

For reference.

Tymon Schroeder

Sunny Yeung

Hailey Nam

Glenn Wintrode

**Implementation details:**

**Replication factor:**  3

**Blocks:**  64 MB

Our nodes communicate with each other through JSON messages using HTTP requests

A typical json would be

**JSONFILE:**

Status message/error:

Type of action ie delete, create file,

Filepath

Block size

DataNodes (pass ips)

**Platforms used:**

EC2 for distribution and nodes

Flask as the framework for EC2

Python

S3 for the file paths

**Client**:

/\*

The client uses the flask framework and is used by the user as a website to send request to our SUFS namenodes and datanodes

\*/

Fields:

functions:

// these functions are needed for saving a file in the file system

create\_directory is include in the createFile function.

create\_file(client path, source filepath); //to put into our SUFS

/\*

This function takes in a source filepath and a user specified file path to put into our hfds directory. It checks whether the filepath exists before creating. It sends filepath to namenode and receives block number and datanode information from namenode. It contacts individual datanode and sends blocked data to them.

\*/

get\_directory(dirPath)

/\*

This function will take in the path to a directory and contact namenode and get a list of all the files inside of that directory and send it back to the client. This list will be displayed on the GUI for client to see

\*/

Read functions:

read\_file(FilePath): client sends name node file path it wants to read

/\*

This function takes in a filepath and contacts namenode to get datanode addresses for the file that the user wants to read. Once it has all the info, if it exists, it will contact the datanodes for the specified blocks and builds the data for read/access.

\*/

Delete functions:

delete\_file(client path)

/\*

This function takes in a filepath that the user wants to delete from our file directory. It will contact namenode and try to get the addresses for the data if it exists, once it gets the information of the datanodes, client sends a delete request to the specified data nodes.

\*/

delete\_directory(dirPath)

/\*

Sends dirPath to namenode and gets back the files in that directory. DeleteFile is then called for the files that were contained in the directory.

\*/

**NameNode:**

Fields:

Metadata Data Structure stores block names and block locations of the data

A dictionary of dictionaries of lists

The first key in the dictionary is the file. Second key is the part of the file. The list contains the ip addresses of replicas.

{FileA: { \_1: [ip address of data block 1, ip address of replica...], ... }, FileB:{}, …}

Function:

create\_file(client filePath, data size in bytes)

/\*

This function creates a key in the metadata structure using the filepath that it got from the client, it checks if it exists before creating. It has a function that divides up the block into the size we specified. Chooses datenodes based on number of blocks based on a round robin algorithm. It requests a heartbeat from the intended datanodes to check for availability before sending the datanode information(addresses, and block amounts) back to the client.

\*/

read\_file(client filepath):

/\*

This function takes in the filepath from the client, it then checks the metadata structure for the key and see if it exists. If it exists the namenode returns back the ips of the datanodes for each block that has the data associated with the filepath.

\*/

recieve\_block\_report(blockreport):

/\*

This function receives blockreports from datanodes and updates the metadata structure based on contents of the block report.

\*/

get\_directory(dirPath)

/\*

Receives a directory path from the client node and returns a list of files in that directory to the client node

\*/

delete\_directory(dirPath)

/\*

Pattern match on all keys in metadata and return the files that have the prefix of the dirPath given.

\*/

**DataNode:**

Fields:

List of datanodes and endpoint

Has a contents directory which stores all the blocks

Functions

trigger\_report(namenode ip)

/\*

This function sends a blockreport to the namenode every 10 seconds regarding its stored blocks on its content directory.

\*/

get\_block(filename):

/\*

This function takes in a filename sent from the client and finds the specified block that the client intends to read. Returns the block to the client.

\*/

delete\_block(filename):

/\*

This function takes in a filename sent from the client and finds the specified block that the client intends to delete. It then removes the file/block from its content directory.

\*/

heartBeat():

/\*

This function sends a response code ok to requester, the namenode, that it is alive and able to take request.

\*/

replicate(block, other DataNode, replicaNum)

/\*

This function is not implemented but will pass block in contents to another datanode along with current copies of block. Current copies will be incremented each times it is passed and it will stop passing data blocks once there are 3 replicas of block

\*/