Semantic Analysis of Spark SQL Statements

A tool to view spark execution plans, cluster and application metrics

Swarnaditya Tamonash Maitra

201701006

Modelling the Project

I - Basic Setup of IntelliJ IDEA, environment variables, and WinUtils

1) Configure IntelliJ IDEA by importing all Spark and Maven dependencies.

2) To test basic working, use Spark Dataframe API to pass simple SQL queries on local datasets, and generate execution plans on the console.

3) Ensure JAVA_HOME, SPARK_HOME, and HADOOP_HOME paths are correctly configured in the environment variables

4) Repeat 2nd step with datasets on the Hadoop Cluster, using WinUtils of the correct Hadoop version.

II - SQL Query Form and Spring Boot

1) The form is made in React.js, and hosted on localhost:3000. Ensure that Node.js is installed, all npm packages are installed, and node_modules is updated.

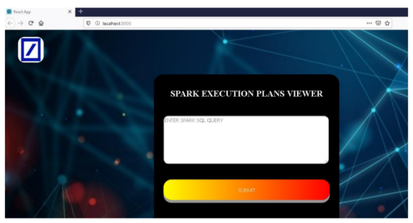
2) Axios Library of Node.js is used to transfer data entered in the react form.

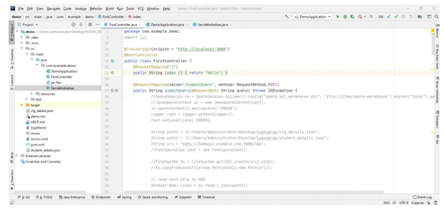
3) The other end of the API is developed using Spring Boot (Java), which works on MVC framework.

4) The outputs corresponding to user query are redirected to an output file

Schematic Representation

Screenshots





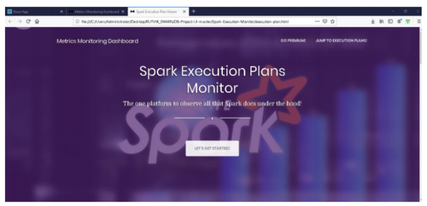
III - Spark Execution Plans Monitor

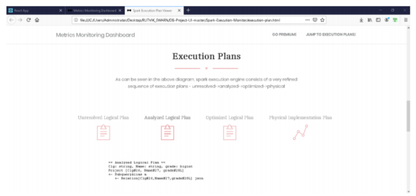
1) The UI is made using HTML, CSS, and JavaScript, which contains links to the Metrics Monitoring UI.

2) The spark java program is appended with another segment, to process the output file, separate the 4 execution plans, and redirect the output to 4 different files.

3) All the 4 files are automatically linked to container elements on the UI.

Screenshots





IV - Configuring core-site.xml, yarn-site.xml, hdfs-site.xml

1) Lookup the Apache documentation for URIs corresponding to installed Hadoop and Spark versions, namely for Hadoop cluster, all cluster applications, all spark applications, and individual spark applications

2) Enable all required URIs for YARN resource manager and Spark Engine. Enable all required permissions for node manager, resource manager, cross-origin resource sharing.

3) Ensure that the port numbers of the URIs in the configuration files, match the corresponding ones in the documentation for the installed versions.

Problem Statement

1) Displaying a semantic analysis of Spark SQL statements, i.e., the most optimal data preprocessing to the user - All Execution Plans

2) Displaying General System Metrics corresponding to the cluster as a whole, as well as each application running on Hadoop.

3) Displaying Specific Metrics about all running Spark Applications

SDLC Followed

Metrics Monitoring UI and Spark Execution Plans Monitor: Front-end Technologies

HTML

CSS

CSS Grid Layout

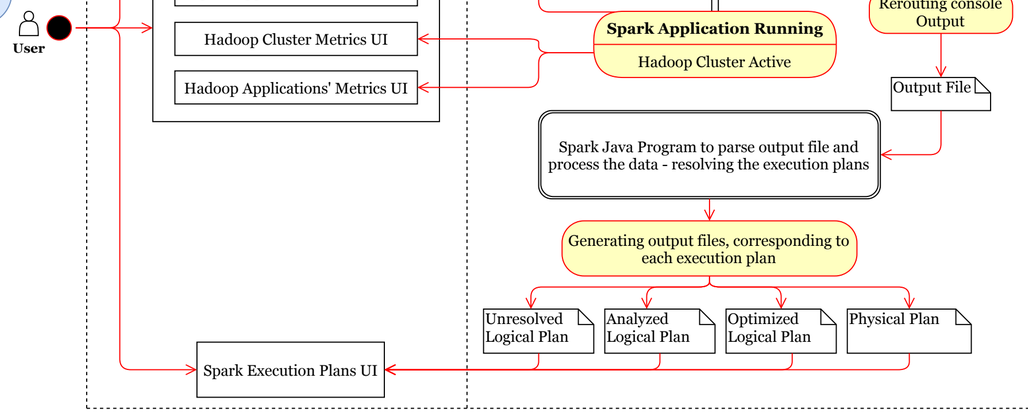
FLEXBOX

Bootstrap 4

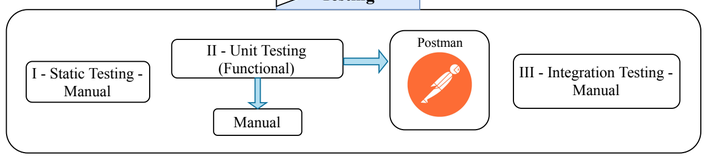
JavaScript

jQuery

Application Architecture



Testing



REST APIs that have been integrated

HADOOP APIs

http://hadoop1.example.com:8088/ws/v1/cluster/apps

http://hadoop1.example.com:8088/ws/v1/cluster/info

http://hadoop1.example.com:8088/ws/v1/cluster/metrics

hadoop1.example.com:8088/ws/v1/cluster/apps/{app-id}

(For example:- hadoop1.example.com:8088/ws/v1/cluster/apps/application_1476912658570_0002)

SPARK APIs

Depending on the number of spark applications currently running the port number may vary from 4040 to 4041, 4042 and so on,...

For example, a spark application running on port number 4042 is given below:

http://hadoop1.example.com:4042/api/v1/applications/

http://hadoop1.example.com:4042/api/v1/applications/local-1590606125695/allexecutors

V - Metrics Monitoring UI

1) The UI is made using HTML, CSS, - grid, flex, bootstrap, and JavaScript - jQuery. REST APIs for the Hadoop and Spark URIs have been integrated.

2) To keep the data updated, HTTP requests are made periodically using JavaScript timer. Both Spark and Hadoop Apps Hub webpages are dynamically generated.

3) We can see the cluster metrics as a whole, generic info about individual Hadoop applications, and detailed metrics about all running Spark Applications

Screenshots