In [6]:
```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *

spark = SparkSession\
    .builder\
    .appName("chapter-14-broadcast-vars")\
    .getOrCreate()

import os
SPARK_BOOK_DATA_PATH = os.environ['SPARK_BOOK_DATA_PATH']

sc = spark.sparkContext
```

In [7]:
```
my_collection = "Spark The Definitive Guide : Big Data Processing Made S
  .split(" ")
words = sc.parallelize(my_collection, 2)  # numSlices = 2
```

In [8]:
```
type(words)
```

Out[8]:
```
pyspark.rdd.RDD
```

In [3]:
```
words.collect()
```

Out[3]:
```
['Spark',
 'The',
 'Definitive',
 'Guide',
 ':',
 'Big',
 'Data',
 'Processing',
 'Made',
 'Simple,',
 'Spark',
 'in',
 'the',
 'Park,',
 'very',
 'powerful']
```

Broadcast

In [9]:
```
# COMMAND ----------

supplementalData = {"Spark":1000, "Definitive":200,
                    "Big":-300, "Simple":100, "Data": 99}
```

In [10]:
```
# COMMAND ----------

suppBroadcast = sc.broadcast(supplementalData)
```

In [11]:
```
# COMMAND ----------

suppBroadcast.value
```

Out[11]:
```
{'Spark': 1000, 'Definitive': 200, 'Big': -300, 'Simple': 100, 'Data':
99}
```

In [13]:
```
# COMMAND ----------

words.map(lambda word: (word, suppBroadcast.value.get(word, 0)))\
  .sortBy(lambda wordPair: wordPair[1])\
  .collect()
```

Out[13]:
```
[('Big', -300),
 ('The', 0),
 ('Guide', 0),
 (':', 0),
 ('Processing', 0),
 ('Made', 0),
 ('Simple,', 0),
 ('in', 0),
 ('the', 0),
 ('Park,', 0),
 ('very', 0),
 ('powerful', 0),
 ('Data', 99),
 ('Definitive', 200),
 ('Spark', 1000),
 ('Spark', 1000)]
```

Accumulator

In [14]:
```
# COMMAND ----------
file_path = SPARK_BOOK_DATA_PATH + "/data/flight-data/parquet/2010-summa

flights = spark.read.parquet(file_path)
```

In [15]:
```
# COMMAND ----------

accChina = sc.accumulator(0)
```

In [16]:
```
type(accChina)
```

Out[16]:
```
pyspark.accumulators.Accumulator
```

In [17]:
```
# COMMAND ----------

def accChinaFunc(flight_row):
  destination = flight_row["DEST_COUNTRY_NAME"]
  origin = flight_row["ORIGIN_COUNTRY_NAME"]
  if destination == "China":
    accChina.add(flight_row["count"])
  if origin == "China":
    accChina.add(flight_row["count"])
```

In [18]:
```
# COMMAND ----------

flights.foreach(lambda flight_row: accChinaFunc(flight_row))
```

In [19]:
```
# COMMAND ----------

accChina.value # 953
```

Out[19]:
```
953
```

verify accumulator via SQL

In [20]:
```
flights.take(3)
```

Out[20]:
```
[Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='Romania',
count=1),
 Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='Ireland',
count=264),
 Row(DEST_COUNTRY_NAME='United States', ORIGIN_COUNTRY_NAME='India', c
ount=69)]
```

In [21]:
```
type(flights)
```

Out[21]:
```
pyspark.sql.dataframe.DataFrame
```

In [22]:
```
flights.count()
```

Out[22]:
```
255
```

In [23]:
```
flights.where("DEST_COUNTRY_NAME='China'").selectExpr("sum(count)").sho
```

```
+----------+
|sum(count)|
+----------+
|       448|
+----------+
```

In [24]:
```
flights.where("ORIGIN_COUNTRY_NAME='China'").selectExpr("sum(count)").sl
```

```
+----------+
|sum(count)|
+----------+
|       505|
+----------+
```

In [25]:
```
flights.where("DEST_COUNTRY_NAME='China' or ORIGIN_COUNTRY_NAME='China'
```

```
+----------+
|sum(count)|
+----------+
|       953|
+----------+
```

RDD.glom()

Return an RDD created by coalescing all elements within each partition into a list.

In [33]:
```
rdd = sc.parallelize([1, 2, 3, 4], 2)
```

In [34]:
```
rdd.collect()
```

Out[34]:
```
[1, 2, 3, 4]
```

In [35]:
```
rdd.glom().collect()
```

Out[35]:
```
[[1, 2], [3, 4]]
```

In [26]:
```
sc.parallelize([0, 2, 3, 4, 6], 5).glom().collect()
# [[0], [2], [3], [4], [6]]
```

Out[26]:
```
[[0], [2], [3], [4], [6]]
```

In [28]:
```
sc.parallelize(range(0, 6, 2), 5).glom().collect()
# [[], [0], [], [2], [4]]
```

Out[28]:
```
[[], [0], [], [2], [4]]
```

In [ ]:
```

```