Agenda

- Apache Spark

Apache Spark

Spark Dataframe Programming

Spark Dataframe Execution

- df.explain()
  - Unresolved Logical plan
  - Resolved/Analyzed Logical plan
  - Optimized Logical plan
  - Physical plan
- Catalyst: Spark optimization engine
  - Dataframe/SQL code into Optimized Logical plan
- Catalog: Spark metastore abstraction

Dataframe creation

- Using DataFrameReader
  - df = spark.read.format("csv").option("key", "value").load()
  - df = spark.read.option("key", "value").csv("path")
  - From Java `List<Row>`, `RDD<Row>`, NamedTuple or Dict
    - df = spark.createDataframe(data, schema)
- Schema can be inferred or can given manually.
  - df = spark.read.schema(my_schema).option("key", "value").csv("path")

CSV options

- https://spark.apache.org/docs/latest/sql-data-sources-csv.html

SUNBEAM INFOTECH

spark.read.option("mode", "...")

- DROPMALFORMED: If data is not matching the schema, drop those rows.
- FAILFAST: If data is not matching the schema, fail read operation.
- PERMISSIVE: If data is not matching the schema, consider it as null value (default).

Spark data formats

- Hive use SerDe to write/read data from hive table.
- Dataframes are created using DataframeReader (spark.read) and can be saved using DataframeWriter (df.write).
- df = spark.read.format('...').option(...).load('/file/path')
- df.write.format('...').save('/file/path')
- Supported formats
    - csv, json
    - text
        - value STRING -- single column
    - orc
        - columnar file format
        - designed & optimized for hive
    - parquet
        - columnar file format
        - designed & optimized for spark
        - default format (i.e. if no format is mentioned)
        - efficient than CSV/JSON data.
        - parquet-cli/parq/parquet-tools is python package to read parquet file format.
            - terminal> python3 -m pip install parquet-cli
            - terminal> parq /tmp/output/part-00000-2d1771de-404f-49ba-83f1-4f4c61558623-c000.snappy.parquet
            - terminal> parq /tmp/output/part-00000-2d1771de-404f-49ba-83f1-4f4c61558623-c000.snappy.parquet -c
            - terminal> parq /tmp/output/part-00000-2d1771de-404f-49ba-83f1-4f4c61558623-c000.snappy.parquet --head 5
    - jdbc
        - read/write data from/to RDBMS.

SUNBEAM INFOTECH

Write Modes

```
df.write.mode("...").save()
```

- APPEND: Append contents of this DataFrame to existing data.
- OVERWRITE: Overwrite existing data.
- ERROR or ERRORIFEXISTS: Throw an exception if data already exists.
- IGNORE: Silently ignore this operation if data already exists.

Dataframe columns

- Columns are expressions.
- Expression can be column name or some arithmetic expression or some sql function processing expression.
  - e.g. "job", "sal", "sal + comm", "sal + ifnull(comm,0)", "1 as one", "*", ...
- Selecting columns/expression
  - df.select(c1, c2, ...)
  - df.selectExpr(e1, e2, ...)
  - df.withColumn("colname", "expression") -- add extra column
- Drop column
  - df.drop("colname")

Dataframe rows

- Internally Dataframe is RDD or Row type.
- Row is StructType.
  - e.g. Row(job='CLERK', sum(sal)=4150.0, sum(comm)=None, sum(income)=4150.0)
- Individual column in Row can be accessed using index [n].
- Dataframe operations
  - DF operations are transformations or actions.
  - Transformations produce new dataframe.
  - Actions cause execution plan preparation & execution.
- Transformations

SUNBEAM INFOTECH

    - select(), selectExpr(), where()
    - orderBy(), sort() -- asc/desc and one/more columns
    - limit(), distinct()
    - groupBy("col").someAggOp("col")
    - join()
    - repartition(), coalesce()
- Actions
    - df.show()
    - df.first(), df.take(), df.collect()
    - df.write.format("csv").option("path", "dirpath").save(), df.write.csv("dirpath")
    - df.write.saveAsTable()

Spark SQL Functions

- Numeric functions
    - abs(), floor(), ceil(), round(), pow(), ...
- String functions
    - substring(), lower(), upper(), concat(), ...
- Null value functions
    - nvl(), isnull(), ...
- Date Time functions
    - from_unixtime(), to_timestamp(), to_date(), ...
    - current_date(), current_timestamp(), ...
    - date_diff(), ...
- Aggregate functions
    - sum(), avg(), count(), min(), max(), stddev_pop(), corr(), ...
- Complex type functions
    - explode(), array_contains(), ...
- Window functions
    - rank(), dense_rank(), ...

Spark SQL

SUNBEAM INFOTECH

- Based on Spark structured API i.e. dataframes.
- Enable writing SQL queries on Spark dataframes as views/tables.
- Before Spark 2.x, SQLContext provides SQL functionality.
- Spark 2.x SparkSession encapsulate SparkContext+SQLContext.
- SQLContext use Hive metastore schema to maintain metadata.

Spark Views

- View is abstraction on spark dataframes.
- Created using df.createOrReplaceTempView("viewName")
- createOrReplaceTempView()
  - Creates view if not available.
  - If available, replace with new view.
- View treats dataframe as in memory table & create a view (like SQL view) to execute SQL queries on it.
- The temporary view is in memory only, its info not stored in metastore. It is attached to current sparkSession.
- df.createGlobalTempView("viewName") creates a global view, which can be shared across multiple spark sessions of the same application.

Assignments

1. Wordcount using Spark Dataframes and find top 10 words (except stopwords). Take file from HDFS/S3.
2. Find max sal, min sal, avg sal, total sal per dept per job in emp.csv file.
3. Find deptwise total sal from emp.csv and dept.csv. Print dname and total sal.
4. Count number of movie ratings per year. Hint: convert time column to TIMESTAMP.
5. Movie recommendation using Spark dataframes.
   - Hint: `config("spark.driver.memory", "4g")`
6. Count number of movie ratings per month using sql query (using temp views).
7. Implement movie recommendation system using temp views.
   - Hint: `config("spark.driver.memory", "4g")`
8. Load Fire Service Calls with pre-defined schema. Repeat all 10 Hive assignments on that dataset. Do the assignments using Dataframe syntax. Use Linux command below to create sample dataset.

SUNBEAM INFOTECH