Introduction to Spark II

Galvanize

Introduction to Spark II

OBJECTIVES

- **Explain** the differences between RDDs and DataFrames, and where Spark development is moving
- **Explain** what persisting/caching an RDD means, and situations where this is useful
- **Define** an out of memory error and why it happens
- **Describe** the difference between narrow and wide transformations
- **Discuss** next steps to expanding Spark to new domains

The Evolution of Spark

Spark is **moving from RDDs to DataFrames**

- Since Spark 2.0, the emphasis has shifted
- MLlib (RDD-based) -> ML (DF-based)
- GraphX (RDD-based) -> GraphFrames (DF-based)
- Focus on DataFrames for future proofing

SparkSQL

Introducing **SparkSQL**

- An in-memory database
- Access it using `SparkSession`
- The primary abstraction is a DataFrame
- Allows for the execution of SQL commands

```
import pyspark as ps

spark = ps.sql.SparkSession.builder \
            .master("local[4]") \
            .appName("df lecture") \
            .getOrCreate()
```

What is a DataFrame?

DataFrames are...

- Immutable collections of data (like RDDs)
- Distributed across nodes in a cluster
- **Organized into named columns**
  - not schema-less rows, like RDDs
  - Schema = Table Names + Column Names + Column Types
- Think of them like an RDD with a schema

Why are they useful?

- They make large data processing easier
- Allow developers to formalize the structure of the data
- Performance parity
  - Unlike RDDs, which are slower on Python than Scala or Java

```
+--------------------+-----+
|                 col|count|
+--------------------+-----+
|        iHeartAwards|27299|
|         BestFanArmy|19892|
|        BestFans2017|12534|
|        OneDBestFans|10829|
|         GagaBestFans| 9164|
|YOU_NEVER_WALK_ALONE| 8229|
|                 BTS| 7566|
|      CamilaBestFans| 7224|
|            Lovatics| 6491|
|  HappyBirthdayHarry| 5391|
|             NOW2016| 5345|
|   BlackHistoryMonth| 5042|
|       RTした人全員フォローする| 4855|
|   ALDUB81stWeeksary| 4567|
|      ALDUBLoveMonth| 4513|
|      BestMusicVideo| 4435|
|   PBBPADALUCKMAYMAY| 4060|
|           ツインテールの日| 4029|
|          사설토토사이트추천| 3933|
|         gameinsight| 3796|
+--------------------+-----+
```

DataFrame Speed Comparison I

DataFrames are **fast**

- Imposing a schema allows for optimization
- **Catalyst Optimizer** is at the heart of Spark SQL
  - Eases the addition of new optimizations
- **Project Tungsten** improves memory/CPU use

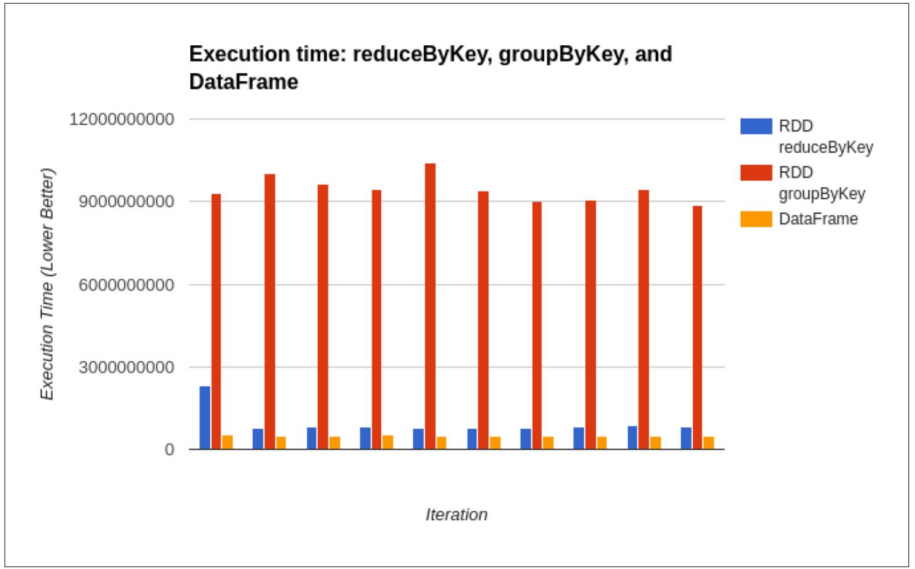


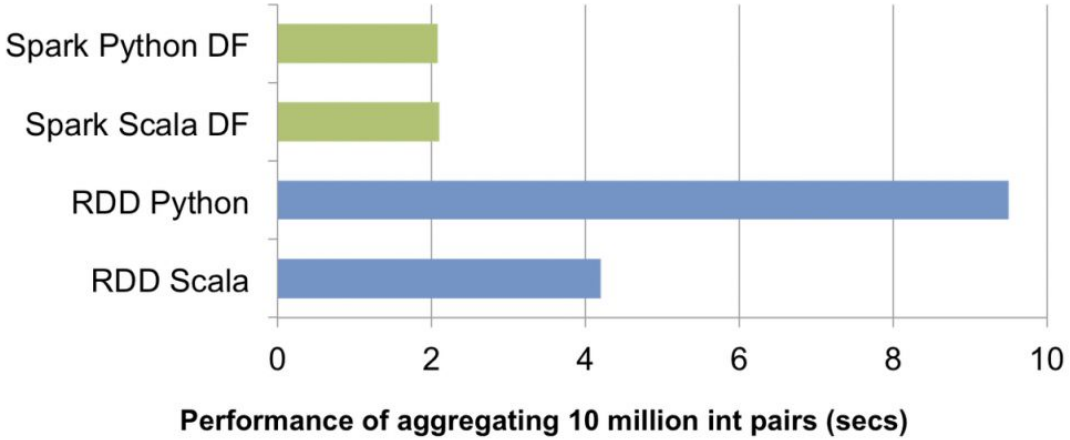
Figure 3-1. Relative performance for RDD versus DataFrames based on SimplePerfTest computing aggregate average fuzziness of pandas

Source: *High Performance Spark*

DataFrame Speed Comparison II

RDDs/Python are **slow**

- There's an overhead between Python and the JVM
- RDDs (especially older versions) suffered from this
- Python is built to be slow (single threaded, not type safe)



Source: Databricks

Creating DataFrames

```
# read JSON
df = spark.read.json('data/sales.json')

# prints the schema
df.printSchema()

# some functions are still valid
print("line count: {}".format(df.count()))

# show the table in a oh-so-nice format
df.show()
```

DataFrames under the hood

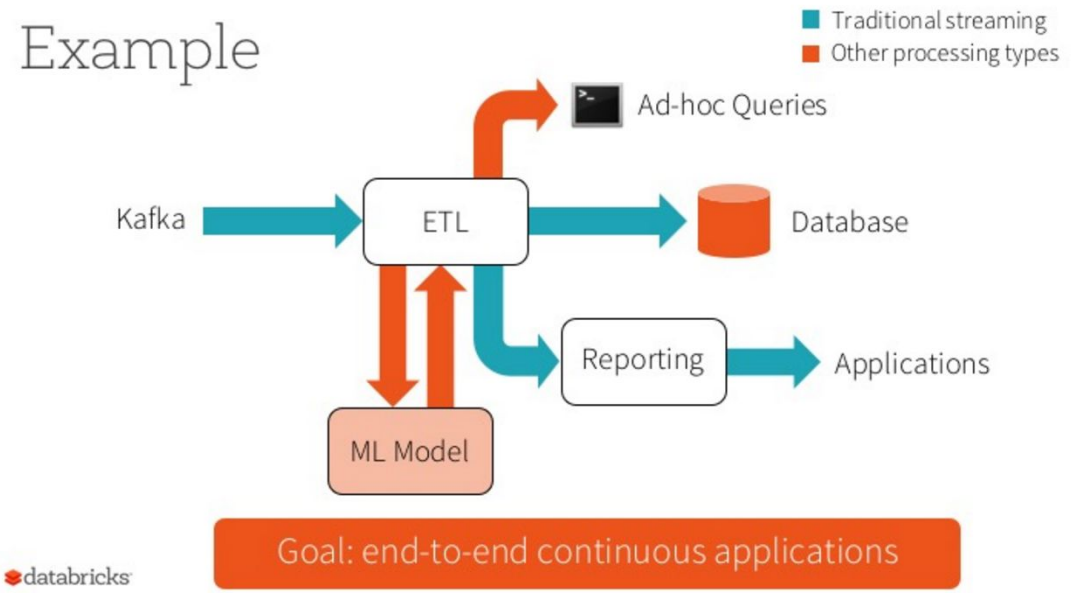
```
# read CSV
df_sales = spark.read.csv('data/sales.csv',
                          header=True,       # use headers or not
                          quote='"',         # char for quotes
                          sep=",",           # char for separation
                          inferSchema=True)  # do we infer schema or
                          not ?

# Now create an SQL table and issue SQL queries against it without
# using the sqlContext but through the SparkSession object.
# Creates a temporary view of the DataFrame
df_sales.createOrReplaceTempView("sales")

result = spark.sql('''
    SELECT state, AVG(amount) as avg_amount
    FROM sales
    GROUP BY state
    ''')
result.show()
```

```
# import the many data types
from pyspark.sql.types import *

# create a schema of your own
schema = StructType( [
    StructField('id',IntegerType(),True),
    StructField('date',StringType(),True),
    StructField('store',IntegerType(),True),
    StructField('state',StringType(),True),
    StructField('product',IntegerType(),True),
    StructField('amount',FloatType(),True) ] )

# feed that into a DataFrame
df = spark.createDataFrame(rdd_sales,schema)

# show the result
df.show()

# print the schema
df.printSchema()
```

The Spark Ecosystem II

Spark can be the **center of your data product**

- Extract Transform Load (ETL)
- Ad-hoc queries (OLAP)
- Transactions (OLTP)
- Stream processing
- Check out the SMACK stack



Source: Databricks

Extending your knowledge of Spark

Possible **future avenues** for understanding Spark

- Probabilistic data structures
- Difference in local and distributed ML algorithms
- Look into the SMACK stack
- Learn Scala to maximize your use of Spark
  - Allows for Datasets API
  - Python can't use Datasets because it's not type safe
- Contribute to Spark core, or community projects

Resources

- Spark Documents: Best stop for the most up to date (and versioned) information
- JF's Walkthroughs: Galvanize instructor who made some helpful walkthroughs
- *Learning PySpark*: An up-to-date intro to the python API to Spark. Good treatment of machine learning
- *Learning Spark*: A good introduction to Spark written on Spark 1.X. Parts of it are still relevant for a high level overview, but most of it is outdated
- *High Performance Spark*: More recent publication by the author of *Learning Spark*
- Databrick's explanation of key terms

Questions?