

Deliverable 2 – Testing of existing file storages

1. IPFS

IPFS is a filesystem which provides a client user interface along with a command line client and also provides great support to developers trying to leverage the filesystem for their applications. It provides limited free storage which comes without any guarantee of persistence of the data. Following is a test case which records the time required to store a file on IPFS and retrieve it.

```
import io.ipfs.api.MerkleNode;
import io.ipfs.api.NamedStreamable;
import io.ipfs.multihash.Multihash;

import java.io.File;
import java.io.IOException;

public class IPFS {

    public static void main(String[] args) {
        long start = 0;
        long end = 0;
        io.ipfs.api.IPFS ipfs = new io.ipfs.api.IPFS( multiaddr: " /ip4/128.0.0.1/tcp/5001 ");
        try {
            File file1 = new File( pathname: " /tmp/testfile.txt ");
            System.out.println("File size: " + file1.length()/1024+ " KB");
            start = System.currentTimeMillis();
            NamedStreamable.FileWrapper file = new NamedStreamable.FileWrapper(file1);
            MerkleNode response = ipfs.add(file).get(0);
            end = System.currentTimeMillis();
            long uploadTime = end - start;
            System.out.println("Time required to upload file to the node: " + uploadTime + " milli Seconds");
            System.out.println(response.hash.toBase58());
            String hash = response.hash.toBase58();
            Multihash multihash = Multihash.fromBase58(hash);
            byte[] content = null;
            start = System.currentTimeMillis();
            while(content == null) {
                content = ipfs.cat(multihash);
            }
            end = System.currentTimeMillis();
            long getFileTime = end - start;
            System.out.println("Content of " + hash + ": " + new String(content).substring(0,10));
            System.out.println("Time required to get file from hash: " + getFileTime + " milli Seconds");
            // ipfs.
        } catch (IOException ex) {
            throw new RuntimeException("Error whilst communicating with the IPFS node", ex);
        }
    }
}
```

```

Library\Java\VirtualMachines\jdk-11.0.16_jdk\Contents\Home\bin\java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=49837:/Applications/IntelliJ IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8
-classpath /Users/ajinkyarajguru/Documents/CS222/FileSystemPerformance/target/classes:/Users/ajinkyarajguru/.m2/repository/com/github/ipfs/java-ipfs-http-client/1.3.3/java-ipfs-http-client-1.3.3.jar:/Users/ajinkyarajguru/.m2/repository/com/github/multiformats/java-multihash/v1.4.1/java-multihash-v1.4.1.jar:/Users/ajinkyarajguru/.m2/repository/com/github/ipfs/java-cid/v1.3.1/java-cid-v1.3.1.jar:/Users/ajinkyarajguru/.m2/repository/com/github/multiformats/java-multihash/v1.3.0/java-multihash-v1.3.0.jar:/Users/ajinkyarajguru/.m2/repository/com/github/multiformats/java-multibase/v1.0/java-multibase-v1.0.jar:/Users/ajinkyarajguru/.m2/repository/io/storj/libstorj-java/0.8.3/libstorj-java-0.8.3.jar IPFS
File size: 993310 KB
Time required to upload file to the node: 6979 milli Seconds
Qmc4eHxYFwzcFCPQT1jWNBvZrdtAZiGqg8MSWoYfaUHQ
Content of Qmc4eHxYFwzcFCPQT1jWNBvZrdtAZiGqg8MSWoYfaUHQ: 0110 =
Time required to get file from hash: 5363 milli Seconds

Process finished with exit code 0

```

2. Storj

Storj is a cloud based decentralized filesystem based on blockchain which uses storj tokens. Along with the web browser interface it provides a command line client which helped me to time the upload process for a file.

Storj is a cloud based decentralized filesystem based on blockchain which uses storj tokens. Along with the web browser interface it provides a command line client which helped me to automate the upload process for a file.

[illegible]

As displayed in the terminal it nearly took 751 seconds i.e more than 12 minutes to store a 390 MB file.

While accessing the file it provides information regarding number of nodes storing the piece of the file and its approximate locations over the globe.



**You're getting this
file from all over
the world.**


Storj Decentralized Cloud Storage splits files into smaller pieces, then distributes those pieces over a global network of Nodes and recompiles them securely on download. This map shows the real-time locations of this file's pieces.

452 Storj nodes are storing a piece of this file




3. Sia and SWARM

Sia and SWARM require a few available tokens to store on their systems. No open-source developer supported options were available to execute performance testing. Swarm provides a simple gateway to store a file up to the size 10 MB.




SWARM GATEWAY

The easiest way to share & access files on the Swarm network.



SHARE




ACCESS

The Swarm Gateway is graciously provided by the Swarm Foundation. This service is under development and provided for testing purposes only. For unlimited use of the Swarm network consider running your own node.

[Swarm Website](#) [FAQ](#) [Ecosystem](#) [Github](#) [Terms & Conditions](#)


FILE

Please double-check before uploading, there is no undo.



Filename: retail.dat
Type: Unknown file type
Size: 4.17 MB

By uploading you agree to the Swarm Gateway [Terms & Conditions](#).



UPLOAD

SHARE


Share your file with a simple web link, or a Swarm hash.

WEB LINK

SWARM HASH

<https://gateway.ethswarm.org/access/857fc4e91e98e20cab4aef3e452f343518db15b1afdf92064ff77f740256a312>

It may take some time for your file to become available on the Swarm network. Please keep in mind that this service is provided for testing purposes only. There's no guarantee of availability.



COPIED

SHARE


Share your file with a simple web link, or a Swarm hash.

WEB LINK

SWARM HASH

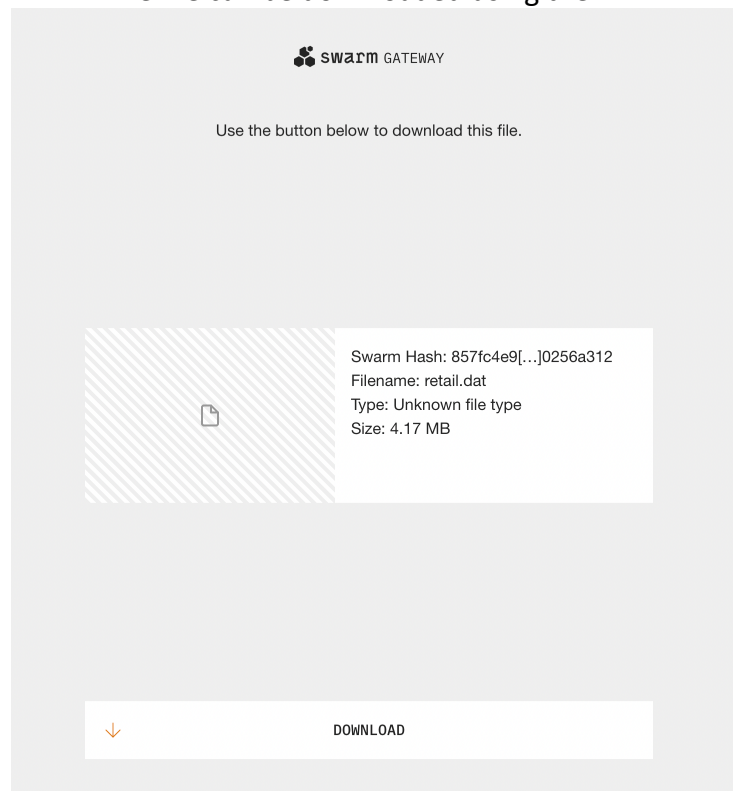
857fc4e91e98e20cab4aef3e452f343518db15b1afdf92064ff77f740256a312

It may take some time for your file to become available on the Swarm network. Please keep in mind that this service is provided for testing purposes only. There's no guarantee of availability.



COPY LINK

The file can be downloaded using the link:



In conclusion IPFS performs better than Storj when it comes to uploading a file as big as 400 MB or 1 GB. It took merely a few seconds for IPFS to store a 400 MB file and about a couple of minutes to store a 1 GB file. On the contrary Storj nearly took more than 12 minutes to store a 400 MB file.