Writing the Application

The example application is an enhanced version of word count, the canonical MapReduce example. In this version of WordCount, the goal is to learn the distribution of letters in the most popular words in a corpus. The application:

1. Creates a `SparkConf` and `SparkContext`. A Spark application corresponds to an instance of the `SparkContext` class. When running a shell, the `SparkContext` is created for you.
2. Gets a word frequency threshold.
3. Reads an input set of text documents.
4. Counts the number of times each word appears.
5. Filters out all words that appear fewer times than the threshold.
6. For the remaining words, counts the number of times each letter occurs.

In MapReduce, this requires two MapReduce applications, as well as persisting the intermediate data to HDFS between them. In Spark, this application requires about 90 percent fewer lines of code than one developed using the MapReduce API.

Scala WordCount

```
import org.apache.spark.SparkContext

import org.apache.spark.SparkContext._

import org.apache.spark.SparkConf

object SparkWordCount {

 def main(args: Array[String]) {

   // create Spark context with Spark configuration

   val sc = new SparkContext(new SparkConf().setAppName("Spark Count"))



   // get threshold

   val threshold = args(1).toInt



   // read in text file and 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 fewer 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(", "))
 }
}
```

Python WordCount

```
import sys
from pyspark import SparkContext, SparkConf
if __name__ == "__main__":
 # create Spark context with Spark configuration
 conf = SparkConf().setAppName("Spark Count")
 sc = SparkContext(conf=conf)

 # get threshold
 threshold = int(sys.argv[2])
```

```
 # read in text file and split each document into words
```

```
 tokenized = sc.textFile(sys.argv[1]).flatMap(lambda line: line.split(" "))


 # count the occurrence of each word
 wordCounts = tokenized.map(lambda word: (word, 1)).reduceByKey(lambda v1,v2:v1 +v2)


 # filter out words with fewer than threshold occurrences
 filtered = wordCounts.filter(lambda pair:pair[1] >= threshold)


 # count characters
 charCounts = filtered.flatMap(lambda pair:pair[0]).map(lambda c: c).map(lambda c: (c,
1)).reduceByKey(lambda v1,v2:v1 +v2)


 list = charCounts.collect()
 print repr(list)[1:-1]
```