Our example app will be a souped-up version of WordCount, the classic MapReduce example. In WordCount, the goal is to learn the distribution of letters in the most popular words in our corpus. That is, we want to:

1. Read an input set of text documents
2. Count the number of times each word appears
3. Filter out all words that show up less than given times
4. For the remaining set, count the number of times each letter occurs

Scala program

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf

object SparkWordCount {
  def main(args: Array[String]) {
    val sc = new SparkContext(new SparkConf().setAppName("Spark Count"))
    val threshold = args(1).toInt

    // split each document into words
    val tokenized = sc.textFile(args(0)).flatMap(_.split(" "))

    // count the occurrence of each word
    val wordCounts = tokenized.map((_, 1)).reduceByKey(_ + _)

    // filter out words with less than threshold occurrences
    val filtered = wordCounts.filter(_._2 >= threshold)

    // count characters
    val charCounts = filtered.flatMap(_._1.toCharArray).map((_, 1)).reduceByKey(_ + _)

    System.out.println(charCounts.collect().mkString(", "))
  }
}
```

Compiling

We'll use Maven to compile our program. Maven expects a specific directory layout that informs it where to look for source files. Our Scala code goes under src/main/scala. That is, we place SparkWordCount.scala in the src/main/scala/org/arkenstone.

Maven also requires you to place a pom.xml file in the root of the project directory that tells it how to build the project. A few noteworthy excerpts are included below.

To compile Scala code, include:

```
<plugin>
 <artifactId>maven-compiler-plugin</artifactId>
 <version>2.3.2</version>
 <configuration>
  <source>1.7</source>
  <target>1.7</target>
 </configuration>
</plugin>
```

which requires adding the scala-tools plugin repository:

```
<pluginRepository>
 <id>scala-tools.org</id>
 <name>Scala-tools Maven2 Repository</name>
 <url>http://scala-tools.org/repo-releases</url>
</pluginRepository>
```

Then, include Spark and Scala as dependencies:

```
<dependency>
    <groupId>org.scala-lang</groupId>
    <artifactId>scala-library</artifactId>
    <version>2.10.4</version>
  </dependency>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.10</artifactId>
    <version>1.6.0-cdh5.7.2</version>
</dependency>
```

Finally, to generate our app jar, simply run:

mvn package

It will show up in the target directory as sparkcsv-1.0-SNAPSHOT.jar.

Run

Before running, place the input file into a directory on HDFS. The repository supplies an example input file in its data directory. To run the Spark program, we use the spark-submit script:

[cloudera@quickstart SparkT]$ spark-submit --class org.arkenstone.SparkWordCount --master local target/sparkcsv-1.0-SNAPSHOT.jar /user/LZ/spark/Count.txt 2

Input:
[cloudera@quickstart SparkT]$ hadoop fs -cat /user/LZ/spark/Count.txt
apple
banana counter
counter one two three
three one
five seven eight
twenty one three five counter six
one siz helga
apple banana fiver

Output:

(e,6), (p,2), (a,4), (t,2), (i,1), (b,1), (u,1), (h,1), (o,2), (n,4), (f,1), (v,1), (r,2), (l,1), (c,1)