Session 21

Assignment 2

|  |  |
| --- | --- |
| **Prepared For:** | AcadGild |
|  |  |
| **Document Approval:** | **AcadGild** |
|  |  |
|  |  |
|  |  |
|  |  |
| **Project Title:** | Session 21 – Assignment 2 |
|  |  |
| **Prepared By:** | Duncan Burgess |
|  |  |
|  | dburgess@duncb.com |
|  |  |
| **Primary Engineer:** | Duncan Burgess |
|  |  |
| **Document Reference:** | **Session 21 – Assignment 2** |
|  |  |
| **Start Date:** | 22/10/2017 |
|  |  |
|  |  |



# 

# Contents

[Contents 2](#_Toc496417969)

[Change History 3](#_Toc496417970)

[1. Problem Statement 4](#_Toc496417971)

[2. Datasets 4](#_Toc496417972)

[3. Solution 5](#_Toc496417973)

[4. Results 7](#_Toc496417974)

# Change History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Document Revision** | **Date** | **Authored By** | **Authorised By** | **Sections Affected** | **Reason for Change** |
| Rev 01 | 22/10/2017 | Duncan Burgess |  | All | Initial release. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Problem Statement

* Join of two or more data sets is one of the most widely used operations you do with your data, but in distributed systems it can be a huge headache. In general, since your data are distributed among many nodes, they have to be shuffled before a join that causes significant network I/O and slow performance.
* Fortunately, if you need to join a large table with relatively small tables you can avoid sending all data of the large table over the network. This type of join is called map-side join in Hadoop community. **In other distributed systems, it is often called replicated or broadcast join.**

**The fact table can be very large, while dimension tables are often quite small.**

**Let’s use the following sample data (one fact and two dimension tables):**

*// Fact table*

*val flights = sc.parallelize(List(*

*("SEA", "JFK", "DL", "418", "7:00"),*

*("SFO", "LAX", "AA", "1250", "7:05"),*

*("SFO", "JFK", "VX", "12", "7:05"),*

*("JFK", "LAX", "DL", "424", "7:10"),*

*("LAX", "SEA", "DL", "5737", "7:10")))*

*// Dimension table*

*val airports = sc.parallelize(List(*

*("JFK", "John F. Kennedy International Airport", "New York", "NY"),*

*("LAX", "Los Angeles International Airport", "Los Angeles", "CA"),*

*("SEA", "Seattle-Tacoma International Airport", "Seattle", "WA"),*

*("SFO", "San Francisco International Airport", "San Francisco", "CA")))*

*// Dimension table*

*val airlines = sc.parallelize(List(*

*("AA", "American Airlines"),*

*("DL", "Delta Airlines"),*

*("VX", "Virgin America")))*

We need to join the fact and dimension tables to get the following result:

**Seattle New York Delta Airlines 418 7:00**

**San Francisco Los Angeles American Airlines 1250 7:05**

**San Francisco New York Virgin America 12 7:05**

**New York Los Angeles Delta Airlines 424 7:10**

**Los Angeles Seattle Delta Airlines 5737 7:10**

# Datasets

See above in the Problem table for **Flights, Airports** and **Airlines**.

# Solution

All comments are in the code and I print out all steps for my understanding.

**Code written**

**package** com.duncb.spark

**object** joining {

**import** org.apache.spark.\_

//import org.apache.spark.\_

//import org.apache.spark.SparkContext.\_

**import** org.apache.spark.sql.\_

**import** org.apache.log4j.\_

**import** org.apache.spark.sql.functions.\_

**import** org.apache.spark.SparkContext

**def** main(args: Array[*String*]) {

// Create a SparkContext using every core of the local machine

//val sqlContext = new org.apache.spark.sql.SQLContext(sc)

**val** sc = **new** SparkContext("local[\*]", "FriendsByAge")

**val** spark = SparkSession

.builder

.appName("SparkSQL")

.master("local[1]")

.config("spark.sql.warehouse.dir", "file:///C:/temp") // Necessary to work around a Windows bug in Spark 2.0.0; omit if you're not on Windows.

.getOrCreate()

Logger.getLogger("org").setLevel(Level.ERROR)

**val** flights = sc.parallelize(List(

("SEA", "JFK", "DL", "418", "7:00"),("SFO", "LAX", "AA", "1250", "7:05"),("SFO", "JFK", "VX", "12", "7:05"),("JFK", "LAX", "DL", "424", "7:10"),("LAX", "SEA", "DL", "5737", "7:10")))

**val** airports = sc.parallelize(List(

("JFK", "John F. Kennedy International Airport", "New York", "NY"),("LAX", "Los Angeles International Airport", "Los Angeles", "CA"),("SEA", "Seattle-Tacoma International Airport", "Seattle", "WA"),("SFO", "San Francisco International Airport", "San Francisco", "CA")))

**val** airlines = sc.parallelize(List(

("AA", "American Airlines"),("DL", "Delta Airlines"),("VX", "Virgin America")))

//Map Side Join

//This will collect the airports and airlines data as map key-value

**val** airportsMap = spark.sparkContext.broadcast(airports.map{**case**(a,b,c,d) => (a,c)}.collectAsMap())

**val** airlinesMap = spark.sparkContext.broadcast(airlines.collectAsMap())

println("--------------------------------------------------------")

println("airports Map results")

println("--------------------------------------------------------")

airportsMap.value.foreach(x => println(x))

//check airports

println("--------------------------------------------------------")

println("airlines Map results")

println("--------------------------------------------------------")

airlinesMap.value.foreach(x => println(x))

//check airlines

//Perform Map side join

println("--------------------------------------------------------")

println("flights Map results")

println("--------------------------------------------------------")

**val** result = flights.map{

**case**(a,b,c,d,e) =>

(airportsMap.value.get(a).get,

airportsMap.value.get(b).get,

airlinesMap.value.get(c).get,d,e)}

result.foreach(x => println(x))

**import** spark.implicits.\_ //to convert RDDto Dataframe import after Spark context is setup

//The SQL Query and convert RDD into Dataframe

//Converting RDDs to Dataframes

flights.toDF("src","dest","airline\_code","distance","time").createOrReplaceTempView("flights")

airports.toDF("airport\_code","airport\_name","location\_name","location\_code").createOrReplaceTempView("airports")

airlines.toDF("airline\_code","airline\_name").createOrReplaceTempView("airlines")

println("--------------------------------------------------------")

println("flight Dataframe results")

println("--------------------------------------------------------")

spark.sql("select \* from flights").show()

//checking table flights

println("--------------------------------------------------------")

println("airport Dataframe results")

println("--------------------------------------------------------")

spark.sql("select \* from airports").show()

//checking table airports

println("--------------------------------------------------------")

println("airlines Dataframe results")

println("--------------------------------------------------------")

spark.sql("select \* from airlines").show()

//checking table airlines

//This will create a table to produce source and destination name

println("--------------------------------------------------------")

println("Source and Destination results")

println("--------------------------------------------------------")

spark.sql("select t1.src,t2.location\_name as srcName,t1.dest,t3.location\_name as destName,t1.airline\_code,t1.distance,t1.time from flights t1" +

" left outer join airports t2" +

" on t1.src = t2.airport\_code" +

" left outer join airports t3" +

" on t1.dest = t3.airport\_code").createOrReplaceTempView("flightsAirports")

spark.sql("select \* from flightsAirports").show()

println("--------------------------------------------------------")

println("The required result")

println("--------------------------------------------------------")

spark.sql("select t1.srcName as source,t1.destName as destination,t2.airline\_name as airline,t1.distance,t1.time from flightsAirports t1" +

" left outer join airlines t2" +

" on t1.airline\_code = t2.airline\_code").show()

//This will produce the required result

}

}

# Results

The results are map correctly with the required outcome.

--------------------------------------------------------

airports Map results

--------------------------------------------------------

(JFK,New York)

(SFO,San Francisco)

(SEA,Seattle)

(LAX,Los Angeles)

--------------------------------------------------------

airlines Map results

--------------------------------------------------------

(AA,American Airlines)

(DL,Delta Airlines)

(VX,Virgin America)

--------------------------------------------------------

flights Map results

--------------------------------------------------------

(San Francisco,New York,Virgin America,12,7:05)

(New York,Los Angeles,Delta Airlines,424,7:10)

(Los Angeles,Seattle,Delta Airlines,5737,7:10)

(Seattle,New York,Delta Airlines,418,7:00)

(San Francisco,Los Angeles,American Airlines,1250,7:05)

--------------------------------------------------------

flight Dataframe results

--------------------------------------------------------

+---+----+------------+--------+----+

|src|dest|airline\_code|distance|time|

+---+----+------------+--------+----+

|SEA| JFK| DL| 418|7:00|

|SFO| LAX| AA| 1250|7:05|

|SFO| JFK| VX| 12|7:05|

|JFK| LAX| DL| 424|7:10|

|LAX| SEA| DL| 5737|7:10|

+---+----+------------+--------+----+

--------------------------------------------------------

airport Dataframe results

--------------------------------------------------------

+------------+--------------------+-------------+-------------+

|airport\_code| airport\_name|location\_name|location\_code|

+------------+--------------------+-------------+-------------+

| JFK|John F. Kennedy I...| New York| NY|

| LAX|Los Angeles Inter...| Los Angeles| CA|

| SEA|Seattle-Tacoma In...| Seattle| WA|

| SFO|San Francisco Int...|San Francisco| CA|

+------------+--------------------+-------------+-------------+

--------------------------------------------------------

airlines Dataframe results

--------------------------------------------------------

+------------+-----------------+

|airline\_code| airline\_name|

+------------+-----------------+

| AA|American Airlines|

| DL| Delta Airlines|

| VX| Virgin America|

+------------+-----------------+

--------------------------------------------------------

Source and Destination results

--------------------------------------------------------

+---+-------------+----+-----------+------------+--------+----+

|src| srcName|dest| destName|airline\_code|distance|time|

+---+-------------+----+-----------+------------+--------+----+

|LAX| Los Angeles| SEA| Seattle| DL| 5737|7:10|

|SFO|San Francisco| LAX|Los Angeles| AA| 1250|7:05|

|JFK| New York| LAX|Los Angeles| DL| 424|7:10|

|SEA| Seattle| JFK| New York| DL| 418|7:00|

|SFO|San Francisco| JFK| New York| VX| 12|7:05|

+---+-------------+----+-----------+------------+--------+----+

--------------------------------------------------------

The required result

--------------------------------------------------------

+-------------+-----------+-----------------+--------+----+

| source|destination| airline|distance|time|

+-------------+-----------+-----------------+--------+----+

|San Francisco|Los Angeles|American Airlines| 1250|7:05|

| Los Angeles| Seattle| Delta Airlines| 5737|7:10|

| New York|Los Angeles| Delta Airlines| 424|7:10|

| Seattle| New York| Delta Airlines| 418|7:00|

|San Francisco| New York| Virgin America| 12|7:05|

+-------------+-----------+-----------------+--------+----+