Assignment 1 Design Document

By:

Group No-11

Meghna Raj- 2019H1120053P

Surbhi Gupta-2019H1120061P

Snigdha Roy- 2019H1120066P

Ayushi Gaur- 2019PHXP0023P

Scope of the Assignment

In this assignment, we are required to develop a REST-based service for the given dataset. This service will accept a query in the form of one of the two templates provided below and translate into MapReduce jobs and also into Spark job.

Service should run these two jobs separately and return the following in a JSON object:

 Time taken for Hadoop MapReduce execution

 Time taken for Spark execution

 Input and output of map and reduce tasks in a chain as they are applied on the data

 Spark transformations and actions, in the order they are applied

 Result of the query

The two standard SQL query templates are :

(1) JOIN TEMPLATE:

SELECT \* FROM <TABLE1> INNER JOIN <TABLE2> ON <CONDITION1> WHERE < CONDITION2>

(2) GROUP BY TEMPLATE:

SELECT <COLUMNS>, FUNC(COLUMN1) FROM <TABLE>

GROUP BY <COLUMNS> HAVING FUNC(COLUMN1)>X

--Here FUNC can be COUNT, MAX, MIN, SUM

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Datasets Used  Description of the dataset is given below: Users file stores the demographic information about the users. The zipcodes file contains the city and state information for the zipcodes. Zipcode is referenced by users file. Movies file stores the information about movies. The last 19 fields in movies table are the genres, a 1 indicates the movie is of that genre, a 0 indicates it is not; movies can be in several genres. Rating file contains the ratings given by users for movies. Userid and movieid refer to the users and movies files respectively.  **Users**: userid | age | gender | occupation | zipcode  **Zipcodes**: zipcode | zipcodetype | city | state  **Movies** table: movieid | title | releasedate | unknown | Action | Adventure | Animation | Children | Comedy | Crime | Documentary | Drama | Fantasy | Film\_Noir | Horror | Musical | Mystery | Romance | Sci\_Fi | Thriller | War | Western |  **Rating**: userid | movieid | rating | timestamp  Pp  Proposed Design  The first step forward is to install Hadoop. There are two ways to install Hadoop, i.e. **Single node** and **Multi node**. We have used Single node cluster.  **Single node cluster** means only one DataNode running and setting up all the NameNode, DataNode, ResourceManager and NodeManager on a single machine.  We’re using REST API with FLASK which will link the shell and the file system and provide a web application framework for running the commands running on the shell as well as fetching the required files .  Spa spark is fast for processing & better than mapreduce as it takes less time to produce the output. Spark uses RDD actions & Transformations to do all sorts of processing on the data.  S   Interface/API Definitions REST is an architectural style, or design pattern, for APIs.  A RESTful web application exposes information about itself in the form of information about its resources. It also enables the client to take actions on those resources, such as create new resources (i.e. create a new user) or change existing resources (i.e. edit a post).  Flask is a lightweight [WSGI](https://wsgi.readthedocs.io/) web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around [Werkzeug](https://palletsprojects.com/p/werkzeug) and [Jinja](https://palletsprojects.com/p/jinja) and has become one of the most popular Python web application frameworks.  Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy. Business Logic **MapReduce JOIN Template**  We have used hadoop-streamer here as we have coded in python.  For Hadoop **MapReduce JOIN Template** execution, we are using the following flow:    Input  Mapper  Reducer  Output    Our SQL query is :  SELECT \* FROM users INNER JOIN rating ON users.userid = rating.userid WHERE gender = "F"   |  |  | | --- | --- | | Aim | Performing the join of the two tables Users and Rating based on the common column useridwhere user is a female. | | Mapper | Key : userid of each record  Value : The rest of the attributes of the table | | Reducer | Joins the two tables based on the key where user is female |   The Mapper takes the input from the two tables in the form of a record and produces key,value pairs which are then sent to the reducer as input which then joins the two tables on the key where the user is female and produces the output in a JSON file.  **MapReduce GROUP BY Template**  We have used hadoop-streamer here as we have coded in python.  For **MapReduce GROUP BY Template** we are going to use Input-Map-Reduce-Output MapReduce Design Pattern as we are using aggregation function here.  Input  Output  Map  Reduce  For example let us suppose our SQL query is  SELECT occupation, sum (age) FROM users GROUP BY occupation HAVING sum (age) > 100   |  |  | | --- | --- | | Aim | Grouping the data on the basis of occupation such that sum of corresponding ages should be greater than 100 | | Map(key , value) | Key : Occupation  Value : Their Age(along with function(here sum) and operator(here > ) | | Reduce | Group by Occupation  And Take summation of ages for each group such that summation is greater than 100. |   The given template was of the form:  SELECT <COLUMNS>, FUNC (COLUMN1)  FROM <TABLE>  GROUP BY <COLUMNS>  HAVING FUNC (COLUMN1)>X  --Here FUNC can be COUNT, MAX, MIN, SUM  According to the given query we will be sending column name which is inside group by as a key and the attribute inside func in value in map function. Here for ease of computation we have even passed the aggregate function , operation and value appended after our attribute of the func.  Whereas in Reduce file after tokenizing the key and value from map function. And here according to the given aggregate function we have applied the logic accordingly and outputted the required result.  **SPARK JOIN Template**  It takes the sql query as an argument tokenises the query. With the help of RDDs we import the csv files needed.Spark is coded in python.  map(“key as your on condition clause”,row)- do this operation on 2 RDDs or the 2 tables  table1.join(table2)  filter(“apply a function to each row ”)  the above resultant rdd is iterated through collect() method to produce JSON with the help of dictionary in python.  **SPARK GROUP BY Template**  It takes the sql query as an argument tokenises the query. With the help of RDDs we import the csv files needed. Spark is coded in python.  map(“key as your group condition”,”aggregating column”)  reduceByKey(function to be applied for reduction such as min,max,count,sum)  filter(“apply a function to each row of the resultant RDD”)  the above resultant rdd is iterated through collect() method to produce JSON with the help of dictionary in python. |