**Final Project Proposal**

**– Dynamic Movie Recommendation System**

Team 1: Yang Bai, Xuejiao Dong, Giang Vu, Linxiu Jiang

# **PROJECT SUMMARY**

In the era of mobile internet, data is growing explosively. Understanding users’ preference plays an important role in providing a better service. In order to do that, we need to make full use of users’ historical data and their interaction. Then we came up with an idea of building a dynamic movie recommendation system. This system is based on the user interaction and their rating history. User interaction will keep influencing our system positively. Based on our recommendations system, we can recommend maybe top 10 movies for each user. Moreover, we virtualize this system, which includes how much data we have ingested, and the accuracy of the model. Our goal of the project is to learn how to implement a dynamic BigData ecosystem, and apply our Scala knowledge learned in class to Spark to solve a common, real-world problem.

# **DATASOURCE INTRODUCTION**

We will use the following dataset(the first dataset in the link) to build the whole project

<https://grouplens.org/datasets/movielens/>

https://files.grouplens.org/datasets/movielens/ml-25m-README.html

## **Scale**

This dataset describes 5-star rating and free-text tagging activity. It contains 25000095 ratings and 1093360 tag applications across 62423 movies. These data were created by 162541 users between January 09, 1995 and November 21, 2019.

## **Files**

**Ratings Data File Structure (ratings.csv)**

Format: userId,movieId,rating,timestamp

**Tags Data File Structure (tags.csv)**

Format: userId,movieId,tag,timestamp

**Movies Data File Structure (movies.csv)**

Format: movieId,title,genres

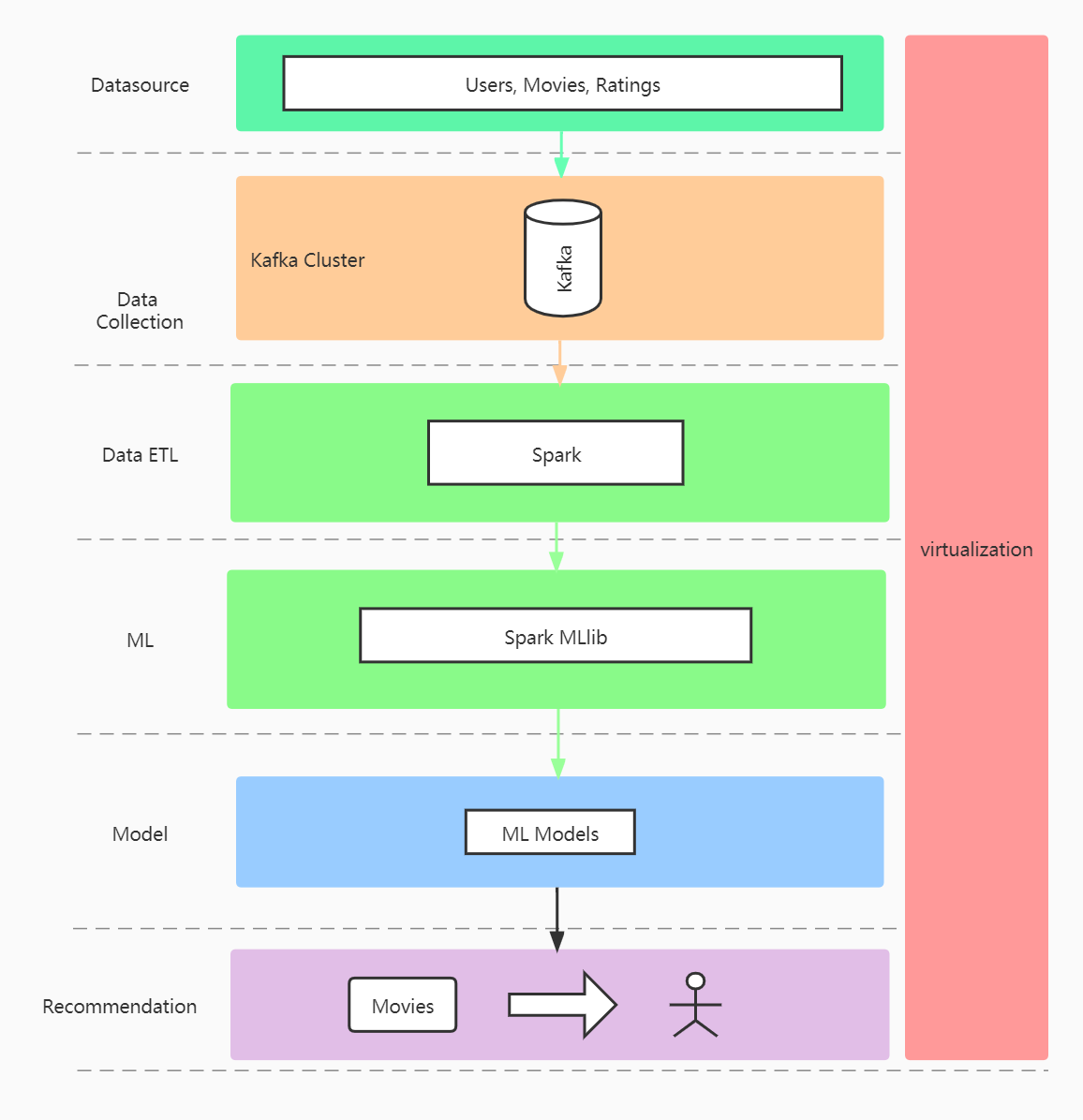
**Links Data File Structure (links.csv)**

Format: movieId,imdbId,tmdbId

**Tag Genome (genome-scores.csv and genome-tags.csv)**

Format: movieId,tagId,relevance

# **FLOW CHART**

****

# **PROCEDURE DESCRIPTION**

## **Step 1: Simulate Dynamic Data stream**

### **Parallel Data Producer**

We create multi-threads as producers to stimulate data flows

### **Stream Processing with Kafka**

We use kafka to build a high-performance streaming platform that enables data pipelines, streaming analytics, and data integration.

* Producer: push data of user actions into Kafka
* Consumer: fetch user actions from Kafka and do streaming analytics by putting analytic results into Graphite

### **ETL and Data Preprocess**

A new Consumer fetches user actions data and does preprocessing including decoding, feature extraction, and ends up with a dataset that can be used for dynamic ML training and modeling.

## **Step 2: Dynamic ML**

The Dataset describes a 5-star rating which can be used to build users' taste rules. Our recommender systems combine “content based filtering” and “collaborative filtering”. We dynamically train ML models based on historical models and new data streams of the dataset, and then come up with an optimal model.

# **ALGORITHMS AND STRATEGIES**

| **Type** | **Definition** | **Example** |
| --- | --- | --- |
| content-based filtering | Uses similarity between items to recommend items similar to what the user likes. | If user A watches two cute cat videos, then the system can recommend cute animal videos to that user. |
| collaborative filtering | Uses similarities between queries and items simultaneously to provide recommendations. | If user A is similar to user B, and user B likes video 1, then the system can recommend video 1 to user A (even if user A hasn’t seen any videos similar to video 1). |

There is still one problem of dynamic models under consideration. Because of the inflow of new data streams, the entire process needs to be repeated for each iteration of the model, which may cause a waste of time. So we may need to optimize the algorithms to reduce the time required for each model iteration.

# **RELATIVE TECHS**

Kafka

Spark: Spark core, Spark Streaming, Spark MLlib

TSDB: Graphite

# **REFERENCES**

1. <https://grouplens.org/datasets/movielens/25m/>
2. <https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109>
3. <https://developers.google.com/machine-learning/recommendation/overview/candidate-generation>
4. <https://towardsdatascience.com/static-machine-learning-models-in-a-dynamic-world-ff1ea1b0892c>