Spark-Broadcast Join

Breaking down Broadcast() function

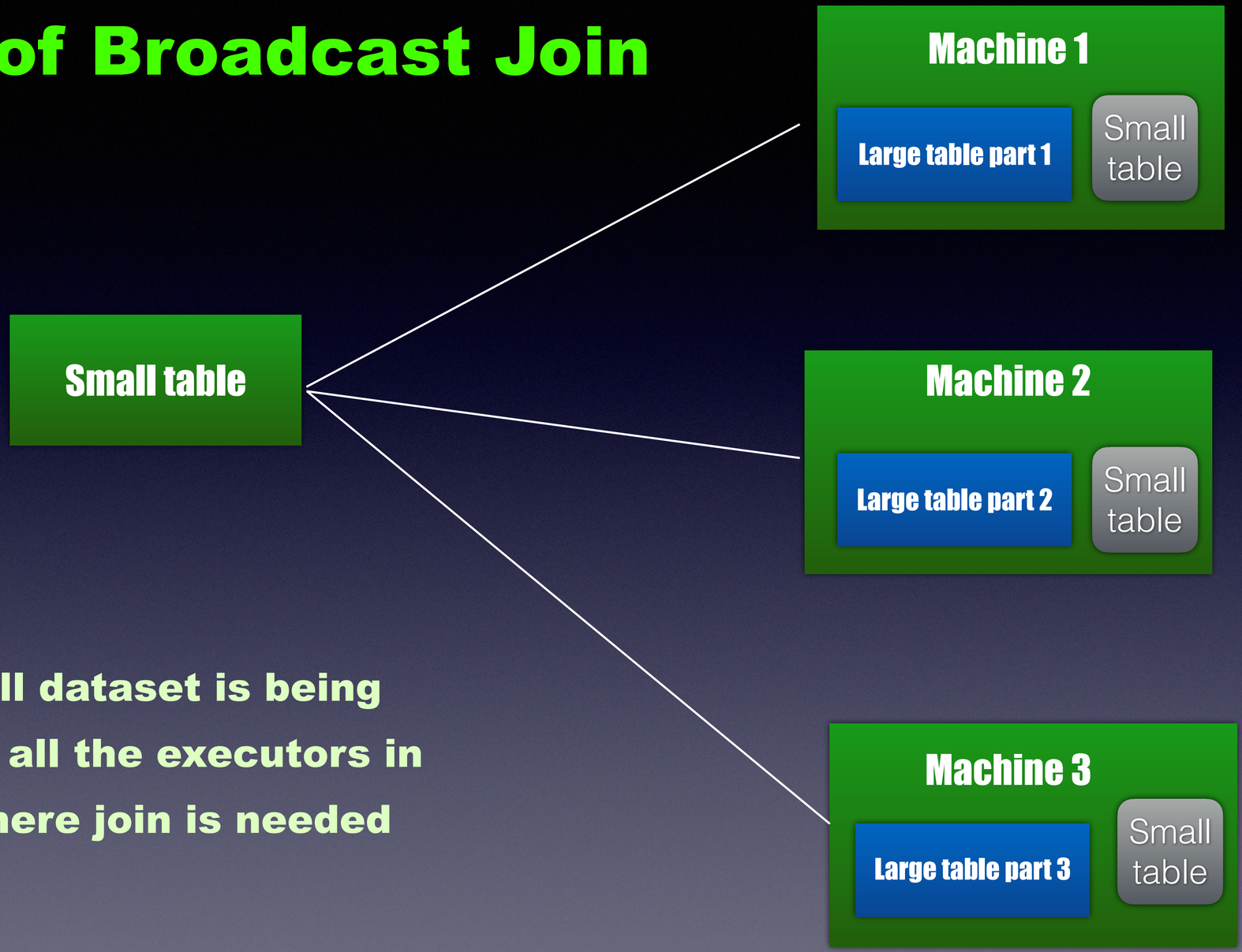
What is Broadcast Join in Spark

- Used for joining a large data frame with a small dataframe
- Cannot be used incase of two large data frames
- It has two phases
- ——>the smaller dataset is broadcasted across the executors in the cluster where the larger table is located
- ——->A standard hash join is performed on each executor

Why Broadcast Join ?

- Spark in default uses Shuffle hash join which involves shuffling both bigger and smaller datasets consuming a lot of time
- In shuffle join, for each join after you finish you have to shuffle it to disk and that can produce many unnecessary intermediate results
- Also data skew occurs as the shuffled data goes into one partition
- Broadcast joins are easier to run on a cluster. Spark can “broadcast” a small DataFrame by sending all the data in that small DataFrame to all nodes in the cluster. After the small DataFrame is broadcasted, Spark can perform a join without shuffling any of the data in the large DataFrame.
- Naturally handles data skew since it doesn’t involve much shuffling

Concept of Broadcast Join



Here the small dataset is being broadcasted to all the executors in the cluster where join is needed

Implementation

- Spark.sql.autoBroadcastJoinThreshold——>this can be configured to set maximum size in bytes for a data frame to be broadcasted
- “-1” will make it disabled
- Default—>10485760—10MB
- joinedTable = largeTable.join(broadcast(smallTable),”equality _operator”)

```scala
def main(args: Array[String]): Unit = {

  val spark = SparkSession
    .builder
    .master( master = "local[*]")
    .appName( name = "BroadcastJoin")
    .getOrCreate()

  //big dataset

  val largeTable = spark.range(1,100000000)  //100 million integers

  //small dataset

  val smallTable = spark.sparkContext.parallelize(List(
    Row(1,"Dhoni"),
    Row(2,"Kohli"),
    Row(3,"Rahul")
  ))
  val rowsSchema = StructType(Array(
    StructField("id", IntegerType),
    StructField("medal", StringType)
  ))

  // small table
  val lookupTable: DataFrame = spark.createDataFrame(smallTable, rowsSchema)


  //normal join
  val joined = largeTable.join(lookupTable,  usingColumn = "id")
  joined.show()

  //broadcast join
  val broadcastJoin = largeTable.join(broadcast(lookupTable),  usingColumn = "id")
  broadcastJoin.show()

}
```

```
kafka
BroadcastJoin › main(args: Array[String])
Run: streamSpark × ConsumerClass × ConsumerClass2 × BroadcastJoin ×
/Library/Java/JavaVirtualMachines/jdk1.8.0_291.jdk/Contents/Home/bin/java ...
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/Users/jagadeesh/Downloads/spark-2.4.7-bin-hadoop2.7%202/jars/slf4j-log4j12-1.7.16.jar!/org/slf4j/impl/StaticLoggerB
SLF4J: Found binding in [jar:file:/Users/jagadeesh/Downloads/spark-2.4.7-bin-hadoop2.7/jars/slf4j-log4j12-1.7.16.jar!/org/slf4j/impl/StaticLoggerBinde
SLF4J: Found binding in [jar:file:/Users/jagadeesh/Downloads/kafka_2.12-2.8.0/libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
+---+-----+
| id|medal|
+---+-----+
|  1|Dhoni|
|  3|Rahul|
|  2|Kohli|
+---+-----+

+---+-----+
| id|medal|
+---+-----+
|  1|Dhoni|
|  2|Kohli|
|  3|Rahul|
+---+-----+


Process finished with exit code 0
```

Output

Code observations

- At default,a sort-merge join on the columns “id” and ”id” (with different identifiers under the hash tag), which requires a sort of the big dataFrame that involves shuffling of big dataFrame
- And a sort,shuffle,filter of the small dataFrame—->costs time—>data skew—->intermediate results
- Whereas BroadcastJoin involves no shuffling as the smaller dataset is broadcasted to all the nodes and join operation is performed across the executors