CS5543 Real-Time Big Data Analytics

MapReduce & Spark Programing

InClassEx-4

9/13/2015

Name: Bill Capps
Class ID:

MapReduce & Spark Programming – Joining two data sets

Two datasets are given as follows:

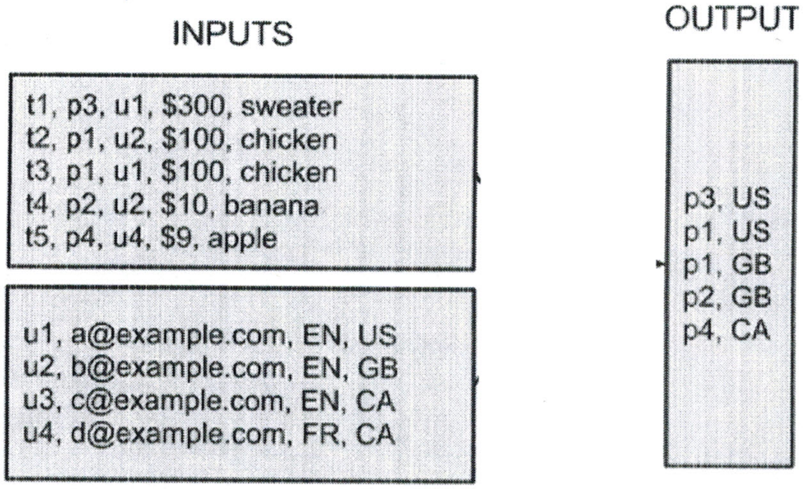
1. User information (id, email, language, location)
2. Transaction information (transaction-id, product-id, user-id, purchase-amount, item-description)

Given these datasets, find the number of unique locations in which each product has been sold.

1) Draw the MapReduce Diagram.
2) Sketch the MapReduce Algorithm.
3) Sketch the Spark Scala implementation

Example

The result is:



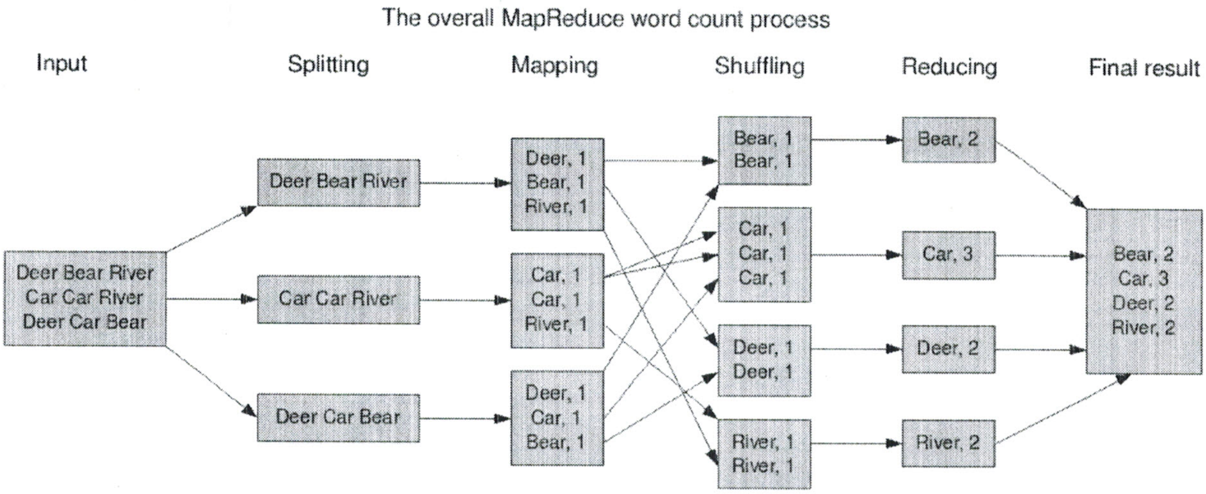
INPUTS

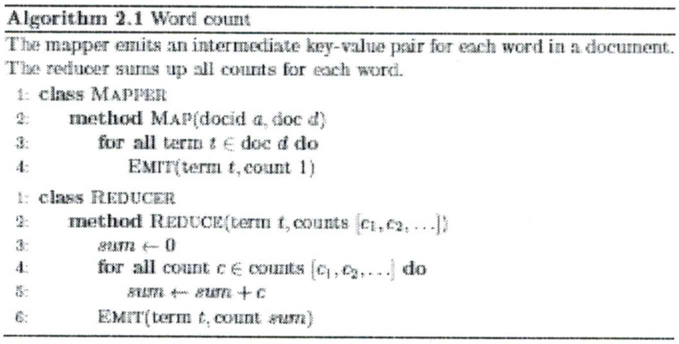
```
t1, p3, u1, $300, sweater
t2, p1, u2, $100, chicken
t3, p1, u1, $100, chicken
t4, p2, u2, $10, banana
t5, p4, u4, $9, apple
```

```
u1, a@example.com, EN, US
u2, b@example.com, EN, GB
u3, c@example.com, EN, CA
u4, d@example.com, FR, CA
```

OUTPUT

```
p3, US
p1, US
p1, GB
p2, GB
p4, CA
```



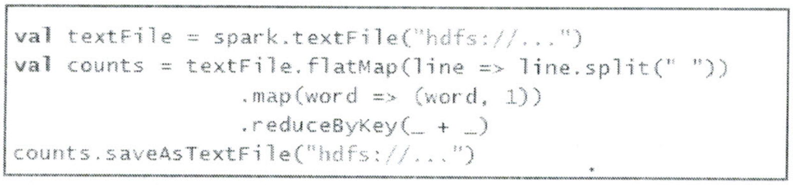


Algorithm 2.1 Word count

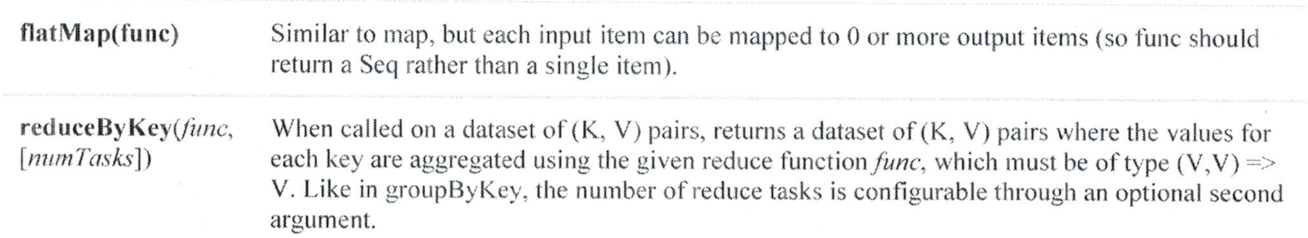
The mapper emits an intermediate key-value pair for each word in a document. The reducer sums up all counts for each word.

```
class MAPPER
    method MAP(docid $a$, doc $d$)
        for all term $t \in$ doc $d$ do
            EMIT(term $t$, count 1)
class REDUCER
    method REDUCE(term $t$, counts $[c_1, c_2, \ldots]$)
        $sum \leftarrow 0$
        for all count $c \in$ counts $[c_1, c_2, \ldots]$ do
            $sum \leftarrow sum + c$
        EMIT(term $t$, count $sum$)
```

Spark Scala Code for WordCount

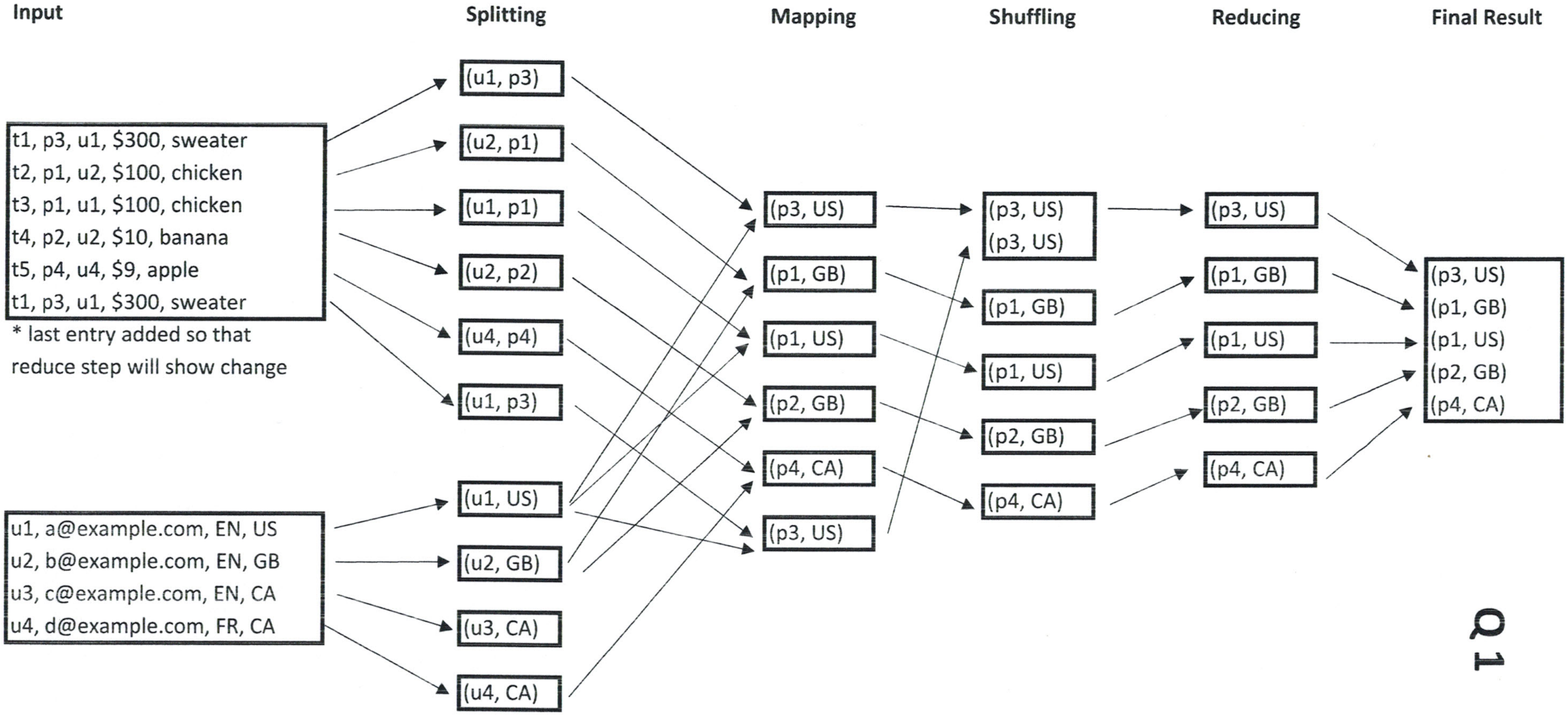


```
val textFile = spark.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```



flatMap(func)	Similar to map, but each input item can be mapped to 0 or more output items (so func should return a Seq rather than a single item).
reduceByKey(*func*, [*numTasks*])	When called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function *func*, which must be of type (V,V) => V. Like in groupByKey, the number of reduce tasks is configurable through an optional second argument.

Q 1



Input

t1, p3, u1, $300, sweater
t2, p1, u2, $100, chicken
t3, p1, u1, $100, chicken
t4, p2, u2, $10, banana
t5, p4, u4, $9, apple
t1, p3, u1, $300, sweater

* last entry added so that reduce step will show change

u1, a@example.com, EN, US
u2, b@example.com, EN, GB
u3, c@example.com, EN, CA
u4, d@example.com, FR, CA

Splitting

(u1, p3)
(u2, p1)
(u1, p1)
(u2, p2)
(u4, p4)
(u1, p3)
(u1, US)
(u2, GB)
(u3, CA)
(u4, CA)

Mapping

(p3, US)
(p1, GB)
(p1, US)
(p2, GB)
(p4, CA)
(p3, US)

Shuffling

(p3, US)
(p3, US)

(p1, GB)

(p1, US)

(p2, GB)

(p4, CA)

Reducing

(p3, US)
(p1, GB)
(p1, US)
(p2, GB)
(p4, CA)

Final Result

(p3, US)
(p1, GB)
(p1, US)
(p2, GB)
(p4, CA)

2)

```
Class MAPPER

        Method MAP(docid a, doc d1, docid b, doc d2)

                For all term t1 ϵ doc d1 do

                        For all term t2 ϵ doc d2 do

                                IF t1.key == t2.key then EMIT(t1.value, t2.value)


Class REDUCER

        Method REDUCE(term t1, term t2)

                For all Distinct(t1, t2)

                        EMIT(t1, t2)
```

3)

```
package ICE4
import org.apache.spark.{SparkConf, SparkContext}
object ICE4 {
 def main(args : Array[String]){
  // administration
  System.setProperty("hadoop.home.dir", "C:\\winutils")
  val config = new SparkConf()
   .setAppName("ICE4")
   .setMaster("local[*]")
  val sc = new SparkContext(config)
  // read in data
  val textTransaction = sc.textFile("src/main/scala/ICE4/input_transaction.txt")
  val textUser = sc.textFile("src/main/scala/ICE4/input_user.txt")
  // map transactions
  val result = textTransaction.map(x => (x.split(", ")(2),x.split(", ")(1)))
   // map users to transactions
   .join(textUser.map(x => (x.split(", ")(0),x.split(", ")(3))))
   // reduce to distinct values
   .values.distinct()
  // output results
  result.saveAsTextFile("src/main/scala/ICE4/output")
   }
}
```