Nubimetrics Challenge

Aquiles Barreto

El desarrollo del Challenge lo llevé a cabo utilizando **Scala 2.12.8**. El código fuente del proyecto está en un **repositorio git** en el siguiente link https://github.com/barretoaquiles/nubimetrics y el listado resultante está en el archivo **cellbrandslisting.csv** ubicado en el mismo repositorio. Este archivo tiene tres columnas: Marca, Monto y Unidades.

Inicialmente revisé el contenido del archivo **cellphoneslisting.csv** y observé que en la columna title no sólo habían datos descriptivos de teléfonos celulares sino de ofertas múltiples de teléfonos tales como:

305 Celulares Basicos Nuevos Alcatel Lg Samsung Blu Tronika 4845.0 5

Para obtener al menos una marca para cada fila del archivo es importante contar con un diccionario de marcas que nos permita buscar cada una de las marcas que se encuentran en el archivo para que sean buscadas en cada fila, se puede crear un arreglo como sigue:

val brands = Array("Asus", "Bgh", "Blackberry", "Blu", "Caterpillar", "Cx", "Go", "Hyundai", "Htc", "Huawei", "Iphone", "Kanji", "K10", "Ken", "Lg", "Maxwest", "Moto", "Motorola", "Nextel", "Nokia", "Panacom", "Philco", "Samsung", "Tcl", "Ulefone", "Xiaomi", "Zte")

Seguidamente, leemos el archivo **cellphoneslisting.csv** y hacemos split de los datos por el separador de filas (**\u0001**), luego nos valemos de un map para instanciar a variables cada columna del archivo. Tomamos el diccionario o array de marcas **brands** y para todas las filas de la columna **title** determinamos si existe al menos una marca y de existir, retorno el índice de la marca en el array **brands**, éste índice lo usaré a continuación. Tomo él índice y obtengo la marca de esa posición y junto al monto y las unidades construyo una tupla de 3 valore para columna. Con el conjunto de estas tuplas armo una lista del tamaño del archivo, todo esto proceso lo podemos observar a continuación:

```
val dataPhones = Source.fromFile("cellphoneslisting.csv").getLines().drop(1)
  .map(_.split("\u0001"))
  .map { x =>
    val v1 = x(0)
    val v2 = {
      brands.indexWhere(p => x(1).contains(p))
    }
    val v3 = x(2).toDouble
    val v4 = x(3).toLong
    if (v2 != -1)
      (brands(v2), v3, v4)
    else
      ("Otros", v3, v4)
  } toList
```

Luego, como tendré marcas repetidas, realizo un agrupamiento por marcas y a partir de esto ejecuto un map-reduce para sumar los montos y unidades de cada una de las marcas, tal como sigue:

```
val listPhones = dataPhones.groupBy(_._1).map {
row => (row._1, row._2.map(_._2).reduce(_+_), row._2.map(_._3).reduce(_+_))}
```

Finalmete se construye el arcvhio csv resultante a partir del map-reduce anterior.

El código fuente completo es:

```
import java.io.{FileWriter, PrintWriter}
import scala.io.Source

object Main {

  val brands = Array("Asus", "Bgh", "Blackberry", "Blu", "Caterpillar", "Cx", "Go", "Hyundai", "Htc", "Huawei", "Iphone", "Kanji", "K10", “Ken", "Lg", "Maxwest", "Moto", "Motorola", “Nextel", "Nokia", "Panacom", "Philco", "Samsung", "Tcl", "Ulefone", "Xiaomi", "Zte")

  def main(args: Array[String]): Unit = {

    val dataPhones = Source.fromFile("cellphoneslisting.csv").getLines().drop(1)

      .map(_.split("\u0001"))
      .map { x =>
        val v1 = x(0)
        val v2 = {
          brands.indexWhere(p => x(1).contains(p))
        }
        val v3 = x(2).toDouble
        val v4 = x(3).toLong
        if (v2 != -1)
          (brands(v2), v3, v4)
        else
          ("Otros", v3, v4)
      } toList

    val listPhones = dataPhones.groupBy(_._1).map {
      row => (row._1, row._2.map(_._2).reduce(_+_), row._2.map(_._3).reduce(_+_))
    }

    val list = new PrintWriter(new FileWriter("cellbrandslisting.csv"))
    list.println("Marca,Monto,Unidades")
    listPhones.foreach(p => (list.print(p._1 + ","), list.print(p._2 + ","), list.println(p._3)))
    list.close()
  }
}
```

El modelo de negocio se basa en el uso efectivo de maps y reducciones para simplificar el tamaño del desarrollo a sólo unas pocas líneas y un cálculo computacional más efectivo. Se probaron otras alternativas y la expuesta resultó la más eficiente computacionalmente.

En pasos, se requiere contar con una estructura que mantenga las marcas existentes en el archivo, luego se desglosan las columnas en variables separadas, para cada fila de la columna de title se busca la marca presente y se construye una lista con una terna de valores, Marca, Monto y Unidades. Seguidamente se agrupa la lista por marcas y se hace un Map-Reduce para sumar los montos y uniades, finalmente se crea el archivo csv con el reporte esperado.

Listado resultante:

```
Marca,Monto,Unidades
Maxwest,115491.31000000001,85
Lg,2.670286735E7,3802
Ken,9998.0,2
Bgh,552883.9299999999,163
Xiaomi,6.0576330839999996E7,5055
Nextel,10853.65,5
Huawei,2.131447445E7,1545
Panacom,106128.54000000001,108
Asus,34450.0,2
Blu,4530646.9799999995,1352
Go,2036566.83,287
Iphone,1.4582383210000005E7,533
Nokia,1588156.18,614
K10,6434744.970000001,1221
Kanji,154712.9,184
Cx,1599.0,1
Samsung,8.558031238999999E7,8682
Zte,168182.76,42
Htc,17249.989999999998,2
Philco,199929.0,72
Blackberry,132654.94,41
Tcl,1086390.0899999999,260
Hyundai,1151612.1,269
Caterpillar,3314833.8800000004,268
Moto,5.5114844789999984E7,5591
Otros,1.4330470220000004E7,2803
Ulefone,115080.0,7
```