Group Project Assignment: Distributed File System

Project description

The *Distributed File System* (*DFS*) is a file system with data stored on a server. The data is accessed and processed as if it was stored on the local client machine. The DFS makes it convenient to share information and files among users on a network. In this project, you will implement a simple *Distributed File System* (*DFS*). Ready-made DFS solutions cannot be used in your project. Files will be hosted remotely on one or more *storage servers*. Separately, a single *naming server* will index the files, indicating which one is stored where. When a *client* wishes to access a file, it first contacts the naming server to obtain information about the storage server hosting it. After that, it communicates directly with the storage server to complete the operation.

Your file system will support file reading, writing, creation, deletion, copy, moving and info queries. It will also support certain directory operations - listing, creation, changing and deletion. Files will be replicated on multiple storage servers. DFS will be fault-tolerant and the data will be accessible even if some of the network nodes are offline.

Storage Servers

The primary function of storage servers is to provide clients with access to file data. Clients access storage servers in order to read and write files. Storage servers must respond to certain commands from the naming server.

Servers and clients need a way to identify files. Each file is identified by its path in the DFS. The storage server is required to put all its files in a directory on the machine that it is running on - this will be referred to as the storage server's local or underlying file system. The structure within this directory should match the storage server's view of the structure of the entire DFS. For example, if a storage server is storing its files in the local directory `/var/storage`, and it is hosting a file whose DFS path is `/directory/README.txt`, then that file's full path on the local File System should be `/var/storage/directory/README.txt`.

The storage server interacts with the naming server to transparently perform replication of files causing multiple storage servers to maintain copies of the same file.

Naming Server

Clients do not normally have direct access to storage servers. Instead, their view of the file system is defined by the file system’s single naming server. The naming server tracks the file system directory tree, and associates each file in the file system to storage servers. When a client wishes to perform an operation on a file, it first contacts the naming server to obtain information about the storage server hosting the file, and then performs the operation on a storage server. Naming servers also provide a way for storage servers to register their presence.

The naming server can be thought of as an object containing a data structure which represents the current state of the file system directory tree, and providing several operations on it.

The naming server interacts with storage server to allow a client perform all operations with files and directories.

Client

Responsible for making the distributed nature of the system transparent to the user. File interface should include next functions:

- **Initialize**. Initialize the client storage on a new system, should remove any existing file in the dfs root directory and return available size.
- **File create.** Should allow to create a new empty file.
- **File read**. Should allow to read any file from DFS (download a file from the DFS to the Client side).
- **File write**. Should allow to put any file to DFS (upload a file from the Client side to the DFS)
- **File delete**. Should allow to delete any file from DFS
- **File info**. Should provide information about the file (any useful information - size, node id, etc.)
- **File copy**. Should allow to create a copy of file.
- **File move**. Should allow to move a file to the specified path.
- **Open directory**. Should allow to change directory
- **Read directory**. Should return list of files, which are stored in the directory.
- **Make directory**. Should allow to create a new directory.
- **Delete directory**. Should allow to delete directory. If the directory contains files the system should ask for confirmation from the user before deletion.

Replication and Sharding

Don’t forget to put storage servers and naming server into private closed network. So that 1) they can find each other automatically 2) their communication will be isolated from the public network. It could be done with VPS provider service or docker’s overlay network.

Each file should be replicated on at least 2 Storage Servers. If one of the Storage Server goes down, files, that is stored should be replicated to another Storage Server.

Files might be split into chunks. Chunk size is under your consideration.

Deliverables

Both write-up and presentation should be uploaded to Moodle by each member of the group (the same pdf files). Pack all the modules (client, naming, storage) of your application in docker images and upload them to DockerHub. Source code should be uploaded to GitHub.

- Work
  - Project start - **Oct, 8th**
  - Not more than **3** person per group
  - Group formation deadline - **Oct, 15th** (Specify the team members here)
  - Deadline - **11:55 p.m. Nov, 25th**
- Write-up
  - Links to GitHub and DockerHub repos
  - Contribution of each team member
- GitHub README
  - Documentation how to launch and use your system
  - Architectural diagrams
  - Description of communication protocols
- Application
  - Any programming language
  - Source code – GitHub
  - Images – DockerHub
- Presentation
  - **Nov, 19th** or **Nov, 26th**. 10 minutes per group.
  - Brief description of contribution of each team member
  - Architectural diagrams
  - Focus on live demo
  - Problems you faced/solved

Defence

Teams present their work during lab class at **Nov 26th**. For presentation your naming server and storage servers should be deployed to several AWS instances (or any other VPS). Prepare backup video of work of your system. But if your presentation doesn't contain live demo – you lose 4 points for presentation.
Live demo content:

- DFS Initialization
- File writing, reading and deleting
- Replication
- Storage server crash
- Storage server restoration

Bonus points

+2 additional points if your team presents work at **Nov 19th**.
+2 additional points if you use docker swarm to deploy and scale you solution.

In total you still can earn only 20 points for the work. But bonuses will help you if you lose points in other parts.

Grading criteria

- Application – 13
- Presentation – 6
- Write-up – 1