IIT Kanpur

CS425 Course Project

Peer to Peer File Distribution System

Nayan Deshmukh
Nilesh Vasita

supervised by
Prof. Dheeraj Sanghi

November 12, 2017

Contents

1 Introduction

As the title suggests, we have implemented a peer to peer distributed file storage system in our project. The basic idea behind the project is to shift the storage of data from a centralized data-center model to a decentralized model in a peer to peer network. The applications related to our project are:

- Apart from distributed storage, the model can serve the purpose of distributed computing as well.
- Allows better utilization of peer-to-peer storage by adding a reliability factor due to the presence of a centralized unit in addition to the peers.
- The model ensures better download and upload speeds by exploiting the distributed organization of the file allowing for the parallelization of the upload and download if the model is used in cloud storage systems.
- Efficiently distributing the files ensures more granular usage of space on the storage devices.

In our model, we have a client, a main server and storage nodes. The client can access his files from any machine which is connected to the peer to peer network. The primary role of the main server is to keep a track of all the clients, storage nodes and the existence of all the files in the network. The storage nodes act as storage servers which serve the files to the client.

2 Objective

To decentralize the existing data storage models and reduce the use of centralized data centers. Files can be stored on multiple nodes connected to the network. Our project tends to meet the following objectives:

- A client can upload any file from anywhere after proper authentication.
- A client can download any file from his account after proper authentication.
- The main server keeps a track of all the files stored by the storage nodes, availability of storage space on various nodes and which all nodes are presently active in the network.
- The main server also keeps a track of all the active nodes. If a node goes down, then the main server instructs other nodes to make more copies of the data which was stored on the node which went down. This ensures that a constant number of copies of all the files are available in the network.
- Token validation in client to node and node to node interactions.
- The network can serve files even if some of the storage nodes go down. This is ensured by keeping duplicates and maintaining the degree of duplicates (by copy operation) whenever a storage node goes down.

3 Assumptions

We have made the following assumptions in the present implementation of our project:

- The main server will never go down.
- There is no adversary in the network and everyone sticks to the given protocol.
- We have not implemented a user authentication and have assumed that the client corresponds to a single user. This is a very simple feature which can be easily incorporated into our design.
- A user only uses a single machine as a client. Because there is certain meta-data which is only available to the client which is not uploaded to the network and hence the same can only access his files from the same client.

4 Architecture

4.1 Nodes

The architecture of our project includes the implementation of the following:

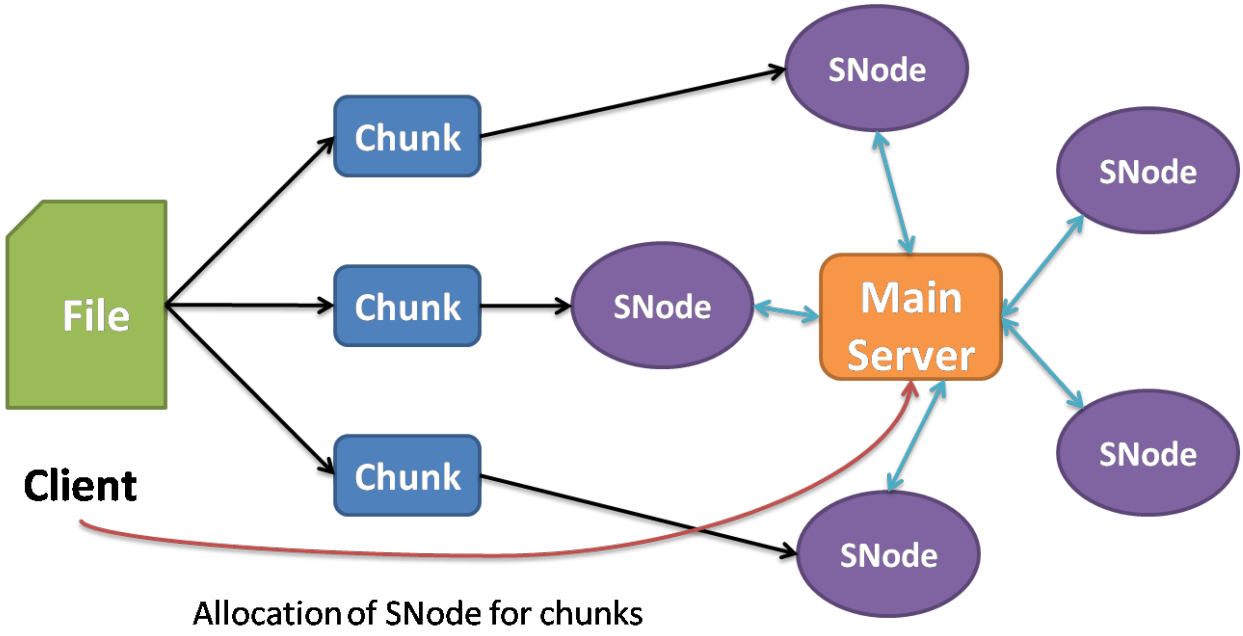

Figure 1: The basic architecture

- **Client:** The program which is used to upload and download file form and to the storage nodes.
- **Main Server:** The program which ensures reliability and does the bookkeeping business in our model. In other words, it is responsible for allocating storage nodes to the client for uploading and downloading a file, keeps a track of the storage node and ensures that there is a backup available for all the data stored on a storage node.
- **Storage Node:** This program stores the files it receives from a client. Also, it regularly updates the main server about the space available with it for further utilization. The details about the program are mentioned in the PROTOCOL below.

4.2 PROTOCOL

P2P-DFS protocol

We have the following connections between the client, main server and the storage nodes as described in Figure 1:

- **Client to Main Server:**
  - **Allocate Storage Node**: In this connection, the client first sends the number of chunks it wants to upload to the storage nodes and then it send hashes of all the chunks to the Main Server. In return, the main server sends back the IP Addresses of the storage nodes where these chunks need to be uploaded.
    Note: Presently the main server uses the Round-robin method for allocation of the storage node IP addresses to the client.
  - **Get Storage Node:** Here the client simply sends the hash of a chunk to the main server and then, in response, the main server sends back the IP Address where the given chunk is uploaded.
- **Client to Storage Node:**
  - **Upload a File:** In this connection, the client sends the hash, token and the chunk to the storage node. If the token validates at the storage node, then the chunk is accepted otherwise it is rejected.
  - **Get File:** In this link, the client sends the hash of a chunk to the storage node. The storage node then checks if it has the given hash in its storage space and then returns the chunk data if the hash is found in the storage space.
- **Storage Node to Main Server:**
  - **Add Me:** In this connection any storage node requests the main server to get added in the p2p network. The Storage node also informs the main server about the storage space it is offering.
  - **Drop Me:** Here, the storage simply notifies the main server that it is going down.
  - **Confirmation of the file uploaded by the client:** In this link, the storage node notifies the main server that the file is uploaded by a client. The main server then sends back the IP addresses of storage nodes where it wants the current storage node to duplicate the file it has received from the client.
    Also, the main server sends a token to the storage nodes whose IP it provides to the current node for verification purposes.
  - **File duplicated successfully:** In this connection the storage node notifies the main server that the above task of duplication is done successfully.
- **Main Server to Storage Node:**
  - Inform the storage node about the token from a client. The token validates the fact that the client which i trying to upload a file on the current node is verified by the main server.
- **Storage Node to Storage Node:**
  - **Upload a chunk for duplication:** In this connection, the storage node sends the chunk to other storage nodes for duplication. During this, the storage node sends the hash, token and the chunk data to the other storage node.

5 Implementation Environment

We have used python as the primary language for the implementation. Python allows quick prototyping of the ideas and allows the use of more abstract features than C. We have used the SocketServer module

for Main Server and Storage node which allows handling of requests at a more abstract level. The client only uses basic socket programming APIs. The model doesn't assume anything about the physical machines on which the main server, storage node, and client are running. Any machine running a Linux distro can potentially play the any/all of the three roles.

6 Summary

We have given a proof of concept of a peer-to-peer storage service which involves a centralized entity to ensure a certain amount of reliability. This allows the peer-to-peer model to provide service which is on par with centralized vendors like Google Drive, Dropbox, etc. The main server is responsible for ensuring reliability by checking the status of the storage nodes and if a node is down then making more copies of the data that was stored in those nodes. In our implementation, the main server does not check the storage nodes to check if they are up. One other limitation is that a user can only use a single machine as a client because there is certain meta-data which is only available to the client which is not uploaded to the network and hence the same user can only access his files from the same client.

7 Future Work

- Incorporate user authentication.
- Upload all meta-data online to allows remote access to files from any machine.
- Better distribution algorithm for the main server to distribute the IP Addresses of the storage nodes to the client.
- Maintaining a constant number of copies of all the files in the network even if any of the node turns down.

8 Group Members Details

Group-35

Member1	Member2
Nayan Deshmukh	Nilesh Vasita
14418	150455