Apache Spark

Andrew Peterson

Big Data in Finance
Baruch Master in Financial Engineering

March 8, 2016

Materials

Materials for this tutorial are at:

`https://github.com/aristotle-tek/cuny-bdif/`

Spark basics

- Alternative to Hadoop map-reduce writing changes to disk
- resilient distributed dataset (RDD): fault-tolerant, read-only storage
- in memory functional programming – lazy execution, logical plan optimization...

Spark Components

- RDDs
- Spark Streaming
- Spark SQL, Datasets, and DataFrames
- MLlib - machine learning library
- GraphX - graph processing

RDDs: Actions & Transformations

- Actions return values, e.g. `count`
- Transformations return pointers to new RDDs, e.g. `filter`

Spark and Public Twitter Data

- In theory, we should be able to generate an RDD directly from the archive on s3 / from HDFS:
  ```
  val tweets = sqlContext.read.json("twitter.tar")
  ```
- In practice, (1) bad lines in load, and (2) slow transfer from s3 ($\sim$ 20 min for 20GB on m3.large)
- Today we will work with sample files.

Data preparation

- (see `AWS/setup.sh` in my github)
- Need to export path to hadoop:

```
export PATH=$PATH:/root/ephemeral-hdfs/bin

hadoop fs -mkdir tweets
hadoop fs -ls
hadoop fs -put *.json tweets/
```

Spark and Spark-Shell

- Run shell with:
  `./spark/bin/spark-shell --driver-memory 7G`
- For s3 access, add `--copy-aws-credentials`
- Can also add jars with `--jars <foo>.jar`

Basic Methods: Transformation examples

- `map`, `filter`
- `mapPartitions` - like map but runs on separate partitions
- `sample`, `union`
- `groupByKey` - dataset of key-value pairs, return reduced using a fn
- `reduceByKey` - dataset of key-value pairs, return dataset using combine fn with a 0 value
- etc

Basic Methods: Action examples

- `count`
- `first`, `take`, `foreach`
- `reduce` - aggregate using a fn
- `collect` - return as an array etc

Basic Methods

- Functions

```
object MyFunctions {
  def func1(s: String): String = { ... }
}
myRdd.map(MyFunctions.func1)
```

- Cache a transformation in memory with `.cache()`