Project Proposal Report

“SMART CHOICE”

Emulation-based file system for Distributed File Storage & Parallel Computation

DSCI551-20223-Group- 128

Group Members:

Name	USC ID	Email id
Ankit Tripathi	4612 6769 99	ankittri@usc.edu
Lakshita Shetty	2162 0892 11	lshetty@usc.edu
Shreya Nayak	8592 8104 56	nayakshr@usc.edu

Aim:

Our goal is to develop a user-friendly online application that shows off emulated HDFS and employs the MapReduce function's partitioning structure to effectively search and analyze various data sets in the emulations.

Description:

In today's digital age, where digital media consumption has a long-lasting impact on consumers' day-to-day behaviour, it's critical for the audience to watch the right TV shows and movies based on their interests. The widespread availability of digital content is both boon and bane at the same time. Many Over-The-Top (OTT) services, such as Netflix, Amazon Prime Video, HBO Max, and others, strive to provide the best available content to their users in order to keep their memberships for extended periods of time. However, from the standpoint of the audience, it is extremely difficult to keep track of their favourite TV shows and movies. They may recall one or two movies that are available on a specific OTT platform, but keeping track of a large number of TV shows and movies is more difficult. In a world when every second counts, it not only wastes customers' time to search every OTT platform for the content they want to see, but it also diminishes their desire to watch it. So the intention of this project is to save the audiences those precious time they would otherwise spend for irrelevant actions.

The project is an emulation-based system for distributed file storage and capable parallel computation. It is a web-application that will provide audiences with a user-friendly interactive environment in which they can not only look for their favourite TV shows and movies, but also which OTT platforms they are available.

Our application ‘Smart Choice’ aims to cater to the varying needs of diverse users and to help them shortlist a movie or a show that they can relax or unwind to after a long tiring day without having to spend a lot of thought on which of the many options they should consider. The user needs to simply choose from the available options whether to check if a particular movie is on Netflix or not.

Motivation:

In the present world where there are innumerable choices for everything. It becomes burdensome to make the right choice. There are too many things to choose from. This habitual phenomenon is because of something termed as the 'Choice paralysis'. Choice paralysis is the situation where an individual has trouble making a decision owing to the massive amount of information and options available. Although having choices comes with its own set of advantages such as flexibility and freedom. It can also lead to a state where an individual does not make a choice at all. This paralysis is partly due to the belief that the individuals want to find the best possible choice such that it makes the most optimum use of their valuable time.

Research also shows that an excess of choices often leads people to be less, not more, satisfied once they actually make a decision. There's a frequent aching feeling that they could have done better. Understanding how to choose could guide people to make better decisions.
People have several criterias in mind when it comes to deciding which movies or TV shows they want to watch. These criterias include lead actor/actress i.e the cast, the genre, the IMDB ratings, time-duration etc. Finding an option that satisfies most of their criterias is a good way of ensuring a suitable choice is made. This is the main inspiration for creating our project " Smart Choice"- which will enable the users to opt for an appropriate selection of their movies or tv shows.

Dataset:

-> Dataset obtained from Kaggle:

This is a dataset that has been collected by a Kaggle user. The dataset lists various different movies & shows and their corresponding rating, genre and many other attributes.

Reference Link:
https://www.kaggle.com/datasets/victorsoeiro/netflix-tv-shows-and-movies

https://www.kaggle.com/datasets/victorsoeiro/amazon-prime-tv-shows-and-movies

https://www.kaggle.com/datasets/victorsoeiro/disney-tv-shows-and-movies

Implementation
Task 1 :

Building an emulated distributed file system (EDFS)
Our EDFS would store metadata (i.e file system structure, file attributes, location of file partition, etc.) in a database and partition data of our file in a second database, much like HDFS, which stores metadata in namenode and actual file data in datanode.

- > Firebase-based emulation

- Consider some (already partitioned) csv files located in user directory

- Metadata can then be stored as the following JSON object in database_1: (i.e HDFS namenode)
- Actual data from file partitions can be stored as the following JSON object in database_2: (i.e HDFS datanode)
- In order to support HDFS-like commands, we can use firebase REST API to perform CRUD operations. A successful request is indicated by a 200 OK HTTP status code. The response contains the data associated with the path.

EDFS will use the following command:
- mkdir: create a directory in file system
- ls: listing content of a given directory
- cat: display content of a file
- rm: remove a file from the file system
- put: uploading a file to file system, will upload a file csv file to the directory in EDFS. The file will be storing in k partitions, and the file system should remember where the partitions are stored.
- getPartitionLocations(file): this method will return the locations of partitions of the file.
- readPartition(file, partition#): this method will return the content of partition # of the specified file. The portioned data will be needed in the second task for parallel processing.

- > MySQL-based emulation

For the MySQL-based emulation, we aim to store file system data in tables enough so one table would contain all the file metadata and another table could be used to store directory structure data. A row in this table will point to a different table that has the data that should be in each partition, and there will also be a table that contains details about the partitions that we will be building. We also intend to investigate the various datasets and discover data-partitioning techniques. The PARTITION BY clause in SQL will be used to make partitions based on features and properties.
For example, for the Netflix dataset, we would partition by the genre which means that the datapoints in the same genre would be placed in the same partition.

For querying, we will be forming queries based on the user input, for example:

SELECT * FROM User
WHERE Genre='Drama'; (where the field '**Genre**' would be the user-choice.)

We will be processing these SQL queries (and conducting the CRUD operations) with the use of Python.

Task 2 :
Implement partition-based MapReduce (PMR) for search/analytics

We can implement our mapPartition function as described in the guidelines (Map() function would process every partition and return the desired list. Reduce() function would return an aggregate of outputs from Map())
Examples of search and analytics queries:

We will implement Search queries and Analytics queries on the data sets mentioned above - for example, searching the movies within a given rating or from a particular genre, comparative study across the 3 platforms etc.

Based on the user choice, our flask framework would take a partition number and use the mapReduce function to send requests/queries to the database and receive a final output for every user operation.

We aim to enable the user and carry out a basic exploratory data search/analysis on the chosen dataset.

Task 3 :
Creating an app that uses the implemented PMR methods.

The web application will be a user-friendly interface that the user can use to achieve the above-mentioned goals.

The application will allow users to select from any of the databases listed via the input/choose box. They can then choose their interest-specific options from a wide range of selection criteria, which can subsequently be analyzed/searched within the database of their choice.

The Frontend of the application will be designed using HTML5/CSS/Javascript and will be integrated with the Flask framework of Python. The application will also be connected to both the databases Firebase and MySQL.

The input/choose boxes will provide users with selection options whose inputs will be utilized to perform CRUD (Create, Retrieve, Update, and Delete) operations similar to those performed in HDFS.

The user will first select the database to which they want to gain access, and then they will be able to execute the following functions:

1. Filter movies and TV Shows based on IMDB ratings.
2. Filter the content by Genre (Action, Thriller, Comedy, Drama)
3. The OTT platform where the corresponding Tv show, Movie is available.

Our initial plan is to take the user inputs from the HTML web page and use Flask to send appropriate queries to the backend databases to perform the necessary CRUD operations and the functions mentioned above.

Timeline:

Task	Member	Timeline
Data Extraction and Data pre-processing	Lakshita Shetty	September 25, 2022
DataSet Generation	Ankit Tripathi	September 25, 2022
Data Preparation (which includes data visualization)	Shreya Nayak, Ankit Tripathi	September 28, 2022
Creating the main EDFS (with a simple and user friendly HTML front end) Decide the flow of the app	Lakshita Shetty, Shreya Nayak, Ankit Tripathi	October 8, 2022
Implementing MapReduce() for basic search/analytics for at least one data set	Lakshita Shetty	October 25, 2022
Development of Flask	Shreya Nayak, Ankit Tripathi	October 25, 2022
Implementing MapReduce() for the rest of the data sets	Lakshita Shetty, Shreya Nayak	November 8,2022
Development of Frontend-UI	Ankit Tripathi, Lakshita Shetty	November 8,2022
Final Compilation and Documentation with video and final report	Ankit Tripathi, Lakshita Shetty, Shreya Nayak	November 20, 2022

Team Members:

Name	Current Degree	Previous Degree	Skills
Ankit Tripathi	MS in Applied Data Science	B.E Electronics and Telecommunication Engineering	Python(Numpy, Pandas, Scikit learn, etc.), SQL, C, C++, HTML & Java, java script, Flask, MATLAB, Big Data Frameworks, Machine Learning
Lakshita Shetty	MS in Applied Data Science	B.E Electronics and Telecommunication Engineering	Python(Numpy, Pandas, Scikit learn, etc.), SQL, R, C, C++, HTML & Java, Power BI, Tableau, MATLAB, Big Data Frameworks, Machine Learning
Shreya Nayak	MS in Applied Data Science	B.E Electronics and Telecommunication Engineering	Python (Numpy, Pandas, Scikit learn, etc.), C, R, SQL, Machine Learning, Big Data Frameworks, HTML, CSS, Tableau, Power BI, MATLAB,SCILAB