John Piontkowsky
Vahe Kilamyan
April Luo
Marc Mendiola
Gueho Choi

TFS Part 2

Design Decisions

1. **Master**
    a. Multi-threaded
    b. Spawns a new thread to service each new request
    c. Contains:
        i. FileSystem.java class (creates, removes, and interacts with directories and files)
        ii. Hashtable to map Strings to a TFSDirectory object
        iii. TFSDirectory class (contains a list of subdirectories and the files within those subdirectories)
            1. ex. “\\” represents the root directory
            2. Stored in a sorted set; lookup time = O(log n)
    d. Each transaction is logged using FSLogger.java, for crash recovery purposes
    e. All transactions other than Read, Write, and Append handled by Master itself
    f. Global unique identifiers: Master hashes the full file path and passes the resulting hash to the chunkservers
    g. ChunkTracker.java (within Master.java)
        i. Maintains a list of live chunkservers and a list of dead chunkservers
        ii. Sends out a “heartbeat” every 60 seconds to ensure existing chunkserver connections are still alive
        iii. If no response, moves the chunkserver to the dead list

2. **Chunkserver**
    a. Each chunkserver “registers” with Master on startup
    b. Registering with Master gives the chunkserver a number (eg, 1,2).
    c. Chunkserver’s files are stored in the folder cs + its number (eg, “cs1”)
    d. Chunkserver attempts to synchronize with other chunkservers by examining its folder for .meta files, which contain the version number of files and the other chunkservers containing them.
    e. Updates its files by issuing “updateFile” commands to other chunkservers contained in the .meta file.

+

3. **Client**
    a. Single-threaded
    b. Forwards requests to either Master or chunkserver

Interfaces and Filesystem Calls

	Input	Output	Data flow
CreateFile	(String fileName, String initialData)	File created on chunkserver using Java File Create and data passed in	Client to Master, Master to Client, Client to Chunkserver Client to Master: Client sends create request to master, specifying a unique file name. Master to Client: Master ensures that filename is unique in this directory. If it is, Master chooses n chunkservers and returns their addresses to the client. If it is not Master returns an error to the client. Client to Chunkserver: Client chooses 1 of n chunkservers and issues a create request to the chunkserver. Chunkserver uses fileName and initialData to create the file. Client proceeds to repeat this process n times until the specified number of replicas have been created.
CreateDir	(String dirName)	Directory created on chunkserver. Return -1 on failure.	Client to Master, Master to Client Client to Master: Client sends create directory request to master, specifying a unique directory name. Master to Client: Master ensures that directory is unique. If it is, it creates the directory and returns success to the client. Otherwise, it returns failure.

Open	(String filename)	Returns List of chunkservers holding a replica of this file.	Client to Master, Master to Client, Client to Chunkserver <u>Client to Master</u>: Client sends an open request to master, specifying a unique file name. <u>Master to Client</u>: Master ensures that the file is valid (exists, is unique in directory). If it is, Master sends back a structure that contains the IP addresses of chunk servers with this file on it on and the chunk value.
Close	(String filename)	Returns boolean indicating success	Client to Master <u>Client to Master</u>: Client sends a close request to master. <u>Master to Client</u>: Master sends back a confirmation indicating success.
Read	(String filename, int filePosition, int numBytes)	Returns struct of data and an error code indicating success or failure.	Client to Chunkserver, Chunkserver to Client <u>Client to Chunkserver</u>: Client sends a read request to chunkserver, specifying a chunk handle. <u>Chunkserver to Client</u>: Chunkserver attempts to read the chunk at the chunk handle. If the chunk exists, the chunkserver sends the data back to the client.
Write	(String filename, int offset, File initialData)	Boolean representing success or failure of write operation.	Client to Chunkserver, Chunkserver to Client <u>Client to Chunkserver</u>: Client uses the chunkserver selected through the open operation as an address of

			the request. Client then sends message to chunkserver specifying filename, file offset location, and data to be written. Chunkserver will then perform the write operation on the file specified. Chunkserver to Client: After successful execution of the write operation chunkserver will send a boolean success(true) to client. Otherwise the chunkserver will send a boolean failure(false).
DeleteFile	(String filename)	Returns boolean success of the delete operation, true if success, false otherwise.	Client to Master, Master to Client, Client to Chunkserver, Chunkserver to Client Client to Master: Client sends filename of file to be deleted to Master. Master to Client: Master returns a list of chunkservers which contain the file to be deleted. Client to Chunkserver: Client sends each member of the list a message to delete the file with the provided filename. Chunkserver to Client: Chunkserver sends confirmation of completion to client after the operation has been completed.
FileSize	(String filename)	Returns integer representation of the file specified	Client to Master, Master to Client, Client to Chunkserver, Chunkserver to Client Client to Master: Client sends the filename to master to determine

			location of the file on chunkservers. Master to Client: Master returns a List of chunkservers to the client which contain the file specified. Client to Chunkserver: Client chooses one of the specified chunkservers to query the size of the file on that chunkserver.
LS	(String directoryname)	Returns data representation of the current file structure being tracked by master.	Client to Master, Master to Client Client to Master: Client sends the request to master specifying a directory name which it would like to receive the contents of. Master to Client: Master takes the directory name and compares it to its internal file structure records. Master then returns the subdirectories and appropriate files contained within the directory specified.