Lab IPFS

October 28, 2024

InterPlanetary File System (IPFS) is a P2P network that realize decentralized data management. It can be used to deploy websites, distribute content, and store data.

In this lab, you will discuss the basic concepts underlying IPFS, customize the IPFS daemon, and review some implementation details of IPFS. At the end of this lab, you need to write a report (max 8 pages) and upload your code to GitLab.

1 Use IPFS to upload and download files (week 1)

Currently, IPFS has multiple implementations. In this lab, we focus on the go implementation - KUBO. As the first step of this lab, you need to download the source code of KUBO from https://github.com/ipfs/kubo/ and build an executable file of KUBO from the source.

Milestone 1: compile kubo from the source

With the built executable file, you need to start a daemon process by 'IPFS daemon'. You can read the document of IPFS to learn how to decide the parameters of the IPFS daemon. If the IPFS daemon is started, it will connect to some bootstrap nodes by default and connect to more peers with the help of bootstrap nodes. Then, you need to upload a file using 'ipfs add filename', which will generate a content identifier (CID) for your file. Later, you should obtain this file from other nodes using 'ipfs cat CID'.

Milestone 2: upload a file using IPFS daemon

2 Customize the IPFS daemon (week 2-5)

You may find that an IPFS daemon is trying to connect to other nodes continuously in the first task. In this task, you are supposed to turn off the bootstrap process by changing the config files and modifying the source code since we want you to construct a private network later.

The IPFS daemon can be configured by modifying the "config" file in its default directory. Read this documentation to learn how to configure the boot-

strapping and connection management. By doing this, you can remove those default bootstrap nodes before the start of an IPFS daemon.
Milestone 3: delete all bootstrap nodes

Unfortunately, changes of config files are not enough to turn off the bootstrap function. The reason is that IPFS enables an mDNS function where nodes continuously find nodes in the local network and connect to them. Although you can set "Discovery.MDNS.Enabled" to false in the config file, it does not stop the bootstrap function entirely. In some cases, a node is uploading a file for example, this node will still connect to other nodes in the local network. This is because every time a new uploading or downing event happens, the IPFS daemon will store the corresponding addresses in a local DHT table and this DHT table is used to route contents. Hence, you need to inspect the source code of KUBO first to see where new connections are constructed. Then, you can modify those codes carefully to stop the automatic new connection construction without influencing the basic function of IPFS (hints: you should pay special attention to go-libp2p-kad-dht). After your modifications, you can rebuild the IPFS daemon from the source. You should now have a version of the daemon that does not automatically start the bootstrap process.
Milestone 4: make IPFS nodes never connect to other nodes automatically

3 Create an IPFS network with specific topologies (week 6-7)

By including a "swarm.key" file in the IPFS's default directory, you are able to run a private IPFS network since only peers with the same "swarm.key" can be connected. If you have built your own IPFS daemon without the bootstrap function. You can create a topology using "ipfs swarm connect [address]" command. You can find more information about the 'ipfs swarm' command on this website.

In this task, we want you to create a private IPFS network for a ring, grid, complete graph, and random graph(Barabási–Albert model) topology with more than 10 nodes.
Milestone 5: create a private IPFS network with a specific topology

4 Simulations of IPFS networks (week 8-10)

Using docker-compose, you are able to run multiple IPFS peers simultaneously (recommended). You can also try to start multiple IPFS daemons locally, but it is more complicated based on my experience. To configure containers for IPFS daemons, you need to write a "docker-compose.yml" file. Here is a template for it, you can modify it according to your needs (if you have changed something in some libraries you will probably need to load them from local directory).
Milestone 6: use docker to run your customized IPFS daemon

After containers are initialized, you can control them by passing commands to docker. Considering there are too many containers, you also need to write a wrapper for those commands using any language you like. As in task 3, you need to connect those IPFS nodes based on specific topologies at first.
Milestone 7: write a IPFS network simulator

Afterwards, you need to simulate the behaviors of IPFS peers. More specifically, you need to let them upload and download some files periodically (you need to vary the size of files from 1mb to 500mb). Additionally, you should do the same simulation with the original IPFS daemons. You do not need to connect peers manually since they are automatically connected with the original IPFS daemon. It is important to record the profiling data of Docker during simulations since you need to analyze them in your report. You need to record the size of successfully uploaded and downloaded files of peers, CUP usage, network bandwidth usage, and memory usage of every node. At last, you need to draw a figure about the average value of each metric for your later report.
Milestone 8: profile different topologies and the original IPFS daemon

With the obtained performance data of different topologies, you need to compare them and explain why they perform differently in your report.

5 Simulations with IPFS clusters (bonus)

IPFS cluster provides a way to replicate and track data across a cluster of IPFS daemons. You should create a private IPFS cluster network and simulate file transferring between peers like in Task 3.

6 Grading criteria

6.1 Implementation

1pt: build the IPFS daemon from the source code and upload a file successfully.

3pt: turn off the bootstrap function correctly by modifying the source code of IPFS

2pt: construct private IPFS networks with the ring, grid, complete graph, and random graph(Barabási–Albert model) topology.

2pt: simulate the uploading and downloading behaviors of IPFS peers correctly

2pt: clear and complete profiling of peers

2pt: reasonable explanation of results

6.2 Report

We have three criteria to grade your report. For each criterion, you can obtain 0-3 pts based on table1.

0pt	not included or explained at all
1pt	incomplete results or lack of explanation
2pt	good results and explanation with only a few unclear points
3pt	perfect explanation with reasonable results

Table 1: Grade of report

6.2.1 Basic ideas of IPFS

Based on the lessons you learned from task 1 and task 2, you need to explain how a file is uploaded and stored, and what happens after the "ipfs cat" command. Secondly, you need to explain how connections between peers are maintained and how IPFS utilizes DHTs (technologies, execution process...) based on the source code.

6.2.2 Implementation

You need to explain the modifications you make to the source code and why these modifications achieve the desired goals. Then, you should elaborate on how you simulate the behaviors of peers, the setup of simulations, and the simulated topologies.

6.2.3 Results

You need to display your results in a clear and understandable way with concrete numbers. Based on the results and profiles you obtained from simulations, you need to compare the performance of different topologies and explain the differences. Then, you need to discuss which topologies result in benefits to network performance. If you also run the IPFS-cluster with multiple peers and compare it with previous results (using a private network), you can get 2 extra points.

7 Links

https://docs.ipfs.tech/
https://docs.libp2p.io/concepts/
https://github.com/ipfs/kubo/
https://github.com/ipfs/kubo/blob/master/docs/config.md
https://dweb-primer.ipfs.io/going-online/find-peers
https://docs.docker.com/compose/
https://raw.githubusercontent.com/ipfs-cluster/ipfs-cluster/master/docker-compose.yml
https://ipfscluster.io/documentation/