## Apache Spark Examples

These examples give a quick overview of the Spark API. Spark is built on the concept of distributed datasets, which contain arbitrary Java or Python objects. You create a dataset from external data, then apply parallel operations to it. The building block of the Spark API is its [RDD API](http://spark.apache.org/docs/latest/programming-guide.html" \l "resilient-distributed-datasets-rdds). In the RDD API, there are two types of operations: transformations, which define a new dataset based on previous ones, and actions, which kick off a job to execute on a cluster. On top of Spark’s RDD API, high level APIs are provided, e.g.[DataFrame API](http://spark.apache.org/docs/latest/sql-programming-guide.html" \l "dataframes) and [Machine Learning API](http://spark.apache.org/docs/latest/mllib-guide.html). These high level APIs provide a concise way to conduct certain data operations. In this page, we will show examples using RDD API as well as examples using high level APIs.

## RDD API Examples

### Word Count

In this example, we use a few transformations to build a dataset of (String, Int) pairs called counts and then save it to a file.

* [Python](http://spark.apache.org/examples.html)
* [Scala](http://spark.apache.org/examples.html)
* [Java](http://spark.apache.org/examples.html)

JavaRDD<String> textFile = sc.textFile("hdfs://...");JavaRDD<String> words = textFile.flatMap(**new** FlatMapFunction<String, String>() {

**public** Iterable<String> call(String s) { **return** Arrays.asList(s.split(" ")); }});JavaPairRDD<String, Integer> pairs = words.mapToPair(**new** PairFunction<String, String, Integer>() {

**public** Tuple2<String, Integer> call(String s) { **return** **new** Tuple2<String, Integer>(s, 1); }});JavaPairRDD<String, Integer> counts = pairs.reduceByKey(**new** Function2<Integer, Integer, Integer>() {

**public** Integer call(Integer a, Integer b) { **return** a + b; }});counts.saveAsTextFile("hdfs://...");

### Pi Estimation

Spark can also be used for compute-intensive tasks. This code estimates π by "throwing darts" at a circle. We pick random points in the unit square ((0, 0) to (1,1)) and see how many fall in the unit circle. The fraction should be π / 4, so we use this to get our estimate.

* [Python](http://spark.apache.org/examples.html)
* [Scala](http://spark.apache.org/examples.html)
* [Java](http://spark.apache.org/examples.html)

List<Integer> l = **new** ArrayList<Integer>(NUM\_SAMPLES);**for** (int i = 0; i < NUM\_SAMPLES; i++) {

l.add(i);}

long count = sc.parallelize(l).filter(**new** Function<Integer, Boolean>() {

**public** Boolean call(Integer i) {

double x = Math.random();

double y = Math.random();

**return** x\*x + y\*y < 1;

}}).count();System.out.println("Pi is roughly " + 4.0 \* count / NUM\_SAMPLES);

## DataFrame API Examples

In Spark, a [DataFrame](http://spark.apache.org/docs/latest/sql-programming-guide.html" \l "dataframes) is a distributed collection of data organized into named columns. Users can use DataFrame API to perform various relational operations on both external data sources and Spark’s built-in distributed collections without providing specific procedures for processing data. Also, programs based on DataFrame API will be automatically optimized by Spark’s built-in optimizer, Catalyst.

### Text Search

In this example, we search through the error messages in a log file.

* [Python](http://spark.apache.org/examples.html)
* [Scala](http://spark.apache.org/examples.html)
* [Java](http://spark.apache.org/examples.html)

*// Creates a DataFrame having a single column named "line"*JavaRDD<String> textFile = sc.textFile("hdfs://...");JavaRDD<Row> rowRDD = textFile.map(

**new** Function<String, Row>() {

**public** Row call(String line) **throws** Exception {

**return** RowFactory.create(line);

}

});List<StructField> fields = **new** ArrayList<StructField>();fields.add(DataTypes.createStructField("line", DataTypes.StringType, **true**));StructType schema = DataTypes.createStructType(fields);DataFrame df = sqlContext.createDataFrame(rowRDD, schema);

DataFrame errors = df.filter(col("line").like("%ERROR%"));*// Counts all the errors*errors.count();*// Counts errors mentioning MySQL*errors.filter(col("line").like("%MySQL%")).count();*// Fetches the MySQL errors as an array of strings*errors.filter(col("line").like("%MySQL%")).collect();

### Simple Data Operations

In this example, we read a table stored in a database and calculate the number of people for every age. Finally, we save the calculated result to S3 in the format of JSON. A simple MySQL table "people" is used in the example and this table has two columns, "name" and "age".

* [Python](http://spark.apache.org/examples.html)
* [Scala](http://spark.apache.org/examples.html)
* [Java](http://spark.apache.org/examples.html)

*// Creates a DataFrame based on a table named "people"// stored in a MySQL database.*String url =

"jdbc:mysql://yourIP:yourPort/test?user=yourUsername;password=yourPassword";DataFrame df = sqlContext

.read()

.format("jdbc")

.option("url", url)

.option("dbtable", "people")

.load();

*// Looks the schema of this DataFrame.*df.printSchema();

*// Counts people by age*DataFrame countsByAge = df.groupBy("age").count();countsByAge.show();

*// Saves countsByAge to S3 in the JSON format.*countsByAge.write().format("json").save("s3a://...");

## Machine Learning Example

[MLlib](http://spark.apache.org/docs/latest/mllib-guide.html), Spark’s Machine Learning (ML) library, provides many distributed ML algorithms. These algorithms cover tasks such as feature extraction, classification, regression, clustering, recommendation, and more. MLlib also provides tools such as ML Pipelines for building workflows, CrossValidator for tuning parameters, and model persistence for saving and loading models.

### Prediction with Logistic Regression

In this example, we take a dataset of labels and feature vectors. We learn to predict the labels from feature vectors using the Logistic Regression algorithm.

* [Python](http://spark.apache.org/examples.html)
* [Scala](http://spark.apache.org/examples.html)
* [Java](http://spark.apache.org/examples.html)

*// Every record of this DataFrame contains the label and// features represented by a vector.*StructType schema = **new** StructType(**new** StructField[]{

**new** StructField("label", DataTypes.DoubleType, **false**, Metadata.empty()),

**new** StructField("features", **new** VectorUDT(), **false**, Metadata.empty()),});DataFrame df = jsql.createDataFrame(data, schema);

*// Set parameters for the algorithm.// Here, we limit the number of iterations to 10.*LogisticRegression lr = **new** LogisticRegression().setMaxIter(10);

*// Fit the model to the data.*LogisticRegressionModel model = lr.fit(df);

*// Inspect the model: get the feature weights.*Vector weights = model.weights();

*// Given a dataset, predict each point's label, and show the results.*model.transform(df).show();

# Additional Examples

Many additional examples are distributed with Spark:

* Basic Spark: [Scala examples](https://github.com/apache/spark/tree/master/examples/src/main/scala/org/apache/spark/examples), [Java examples](https://github.com/apache/spark/tree/master/examples/src/main/java/org/apache/spark/examples), [Python examples](https://github.com/apache/spark/tree/master/examples/src/main/python)
* Spark Streaming: [Scala examples](https://github.com/apache/spark/tree/master/examples/src/main/scala/org/apache/spark/examples/streaming), [Java examples](https://github.com/apache/spark/tree/master/examples/src/main/java/org/apache/spark/examples/streaming)