Decentralised Storage

Anurag Dashputre
Senior Developer, ConsenSys India

Agenda

What is Decentralisation + Storage?

Decentralisation is understood as the transfer of authority from a central entity to a more localised and ‘liberal’ system.

Storage is defined as the retention of retrievable data on a computer or other electronic system.

What is Decentralised Storage?

Decentralised storage is a system of being able to store your files without having to rely on large, centralized silos of data that don’t undermine important values such as privacy and freedom of your information.

It is Content-Addressable, rather than Location-Addressable. Every file has a unique hash of its content.

Why Decentralised Storage?

- Availability
  - Censorship Resistant
  - Data geographically spread
  - No "404 Page Not found" error
- Security & Privacy
  - No centralised server storage hence very difficult to hack and breach data
  - Files are not stored directly but as chunks of data spread across multiple nodes
- Cost reduction due to more efficiency

Decentralised Storage - Current Projects

- Swarm
- IPFS (Inter Planetary File System)
- Sia
- Storj

Swarm

Swarm is a distributed storage platform and content distribution service, a native base layer service of the ethereum *web3* stack.

The primary objective of Swarm is to provide a sufficiently decentralized and redundant store of Ethereum's public record, in particular to store and distribute dapp code and data as well as blockchain data.

Refer Swarm Documentation for further details

IPFS - Deep Dive

What is IPFS?

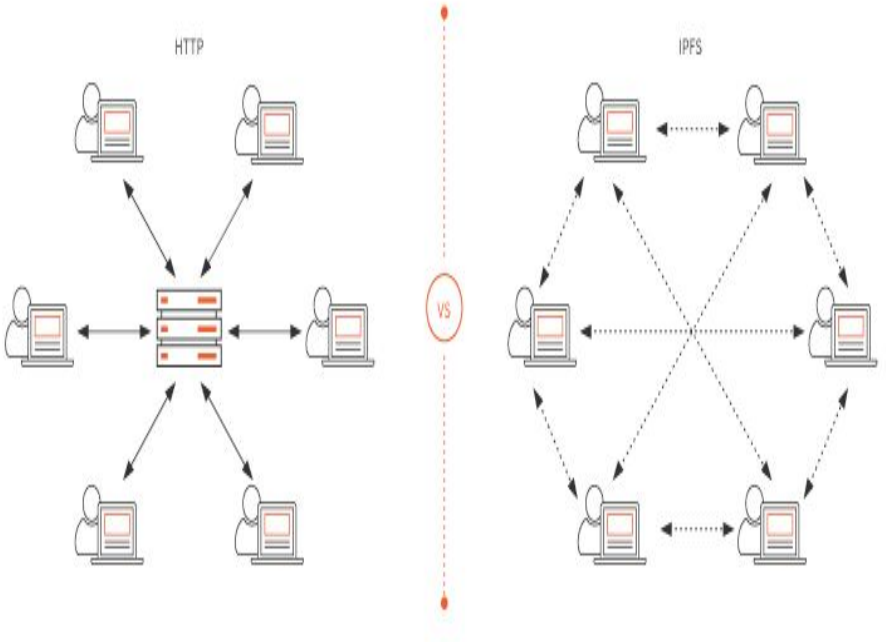
IPFS is a distributed peer-to-peer (p2p) file sharing system for storing and accessing files, websites, applications, and data.

IPFS aims to replace HTTP and build a better web for all of us.

HTTP v/s IPFS

Today, the Internet is based on HyperText Transfer Protocol (HTTP). **HTTP** relies on location addressing which uses IP addresses to identify the specific server that is hosting the requested information. This means that the information has to be fetched from the origin server.

IPFS is meant to be a replacement for HTTP. Most notably, IPFS never has a **single point of failure**. It's a peer-to-peer distributed file system that would decentralize the Internet and make it much more difficult for a service provider or hosting network to pull the plug and make published information suddenly disappear.



HTTP vs. IPFS [Source: https://www.maxcdn.com/one/visual-glossary/interplanetary-file-system/]

How IPFS works?

IPFS works by connecting all computing devices with the same system of files via a system of nodes. It uses a “distributed hash table, an incentivized block exchange, and a self-certifying namespace.”

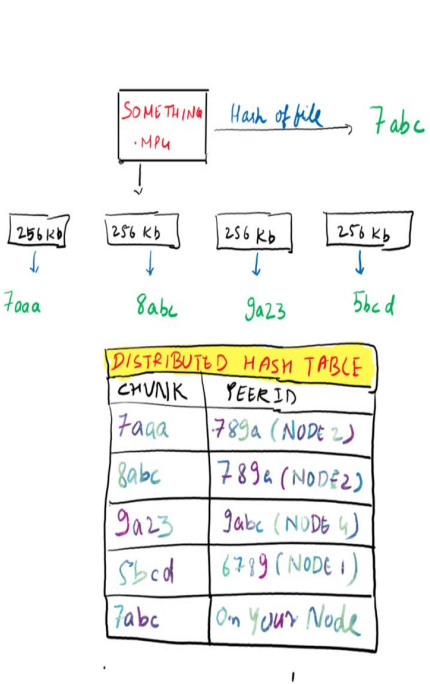
In simpler terms, it acts similarly to a torrent system, except that instead of sharing and exchanging media, IPFS exchanges git objects. This means that the whole system is based around a simple key-value data store. Any type of content can be inserted, and it will give back a key that can be used to retrieve the content again at any time. This is what allows for content addressing instead of location addressing: The key is completely independent of the origin of the information and can be hosted anywhere.

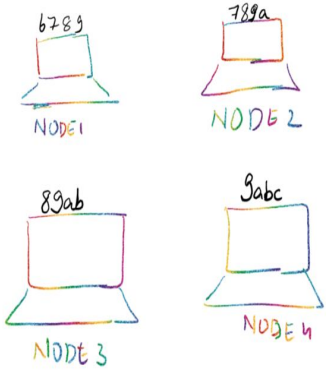
How IPFS stores data?

When you add any content on IPFS network, the data is split into chunks of 256Kb. Each chunk is identified with it’s own hash. These chunks are then distributed to various nodes on network which have there hash closest to peerId.

How IPFS stores data? ...Continued

1. Let us assume that there are 4 nodes with peerId *6789*, *789a*, *89ab*, *9abc* respectively
2. We try to add a file name(size= 1Mb) *something.mp4*. Your node first calculates that hash of the file, say *7abc*. Additionally the file is broken into 4 chunks of 256 Kb each. Your node then calculates the hash of the each chunk, say (*7aaa*, *8abc*, *9a23*, *5bcd*)
3. Now node broadcasts the each chunk to node with has the closest peerId numerically. In our mentioned example chunk with hash *7aaa* it closest to hash *789a*. Hence this chunk is send to node with peerId *789a*.
4. Similarly, all chunks are send and there address in updated in DHT.
5. Lastly, the object root hash i.e *7abc* is stored, (*Root hash can be stored anywhere, it is assumed that in current example it is stored in our system)* and hashes that it links to *i.e 7abc → [7aaa*, *8abc*, *9a23*, *5bcd]*





[Source: https://medium.com/@akshay_111meher/how-ipfs-works-545e1c890437]

How data is divided and stored. Root hash is assumed to be stored on your node. It is however stored in same way chunks are stored. It could be on anyone, including yours.

IPFS File Storage - Live Demo

How data is retrieved?

On IPFS network, the file is identified solely by it HASH (root hash), in our case ***7abc.*** Once the user requests a file, the request traverses to nodes where hash is existing using the DHT. If the data points to other chunks (***like in our case***), even they are searched same way. Once all chunks are obtained, all of them are simply concatenated to obtain the main object.

Basic Commands

- Install IPFS
- Get IPFS version
- Initialize the IPFS repository
- Get IPFS node id
- Start IPFS node
- Check Peer Nodes

Get IPFS version

- Install IPFS
- **Get IPFS version**
- Initialize the IPFS repository
- Get IPFS node id
- Start IPFS node
- Check Peer nodes

```
ipfs version
ipfs version 0.4.18
```

Initialize the IPFS repository

- Install IPFS
- Get IPFS version
- **Initialize the IPFS repository**
- Get IPFS node id
- Start IPFS node
- Check Peer nodes

```
ipfs init
initializing ipfs node at
/Users/jbenet/.go-ipfs
generating 2048-bit RSA
keypair...done
peer identity:
Qmcpo2iLBikrdf1d6QU6vXuNb6P7hwrbNPW9
kLAH8eG67z
to get started, enter:

  ipfs cat
/ipfs/QmYwAPJzv5CZsnA625s3Xf2nemtYgP
pHdWEz79ojWnPbdG/readme
```

Get IPFS node id

- Install IPFS
- Get IPFS version
- Initialize the IPFS repository
- **Get IPFS node id**
- Start IPFS node
- Check Peer nodes

```
ipfs id

"ID":
"QmP7JssmhNTpayGoK5ZhBt78hRRBi3VBYQ
yqwMsBsSZBSW"
```

Start IPFS node

- Install IPFS
- Get IPFS version
- Initialize the IPFS repository
- Get IPFS node id
- **Start IPFS node**
- Check Peer nodes

```
ipfs daemon

Initializing daemon...

go-ipfs version: 0.4.18-

Repo version: 7

System version: amd64/darwin

Golang version: go1.11.1

Successfully raised file descriptor limit to 2048.

Swarm listening on /ip4/127.0.0.1/tcp/4001

.

Swarm announcing /ip4/127.0.0.1/tcp/4001

.

Daemon is ready
```

Check peer nodes

- Install IPFS
- Get IPFS version
- Initialize the IPFS repository
- Get IPFS node id
- Start IPFS node
- **Check Peer nodes**

```
ipfs swarm peers
```

Advanced Commands

- Check IPFS repository statistics
- Add a file to IPFS
- Pin objects to local storage
- Remove pinned objects from local storage
- Download IPFS objects
- Show IPFS object data
- List objects pinned to local storage

Check IPFS repository statistics

- **Check IPFS repository statistics**
- Add a file to IPFS
- Pin objects to local storage
- Remove pinned objects from local storage
- Download IPFS objects
- Show IPFS object data
- List objects pinned to local storage

```
ipfs stats repo

NumObjects: 4817

RepoSize:   127963949

StorageMax: 10000000000

RepoPath:   /Users/anuragd/.ipfs

Version:    fs-repo@7
```

Add a file to IPFS

- Check IPFS repository statistics
- **Add a file to IPFS**
- Pin objects to local storage
- Remove pinned objects from local storage
- Download IPFS objects
- Show IPFS object data
- List objects pinned to local storage

```
ipfs add temp

added
QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy
47CnJDgvs8u temp

 12 B / 12 B
[================================
================================
================================
==============================]
100.00%
```

Pin objects to local storage

- Check IPFS repository statistics
- Add a file to IPFS
- **Pin objects to local storage**
- Remove pinned objects from local storage
- Download IPFS objects
- Show IPFS object data
- List objects pinned to local storage

```
ipfs pin add
QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy
47CnJDgvs8u

pinned
QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy
47CnJDgvs8u recursively
```

Remove pinned objects from local storage

- Check IPFS repository statistics
- Add a file to IPFS
- Pin objects to local storage
- **Remove pinned objects from local storage**
- Download IPFS objects
- Show IPFS object data
- List objects pinned to local storage

```
ipfs pin rm
QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy
47CnJDgvs8u

unpinned
QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy
47CnJDgvs8u
```

Download IPFS objects

- Check IPFS repository statistics
- Add a file to IPFS
- Pin objects to local storage
- Remove pinned objects from local storage
- **Download IPFS objects**
- Show IPFS object data
- List objects pinned to local storage

```
ipfs get
QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy
47CnJDgvs8u

Saving file(s) to
QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy
47CnJDgvs8u

 20 B / 20 B
[================================
================================
================================
============================]
100.00% 0s
```

Show IPFS object data

- Check IPFS repository statistics
- Add a file to IPFS
- Pin objects to local storage
- Remove pinned objects from local storage
- Download IPFS objects
- **Show IPFS object data**
- List objects pinned to local storage

```
ipfs cat
QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy
47CnJDgvs8u

Hello World

ipfs cat
QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy
47CnJDgvs8u > temp1

cat temp1

Hello World
```

List objects pinned to local storage

- Check IPFS repository statistics
- Add a file to IPFS
- Pin objects to local storage
- Remove pinned objects from local storage
- Download IPFS objects
- Show IPFS object data
- **List objects pinned to local storage**

```
ipfs pin ls
```

Demo

Appendix

- https://github.com/ethereum/go-ethereum/tree/master/swarm
- https://docs.ipfs.io/introduction/usage/
- https://itnext.io/build-a-simple-ethereum-interplanetary-file-system-ipfs-react-js-dapp-23ff4914ce4e
- https://github.com/mcchan1/eth-ipfs

Thank You