Secondary Sort

Objective

Technique for preferential sorting per-partition

RepartitionAndSortWithinPartitions

To Remember

RepartitionAndSortWithinPartitions

- combines preferential partitioner + preferential sorting by key
- embeds sorting in the shuffle process
- is faster than a repartition + mapPartitions with sort
- works best with an I2I transformation right after

Technnique: secondary sort in key-value RDDs

```
class MyPartitioner(override val numPartitions: Int) extends Partitioner {
     override def getPartition(key: Any) = {
          val (k, _) = key.asInstanceOf[(String, Double)]
          Math.abs(k.hashCode()) % numPartitions
     }
}
```

```
val metrics = df.rdd
     // add dummy values
     .map(row => ((row.getString(0), row.getDouble(1)), 1))
     // repartition by the hash of the key, sort by the value - in the SAME STEP
     .repartitionAndSortWithinPartitions(partitioner)
     .mapPartitions { pairs =>
          // process all the tuples as you see fit, with an I2I transformation
     }
```

Spark rocks