Studying the workload of a fully decentralized Web3 system: IPFS *

Pedro Ákos Costa[1], João Leitão[1], and Yannis Psaras[2]

[1] NOVALINCS & NOVA University of Lisbon
pah.costa@campus.fct.unl.pt, jc.leitao@fct.unl.pt
[2] Protocol Labs
yiannis@protocol.ai

Abstract. In this paper we present a study of the workload of the InterPlanetary File System (IPFS), a decentralized file system which is a key enabler of Web3. Our study focuses on the access patterns observed from one of the most popular IPFS gateways located in North America, and analyzes these access patterns in light of one of the most common assumptions made in regard to the access pattern of decentralized content sharing systems: that the access patterns are mostly geographically localized. However, through our study, we show that the access patterns are content-dependent rather than geographically localized. In our study, we found that access patterns mostly target a small set of popular content, which is provided by nodes in the North American and European regions, despite the location of the requester. Furthermore, we found that, interestingly, this popular content is only provided by a few nodes in the system, suggesting a significant imbalance both in content providers and in the access patterns of the system to the content. This in turn suggests that the system is significantly centralized on these few node providers.

Keywords: Web3 · Distributed Systems · Measurements

1 Introduction

The Internet nowadays is supported mostly by a few large cloud providers that include Google, Amazon, Microsoft, and Cloudflare. These providers host a wide variety of web services that operate at a large scale serving a huge number of users scattered throughout the World. Nevertheless, this paradigm forces application providers to fully rely and trust on the operators of these centralized cloud infrastructure, which dictate the terms of service with little to no competition. Moreover, in most application scenarios using cloud infrastructures the control of user data is relinquished, in some way, to these operators, which is undesirable

* This work was partially supported by FCT/MCTES grant SFRH/BD/144023/2019 and by the European Union's Horizon Europe Research and Innovation Programme under Grant Agreement No 101093006. Artifacts available in https://doi.org/10.5281/zenodo.7850384.

(and being target of legislation such as the European GDPR), specially if we consider the susceptibility of attacks to cloud infrastructures [12, 18]. To address this, and partially motivated by the increased popularity and use cases enabled by blockchain technologies [27, 36], the concept of Web3 [16] has emerged. Web3 aims at decentralizing web technologies to improve user security and privacy as well as providing guaranteed ownership of user data, through the use of a combination of existing and novel peer-to-peer protocols [20, 32].

However, Web3 is still in its early stages and has yet to become competitive with modern cloud infrastructures, in terms of flexibility, application development, security/privacy, and performance. This is due to the current Web3 main technology enablers: blockchain, that maintains and replicates the system state; libp2p [30], that is used to develop decentralized applications (dApps) and their support; and IPFS [3, 34], that is used as an entry-point to most dApps, that are still restricted to the domains of content creation and sharing [10, 28], decentralized financing [35], decentralized communication [4, 9], among a few others. This is because, blockchain, although important for decentralization, also limits the amount of interactions applications can have, as these are mostly made through the replicated state machine materialized by blockchains, which have limited throughput. Furthermore, IPFS still has a large space for performance improvement, as recent studies show that searching for content on IPFS can take up to $2,5$ hours [5].

IPFS relies on a distributed hash table (DHT) to make content available to users in the network. The DHT organizes nodes and content according to a uniform distribution of identifiers that are assigned both to nodes and content. This however, leads the topology of the DHT to not match the physical network topology, which can cause routing to be performed across large distances for content that is published near the requesters. This is commonly known as the topology mismatch problem [21–23]. To address this issue, there are a number of works in the literature [1, 7, 13, 17, 26, 31] that try to optimize and scale DHT designs by assuming that content access patterns presents a high level of locality, meaning that content is mostly accessed by users located in the (geographical) vicinity of users that published it.

In this paper we present an in-depth analysis of the workload on IPFS to verify if IPFS can benefit from such approaches that optimize for locality of content access. To this end, we have gathered two weeks worth of logs from one of the most popular IPFS gateways located in North America. These logs contained the (access) requests made from IPFS users across the World to large amounts of content stored in IPFS. We analyzed these logs and performed the same (valid) requests that were observed in the logs to fetch the information regarding the providers of the content.

While one of the contributions of this paper is the presentation of the novel methodology that we developed to be able to study the logs of a large-scale peer-to-peer system regarding its workload, our study allows us to make two additional contributions. First, to correlate the amount of content that is being provided on IPFS by different peers (providers) considering content that was

requested through one of the most popular public IPFS gateways. Second, and through the use of the MaxMind [24] database that matches IP addresses to geolocation information, to identify the relation between the location of the origin of requests and location of providers of the requested content. With this data, we analyzed the IPFS workload and found that most content is provided only by a few providers that are mostly located on North America and Europe, and disprove the locality assumption commonly made in many peer-to-peer systems [1, 13].

The remainder of the paper is structured as follows: Section 2 provides a brief description of IPFS and how it operates. In Section 3 we detail our methodology to gather and analyze data used in this study. Section 4 presents our results, providing insights on the workload of the IPFS network. Section 5 discusses related work, and finally, Section 6 concludes the paper with final remarks and observations regarding the obtained results.

2 IPFS

IPFS is a large scale peer-to-peer distributed system that aims at connecting computing devices offering a shared file system. To enable this, IPFS relies on libp2p [30] to handle networking aspects. To provide a membership and lookup service among peers in the network, libp2p relies on a distributed hash table (DHT), implemented as a variant of the Kademlia protocol [25]. IPFS leverages this DHT to distribute and search content and locate peers in the system. Content in IPFS is immutable, with each individual piece of content (e.g., file, data block, etc) being associated with an identifier (*CID*), that is a collection of hashes referred as a *multi-hash*, that includes the hash of the content and hashes of metadata describing the hashing mechanism. Similarly, each peer in IPFS also has an identifier (*peerId*) that is a multi-hash of the peer's public key. Peers organize themselves in the DHT according to a SHA-256 hash of their peerId and store content pointers according to the SHA-256 hash of the CID. Furthermore, each peer has associated to it a list of multi-addresses, that describe the Internet addresses of the peer (this can be ipv4, ipv6, and the transport protocols supported by that peer and ports).

IPFS peers do not store the content itself but only a pointer to the peer providing the content (the one that published the content). As such, for a peer to publish content on IPFS, the peer effectively announces to the network that it provides the content by storing in the IPFS DHT a *provider record*. A provider record contains a mapping of a CID to peerIds, i.e., the content providers. As per the Kademlia operation, this provider record will be stored in the k closest peers to the hash of the CID on the DHT. In IPFS k has a value of 20. Note that the same content can be provided by multiple peers.

To fetch content, IPFS uses a protocol named *Bitswap* [8] that performs both discovery and data transfer. For Bitswap to discover content it begins to perform a local one-hop flood request for the CID. This will send a message to all neighboring peers asking if they have the contents of the CID locally. Note that this process is highly optimistic as Bitswap leverages the fact that an IPFS node is connected to hundreds of other nodes (which are the nodes managed by

DHT protocol and cached nodes that were used in previous DHT searches). If the answer is positive, Bitswap begins transferring the content with a process akin to BitTorrent [11]. If the response is negative and the content cannot be found in a neighboring peer, it resorts to the DHT to find the provider records of the CID. Once Bitswap has obtained one provider record, it will start to try to transfer the content from providers indicated in that record.

IPFS has two modes of operation: as a server or as a client node. Server nodes are (usually) publicly reachable in the Internet and participate actively in the DHT to enable routing among peers and serve content to the network. Client nodes connect to the DHT, but do not maintain the DHT, meaning that client nodes can only perform requests to the DHT and do not participate in the routing process. Additionally, an IPFS peer acting as server can also act as an HTTP gateway. In this case the IPFS node also runs a web server that grants access to IPFS via a browser interface to users. In more detail, a gateway node is able to transform an HTTP request into a valid IPFS request, that will either trigger a Bitswap and/or DHT operation, allowing to serve the content to clients that are not running an IPFS node, and instead access to content via their browsers.

As it is common in many P2P systems, not all IPFS nodes are publicly reachable. This is the case for nodes that are behind a NAT. In this case, an IPFS node can request a publicly reachable IPFS node to relay traffic for itself. Furthermore, IPFS hosts third-party pinning services that host content for users on IPFS servers controlled by the pinning service provider for a fee.

3 Methodology

In this section we provide a detailed description of our methodology to study and characterize the IPFS workload. In summary, we collected two weeks worth (from March 7th to March 21st of 2022) of logs [6] from one of the most popular IPFS gateway – *ipfs.io* – that is located in North America. These logs were produced by a NGINX reverse proxy that logged every HTTP request made. Each entry in the log contains information about an HTTP request made by a user to that IPFS gateway.

To process these logs, we filtered all non-valid HTTP requests (e.g., POST operations and out of format entries), and extracted the CID in each valid HTTP request. From this filtered subset, we resorted to IPFS to obtain all the available provider records for each CID. To obtain geolocation information from both the requests and providers, we matched the IP addresses in the gateway logs and the provider record respectively against the MaxMind GeoLite2 Free Geolocation Database [24]. Note that, the dataset provided was anonymized by replacing the IP address of requesters with an identifier that maps to the geographic location of the requester.

Finally, we combined both datasets on the requested and provided CID to produce a global view that shows where content is requested and where that content was being served from. In the following, we describe this process in more detail.

3.1 Processing the requests

The first step in analyzing the IPFS workload is to extract the requested CIDs. To do this, we parsed the gateway logs into a structured data format that can be easily processed. The gateway logs are generated by an NGINX reverse proxy, that serves as a frontend for an IPFS server node that acts as the IPFS gateway. Each log entry contains information about the received HTTP requests by the IPFS gateway. In particular, we are interested in the following information of the HTTP requests: the IP address of the requester, to extract geolocation information; the HTTP operation and the HTTP status, to filter unwanted HTTP requests for our study (e.g., POST operations and GET operations whose answer was had a 400 HTTP status); and the HTTP target and the HTTP host, that effectively contain the requested CID. In total the logs we collected contained $123,959,912$ requests.

We begin by filtering logs that are out of format. This amounts to 0.001% of all requests and include requests that have unparseable HTTP targets which contained hexadecimal codes in that field. Next, we remove all requests that are not GET HTTP requests, as only GET requests actually make requests to IPFS (either through BitSwap or the DHT). This removed about 19% of all requests contained in the logs, leaving almost 81% of log entries with only GET operations.

However, not all GET operations are of interest to study the workload. Such is the case for GET operations that did not succeed (i.e., where the reply had an HTTP status code of 400) or that do not contain a CID in their request. We filter GET operations that did not succeed due to these having a high probability that the content is either invalid (i.e., it was never available on IPFS) or the content was no longer available at the time of the request. As for the CIDs in requests, these appear in the full `url` of the request that is obtained by concatenating the HTTP host field with the HTTP target field. Note that the HTTP host in this case can be `ipfs.io` (i.e., the gateway host), in which case the CID will appear on the HTTP target; or can be in the form of <`CID`>`.ipfs.dweb.link`, in which case the CID is in the HTTP host part. Note as well, these `url` contain a file system path to the requested content, which means that the CID might represent a folder containing multiple files. In the case a `url` contains multiple CIDs, we only consider the first CID, as effectively the gateway will only search for the first CID in the `url`, as the remaining accessed content can be found below the first CID in the (remote) file system path.

With this step, we filter out 41% of all GET operations, where 17% of these were GET operations that did not succeed and 24% were requests that did not contain a CID. With this, 47% of the total requests remained as valid GET operations, which were the ones consider in our study. Table 1 summarizes the number of requests we processed in each step for our study described above.

3.2 Locating the content providers

The second step in studying and characterizing the IPFS workload is to gather information on the providers of the requested content, as to understand where

Table 1: Requests processed summary.

	Number of entries	Percentage
Total	123,959,912	100%
Out of Format	2,165	0.001%
Not GETs	24,298,396	19.602%
All GETs	82,439,744	80.396%
Valid GETs	58,869,788	47.491%

Table 2: Providers processed summary.

	Number of entries	Percentage
Total CIDs	4,009,575	100%
CIDs w/out provider	2,175,608	54.26%
CIDs w/ provider	1,833,967	45.74%
Providers	55,830	100%
Providers w/out address	32,968	59%
Providers w/ address	22,862	41%
Providers w/ address after find	26,886	48%

and by how many peers is the content served. To achieve this, we developed a simple libp2p application that connects to IPFS and requests all provider records for a given CID. Our application leverages the fact that IPFS uses the same networking software stack and DHT provided by libp2p (which by default connects to the IPFS network), to execute FindProviders API calls to libp2p DHT to gather information for all providers of CIDs.

Out of the 58,869,788 valid GET operations, a total of 4,009,575 different CIDs were requested. We requested the providers of all these CIDs through our libp2p application. We found we were unable to locate the providers of 54% of all CIDs. This can be due to the fact that the content was no longer available on the network. Note that this study was perform about 6 months after the requests were recorded by the gateway. This enables us to focus our study on non-ephemeral content on IPFS which we argue is more representative of the workload of the system. From the CIDs with providers we discovered 55,830 different providers however, 59% of these did not have any addressing information associated to them. This means that the peers storing the provider record did not receive any update on the provider (from any kind of traffic) for a window of time longer than 30 minutes, as such, the peers storing the provider record assumed the provider might have moved and changed its network address, thus deleting the previously known provider multi-address. To fetch the multi-address in these cases, we queried the DHT for the multi-address of the provider, and managed to find the multi-address of 4,024 more providers (an additional 7% of providers regarding those obtained directly from provider records). Table 2 summarizes the numbers of processed CIDs and found providers.

3.3 Analyzing the data

The final step to study and characterize the IPFS workload is to join both request and providers data to map from where in the World are requests being performed

and where in the World is the content provided/available. This required us to extract geo-locality data from the gathered data. To this end, we use the MaxMind GeoLite2 Free Geolocation Database [24], that provides the geolocation of a vast collection of public IP addresses. However, this database is not complete and may have IP addresses whose geolocation is unknown. Fortunately, for the request data all IP addresses had geolocation information. On the other hand, only 88% of providers with addresses had geolocation information.

Note that a provider is identified by a peerId and has multiple multi-addresses. To get the geolocation information of a provider we had to extract the (public) IP address of multi-addresses. For multi-addresses that contained protocols `ip4` and `ip6` this procedure is straightforward. This amounts to 98% all observed multi-address (excluding local address, such as `127.0.0.1` and `::1`); 0.6% of multi-addresses were DNS addresses, that we resolved with local DNS resolvers; and the remainder 1.4% of multi-address were relay multi-addresses, and hence the provider did not have a public reachable IP address, which we ignored in this study. For providers that had multiple locations (probably due to the use of VPN services), we considered the last observed location. These were just a few cases that do not impact significantly our study.

We have inserted both datasets into a PostgreSQL database for ease of analysis. This database has 2 tables, one containing the requests and another containing the providers. The requests table stores a request identifier (reqId), the timestamp the request was originally made to the gateway, the CID requested, and the location information of the requester. The requests table has as key the reqId, that is a hash of the request log entry, to avoid processing duplicate request entries from the log. The providers table stores: the CID provided, the peerId of the provider, and the location information of the provider. The provider table has as key the CID and peerId. This uniquely identifies each provider entry, since each CID can have multiple providers, and a provider can provide multiple CIDs. Notice that the CID in the providers table is a foreign key of the CID in the requests table.

By performing a join over the requests and providers table we can compute a mapping from where requests are performed to where they are provided. Before presenting the results in the next section, we follow by providing some implementation details on the mechanisms we employed to process and find content providers information.

3.4 Implementation details

The code and scripts that were used to process the data for this study can be found in `https://github.com/pedroAkos/IPFS-location-requested-content`. The processing of data required fine-tuning of the parallelization of queries to IPFS. This was required because IPFS can take some time to retrieve the provider records from the DHT; from our study the average latency was about 6 seconds, with the maximum latency reaching 1.5 hours; we made parallel queries to IPFS to fetch provider records. However, libp2p can be extremely taxing on the network, as a libp2p node can maintain hundreds, or even thousands, of connections and perform thousands of requests. To put this in perspective, a process executing

100 queries in parallel to find providers would produce almost 10,000 packets per minute. The process to resolve all 4,009,575 distinct CIDs took approximately 40 hours.

4 Results

In this section we analyze the results from our study to understand if IPFS would benefit from a DHT design that assumes requests to have geographic locality. By doing so, our analysis aims to answer the following questions:

1. *How many requests are made to IPFS on average per day?*
2. *How is the request frequency distributed over different CIDs in the system?*
3. *How are providers geo-distributed in the system?*
4. *How is provided content distributed across providers in the system?*
5. *How does the location of requested content correlate with the location of providers for requested content?*

To answer these question, we analyze the data first from the point of view of requests by considering the requests data extracted from the gateway logs (Section. 4.1). We then analyze the data from the point of view of providers using the data we extracted directly from IPFS (Section 4.2). Finally, we combine information from both requests and providers data to produce a correlation between the location of requests' origins and content providers (Section 4.3).

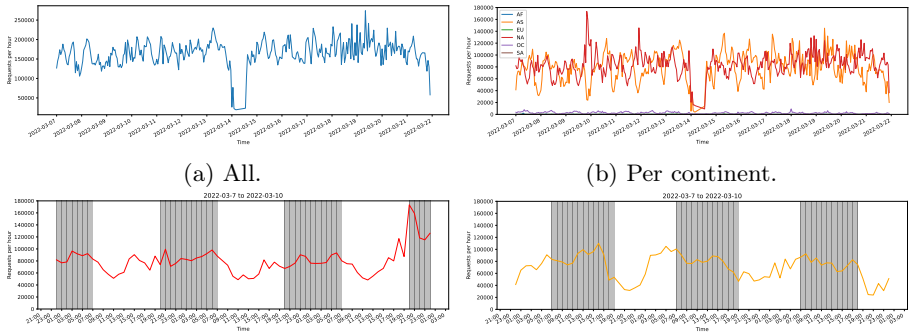
4.1 Requests

In this section we analyze the results from the perspective of content fetchers. With this, we aim to answer the first two questions of our analysis. *How many requests are made to IPFS on average per day?* and *How is the request frequency distributed over different CIDs in the system?* We begin by answering the first question.

Figure 1 represents the client requests processed by the gateway per hour. Notice that Figure 1a captures all requests made during the period of two weeks (x-axis), Figure 1b captures the same requests but characterized by continent, and Figures 1c and 1d focuses on the request traffic for the two regions with most traffic, North America (Fig. 1c and Asia (Fig. 1d), for only the first 3 days of the analysis period, with the night hours shaded on timezones that align with each region (GMT-7 and GMT+8 respectively).

Figure 1a shows that, on average, more than 150,000 requests per hour are made to the IPFS gateway, reaching a maximum of almost 275,000 requests per hour. Notice that on day 2022-03-14 the requests suddenly drop. We verified this, and indeed the logs we have from that day abruptly stop after a few hours (just before the 5 hour mark). Most likely, this was due to an issue with the gateway that day that made it unreachable for about 9 hours, which after then resumed processing requests regularly.

From Figure 1b we can see that most of the gateway traffic is split from North America (NA) and Asia (AS) with more than an average of 75,000 requests



(a) All. (b) Per continent.

(c) Three days of NA trafic on GMT-7. (d) Three days of AS traffic on GMT+8.

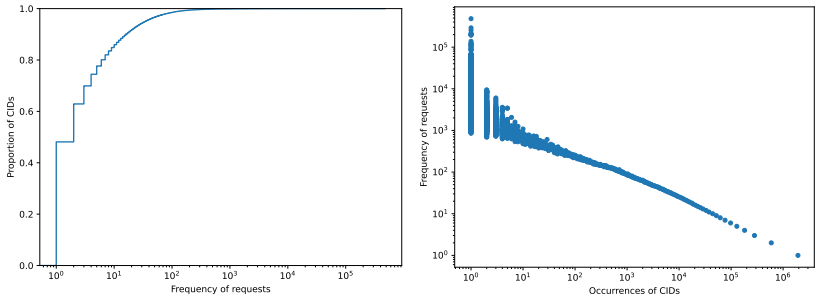
Fig. 1: Requests over time.

per hour with origin on each region. The third region with the most requests per hour is Oceania (OC) with an average of around 2,500 requests per hour. This is followed by Europe (EU) with an average of around 85 requests per hour, Africa (AF) with an average of around 57 requests per hour, and South America (SA) with an average of only 3 requests per hour. From these results we conclude that this IPFS gateway handles predominantly traffic from North America and Asia with a high volume of requests. Note that `ipfs.io` has an anycast DNS record [34], which means that there are multiple instance of the gateway located in the world (most likely there is an instance in Europe that handles the European traffic). Nevertheless, we argue that we have sufficient data to analyze if there is geographic locality in requested content.

To understand if this high volume of traffic has a day/night pattern, on Figures 1c and 1d we plot the requests per day of the first 3 days of our analysis, and shaded the areas of the plot that represent the night cycle (between 21h and 7h). In Figure 1c we plot the North American traffic and shaded the night hours on the gateway's timezone (GMT-7), and in Figure 1d we plot the Asian traffic and shaded the night hours on the Asian timezone (GMT+8). From these results, there appears to be no obvious day/night pattern for the North American and Asian traffics. However, there is a slight tendency towards having more traffic during the night, although marginal. From these results we conclude that the IPFS gateway has a steady high volume of traffic that is not driven by geographical region nor by day/night cycles.

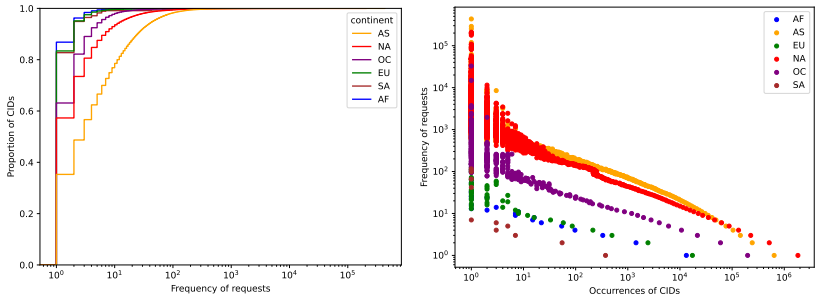
Figures 2 and 3 represent the frequency of requests performed for a CID (i.e., how many times was a CID requested from the gateway by a user). These results serve to answer question #2: *How is the request frequency distributed over different CIDs in the system?*

Figure 2a shows an Empirical CDF (ECDF) for all requested CIDs. Notice that the x-axis (representing the frequency of requests) is in logarithmic scale. The y-axis captures the proportion of requested CIDs with at most that amount of accesses. We notice that almost half of all CIDs are only requested once. After



(a) ECDF. (b) Distribution.

Fig. 2: All requested CIDs frequency.



(a) ECDF. (b) Distribution.

Fig. 3: Per continent requested CIDs frequency.

that, the frequency increases with decreasing increments on the proportion of CIDs, where about 90% of all CIDs are requested at most 10 times and about 99% of all CIDs are requested at most 100 times. Figure 2b complements the ECDF showing the distributions of frequency of requests (shown on the y-axis) over the number of CIDs (shown on the x-axis). Each point in this distribution represents how many CIDs where requested how many times. Note that both axis of this figure are in logarithmic scale. From this figure we can see the tendency on the frequency of requests over the number of CIDs, which resembles a typical Zipf distribution. Table 3 summarizes the frequency of the top 10 requested CIDs. We can further add, that most of these top 10 most requested CIDs were Non-Fungible Tokens (NFT) related data, suggesting that this is a primary use case for IPFS.

Figure 3 shows the same information but characterized by the following regions: Africa (AF), Asia (AS), Europe (EU), North America (NA), Oceania (OC), and South America (SA). Figure 3a shows an ECDF for the frequency of requested CIDs (on the x-axis in logarithmic scale) over the proportion of requests (on the y-axis), discriminated by region. We notice that almost 60% of requests originating from Asia, request at most the same CID thrice, whereas

Table 3: Top 10 summary of data in descending order. The first column represents the amount of requests to each CIDs. The second column represents the amount of replicas of each CIDs. The third column represents the amount of different CIDs provided by each provider node. Each column is independent, encoding different CIDs and providers.

Requested CIDs	Replicated CIDs	CIDs per Provider
482,620	12,610	869,734
290,485	5,274	213,837
254,132	4,663	202,947
213,019	2,047	200,373
209,913	1,876	176,180
203,510	1,822	174,803
199,628	1,404	173,315
198,500	1,372	144,422
193,432	1,283	108,023
138,084	1,272	107,885

60% of requests originating from North America request the same CID only once. This shows that content requested from Asia has a higher popularity (i.e., the same CIDs are requested more often) than in the remainder of regions. Figure 3b complements the ECDF with the distributions of frequency of requests (shown on the y-axis in logarithmic scale) over the number of CIDs (shown on the x-axis in logarithmic). However, this shows that all regions seem to present a similar Zipf distribution albeit, with different proportions, that is proportional to the request rate originating from that region.

4.2 Providers

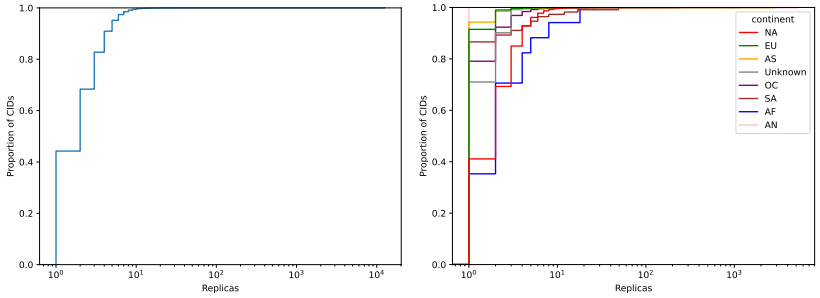
In this section we analyze the results from the perspective of providers. With this, we aim to answer questions #3 and #4 of our analysis. *How are providers geo-distributed in the system?* and *How is provided content distributed across providers in the system?*

The first question is answered by the results presented in Table 4 which shows the number of providers per continent. Notice that North America (NA) has almost as many providers as Europe (EU) and Asia (AS) together. This shows, that North America composes the largest portion of the content providers on the IPFS system (which is corroborated by previous studies [2, 14]). Furthermore, notice the last two columns in the table, that represent providers that only had a relay address (labeled as Rl), meaning they were behind a NAT without a public address; and providers whose location information was unknown (labeled as Un), meaning there was no entry on the MaxMind database for those providers public IP address. As the location of Rl nodes is also unknown, from this point on all Rl nodes are considered as belonging to the Un category.

To answer question #4: *How is provided content distributed across providers in the system?* we analyze both the amount of content replication in the system (Figure 4) and the amount of (different) content provided by each individual provider node in the system (Figure 5).

Table 4: Providers geo-distribution.

	AF	AS	EU	NA	OC	SA	AN	Rl	Un
Providers	40	4959	5789	10983	431	104	1	2473	689

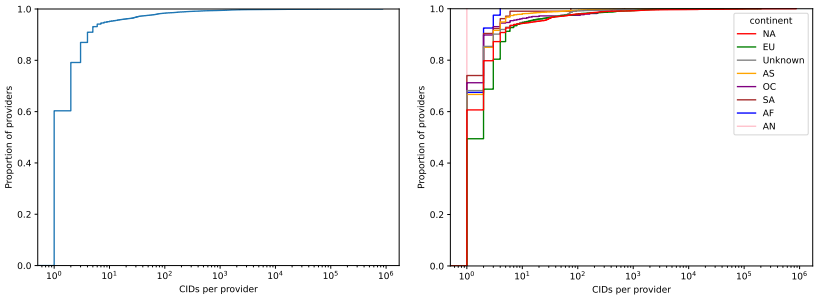


(a) All. (b) Per continent.

Fig. 4: CID replicas.

Figure 4 reports on the amount of replicated content over different providers. Figure 4a shows an ECDF that captures the amount of replicas (on the x-axis in logarithmic scale) that were found for the proportion of CIDs in that were requested through the gateway (on the y-axis). We note that almost 70% of all CIDs are replicated at most twice (i.e., provided by at most two different providers in IPFS). Only a very small proportion of CIDs are replicated by a large number of providers. Table 3 summarizes the amount of replicas of the top 10 replicated CIDs, where, after looking into these CIDs, we found that most of them are IPFS manual pages. We verified if theses highly replicated CIDs were also the most requested CIDs and found that this was not the case. In fact, the top 10 requested CIDs are not highly replicated, having only a few providers (only 3 of these CIDs have more than 10 providers). Figure 4b breaks down the CID replicas by region. Here we notice that Africa has the most replicas of CIDs although, this does not represent a large number as there are only a few providers in that region. Although it is not visible in the plot, there is a small percentage of CIDs that is highly replicated in North America. This is not surprising, as North America has the largest number of content providers. Nevertheless, these results suggest that there is a very limited high availability of requested content through replication. This can be mostly explained by the way content is replicated, where there needs to be an explicit (re)provide action by the user after fetching a copy of the content from other provider(s).

Figure 5 shows the amount of different (requested) CIDs each provider provides. Figure 5a presents an ECDF of the proportion of providers (on the y-axis) that provide different amounts of CIDs (on the x-axis in logarithmic scale), here we can see that 60% of providers only provide a single CID. We also note that less than 10% of providers provide at least 10 CIDs, with a very small proportion



(a) All. (b) Per continent.

Fig. 5: CIDs per provider.

of providers providing at least 1,000 CIDs. The providers that provide more CIDs amount to the largest part of provided CIDs, meaning that most CIDs are provided by the same small set of providers. This suggests that pinning services are the main providers of content in IPFS. Table 3 summarizes the top 10 providers with the most CIDs. Some of these providers had DNS multi-address, which we verified pointed to DNS records suggesting these providers belonged to `nft.storage`, which is a popular storage service for NFT content in IPFS. Figure 5b analyzes the proportion of providers (on the y-axis) that provide different amounts of CIDs (on the x-axis in logarithmic scale) categorized by continent, which shows that the large providers are mostly located in North America (NA), Europe (EU), and Oceania (OC). The fact that the biggest portion of CIDs is provided only by a small set of providers suggests that although IPFS is a decentralized content network system, the content stored in IPFS presents a high degree of centralization in this small set of providers.

4.3 Requested content vs. provided content

Finally, in this section we combine gathered data from the requests and the providers to obtain a global view of the workload, and to answer the last question: *How does the location of requested content correlate with the location of providers for requested content?*

To this end, we matched the request's origin location to the providers' location, generating a heatmap (presented in Figure 6) that matches the location from where each request was made to the location of provider(s) that had the requested content. In Figure 6 the rows represent the location of the origin of (all) requests while the columns present the location of providers, including a column (and row) labelled as *Unknown* that encodes the percentage of requests to content whose provider was not found or did not have (valid) geolocation information. Note that a single request can be served by multiple locations, as per the CID replication factor we discussed previously. We normalized the requests per region to eliminate the disparity in quantity of requests, showing on the heatmap the percentage of all requests made from one region to any other region (including itself).

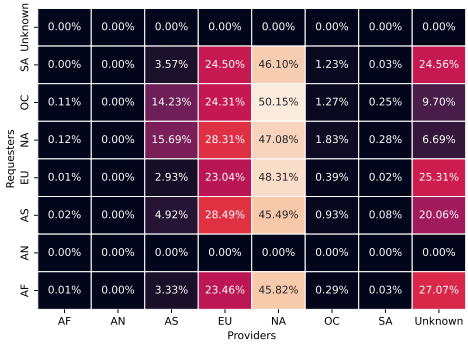


Fig. 6: Request locality of all requested content through the gateway.

By analyzing the heatmap, we notice that the vast majority of requests from all regions are either provided by providers in North America or in Europe. This in fact suggest that there is very little geographic locality access pattern in IPFS, as the heat is concentrated in North America and Europe, rather than on the diagonal of the heatmap. However, from our previous observations, this is also to be expected as the vast majority of content is located in these regions. Furthermore, from this heatmap we can conclude that the North American region contains the most popular CIDs, which from the previous observation that requests follow a Zipf distribution, we may also conclude that the Zipf is not independent per region. Finally, one last conclusion we can draw from our results is that the IPFS access patterns seems to be driven more by the content's popularity than by the local interests of users, and hence the locality assumption made by many P2P solutions is not applicable in this context. Indirectly, these findings also indicate that current dApps that really on IPFS for content distribution have a global expression, whose service providers are mostly located in North America and Europe.

5 Related Work

Measuring and understanding the behavior of large scale and decentralized systems has always been an important endeavour, with a vast amount of studies being made for peer-to-peer systems in the early 2000's. The main challenge in understanding how these systems operate in the wild derive from their decentralized nature, which makes it hard to have vantage points to collect enough information about events happening in the system. In particular, we highlight two studies over peer-to-peer systems, one that also analyzes the traffic of large scale systems, and a second that characterizes the workload of BitTorrent. The first [33] analyzes the peer-to-peer traffic across large networks such as FastTrack, Gnutella, and DirectConnect, from the point of view of a single ISP. Their findings are similar to ours in the sense that they also observe most of the traffic being generated by a very small amount of hosts. The second study [29] analyzes the popular BitTorrent peer-to-peer file-sharing system. In this study, the authors

perform active measurements in the systems to obtain data related to upload and download speeds. On the other hand, our study focuses on complementary aspects, such as the distribution and popularity of content published and accessed through IPFS and the access patterns to that content.

More recent studies on peer-to-peer systems include blockchain systems such as Bitcoin and Ethereum. Here we highlight two studies [15, 19], that focus on the transactions made within Bitcoin and Ethereum blockchains. These studies characterize the performance of blockchain systems but fail to provide insights over system network properties, such as the number of peers per region or workload distribution over peers. Our study complements these by focusing on IPFS, an increasingly relevant building block in the Web3 ecosystem.

IPFS has been the subject of several recent studies [2, 14] that are complementary to our own study, where the authors analyze peer distribution over geographical regions and Bitswap traffic inside IPFS. Furthermore, IPFS was extensively studied in [34], where the authors analyze the performance of the system in general. Our own study complements the previous findings through an analysis on the geographical relationship between IPFS web clients and content providers that previous studies did not accomplish, with the aim to characterize client access patterns to guide future research on IPFS and decentralized Web3 systems in general.

6 Conclusion

In this paper we presented a study over the traffic processed by one of the most popular public IPFS gateways to identify characteristics of the workload of a popular decentralized system and understand if IPFS would benefit from DHT designs that optimize a content sharing network assuming that there exists geographic locality on access patterns. In our study, we observed that the IPFS gateway mainly processes requests incoming from North America and Asia that target mostly the same content, independently of the location of the requester. To understand where was the content provided, we queried the network for the content provider records and discovered surprisingly that the most popular content fetched through the public IPFS gateway is provided by only a few nodes in the network. Our results suggest that IPFS is an imbalanced system centered on these few provider nodes, which would not benefit from a DHT design that access patterns follow geographic locality, as the access patterns seem to be driven by content popularity rather than geographic interest. On the other hand, this also points to studying novel load balancing schemes on IPFS that encourage IPFS (server) users to replicate popular content. As future work, we plan to extend our study to other public IPFS gateways as well as other Web3 networks, such as Ethereum Swarm and StorJ, to understand if our findings are generalizable to other decentralized Web3 systems. Furthermore, we plan to complement this study with a workload generator that produces client requests based on the observations of our study, to enable the research and practitioners community to evaluate Web3 (prototype) systems under realistic workloads.

References

1. F. Araujo and L. Rodrigues. Geopeer: a location-aware peer-to-peer system. In *Third IEEE International Symposium on Network Computing and Applications, 2004. (NCA 2004). Proceedings.*, pages 39–46, 2004.
2. Leonhard Balduf, Sebastian Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. Monitoring data requests in decentralized data storage systems: A case study of ipfs. In *2022 IEEE 42nd ICDCS*. IEEE, 2022.
3. Juan Benet. IPFS - Content Addressed, Versioned, P2P File System. Technical Report Draft 3, 2014.
4. Berty. Berty: The privacy-first messaging app. `https://berty.tech/`, 2022. Accessed October 2022.
5. ConsenSys. Consensys: Ipfs look up measurement. `https://github.com/ConsenSys/ipfs-lookup-measurement/`, 2022. Accessed February 2022.
6. Pedro Ákos Costa, João Leitão, and Yannis Psaras. Anonymised IPFS gateway logs from 7th of March of 2022 to 21st of March of 2022. `https://doi.org/10.5281/zenodo.7876622`, April 2023.
7. Matteo D'Ambrosio, Christian Dannewitz, Holger Karl, and Vinicio Vercellone. Mdht: A hierarchical name resolution service for information-centric networks. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ICN '11, page 7–12, New York, NY, USA, 2011. Association for Computing Machinery.
8. Alfonso de la Rocha, David Dias, and Psaras Yiannis. Accelerating content routing with bitswap: A multi-path file transfer protocol in ipfs and filecoin. Technical report, 2021.
9. Discussify. Discussify. `https://github.com/ipfs-shipyard/pm-discussify`, 2022. Accessed October 2022.
10. Fleek. Fleek: Build on the new internet. `https://fleek.co/`, 2022. Accessed October 2022.
11. BitTorrent Foundation. Bittorrent (btt) white paper. `https://www.bittorrent.com/btt/btt-docs/BitTorrent_(BTT)_White_Paper_v0.8.7_Feb_2019.pdf`, 2019.
12. Bernd Grobauer, Tobias Walloschek, and Elmar Stocker. Understanding cloud computing vulnerabilities. *IEEE Security & Privacy*, 9(2), 2011.
13. Christian Gross, Dominik Stingl, Björn Richerzhagen, Andreas Hemel, Ralf Steinmetz, and David Hausheer. Geodemlia: A robust peer-to-peer overlay supporting location-based search. In *2012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P)*, pages 25–36, 2012.
14. Sebastian Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. Mapping the interplanetary filesystem. In *2020 IFIP Networking Conference (Networking)*, 2020.
15. Hwanjo Heo and Seungwon Shin. Behind block explorers: Public blockchain measurement and security implication. In *2021 IEEE 41st ICDCS*, 2021.
16. Gaurish Korpal and Drew Scott. Decentralization and web3 technologies. Technical report, 5 2022.
17. A. Kovacevic, N. Liebau, and R. Steinmetz. Globase.kom - a p2p overlay for fully retrievable location-based search. In *2007 7th International Conference on Peer-to-Peer Computing*, pages 87–96, Los Alamitos, CA, USA, sep 2007. IEEE Computer Society.

18. Sarvesh Kumar, Himanshu Gautam, Shivendra Singh, and Mohammad Shafeeq. Top vulnerabilities in cloud computing. *ECS Transactions*, 107(1), apr 2022.
19. Xi Tong Lee, Arijit Khan, Sourav Sen Gupta, Yu Hann Ong, and Xuan Liu. Measurements, analyses, and insights on the entire ethereum blockchain network. In *Proceedings of The Web Conference 2020*, WWW '20, 2020.
20. J. Leitão. Gossip-based broadcast protocols. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2007.
21. João Leitao, João Pedro Marques, José Orlando Pereira, and LuÍs Rodrigues. X-bot: A protocol for resilient optimization of unstructured overlays. In *2009 28th IEEE International Symposium on Reliable Distributed Systems*, pages 236–245, 2009.
22. João Leitão, João Pedro Marques, José Orlando Pereira, and Luís Rodrigues. X-bot: A protocol for resilient optimization of unstructured overlay networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(11):2175–2188, 2012.
23. João Leitão. *Topology Management for Unstructured Overlay Networks*. Phd thesis.
24. MaxMind. Maxmind - geolite2 free geolocation data. `https://dev.maxmind.com/geoip/geolite2-free-geolocation-data?lang=en`, 2022. Accessed October 2022.
25. Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*. Springer, 2002.
26. J. Monteiro, P. A. Costa, J Leitão, A. de la Rocha, and Y. Psaras. Enriching kademlia by partitioning. In *Proceedings of the 1st Workshop on Decentralized Internet, Networks, Protocols, and Systems (DINPS'22) colocated with ICDCS*, 2022.
27. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 2008.
28. NFT.STORAGE. Nft storage: Free storage for nfts. `https://nft.storage/`, 2022. Accessed October 2022.
29. Johan Pouwelse, Paweł Garbacki, Dick Epema, and Henk Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *International workshop on peer-to-peer systems*, pages 205–216. Springer, 2005.
30. Protocol Labs. libp2p: A modular network stack. `https://libp2p.io`, 2022. Accessed February 2022.
31. Saurabh Ratti, Behnoosh Hariri, and Shervin Shirmohammadi. Nl-dht: A non-uniform locality sensitive dht architecture for massively multi-user virtual environment applications. In *2008 14th IEEE International Conference on Parallel and Distributed Systems*, pages 793–798, 2008.
32. Rodrigo Rodrigues and Peter Druschel. Peer-to-peer systems. *Commun. ACM*, 53(10), October 2010.
33. Subhabrata Sen and Jia Wang. Analyzing peer-to-peer traffic across large networks. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, 2002.
34. Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. Design and evaluation of ipfs: A storage layer for the decentralized web. In *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, 2022.
35. Uniswap. Uniswap protocol: swap, earn, and build on the leading decentralized crypto trading protocol. `https://uniswap.org/`, 2022. Accessed October 2022.
36. Gavin Wood. Ethereum: a secure decentralised generalised transaction ledger. Technical report, 2014.