTP refer, time, CID, cache hit/miss status, and other information.

Figure 3 illustrates the hourly request rate at the gateway during these two weeks. From this figure, we can observe a mild diurnal daily pattern. Note the significantly lower number of requests on 03/14/2022 is from the original dataset we obtained, and is not due to our processing.

Based on these requests, we further process the CIDs and the `HTTP refer` (if existing), and obtain 4,300,909 unique and valid CIDs. Table 1 summarizes these CIDs. Based on the obtained CIDs, we retrieve and obtain the original files, and store them in our AWS servers. Table 1 shows that for Gateway-22, most of the CIDs (> 90%) correspond to the internal nodes in the Merkle tree. Only a small percentage corresponds to raw files (leaf nodes in the Merkle DAG).

Network Dataset (Network-22). In addition to the gateway log, we also obtained a network dataset that includes Bitswap [4] messages. This network dataset is shared by the authors of another study [30]. The network data was collected by deploying an IPFS client participating in the IPFS network, and over time, it logged all the Bitswap messages going through itself. The time period of the messages was from 2022-03-10 UTC 00:00:00+00:00 to 2022-03-21 UTC 23:59:59+00:00.

Each Bitswap message corresponds to a tuple of `<timestamp, peerID, address, request_type, CID>`, where the address is mapped to the city in our data collection. Here the `request_type` refers to Bitswap signals such as `Cancel`, `WANT_HAVE`, `WANT_BLOCK`. Note that this message cannot tell whether the file corresponding to that CID was successfully downloaded or not. In addition, most of the CIDs in the Bitswap message correspond to a chunk of some file, which corresponds to the leaf node or body node in the Merkle tree [9] as shown in Figure 2.

From the network dataset, we first extracted the CIDs. We also decoded the CIDs (including chunk file CIDs) and Table 1 shows the result. We can observe that 90% of the CIDs in the Gateway-22 are likely to be the root node, as compared to nearly half of the CIDs being raw data in Network-22. This is expected as the gateway users often request an entire file while in the network log, chunks of files are usually requested. Given the large amount of CIDs in Network-22 (over 100 million messages per day on average), we sampled from the entire CID set of Network-22 with a roughly equal number of unique CIDs as in Gateway-22. To align with Gateway-22, we merely extracted the root CIDs. During our analysis, we also collected and extracted some other gateway and network traces with shorter duration. We will introduce them when we present their analysis.

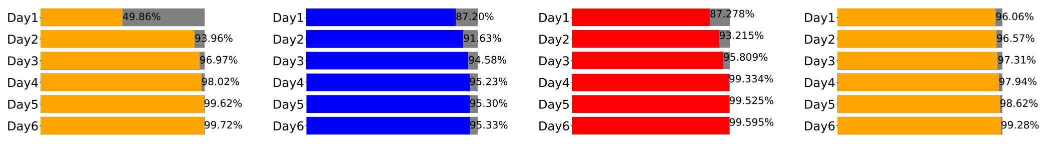
Downloaded Files. For IPFS content analysis, we retrieved all available CIDs from Gateway-22, obtaining 4,070,936 objects. We also sampled nearly 4 million CIDs from Network-22, resulting in 3,770,516 objects. Our analysis shows that a majority of them are either NFT-related files, for example, NFT metadata and digital artworks, or video streaming files. Table 2 and Table 3 give an overview of the CID type distribution. We will analyze them with more detail in the next section.

Table 2. File types in Gateway-22 (by CID).

File type	Count (%)	Total size in GB (%)
NFT	1,595,663 (39.20%)	5402.87 (64.97%)
stream	1,228,821 (30.19%)	1,635.85 (19.67%)
Other	1,246,452 (30.62%)	1277.41 (15.36%)

Table 3. File types in Network-22 (by CID).

Category	Count (%)	Total size in GB (%)
NFT	805,164 (21.35%)	1315.50 (59.75%)
stream	618,854 (16.42%)	76.02 (3.45%)
Other	2,346,498 (62.23%)	810.31 (36.80%)



(a) Gateway-23 (One-year old). (b) Network-23 (Six-month old). (c) Gateway-23 (One-month old). (d) DHT-23 (Zero-day old).

Fig. 4. Downloading progress across different ages of datasets.

4 ANALYSIS: WHAT IS STORED IN IPFS?

This section aims to answer the following questions: what is stored on IPFS and who is providing these files. To do so, we retrieve the files from IPFS. This enables us to also study the file accessibility on IPFS.

4.1 File Accessibility

Based on the valid 4,300,909 CIDs we extracted from the Gateway-22 dataset, we retrieved the corresponding files for these CIDs in parallel using 22 AWS VMs starting from April 15th, 2023. Figure 4a shows the retrieval progress that indicates how many files we have successfully downloaded over a 1-week period.

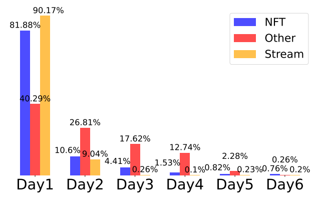


Fig. 5. Retrieval progress of NFT and Non-NFT (Gateway-22)

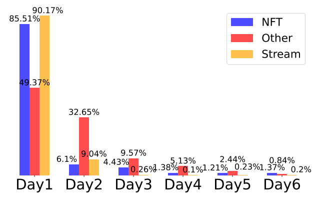


Fig. 6. Retrieval progress of NFT and Non-NFT (Network-22)

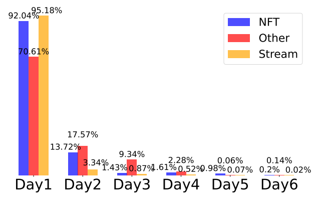


Fig. 7. Retrieval progress of NFT and Non-NFT (Gateway-23)

We started the retrieval process with 22 VMs with the aim of downloading all of the CIDs within one day. As previous studies [54, 72] indicate that the file retrieval time in IPFS is often less than 5 seconds, we set up a time threshold of 60 seconds for the first retrieval attempt. This threshold means if the content provider is not discovered in 60 seconds, the retrieval process will be terminated. However, on Day one, only about 49.86% of the files could be accessed and were successfully downloaded within the 60-second threshold. So for the CIDs we failed to retrieve on Day one, we repeated the process on the second day, continuing until Day six. After Day six, we extended the retrieval threshold to 24 hours for all unsuccessful attempts, but even then, we encountered further retrieval failures. Thus, we consider those files no longer available in IPFS[2].

As shown in Figure 4a, on Day one and Day two combined we successfully retrieved 93.96% of the total CIDs. Subsequently, we continued our file retrieval in another four attempts and achieved a 99.72% success rate, with 12,101 CIDs remaining inaccessible after six days.

Intuitively, file accessibility in a decentralized storage system such as IPFS is related to the age and access recency of files. Especially considering Gateway-22 is one-year old since it was created, one might think it is natural that certain files in Gateway-22 may no longer be hosted in IPFS. To validate if the file accessibility may be better if they have been accessed more recently, we managed to get access to or directly collected the other three “younger” datasets and performed a similar process on them. Note that the “age” of these CIDs is not their true age since they were created. Instead, it reflects the time duration between when the CIDs were found in the dataset and the time when we performed the retrieval using these CIDs.

These new datasets are as follows: 1. **Network-23:** It is a *six-month* old network dataset of 2 weeks long, covering the data from May 14, 2023 to May 28, 2023, shared by the same author [30]. Similar to how we processed Network-22, we sampled Network-23 and obtained a total of 3,862,785 unique and valid CIDs. 2. **Gateway-23**: It is a *one-month* old dataset, containing a one-day gateway

[2]For those retrievals that failed after six days, we also tried retrieving them from the gateway, considering the possibility of those files being cached at the gateway. However, we were unable to retrieve those files from the gateway either.

Table 4. Number of reachable peers.

Status	Network-22	DHT-23
Reachable	817	6,145
Non-reachable	16,910	14,232

log from the study [72] on March 15, 2023 UTC+0. We followed the same method as Gateway-22 to process Gateway-23 and obtained a total of 2,339,954 unique and valid CIDs. 3. **DHT-23**: It is a *zero-day* old dataset, which was immediately collected using a DHT crawler from November 1 to December 31, 2023. This dataset ensures that as long as we obtain a CID, we try to retrieve it immediately with a threshold of 60 seconds. It is worth noting that a CID appearing on the DHT is supposed to be up-to-date as a provider has to republish its records periodically (every 12 hours); this is why the "age" of this dataset is zero day. Our DHT crawler captured incoming Bitswap requests at our deployed nodes within the IPFS network and subsequently verified the presence of the requested CIDs in the DHT. The collection process lasted two months and we obtained 1.8 million CIDs in total. If a CID could not be instantly accessed, we performed another retrieval in the next day and similar operations afterward. During the operation, we ensure each CID in these datasets was unique.

Figure 4b presents the result for the six-month old dataset (Network-23). Figure 4b shows that there are still 180,392 CIDs unavailable after six days. Figure 4c presents the result of the one-month old dataset (Gateway-23). Compared to the results shown in Figure 4a, both of which are gateway logs, Figure 4c shows that a slightly larger percentage of files can be retrieved in the first attempt. Also similar as before, there is still a number of files (9,482 in total) that can no longer be retrieved. Figure 4d presents the result of the zero-day dataset DHT-23. It shows a similar result that nearly 4% of CIDs are not instantly accessible and 0.72% of CIDs are no longer considered as available. Note that even in DHT-23, the CIDs we extracted from the latest provider records in DHT are not necessarily “younger” than the CIDs we extracted from the other datasets. However, collectively, our results show a similar pattern: a non-trivial number of files can not be instantly accessed. It is also worth noting that these numbers are still estimate as they may be affected by the behaviors of the providers as well. For instance, a live streaming app may choose to make its objects available for only a short period for some reasons.

While IPFS is not inherently designed for persistent file storage, modern web applications like internet streaming and NFT transactions demand instant access for optimal user experiences. This is mostly achievable under current Web 2.0. To understand how IPFS performs compared to the current practice, we further divide the files into two types, NFT and video stream, as NFT platforms like OpenSea [3] commonly use IPFS as a storage solution. Figure 5, Figure 6, and Figure 7 show the accessibility of NFT, video stream files, and other files during the process of the above six-day retrievals, which are conducted on Gateway-22, Network-22 and Gateway-23, respectively. Compared to other files, NFTs do enjoy a much better accessibility[3], but a non-trivial percentage of NFT files are still not instantly accessible, potentially blocking the NFT transactions. Similar results are also shown for video streaming files: multiple attempts may be needed to access the streaming files, which can interrupt the process of video streaming.

As a comparison, Figure 8 further shows the accessibility of our current web. In August 2023, we accessed the top 1 million websites ranked by Alexa [18] in February 2023 to see if these websites were instantly accessible after six months. As shown in Figure 8, even the top 1 million websites offer better accessibility than IPFS for NFTs.

These findings show that while the majority of the files requested in March 2022 can still be retrieved in April 2023, many files are not instantly available and accessible upon request and demand a second

[3]Our following analysis will explain the reason.

or a third try in subsequent days. This is intuitive because files are stored in peers, while peers in IPFS may come and go from time to time. However, given that multiple copies of the same file may exist, one may expect that the overall availability of the files should be comparable to that of using a traditional (centralized) storage service [19, 21, 24]. This comparison suggests that for applications using IPFS for storage, there is a need to address the issue of accessibility. This responsibility could lie either with the applications or with IPFS itself, especially if the goal is to achieve performance on par with current Web 2.0 standards.

On the other hand, in IPFS, file accessibility is directly impacted by peer accessibility. Compared to traditional storage services, peers in IPFS may come and go from time to time, leading to the phenomena we have observed above. While this is expected, peers' departure pattern affects file accessibility: the sooner peers leave the system, the poorer the file accessibility. To get an idea on how quickly peers may leave in IPFS, we extracted unique peerIDs from Network-22 and tried to connect to these peers. Table 4 shows that over 90% of peers are no longer reachable in Network-22.

We did the same experiment with our zero-day dataset, where as long as we obtain a Bitswap request, we immediately try to connect to the peerID of the sender. As shown in Table 4, about 70% of peers are no longer reachable within several minutes. Although this shows a relatively higher level of peer accessibility, the fact that a majority (70%) of peers were unavailable within a couple of minutes raises concerns regarding overall file accessibility in general. *While our experiments only touched a subset of peers, and these numbers may fluctuate from time to time, we do believe that the poor peer accessibility may have influenced the observed, mixed file accessibility results. Although current IPFS does provide some optional functionality such as pinning [8] to improve general accessibility, it still demands further improvement on file accessibility.*

Takeaway #1: *Through the file accessibility measurement and analysis, we find that nearly half of the requested files are not instantly available. Given the inherent peer churn in P2P systems, IPFS is not designed to provide persistent file accessibility guarantees. However, this can hamper the success of applications running on IPFS. For applications requiring reliable storage, such as NFT transactions and streaming services, storage persistence and accessibility become crucial. That is, either the application developers need to address this accessibility issue or the IPFS needs to include some mechanism to provide better accessibility. In other words, IPFS may not be the best choice for some applications [42, 75] that have a high and stringent requirement on file accessibility. On the other hand, this also calls for more enhancement of file accessibility in IPFS in order to replace our current web.*

4.2 File Types

After retrieving the files, we were able to examine the file types. To do so, we used a combination of different categorization methods, such as file name (including directory name) extension, the magic number of the file, the encoding, the keyword, and manual inspection. For example, to determine if a file is `JSON` file, we checked if it follows a certain pattern that objects are surrounded by curly brackets `{.}` and the data is organized as key-value pairs. `normal` refers to *pdf, txt, doc, and other commonly-used files.* `video/stream` refers to video or audio files or chunks that are often used for video and streaming, including .mp4, .mp3 .m3u8, .ts, etc.

Identifying NFT files is challenging as a file of any type, ranging from a figure, a video, or a recording, can be an NFT and an NFT is not labeled based on a category. Therefore, we use a combination of different methods. The first is to examine the File Pattern. When an NFT is downloaded, its metadata (a `JSON` file indicating its token ID and other attributes) and the digital files are often downloaded together. Thus if the name of the digital file is aligned with the token ID indicated by the metadata within the same CID, we regard it as an NTF file. The second is to find ground truth values of NFT files by NFT metadata crawling in order to obtain the URL field (pointing to the CID, if any) of an NFT digital artwork. This crawl is done by (1) utilizing the API of a marketplace to access the NFT assets (unfortunately, it usually has a rate limit, for example, 1K per day for OpenSea), (2) accessing a marketplace by web scraping, and (3) collecting on-chain data of Ethereum by Etherscan [23]. Note that it is highly unlikely to obtain all the CIDs of the NFTs stored in IPFS by this method. By the end of January 2024, we managed to obtain more than 5.67 million ground truth CIDs. Thirdly, we directly analyze the content to identify a figure as an NFT. It is worth noting that intuitively most NFTs are selfies of a digital character. As such, we can determine if a file is NFT by training a Convolutional Neural Network (CNN) classifier. We use the Resnet101 as the model. The image pixel values are normalized and scaled to values between -1 to 1 and are grouped into batches of 128. We use the cross-entropy loss function. We fed the classifier 640,000 images as the training data with 80% training set and 20% test set (we collected the images by crawling the NFT marketplaces in step two, and manually filtered out those character selfies). We finally obtained an accuracy rate of 91.83% on the test set.

Table 5 lists the breakdown of files by types. We observe that the most popular type is `JSON` file. After looking into the content of the `JSON` files, we confirm that they correspond to the NFT [74] metadata by examining the keyword. Here keyword refers to metadata standards such as ERC-721 [15]. If a metadata follows such a standard, it will have certain keys including `like`, `name`, `description` and `external_url`, etc.

NFT is an ownership certificate that is generated from smart contracts, originally by Ethereum [33]. NFTs can certify the ownership of digital or physical assets, such as artworks, images, music, video, game props, etc. NFTs have gained immense popularity on the Internet since 2019 and have found applications in various contexts. For example, DappRadar reported that NFT games formed nearly 51% of the blockchain industry usage in August 2022 [46]. In 2021, IPFS announced that OpenSea [3], the largest NFT market platform, stores NFTs in IPFS and FileCoin [11]. Storing NFTs in IPFS makes NFT objects resilient to data loss and attacks such as metadata tampering [73].

For an NFT, it comprises a digital asset (e.g., an image) and its corresponding metadata (e.g., a `JSON` file). The first two rows in Table 5 indicate that our retrieved files are indeed NFT dominant. Compared to the actual digital assets, the `JSON` files are of small sizes. Together, they are more than 40% in terms of file counts.

Table 5 also shows that video files, while only occupying a small portion in terms of file counts, take more than 30% of all the files in terms of size. This is reasonable as video files are usually larger, compared to other files.

Table 5. File types in Gateway-22 (by file).

File type	Count (%)	Total size in GB (%)
JSON	3,985,956 (40.24%)	7.02 (0.09%)
image	3,446,190 (34.79%)	4,866.08 (58.4%)
unknown	563,487 (5.69%)	590.33 (7.09%)
video/stream	1,367,262 (13.8%)	2,798.55 (33.59%)
normal	517,400 (5.23%)	63.72 (0.76%)
HTML	24,225 (0.25%)	6.12 (0.07%)

Table 6. File types in Gateway-22 (by CID).

Category	Count (%)	Total size in GB (%)
NFT	1,812,071 (44.51%)	5921.84 (71.21%)
media	1,236,408 (30.37%)	1740.8 (20.93%)
normal	486,988 (11.97%)	65.38 (0.79%)
unknown	535,469 (13.15%)	588.15 (7.07%)

Table 7. File types in Network-22 (by file).

Category	Count (%)	Total size in GB (%)
text/HTML	1,455,419 (26.68%)	43.25 (1.98%)
JSON	1,642,708 (30.11%)	3.40 (0.16%)
image	1,120,051 (20.53%)	949.62 (43.38%)
video/stream	535,469 (9.81%)	547.6 (25.01%)
unknown	534,740 (9.62%)	330.4 (15.09%)
normal	177,325 (3.25%)	314.88 (14.38%)

Table 8. File types in Network-22 (by CID).

Category	Count (%)	Total size in GB (%)
NFT	907,171 (24.06%)	1,444.23 (65.59%)
media	771,126 (20.45%)	82.30 (3.74%)
normal	1,610,237 (42.70%)	360.16 (16.36%)
unknown	481,982 (12.79%)	315.17 (14.31%)

While Table 5 summarizes the file breakdown based on individual files, in IPFS, a CID may correspond to a directory, which sometimes contains up to 10 thousand files. We find that such a directory often corresponds to a bunch of NFT metadata that belongs to a certain NFT collection [13], where the digital asset could be images, videos, or normal files. Therefore, we further categorized the files or directories based on CIDs, as shown in Table 6. For example, if a majority of files contained in a directory are NFT metadata and corresponding images, we consider that CID represents an NFT (or an NFT collection). Thus, some audio and video files are classified into NFTs. The multimedia files that do not belong to any NFT or NFT collections are referred to as `media` in the table. Table 6 confirms that the dominant applications using IPFS are NFT and media, mainly videos.

To see if this trend is also true from Network-22, we conducted a similar analysis. However, unlike files retrieved in Gateway-22, in the network trace, a large portion of CIDs correspond to just a chunk of a file. Analyzing the types of file chunks is challenging. Thus we first filtered the CIDs to obtain the root nodes in the Merkle tree that represent the complete file. Accordingly, we sampled 3,906,560 and successfully extracted such 3,770,518 CIDs from Network-22. Table 7 shows the file type result. As we can observe in this table, there are more requests to HTML files in Network-22 than that in Gateway-22. By looking into these HTML files, we found that many of them are websites, e.g., personal websites.

Besides more text/HTML files accessed via the network, Table 7 shows that NFT and video still occupy a significant portion in terms of both file counts and the total file size. Table 8 shows the corresponding results based on CIDs. This is consistent with what we have observed from the gateway trace, shown in Table 5 and Table 6.

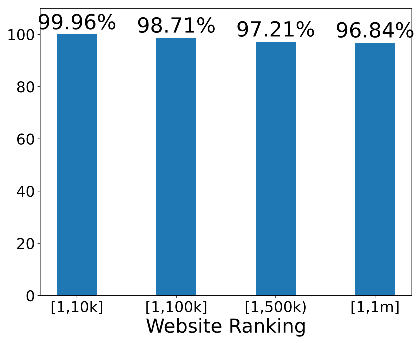


Fig. 8. The accessibility of top websites (Alexa website ranking in Feb 2023).

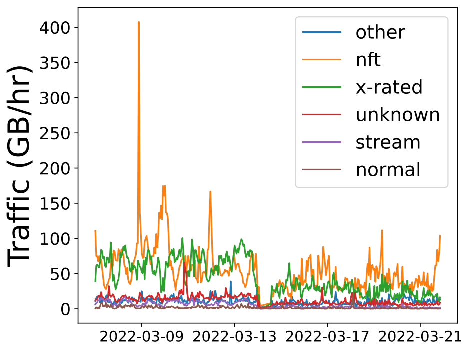


Fig. 9. The hourly traffic rate of file types in a period of two weeks.

To further assess the file types associated with a particular CID, we also analyzed the `http_refer` field of each request in Gateway-22, if it was present, to determine the service from which the request was originated. To determine what kind of website it is, we started by collecting website content in the NFT-related category to train a naive Bayes classifier so that we can determine if a website is indeed NFT-related. Our classifier takes in the NFT-related keywords (wallet, NFT, collections, etc.) as features and outputs the category of a website based on the occurrence of these words. After successfully filtering out NFT-related websites, we assigned categories to each of them manually. Websites that were not reachable were labeled as unknown. In Gateway-22, 20.48% requests came with a valid `http_refer` field. Table A1 (in Appendix) shows the top 10 websites with most requests. We observe that the top 10 websites are either NFT-related or online streaming videos. By checking the web pages and content of these video websites, we find that there are four out of the top 10 websites featuring x-rated movies. To identify the x-rated movies and pirate videos, we first match the CID to the blacklist [22] and then use keywords to check the web pages and the content of these video websites. We observe that 680, 745 (16.72%) CIDs are related to

Table 9. Video file categories of Gateway-22.

Category	Count (%)	Total size in GB (%)
x-rated	680,745 (55.06%)	860.59 (49.43%)
normal	555,663 (44.94%)	880.21 (50.57%)

x-rated videos and pirated movies. As shown in Table 9, the x-rated video files and pirated movies account for over 50% of the video files. For the remaining video files, we do not know their content.

The file sizes and the number of files in each category are static information. An NFT or a video file may be requested multiple times. While NFT and video files are dominant in terms of file counts and sizes, they may not dominate in terms of generated traffic. To investigate this, we further checked the hourly traffic rate for each type of files over the 2-week period in Gateway-22. As shown in Figure 9, while the NFT traffic rate was clearly leading the traffic, video traffic was the second largest. *This raises concerns regarding the misuse of IPFS as a non-trivial portion of such video traffic involves x-rated videos and pirated movies.*

Takeaway #2: *Our findings indicate that IPFS is mainly used for NFT storage, which accounts for about half of the total file sizes in Gateway-22.* This observation aligns with the fact that the largest NFT marketplace, OpenSea, uses IPFS for storage. Given the file immutability supported by IPFS, NFT files can enjoy this property and such files on IPFS would be less susceptible to loss or attacks [73]. Surprisingly, over 50% of video-related CIDs are associated with either unlawful or x-rated content. This observation raises an important concern that current IPFS lacks mechanisms to scrutinize malicious or unlawful content. Even though employing such a mechanism could be hard and against the decentralized nature of Web3, the prevalence of unlawful content underscores a critical need for balancing its design goals and responsible content management.

4.3 Content Providers

The previous section shows *what* type of files are stored in IPFS. Next, we seek to understand *who* are requesting and *who* are providing the content. Since IPFS is P2P based, any node that has a copy of the requested file may share the file, and thus could be a content provider. Since Gateway-22 contains client location, we start with analyzing the geographical distribution of these clients. Table 10 shows the geographical distribution of clients in the Gateway trace. We see that over 95% of the requests to the gateway originated from Asia and North America, and a high concentration of requests were from the United States and China/Hong Kong, together accounting for about 90.99% of the total requests. Table 11 shows the client (we count client per file request instead of CID) geographical distribution of Network-22. It shows a similar trend of centralization: over 85% of the requests are from Europe and North America. Unlike Gateway-22, the requests from Asia drop dramatically. A possible reason is that the accesses from Asia are mainly from third-party websites via gateways and there lacks direct participation within the network, as we found that many third-party video and NFT transaction websites in Gateway-22 are located in Asia.

Table 10. Client geo distribution of Gateway-22.

Country/Region	Count	Percentage
US	32,727,770	46.69%
CN	25,823,483	36.83%
HK	5,229,562	7.47%
JP	2,049,326	2.92%
CA	1,772,970	2.53%
Others	3,527,792	3.56%

Table 11. Client geo distribution of Network-22.

Country/Region	Count	Percentage
US	6,823,051	54.32%
DE	1,635,297	13.02%
CA	1,492,477	11.89%
NL	925,147	7.37%
SG	684,018	5.45%
Others	995,170	7.94%

The high concentration of requests from a small number of regions motivates us to further investigate the situation of content providers. Since we retrieve IPFS files based on the extracted

Table 12. Categories of content provider IPs Gateway-22.

Category	Count (unique files served)
Storage nodes	383 (20,718,126)
Cloud nodes	2792 (13,903,521)
NAT nodes	2602 (839,040)
Other	655 (261,233)

Table 13. Categories of content provider IPs Network-22.

Category	Count (unique files served)
Storage nodes	302 (8,110,306)
Cloud nodes	2613 (6,187,712)
NAT nodes	2419 (666,777)
Other	387 (304,044)

CIDs, we are able to find out the content providers for all these CIDs from the log of the Bitswap messages and the corresponding IP [4] by Wireshark/tcpdump or DHT query. In contrast to previous studies [31, 39] that analyze content providers using DHT provider records only, our method involves directly retrieving these CIDs. This approach offers several benefits. First, it avoids the issue of potentially stale DHT records, as direct retrieval can ensure the data is currently held by the provider. Second, it provides more accurate IP address information of the provider. Merely querying the DHT might not always yield the provider's IP, especially in cases involving NAT (network address translation) nodes. Third, direct retrieval inherently involves DHT queries. In case the provider of some CID may not appear in the DHT, which is possible if the record was deleted previously and it is not updated by the provider yet, the provider can be found via Bitswap. Accordingly, we collected these content providers' IPs and hostnames. For those CIDs unreachable over a one-week period, we consider them inaccessible.

Table 12 summarizes the types of content providers encountered and the number of unique file chunks (based on CIDs) served by each type of providers. A storage node is, for instance, an NFT storage server. It is a storage service allowing users to upload their off-chain NFTs to make them accessible to the public [17]. These storage nodes are often hosted in cloud data centers. The cloud nodes are those owned by tech companies such as AWS, Cloudflare, and Google. NAT nodes are those who have no public IP address or only relay address and are not reachable by the DHT.

We make the following interesting observations from Table 12. First, while there was a significant number of NAT (without public address) providers during our measurement, they served only a very small portion of files. Second, on the contrary, the majority (> 95%) of files were served from storage nodes and cloud servers, where the storage nodes alone served over 65% files. Since the NAT nodes are more likely belonging to ordinary users while cloud and storage nodes are more likely belonging to influential users or companies, it indicates a trend that the current IPFS also heavily relies the cloud services.

Moreover, we find that the top-50 content providers are either storage nodes or cloud servers and they serve 95% of files. Table A2 lists the first 20 among them. These top-20 providers served the same amount of files as the remaining providers combined, and about 4 million CIDs were retrieved from about 6,000 providers as shown in Table 12. This finding not only verifies previous studies [31] that most CIDs are held in cloud servers, but also further confirms that users are actually accessing these cloud servers most of the time.

We conducted a similar analysis with Network-22. The IP categories are listed in Table 13 and the top-20 providers are listed in Table A3 (in Appendix) . Table 13 shows a similar trend to Table 12: over 90% of the files are from cloud or storage nodes while the number of those nodes is quite small with respect to the network size. Interestingly, we find that there are 11 providers that are overlapped in the top-20 list between Gateway-22 and Network-22 (as highlighted in blue-colored text in Table A2 and Table A3 in Appendix) and there are 35 providers that overlap in top-50 list. *This suggests that, compared to the tens of millions of unique peers in IPFS [72], the content providers,*

[4]We utilize IPinfo and MaxMind for mapping IPs to physical locations, with both services yielding consistent results. We only map IPs to country level where the geolocation databases can achieve 95% accuracy [61].

however, are highly concentrated on cloud servers or storage nodes maintained by specific services. These results are counter-intuitive given the design goal of IPFS. Our measurement and analysis so far hint at the following reasons that may attribute to the high concentration trend we observed.

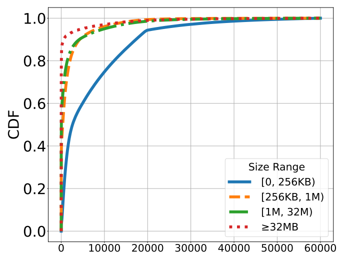
- **Insufficient network participation from ordinary users.** As illustrated in Table 10 and Table 11, there is a noticeable lack of participation from certain global regions both within the network and gateway. For example, among the requests from Asia, the requests made from the network are significantly lower than the requests made by gateway. Meanwhile, 92.11% of requests of Network-22 are made by cloud nodes and storage nodes. These indicate that ordinary users are used to traditional Web access model to access the content instead of establishing an active IPFS node.
- **Limited and skewed types of hosted content.** Compared to the various content hosted on our current Web, Section 4.2 highlights that the content on IPFS is dominated by NFT files. Given that Pinata, an NFT distribution platform, and Opensea, an NFT marketplace, have experienced remarkable growth and attracted millions of users, users tend to rely on those services to store their NFTs. Therefore, it is not surprising that NFT storage nodes contribute a significant portion of the files on the network. This trend, however, leads to a concentration of providers in the network.
- **Limited NAT support and adoption.** Initially, nodes behind NAT in the early versions of IPFS were restricted to client roles, unable to serve as servers or content providers. Despite the introduction of autoNAT [6, 72], enabling nodes behind the network to be accessible and serve as content providers, its adoption rate among peers remains unclear. The lack of widespread adoption of autoNAT could be a contributing factor to the limited number of active content-serving nodes, further exacerbating the network's concentration issue.

However, from a user's perspective, a centralized service is more likely to be trusted and less susceptible to peer churning, thus becoming a better candidate for serving content to others. *In absence of precise information about why, the fact that a majority of files were served from dedicated nodes in clouds may raise controversies and security concerns, indicating a big gap between the intended design goal of IPFS and the existing reality.*

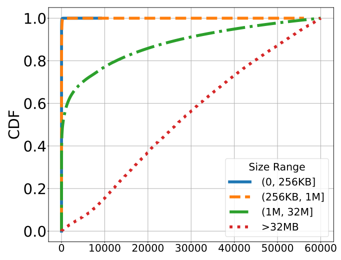
Takeaway #3: *We observe a high concentration of content providers. Over half of the providers are concentrated on dedicated storage nodes or cloud servers, and these servers store a substantial amount of files in IPFS. This essentially means that accessing IPFS is, in reality, accessing a small number of concentrated storage or cloud servers, which resembles more traditional client-server based web accesses. This observation somewhat contradicts IPFS' goal of decentralization and clearly indicates a big gap between the original design objective and the current practice. Without bridging this gap, currently many Web3 applications are observed to be not fully decentralized [52, 59, 63]. In the case of IPFS, the reason for this gap can be attributed to limited ordinary user participation and a lack of foundational infrastructure support.*

5 PERFORMANCE ANALYSIS

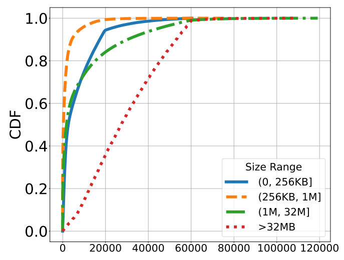
During the file retrieval from IPFS, we timestamped the end-to-end latency of the process, enabling us to analyze the performance of file retrieval on a large scale. This is different from prior studies that often use microbenchmark workloads to test the file access performance [68, 72], where the content providers are designed or deployed by the authors in certain locations. Statistically, we hope our performance analysis can provide a more realistic and representative view of IPFS when considering file access. Our following analysis will mainly focus on the 4+ million files retrieved based on the CIDs extracted from Gateway-22 unless noted otherwise.



(a) Lookup time (ms).



(b) Downloading time (ms).



(c) End-to-end retrieval time (ms).

Fig. 10. CDF of the lookup time, downloading time, and end-to-end retrieval time.

5.1 File Size Distribution

Based on the CIDs extracted from Gateway-22, we retrieved 4,070,936 objects in total, with a size of 8.12 TB. Figure 11 presents the file size distribution. As we can observe from this figure, the majority of files (70.50%) are above 0.25 MB, and more than 99% of them are smaller than 32 MB, indicating that small files are dominant in the IPFS network.

The size of the files also shows two clusters: around 1 KB (7.15%) and 1 MB (31.12%). By looking into the files, we found that the 1 KB cluster mainly contains the JSON files. They are mainly the metadata for NFTs. On the other hand, the 1 MB cluster primarily includes video chunks. These are consistent with our findings that NFT and video are the two leading applications using IPFS.

5.2 Content Retrieval Performance

End-to-End Latency. As we discussed before, retrieving a file associated with a CID consists of two steps: lookup and download the content. Here we define the lookup time as the time spent on identifying the root CID block. Figure 10c shows the end-to-end latency to retrieve the CIDs. As can be seen in this figure, the median time to retrieve a CID is 4,383 ms for files in the size range of (0,256KB], 1,465 ms for files in the size range of (256KB,1M], 3,595 ms for files in the size range of (1M,32M], 25,586 ms for files in the size range of ≥32M, respectively. It is surprising that retrieving a file in the size range of (0,256KB] on average is much slower than retrieving a file in the size range of (256KB,1M].

To figure out the potential reasons, Figures 10a and 10b show the end-to-end latency breakdown for the lookup phase and the downloading phase. As shown in Figure 10b, the downloading time for files up to 1MB is fast, and 99% of files that fall into this size range can be downloaded within 10 ms. On the other hand, the lookup time, as shown in Figure 10a, dominates the end-to-end retrieval time for small files. This can be observed for files in the aggregate size range of (0,1M]. Our conjecture is that since the files are often provided by storage nodes and cloud servers in data centers, the downloading operation is fast. On the other hand, the lookup operation often involves routing via ordinary peers that may not always be available, resulting in multiple rounds and thus longer time.

Lookup Time Analysis. The analysis of the end-to-end retrieval latency further motivates us to take a closer look at the lookup time. To do so, we break the file size into finer granularity and Figure 12 shows the result. As shown, smaller files experience clearly longer lookup time, while larger files tend to have a shorter lookup time. For example, the average lookup time of large files with sizes >256KB is 1644.58 ms while the average lookup time of smaller files ≤256KB is 6632.49 ms. Figure 12 further shows that the lookup time varies significantly across different file sizes, ranging from 8 ms to more than 5.4 seconds in terms of median value. In general, the lookup time for smaller files tends to be much larger than that for larger files. Given that file sizes intuitively should not affect lookup time, it motivates us to look into the file types of the small files and large files, and their corresponding providers. Figure 14 shows the ratio of the file types in different file size intervals, and Figure 15 shows the provider types in different file size intervals. As shown in Figure 15, a higher percentage of the providers of the small files ($\leq 256KB$) are from NAT nodes while a higher percentage of the providers of large files ($> 256KB$) are from cloud nodes or storage nodes. We further show the provider distribution of different file types in Figure 16, which demonstrates that NFT files are more likely to be stored on cloud nodes and storage nodes. Meanwhile, Figure 17 shows the lookup time across different provider types. The lookup time of cloud nodes and storage nodes has an average of 285.87 ms compared to 16367.23 ms of NAT nodes. That is, large files contain a higher percentage of NFT-related files, which are stored on cloud nodes or storage nodes and thus have a relatively lower average lookup time. In contrast, small files contain more unknown files, web HTML files, and ordinary files (which we denote as normal in Section 4.2), and those files have a higher portion of providers from NAT nodes. Therefore, those small files experience a higher average lookup time.

The variation in lookup time motivates us to further look into why the NAT nodes offer worse lookup performance. In general, a lookup operation in IPFS consists of two phases: (1) peer resolving and (2) peer routing. In the peer resolving phase, the requestor first sends Bitswap messages to all its neighbors, asking for desired chunks with a cap of 1 second. If the Bitswap times out and/or fails to retrieve the content, then the requestor triggers a DHT walk to resolve the provider's nodeID. In the peer routing phase, the requestor will launch another DHT walk to get the multi-address of the provider. Upon resolving the multi-address, the requestor will connect to the provider.

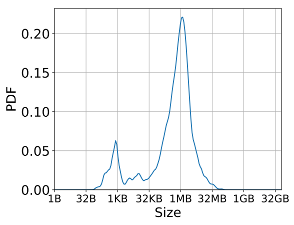


Fig. 11. File size distribution (PDF). The X-axis is in log-scale.

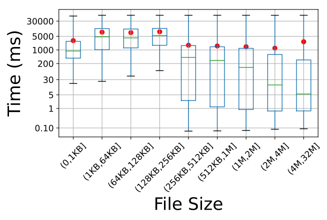


Fig. 12. The lookup time. The Y-axis is in log-scale.

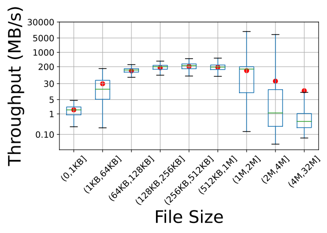


Fig. 13. The downloading throughput. The Y-axis is in log-scale.

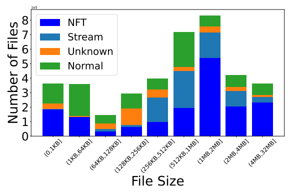


Fig. 14. Type distribution based on file size.

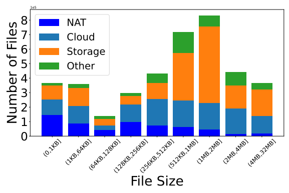


Fig. 15. Provider distribution based on file size.

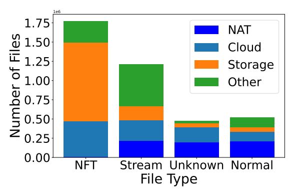


Fig. 16. Provider distribution based on file type.

To investigate the difference in terms of lookup time for different provider types, we randomly sampled 400K CIDs from each group and retrieved them using two AWS instances. Upon resolving the providers, we recorded the time spent in each stage and labeled them with cloud nodes or NAT

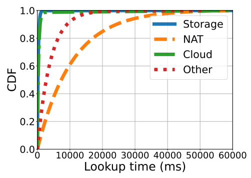


Fig. 17. Lookup time based on file type.

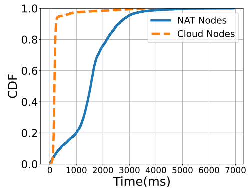


Fig. 18. The time of peer resolving.

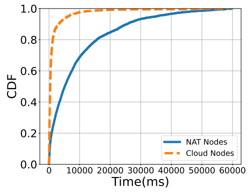


Fig. 19. The time of peer routing.

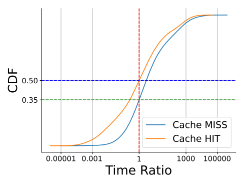


Fig. 20. The ratio of average request duration via gateways to that via the network for all bundled CID accesses.

nodes. Figure 18 illustrates the combined overhead of Bitswap and the initial DHT walk. Once the Bitswap manages to find the provider, the DHT walk will not be needed. Figure 19 shows the overhead of phase 2, resolving the multi-address and connecting to the provider. Note that the requestor queries the DHT only if the Bitswap fails.

Figure 18 shows that the NAT node group incurs a higher cost in the first phase, with over 80% of these lookups exceeding 1 second. In contrast, the cloud node group has more than 97% of lookups taking less than 1 second, indicating that the NAT node group has a higher likelihood of failure possibility with Bitswap. Additionally, the NAT node group suffers from a higher average connection time, as shown in Figure 19, exacerbating the problem.

The above analysis indicates that the difference in lookup time between providers and the file distributions contribute to the difference between the lookup time of small files and large files. The fact that lookup performance varies between cloud nodes and NAT nodes is not surprising, especially considering our analysis in Section 4.3 that top-50 providers account for over 95% of files and they are all cloud nodes or storage nodes. In summary, the variation of lookup time is possibly due to the following reasons.

- **Cloud nodes have a better chance to be connected to.** Cloud nodes host a large portion of content across the network, making them more likely to be connected by other nodes. Meanwhile, when forwarding Bitswap messages, a node prioritizes other nodes it is connected to. These two factors make the cloud nodes more easily detected in the Bitswap discovery stage, leading to a lower lookup time.
- **NAT nodes experience a higher routing time.** NAT nodes in the IPFS network either use non-NAT nodes as a reverse proxy or get connected through a relay (NAT hole-punching). In either case, lookup involving NAT nodes experiences higher routing delays.

The above analysis indicates that the lookup time is dependent on the accessibility of the content provider. If the provider has bad connections or is less involved in the network, the likelihood of having a successful Bitswap decreases. To address this issue, intuitively, we can explore by (1) parallelizing the Bitswap and DHT walk process when resolving the provider; (2) employing more dedicated nodes to assist lookup.

As implementing these ideas requires modification of the original protocol, we did not conduct experiments in IPFS to evaluate their effectiveness. We note that this is purely based on our experiments on file access performance. The applications running on IPFS (e.g., streaming) may have other considerations that need to be taken into account as well. Employing more dedicated nodes can also be used to improve file accessibility, which is another concern identified via our measurement (Section 4.3).

Throughput Analysis. Figure 13 plots the throughput distribution across different file size ranges collected from our retrievals. As demonstrated, the downloading throughput increases along with the file sizes up to 1 MB. Then we observe a decline in average throughput and a significant increase in variation. This fluctuation can be attributed to two facts: (1) the downloading process of IPFS needs to recursively traverse the Merkle tree structure in order to resolve dependencies: the file chunk residing in the lower level in the Merkle tree has to wait to be retrieved until its ancestors are fetched. As the file size increases, the structure of the Merkle tree becomes more complicated in terms of both depth and width, making the downloading process less paralleled; (2) during downloading, some chunks may be downloaded faster than others because the slowest connection will be the bottleneck that affects the overall throughput. The larger the file, the more chunks to download, and the more likely to have bottleneck connections that lead to a smaller throughput on average.

Comparing Gateway Access with Network Access. The above analysis is based on our file retrieval process via the network. IPFS provides an alternative for data access via gateways. For the gateway access, we can obtain a metric called `request_time`, which refers to the total request duration. Note that a CID might correspond to multiple requests, and a request could hit the cache as IPFS gateways employ content caching to speed up file access. We refer to all requests linked to a single CID as a bundled CID access. Thus, in Gateway-22, for each bundled CID access, we extracted the average request duration with a cache hit or a cache miss, respectively. Then we compute the ratios of the average request duration obtained via the gateway (extracted from Gateway-22) to the corresponding duration via the network during our retrieval process for all CIDs.

Figure 20 presents the CDF of the ratios. We observe that accessing via the gateway does not necessarily improve the overall access performance even with the presence of a content cache at the gateway side: for almost half of the bundled CID access, retrieval via the network offers a better performance. We acknowledge that comparing gateway accesses from 2022 and network accesses from 2023, a year later, might not provide a fair comparison, given that the IPFS network might have undergone numerous updates during this time. Nonetheless, the results of the relative ratios underscore the significant variation in file access performance, whether accessed via gateways or the IPFS network.

Takeaway #4: *The analysis of both lookup and downloading throughput reveals that accessing files in IPFS, whether through gateways or directly via the network, can result in significant performance variations. Cloud nodes, as content providers, offer better lookup performance compared to NAT nodes. As such, this disparity in performance results in smaller files having a higher lookup time on average, as they are less frequently hosted by cloud nodes. Retrievals of larger files exhibit higher throughput variation, potentially stemming from the current block fetching mechanism employed by Bitswap. These performance variations may cause quality of experience issues for future web applications relying on IPFS for storage, thus, calling for further research and improvements.*

5.3 The Impact of Chunk Size

The measurement results motivate us to explore potential improvements for IPFS. In the following, we evaluate how chunk sizes affect the I/O performance of IPFS, aiming to improve the downloading throughput in IPFS, especially for large files. In IPFS, the default chunk size is set at 256 KB. To investigate this, we conduct an experiment by downloading a 128 MB file with five different chunk sizes from a server. The experiment is done at 6 different locations across the globe. For each location, first, we have the requestor and the provider in the same region. We use AWS instances (t2.micro, 1vcpu, 1GB memory). For each chunk size, the downloading is repeated 1000 times.

Figure 21 shows the result conducted in Central EU. This figure indicates that, for a file of 128 MB, a 1MB chunk size is more favorable in terms of the downloading time when compared to the default settings of IPFS (a chunk size of 256KB). We also report the results of other regions

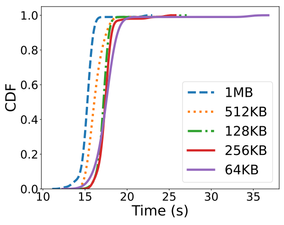


Fig. 21. The downloading time of a 128 MB file with various chunk sizes (Conducted in Central EU).

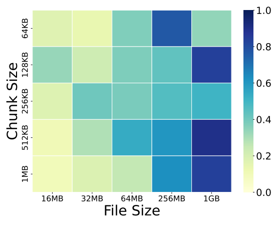


Fig. 22. Heatmap of downloading throughput with different file size and chunk size combinations (Conducted in Central EU).

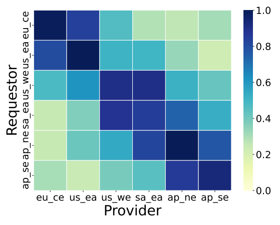


Fig. 23. Heatmap of downloading throughput with different combination of regions (1 GB file size and 1 MB chunk size)

as shown in Figure A1 (in Appendix A.3). Among all the regions, the average and 99^{th} percentile downloading time with a chunk size of 1MB are 17.30 seconds and 18.37 seconds, respectively, while they are 21.82 seconds and 26.94 seconds for the default chunk size of 256 KB. Additionally, the US and EU show superior downloading performance compared to other regions. Specifically, the average retrieval time in EU with a chunk size of 1 MB is 15.36 seconds while the average retrieval time in other regions with 1 MB chunk size is 17.69 seconds.

These experiments indicate that a uniform chunk size is sub-optimal for file downloading. By conducting more experiments with different chunk sizes and different file sizes, we seek to determine an optimal chunk size for large files (e.g., $>$ 16 MB) as their downloading time is more substantial.

To further investigate how the file size and chunk size impact the downloading throughput, we conduct a more comprehensive experiment across the globe. Figure 22 presents the result and illustrates the impact of chunk sizes on downloading throughput (i.e., file size over downloading time) in Central EU. The results of other regions are shown in Figure A2 (in Appendix A.3). The experiment was conducted on the same set of AWS instances as mentioned above. Each chunk size and file size combination was repeated 1000 times. Both requestor and provider were located in the same region. In Figure 22, the value in each box is the normalized average throughput. A darker color indicates a better retrieval performance. As can be seen in this figure, for large file sizes (1GB and 256MB for example), a larger chunk size can boost the downloading throughput. Meanwhile, we can also observe the downloading performance in US and EU outperforms other regions. We further conduct similar experiments by having the providers and requestors located in different regions, for each file size and chunk size combination. Figure 23 displays the result under 1 MB chunk size and 1 GB file size. The x-axis and y-axis represent the placement of provider and requestor, respectively. Other 24 results of different combinations of file sizes and chunk sizes are presented in Figure A3 (in Appendix A.3). Those results under different providers and requestors confirm our finding that a default chunk size of 256 KB may not necessarily be an optimal choice in terms of downloading performance and a larger chunk size suits large files better. Figure A4 (in Appendix A.3) further shows the downloading throughput variations during different times of the day. For this experiment, we periodically retrieve a 256MB file over 5 days using two chunk sizes: the default size of 256KB and 1MB. This figure indicates a better choice of chunk size has the potential to improve the retrieval performance.

The results suggest that the default chunk size of 256 KB is not necessarily the best option in terms of retrieval performance. Intuitively, based on our empirical results, we suggest that a dynamic chunk-sizing strategy is more desirable when downloading large files in order to accelerate the downloading process.

- For relatively small files sizing from [16M, 32M], a 128 KB chunk size is preferable;
- For medium-sized files sizing from [32M, 64M], a 256 KB chunk size is a better option;
- For large files (>64M), using 1 MB as the chunk size is recommended.

6 DISCUSSION

Limitation of Our Study. Measuring and characterizing a massive-scale, real-world P2P system deployed in production is challenging. The difficulty stems from the absence of centralized points that allows for one to obtain a global view of the system at any given time. Furthermore, in P2P systems, peers frequently join and exit at random, resulting in churn or temporary inaccessibility during the data collection phase. This makes it extremely difficult to take an accurate snapshot of the systems [71]. The characterization of IPFS is no exception given that it is a P2P-based file-sharing system. Prior studies on IPFS [39, 72] took advantage of the roles as key IPFS project code contributors and maintainers; by directly collecting data from IPFS gateways over time, they were able to provide a relatively comprehensive picture of the entire system.

Our study started from the CIDs extracted from the dataset shared by prior work [39, 72]. However, the CIDs we were able to extract only reflect the state of IPFS in 2022. We do not know what portion of the entire CIDs, files, or traffic they represented at that time, let alone today. As such, the findings reported from our study offer only a partial view of the system, and we provide additional thoughts as follows.

File Accessibility/Availability. Our findings on *file accessibility* come to a mixed outcome: while most files can still be retrieved from IPFS, a successful retrieval often requires multiple attempts over an extended period (up to a week). Given that IPFS is not designed to be a persistent file storage and IPFS peers have a high churn rate [72], this seems unsurprising. However, if this trend persists in the future, it may hinder the adoption of IPFS in new Web3 applications and also poses challenges for migrating current Web2.0 applications such as video streaming into IPFS where steady and reliable accessibility is crucial. Moreover, it appears that IPFS might be better suited as a backup store, rather than a primary storage for time-sensitive applications that have stringent availability requirements. Enabling an availability guarantee necessitates the design and implementation of a data replication model, or an SLA, which statistically ensures one copy of each file is contiguously available during a certain time unit (e.g., the duration of one month). This requires a better protocol for handling dynamic peer departure (for example, when a node is about to leave the network, it should delegate its content (and DHT records) to its neighbors.) or the approach of employing more dedicated storage nodes as discussed in Section 5.3.

File Types. Regarding our findings on *file types*, our study reveals that a majority of IPFS usage has been dedicated to NFTs, followed by a substantial portion of usage for video files. The fact that IPFS is being utilized for storing x-rated and pirated movies raises concerns. As a storage service, it should be agnostic to the content it stores. Yet, if this trend continues, IPFS could inadvertently become the stepstones for malicious activities (*e.g.,* malware storage [1] or the command and control channel for botnets [37]). Deterrence of such misuse requires careful deliberation and mechanisms. Designing such mechanisms that can not only prevent unlawful content but also preserve the decentralized nature of IPFS is challenging, which calls for further investigation.

High Concentration. Regarding our findings on a *high concentration* of content providers, where a small percentage of participating nodes, which are either cloud nodes or storage nodes, contributed the majority of the content, we find that this trend has dual aspects. On the one hand, it is in stark contrast to the design goal of IPFS, as the current practice resembles more the traditional client/server-based web and storage services. The concentration may be rooted in the fact that (1) some participating peers may be still using the old version of the software without proper support of autoNAT [51]; (2) low participation from ordinary users further exacerbates the gap between NAT nodes and cloud nodes; and (3) IPFS hosts a substantial amount of NFT files, leading to a more dominant role of cloud nodes and storage nodes. On the other hand, retrieving files from storage nodes or cloud servers offers a more stable and better user experience, but with a concentration of providers. Thus, a trade-off between them deserves further investigation. Furthermore, a proper incentive mechanism, such as the tit-for-tat method [38] used in BitTorrent, could effectively encourage normal peers to stay in the network for extended periods. This could lead to truly decentralized content providers and improve overall file availability.

7 CONCLUSION

IPFS has emerged as a pioneer distributed storage system for Web 3.0. Its early implementation has attracted lots of users and applications and garnered attention from the research community. While prior studies have characterized IPFS on its design and implementation, geographical participation, and file storage and retrieval performance, this paper has conducted a study aiming to gain a better understanding regarding what content is currently stored on IPFS, what applications are actively using the content served via IPFS, and who are providing the services. Our findings unveil several trends that demand more deliberation and mechanisms for improvements so that IPFS can realize its envisioned goals in the future. By no means our study offers a complete picture of IPFS, a system that continues to evolve actively. However, we hope the trends we have identified will provide valuable insights into the design, implementation, and optimization of a large, decentralized storage service in the era of the next-generation web.

ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers and our shepherd Stefan Schmid for their insightful comments. We would also like to thank Leonhard Balduf (researcher at the Technical University of Darmstadt and the Weizenbaum Institute) and Dennis Trautwein (research engineer at Protocol Labs) for generously providing some of the datasets used in this work. This work was supported in part by NSF grants CCF-1919113, CNS-2007153, CCF-2318628, and CNS-2322860.

REFERENCES

[1] 2015. Bitcoin's Blockchain Offers Safe Haven For Malware And Child Abuse, Warns Interpol. https://www.forbes.com/sites/thomasbrewster/2015/03/27/bitcoin-blockchain-pollution-a-criminal-opportunity/?sh=73cf7c43207b.
[2] 2017. Filecoin: A Decentralized Storage Network. https://filecoin.io/filecoin.pdf.
[3] 2018. OpenSea, the largest NFT Marketplace. https://opensea.io/.
[4] 2019. Bitswap. https://docs.ipfs.tech/concepts/bitswap/.
[5] 2019. How CIDs are created. https://docs.ipfs.tech/concepts/content-addressing/#how-cids-are-created.
[6] 2019. IPFS doc. https://docs.ipfs.tech/how-to/nat-configuration/.
[7] 2019. IPFS Gateway. https://docs.ipfs.tech/concepts/ipfs-gateway/.
[8] 2019. IPFS Pinning Files. https://docs.ipfs.tech/how-to/pin-files/.
[9] 2019. Merkle Directed Acyclic Graphs (DAGs). https://docs.ipfs.tech/concepts/merkle-dag/.
[10] 2021. IPFS in 2021: The Backbone of Web3's Mainstream Momentum. https://blog.ipfs.tech/2022-01-11-IPFS-in-2021/.
[11] 2021. OpenSea stores NFTs with IPFS and FileCoin. https://blog.ipfs.tech/2021-06-17-opensea-ipfs-filecoin/.
[12] 2022. IPFS for Nextcloud. https://apps.nextcloud.com/apps/files_external_ipfs.
[13] 2022. NFT Collections Explained. https://www.nftgators.com/nft-collections/.
[14] 2023. Dtube. https://d.tube/.
[15] 2023. ERC-721 NON-FUNGIBLE TOKEN STANDARD. https://ethereum.org/en/developers/docs/standards/tokens/erc-721/.
[16] 2023. ipfs-gateway-doc. https://blog.ipfs.tech/2022-06-30-practical-explainer-ipfs-gateways-2/#debugging-ipfs-content-discovery-and-retrieval.
[17] 2023. NFT-storage-service. https://nft.storage.
[18] 2024. Alexa Website Ranking. https://www.alexa.com/.
[19] 2024. Amazon S3 Cloud Storage. https://aws.amazon.com/s3/.

[20] 2024. Amazon S3 Service Level Agreement. https://aws.amazon.com/s3/sla/.

[21] 2024. Azure Blob Storage. https://azure.microsoft.com/en-us/products/storage/blobs.

[22] 2024. content-blacklist. https://badbits.dwebops.pub/.

[23] 2024. Etherscan:. https://etherscan.io/.

[24] 2024. FPT Cloud Object Storage. https://fptcloud.com/en/product/object-storage-2/.

[25] 2024. IPFS powers the Distributed Web. https://ipfs.tech/.

[26] 2024. Protocol Labs. https://protocol.ai/.

[27] 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.

[28] Dadepo Aderemi and Woudt van Steenbergen. 2020. An Evaluation of IPFS as a Distribution Mechanism for RPKI Repository. (2020).

[29] Onur Ascigil, Sergi Reñé, Michał Król, George Pavlou, Lixia Zhang, Toru Hasegawa, Yuki Koizumi, and Kentaro Kita. 2019. Towards peer-to-peer content retrieval markets: Enhancing IPFS with ICN. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*. 78–88.

[30] 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.

[31] 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. *arXiv preprint arXiv:2309.16203* (2023).

[32] D. Barkai. 2001. Technologies for sharing and collaborating on the Net. In *Proceedings First International Conference on Peer-to-Peer Computing*. 13–28.

[33] Vitalik Buterin et al. 2014. A next-generation smart contract and decentralized application platform. *white paper* 3, 37 (2014), 2–1.

[34] A. Carzaniga, M.J. Rutherford, and A.L. Wolf. 2004. A routing scheme for content-based networking. In *IEEE INFOCOM 2004*.

[35] Ruizhi Cheng, Nan Wu, Songqing Chen, and Bo Han. 2022. Will metaverse be nextg internet? vision, hype, and reality. *IEEE Network* 36, 5 (2022), 197–204.

[36] Ruizhi Cheng, Nan Wu, Matteo Varvello, Songqing Chen, and Bo Han. 2022. Are we ready for metaverse? A measurement study of social virtual reality platforms. In *Proceedings of the 22nd ACM Internet Measurement Conference*. 504–518.

[37] Chia Yuan Cho, Domagoj Babi ć, Eui Chul Richard Shin, and Dawn Song. 2010. Inference and Analysis of Formal Models of Botnet Command and Control Protocols. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*.

[38] Bram Cohen. 2003. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer systems*, Vol. 6. Berkeley, CA, USA, 68–72.

[39] 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).

[40] Erik Daniel and Florian Tschorsch. 2022. Passively Measuring IPFS Churn and Network Size. In *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 60–65.

[41] 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.

[42] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: Amazon's Highly Available Key-Value Store. *SIGOPS Oper. Syst. Rev.* 41, 6 (oct 2007), 205–220. https://doi.org/10.1145/1323293.1294281

[43] Trinh Viet Doan, Tat Dat Pham, Markus Oberprieler, and Vaibhav Bajpai. 2020. Measuring decentralized video streaming: A case study of dtube. In *2020 IFIP Networking Conference (Networking)*. IEEE, 118–126.

[44] 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.

[45] R Fielding, M Nottingham, and J Reschke. 2022. RFC 9110: HTTP Semantics.

[46] Savannah Fortis. 2022. url=https://cointelegraph.com/news/gaming-makes-up-over-half-of-blockchain-industry-usage-dappradar.

[47] Barbara Guidi, Marco Conti, Andrea Passarella, and Laura Ricci. 2018. Managing social contents in Decentralized Online Social Networks: A survey. *Online Social Networks and Media* (2018).

[48] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. 2005. Measurements, Analysis, and Modeling of BitTorrent-like Systems. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement* (Berkeley, CA) *(IMC '05)*.

[49] Anaobi Ishaku Hassan, Aravindh Raman, Ignacio Castro, Haris Bin Zia, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. 2021. Exploring Content Moderation in the Decentralised Web: The Pleroma Case. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*.

[50] Jim Hendler. 2009. Web 3.0 Emerging. *Computer* 42, 1 (2009), 111–113.

[51] Sebastian Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. 2020. Mapping the interplanetary filesystem. In *2020 IFIP Networking Conference (Networking)*. IEEE, 289–297.

[52] Binbing Hou and Feng Chen. 2020. A study on nine years of bitcoin transactions: Understanding real-world behaviors of bitcoin miners and users. In *2020 IEEE 40th international conference on distributed computing systems (ICDCS)*. IEEE, 1031–1043.

[53] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson. 2010. Privacy-Preserving P2P Data Sharing with OneSwarm. *SIGCOMM Comput. Commun. Rev.* (2010).

[54] Aisyah Ismail, Mark Toohey, Young Choon Lee, Zhongli Dong, and Albert Y Zomaya. 2022. Cost and Performance Analysis on Decentralized File Systems for Blockchain-Based Applications: State-of-the-Art Report. In *2022 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 230–237.

[55] Guoli Li, Vinod Muthusamy, and Hans-Arno Jacobsen. 2008. Adaptive Content-Based Routing in General Overlay Topologies. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*.

[56] Ying Li, Yaxin Yu, and Xingwei Wang. 2022. Three-tier Storage Framework Based on TBchain and IPFS for Protecting IoT Security and Privacy. *ACM Transactions on Internet Technology (TOIT)* (2022).

[57] 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.).

[58] David Mazières and M. Frans Kaashoek. 1998. Escaping the Evils of Centralized Control with Self-Certifying Pathnames. In *Proceedings of the 8th ACM SIGOPS European Workshop on Support for Composing Distributed Applications*.

[59] Satoshi Nakamoto. 2008. Bitcoin whitepaper. *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)* (2008).

[60] Shirish Patel and Philip J Rhodes. 2021. Decentralized Storage for Scientific Data. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 3760–3769.

[61] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. 2011. IP geolocation databases: Unreliable? *ACM SIGCOMM Computer Communication Review* 41, 2 (2011), 53–56.

[62] Bernd Prünster, Alexander Marsalek, and Thomas Zefferer. 2022. Total Eclipse of the Heart - Disrupting the InterPlanetary File System. In *USENIX Security Symposium*.

[63] Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. 2019. Challenges in the decentralised web: The mastodon case. In *Proceedings of the internet measurement conference*. 217–229.

[64] Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth R. Sastry, and Gareth Tyson. 2019. Challenges in the Decentralised Web: The Mastodon Case. *Proceedings of the Internet Measurement Conference* (2019).

[65] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. 2001. A Scalable Content-Addressable Network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*.

[66] Antony Rowstron and Peter Druschel. 2001. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Middleware 2001*.

[67] Recep Ahmet Sarıtekin, Eren Karabacak, Zübeyir Durgay, and Enis Karaarslan. 2018. Blockchain based secure communication application proposal: Cryptouch. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*. 1–4. https://doi.org/10.1109/ISDFS.2018.8355380

[68] 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)*.

[69] 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.

[70] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*.

[71] D. Stutzbach and R. Rejaie. 2005. Capturing accurate snapshots of the Gnutella network. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*.

[72] 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.

[73] 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).

[74] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[75] 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*.

[76] Quanqing Xu, Zhiwen Song, Rick Siow Mong Goh, and Yongjun Li. 2018. Building an ethereum and ipfs-based decentralized social network system. In *2018 IEEE 24th international conference on parallel and distributed systems (ICPADS)*. IEEE, 1–6.

[77] B. Y. Zhao, Ling Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. 2006. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *IEEE J.Sel. A. Commun.* (2006).

[78] Qiuhong Zheng, Yi Li, Ping Chen, and Xinghua Dong. 2018. An innovative IPFS-based storage model for blockchain. In *2018 IEEE/WIC/ACM international conference on web intelligence (WI)*. IEEE, 704–708.

A ADDITIONAL EXPERIMENTS

A.1 File types

Table A1. Top 10 websites based on `http_refer`.

`http_refer`	# requests	Category
d***p	428,068	x-rated
d***p	136,321	x-rated
teia.art	76,125	nft
n***p	72,105	x-rated
nftexplorer.app	50,067	nft
nftbiker.xyz	43,092	nft
hicetnunc.art	35,440	nft
d***m	31,056	x-rated
hicetnunc.xyz	26,477	nft
paintswap.finance	19,324	nft

Table A1 presents the Top 10 websites with the most requests in Gateway-22 and their corresponding category. It shows that the Top-10 sites with a domain name are either NFT-related or associated with x-rated activities.

A.2 Content Providers

Table A2. Top 20 providers serving requests to content in Gateway-22. The PeerID is partially replaced with *. Providers that overlap between Gateway-22 and Network-22 are highlighted in blue. In hostname column where IPs are included, the IPs are anonymized by *.

PeerID	Hostname	# of unique files served
***nx5EnZ6ZmC	elastic.dag.house	2,856,794
***sJa8vXjt3yW	nft3-storage-***	1,905,235
***PWwUT24qK4	nft3-storage-***	1,850,467
***zybsm2J2cP	nft3-storage-***	1,780,256
***RqNMRi8YHF	collab-cluster-***	1,535,774
***PgRwrMp2	nft3-storage-***	1,456,701
***PRbVHRhobC	nft3-storage-***	1,454,030
***MvybeTnwPy	collab-cluster-***	1,397,218
***6kLattaMnE4	nft3-storage-***	1,316,598
***AeBQceNrf4G	nft3-storage-***	1,307,287
***3yA8Z2DJZ	nft3-storage-***	1,259,163
***mSmCM7VkrE	ec2-*.amazonaws.com	1,235,489
***tWS8yKfeWmcr	ec2-*.amazonaws.com	1,154,694
***v9ahjhehXZ	ec2-*.amazonaws.com	1,118,412
***3n9DFyuqUTb	ec2-*.amazonaws.com	1,111,825
***KB9Z34Y2gV	ec2-*.amazonaws.com	1,099,531
***2roPNMXYunG	ec2-*.amazonaws.com	1,050,875
***Cs5uSfkoHK	nft3-storage-***	943,574
***1Z5pXUYDRH	ec2-*.amazonaws.com	900,732
***UdZtDHCiDDi	ec2-*.amazonaws.com	894,630

Table A3. Top 20 providers serving requests to content in Network-22. The PeerID is partially replaced with *. Providers that overlap between Gateway-22 and Network-22 are highlighted in blue. In hostname column where IPs are included, the IPs are anonymized by *.

PeerID	**Hostname**	**# of unique files served**
***nx5EnZ6ZmC	elastic.dag.house	963,106
***PRqNMRi8YHF	collab-cluster-***	870,610
***MvybeTnwPy	collab-cluster-***	852,455
***RbVHRhobC	nft3-storage-***	852,080
***PgRwrMp2	nft3-storage-***	791,633
***zybsm2J2cP	nft3-storage-***	751,387
***PWwUT24qK4	nft3-storage-***	707,653
***sJa8vXjt3yW	nft3-storage-***	659,109
***7n6vQ1y2Jdi	collab-cluster-***	658,145
***EsbC1C9nPqce	collab-cluster-***	609,532
***HahWkUjFaf	nft3-storage-***	600,901
***wJHanEeuvBa	nft3-storage-***	563,427
***wijHMxTj4E7f	ec2-*.amazonaws.com	512,784
***Jrvgc8eCbEo8	ns1016489.*.us	548,467
***3yA8Z2DJZ	nft3-storage-***	487,639
***UdZtDHCiDDi	ec2*.amazonaws.com	479,791
***mSmCM7VkrE	ec2-*.amazonaws.com	429,322
***RcXunZ8xEWF2y	ec2-*.amazonaws.com	401,375
***M5Bw3Gf8p7b	Forest Net LTD	373,679
***sBAfXAdCNjXQ	ec2-*.amazonaws.com	333,081

Table A2 and Table A3 present the Top-20 providers that serve the content most during our retrievals of Gateway-22 and Network-22, respectively. As we can see in these two tables, the top providers are almost all cloud-based nodes.

A.3 The Impact of Chunk Size

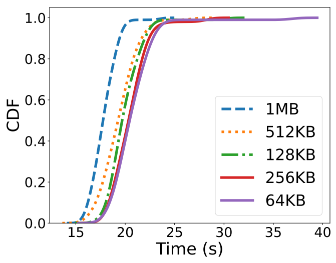
Figure A1 presents the CDF of downloading performance of a 128 MB file in 6 different regions. Each subfigure represents the result of a certain region. In each experiment, we place the requestor and provider in the same region and log the time cost of downloading a 128 MB file in this region with five different chunk sizes. For each chunk size, the downloading is repeated 1000 times.

Figure A2 presents the throughput by downloading a file with different combinations of file size and chunk size across all 6 regions. For each combination, the downloading is repeated 1000 times. Subsequently, we normalize the average throughput of all the combinations of region, file size and chunk size, into (0,1] interval. This normalized value is then represented in the heatmap, where a darker color indicates higher downloading throughput.

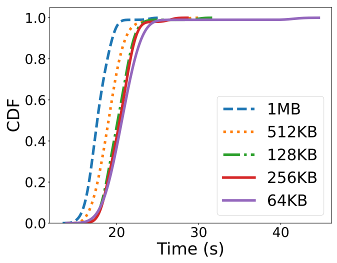
Figure A3 extends the experiment of Figure A2 into different requestor and provider placements. Each subfigure represents the average throughput with various placements of requestor and provider for a certain combination of file size and chunk size.

Figure A4 presents the results of the experiment designed to analyze the variations in downloading throughput at different times of the day. In this study, a 256 MB file was periodically retrieved for five days using two distinct chunk sizes: the default size of 256KB and a larger size of 1MB.

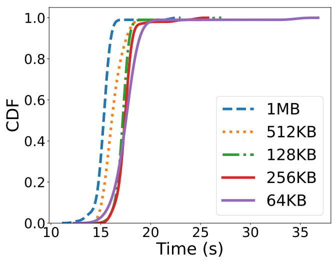
Received August 2023; revised April 2024; accepted April 2024



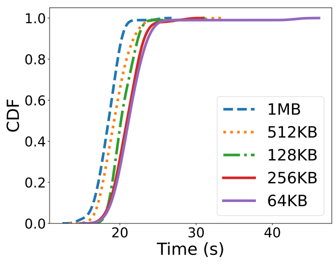
(a) US East Coast.



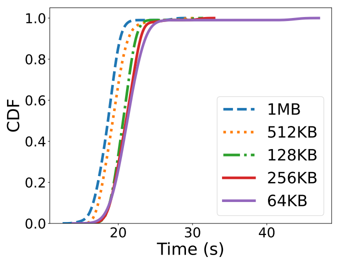
(b) US West Coast.



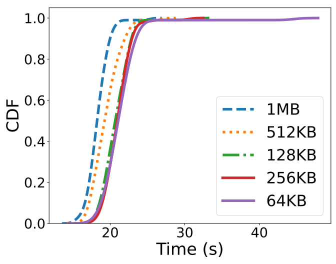
(c) Central EU.



(d) South America.

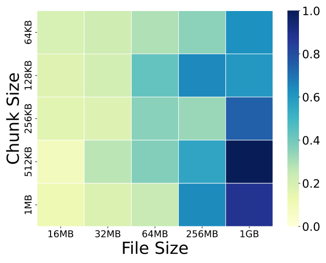


(e) Asian Pacific Northeast.

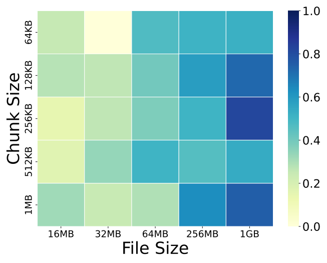


(f) Asian Pacific Southeast.

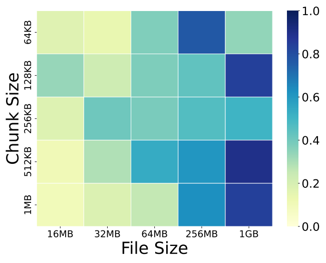
Fig. A1. CDF of Time cost on downloading a 128 MB file across different regions with various chunk sizes



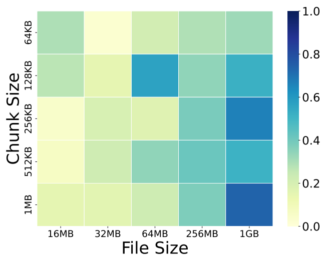
(a) US East Coast.



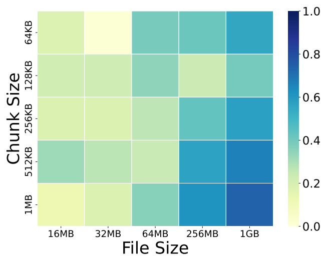
(b) US West Coast.



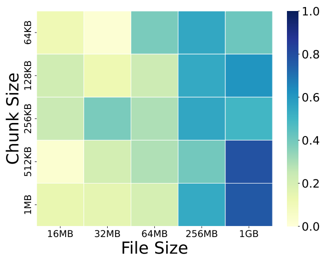
(c) Central EU.



(d) South America.



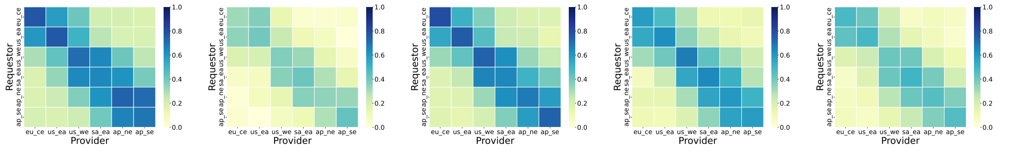
(e) Asian Pacific Northeast.



(f) Asian Pacific Southeast.

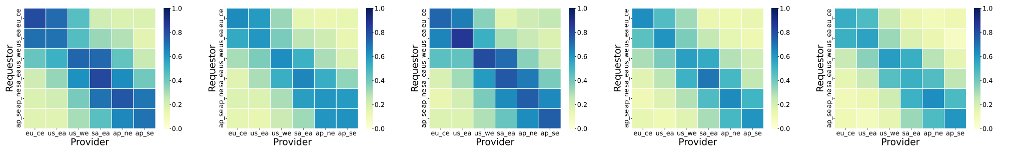
Fig. A2. Heatmap of normalized throughput across different regions with various chunk size and file size

Chunk Size: 64KB



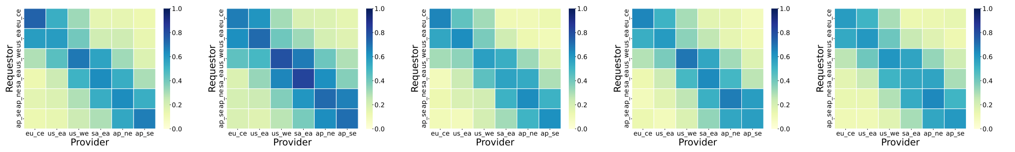
(a) File Size: 16MB (b) File Size: 32MB (c) File Size: 64MB (d) File Size: 256MB (e) File Size: 1GB

Chunk Size: 128KB



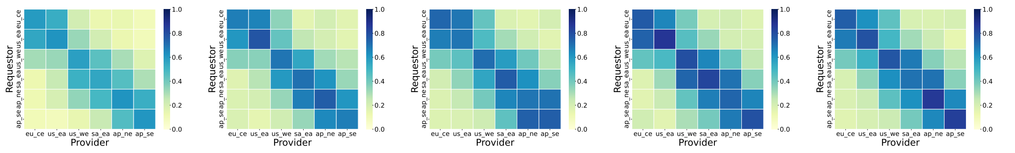
(f) File Size: 16MB (g) File Size: 32MB (h) File Size: 64MB (i) File Size: 256MB (j) File Size: 1GB

Chunk Size: 256KB



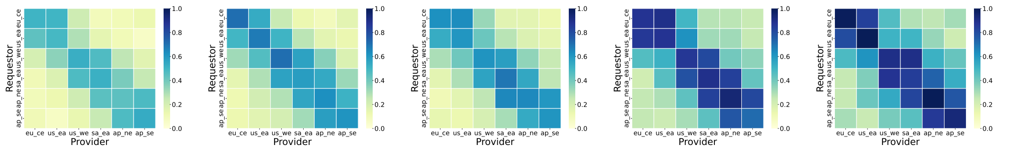
(k) File Size: 16MB (l) File Size: 32MB (m) File Size: 64MB (n) File Size: 256MB (o) File Size: 1GB

Chunk Size: 512KB



(p) File Size: 16MB (q) File Size: 32MB (r) File Size: 64MB (s) File Size: 256MB (t) File Size: 1GB

Chunk Size: 1MB



(u) File Size: 16MB (v) File Size: 32MB (w) File Size: 64MB (x) File Size: 256MB (y) File Size: 1GB

Fig. A3. The impact of chunk size with requestor and provider located in different regions.

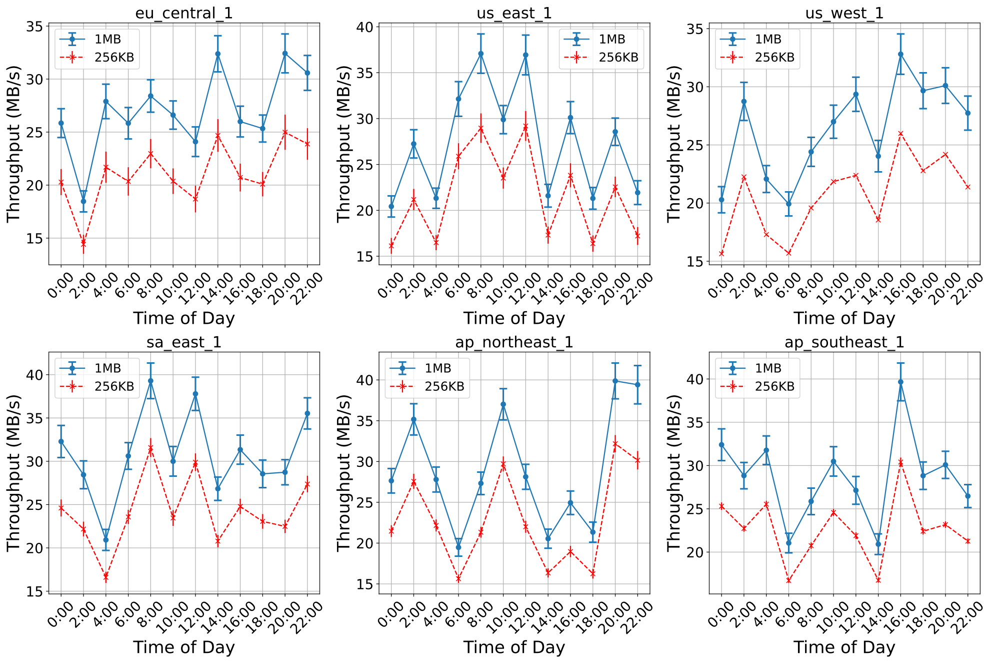


Fig. A4. The throughput of downloading across locations at different time (all local time).