Big Data Technologies

Agenda

- Broadcast and Accumulators
- RDD persistence
- Spark Dataframes
- Spark SQL

Spark cluster & RDD

- Spark Cluster
  - https://www.linkedin.com/posts/nilesh-g_understanding-apache-spark-cluster-activity-6965544536093655040-LE4t
- Spark RDD Word Count
  - https://www.linkedin.com/posts/nilesh-g_bigdata-spark-learning-activity-6962970846302748672-LXWs

Spark RDD Partitions

- For spark cluster defaultParallelism
  - defaultParallelism = Number of CPU cores in local mode.
- sc.parallelize(list, numSlices) will take numSlices or defaultParallelism.
- sc.textFile() will take defaultMinPartitions as minimum of defaultParallelism and 2.
  - https://github.com/apache/spark/blob/e9f983df275c138626af35fd263a7abedf69297f/core/src/main/scala/org/apache/spark/SparkContext.scala#L2329
- sc.textFile() will make number of partitions as the maximum between minPartitions and the number of splits computed based on hadoop input split size divided by the block size.
- sc.textFile() calls sc.hadoopFile(), which creates a HadoopRDD that uses InputFormat.getSplits() under the hadoop [Ref. InputFormat documentation].

```
InputSplit[] getSplits(JobConf job, int numSplits) throws IOException : Logically split the set of input
files for the job. Each InputSplit is then assigned to an individual Mapper for processing. Note: The split
is a logical split of the inputs and the input files are not physically split into chunks. For e.g. a split
```

```
could be tuple. Parameters: job - job configuration. numSplits - the desired number of splits, a hint.
Returns: an array of InputSplits for the job. Throws: IOException.
```

Spark Submit

- spark-submit --help
- spark-submit app.py
  - Submit the job to the master defined in app.py (setMaster(...)).
  - If setMaster() is not given, the job will be submitted to spark master defined in spark-defaults.conf on current machine (spark.master).
- spark-submit --master local app.py
  - local -- local mode with 1 core
  - local[*] -- local mode will all CPU cores on current machine
  - local[n] -- local mode will "n" CPU cores on current machine
- spark-submit --master spark://master:7077 app.py
  - submit the job to master given on command line (no setMaster() to be written in application)
  - The default --deploy-mode is client. For python applications, --deploy-mode=cluster is not supported.
  - By default, it uses all CPU cores in the cluster.
- spark-submit --master spark://master:7077 --total-executor-cores 8 app.py
  - Submit the job to master given on command line (no setMaster() to be written in application).
  - It will use "8" CPU cores from the cluster. Remaining cores can be used for executing other applications.
- spark-submit --properties-file app.conf app.py
  - Here app.conf file contains application config.

```
spark.driver.memory     2g
spark.cores.max         8
```

Broadcast Variables and Accumulators

- Refer demo and slides

RDD Repartitioning

- pyspark --master local[2]

```
rdd1 = sc.parallelize(range(1,100), 4)
rdd1.getNumPartitions()
rdd1.saveAsTextFile('file:///tmp/rdd1')

rdd2 = rdd1.repartition(6)
rdd2.getNumPartitions()
rdd2.saveAsTextFile('file:///tmp/rdd2')

rdd3 = rdd1.repartition(3)
rdd3.getNumPartitions()
rdd3.saveAsTextFile('file:///tmp/rdd3')

rdd4 = rdd1.coalesce(2)
rdd4.getNumPartitions()
rdd4.saveAsTextFile('file:///tmp/rdd4')
```

RDD persistence

- Refer slides and demos

Spark Dataframes

- https://spark.apache.org/docs/2.3.0/sql-programming-guide.html
- https://spark.apache.org/docs/latest/sql-programming-guide.html
- terminal> pyspark --master local

```
file_path = 'file:///home/nilesh/sep22/dbda/bigdata/data/books_hdr.csv'
df1 = spark.read\
        .option('delimiter', ',')\
        .option('header', 'true')\
        .option('inferSchema', 'true')\
        .csv(file_path)
df1.printSchema()

df2 = df1.select('subject', 'price')
df2.printSchema()

df3 = df2.groupBy('subject').sum('price')
df3.printSchema()

df4 = df3.orderBy('subject')
df4.printSchema()

df4.show()
```

```
file_path = 'file:///home/nilesh/sep22/dbda/bigdata/data/books_hdr.csv'
books = spark.read\
        .option('delimiter', ',')\
        .option('header', 'true')\
        .option('inferSchema', 'true')\
        .csv(file_path)

result = books\
        .select('subject', 'price')\
        .groupBy('subject').sum('price')\
        .orderBy('subject')

result.show()
```

CSV options

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

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).

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.

JDBC format

- Add mysql JDBC driver jar into pyspark/jars directory.
  - Download from https://mvnrepository.com/artifact/mysql/mysql-connector-java/8.0.30
  - Copy into /home/nilesh/.local/lib/python3.8/site-packages/pyspark/jars
- Implement spark application to write data into MySQL tables and execute it.
- mysql> USE test;
- mysql> SELECT * FROM emp_summary;

Parquet CLI

- 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

Assignments

- Wordcount using Spark Dataframes and find top 10 words (except stopwords).
- Find max sal per dept per job in emp.csv file.
- Find deptwise total sal from emp.csv and dept.csv. Print dname and total sal.
- Count number of movie ratings per year.
- Movie recommendation using Spark dataframes.
- Clean NCDC data and write year, temperature and quality data into mysql table.
- Read ncdc data from mysql table and print average temperature per year in DESC order.
- Count number of movie ratings per month using sql query (use temp views).