Movie Recommender system using Spark and ElasticSearch

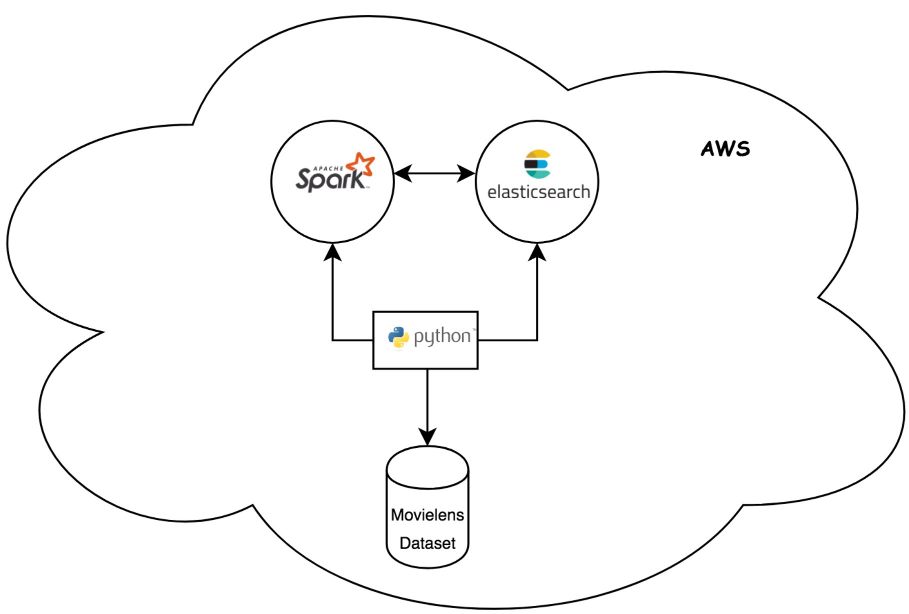
Pranav Kumar Sivakumar, Shemal Somil Lalaji

Problem Description:

Recommendation engines are one of the most commonly used machine-learning based application that can be easily implemented. But the issue of huge datasets makes it harder to develop one of them.

In this project, we propose an approach to provide real-time recommendations, by constructing a large-scale recommender engine using big-data technologies like Apache Spark and ElasticSearch. This approach is highly scalable as it involves frameworks based on distributed systems and cluster computing. So we will be utilizing the related distributed computing services provided by Amazon Web Services (AWS) to accelerate our implementation.

Apache Spark is a unified analytics engine that also has an inbuilt module for implementing machine learning based computations called MLlib. So we will be using it in our project as well for training the model in our recommendation system.



Solution Approach (in High level):

The following steps are the tentative sequence of tasks that we plan to follow in our project,

1. The first part is the data preprocessing which will involve some basic text manipulation.

2. After preprocessing the dataset that contains the movie ratings, we plan to train a collaborative filtering model in Spark using RDD operations and MLlib.

3. We plan to load the data in ElasticSearch after designing a schema for the database. We will be using the Python wrapper for Elasticsearch and probably a connector for Spark and ElasticSearch. We would have to index for mappings for users, movies, and ratings.

3. After successfully loading it, we'll be training the Collaborative filtering model on that data. We chose the collaborative filtering method as it is a sort of technique that enables to model the similarity between the items(movies in our project) based on other similar user's preferences. For this, we would have to explore libraries/methods like ALS(Alternating Least Squares) in MLlib. Our main emphasis in this project is on Spark and ElasticSearch instead of Machine Learning since it is a very small component of our project.

ML part may still take some time.

4. Load the trained model's factors into Elasticsearch and probably perform some Dataframe operations for indexing.

5. The last step will consist of generating the recommendations. To do this, we will create some utility functions that handle different tasks such as finding similar movies based on another movie, or based on user preferences, fetching movie meta-data etc. After getting the results from the queries, we plan to display them effectively. In case we finish the project early, we plan on using Kibana to display the results in a more better way.

how?

try for this

Dataset:

The dataset will be the Movielens 20M dataset that consists of ratings given by a set of users to movies, combined with their metadata information. It was collected and made available by the GroupLens Research.

It is a standard benchmark dataset with 20 million ratings. It includes tag genome data with 12 million relevance scores across 1,100 tags. The dataset describes 5-star ratings and other tags. It contains 20000263 ratings and 465564 tag applications across 27278 movies

The data are contained in six files: genome-scores.csv, genome-tags.csv, links.csv, movies.csv, ratings.csv, and tags.csv.

For data collection, the users were selected at random for inclusion and they had to rate at least 20 movies. User ids are consistent between ratings.csv and tags.csv (i.e., the same id refers to the same user across the two files).

Considering movies column, only those movies with at least one rating or tag are included in the dataset. Movie ids are also consistent between ratings.csv, tags.csv, movies.csv, and links.csv (i.e., the same id refers to the same movie across these four data files).

The data is a static and open source dataset, so there is no need for a developer account as it can be easily accessed. This dataset, as well as the others, are available for free download at http://grouplens.org/datasets/.

The data will be downloaded and processed on our desktop. The data require some preprocessing steps such as removing white spaces and separating some data columns in the dataset. After that, we can use the data in our project.

Challenges:

The problem is hard in the sense that it has many components attached to it which we need to solve as well as issues in integration with ElasticSearch, Spark and Machine Learning.

Since both of us are new to Elasticsearch, we may come across components of ElasticSearch that we may take time to understand. There will be situations where we need to resolve issues in our machine and try to install ElasticSearch in the right way in our machines.

Another issue is that the dataset is huge and using it on a complex architecture of Elasticsearch and Machine Learning will add to the preprocessing complexities of the project.

One of the major components of this project is integrating Machine Learning with ElasticSearch and Spark. Our goal is to give the best recommendations possible based on the given preferences in real-time.

Timeline:

Assuming that we are going to start working on the project from October 31st and will have to submit it in the middle of December, this is the predicted timeline that we have come up with.

The project consists of the following parts and the expected Deadline of each of them:

1. Load the MovieLens 20M dataset into Spark. might be lots of work, start early
2. Make use of Spark DataFrame operations to clean up the dataset. After the cleaning is finished, it would be loaded into Elasticsearch.

   Both these parts will be done in 1-2 weeks

3. Using Spark MLlib, train a recommendation system based on collaborative filtering.

   This is the Machine Learning part of the project and will take around two weeks.

4. Ingest the trained model into Elasticsearch.
5. Using Elasticsearch queries and some custom scoring methods, generate sample recommendations and test the recommender system.

Both of these will take the remaining time of around 2-2.5 weeks left for the project.

It is not feasible in this project to do two different operations in parallel, so we plan on doing the project by sitting together until completion.

Project proposal looks good.