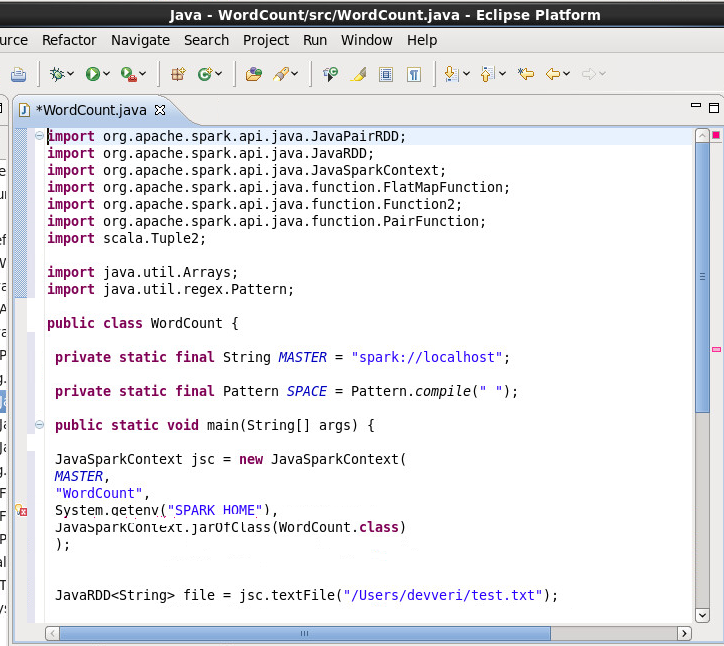
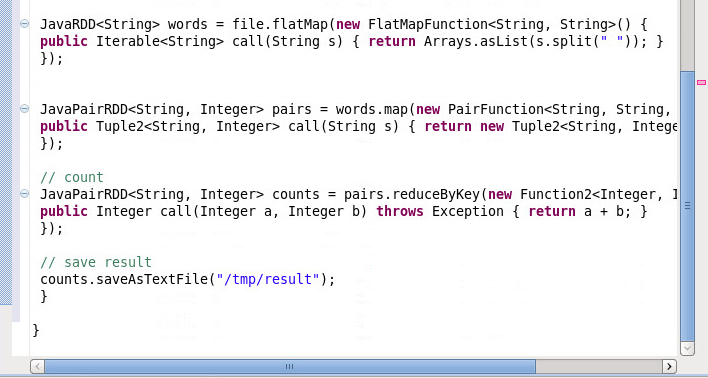
I started to learn Python this week and I was thrown into the Apache Spark. Since I already know the java language, I think it would be easier with it. However, after exploring the Python, Java and Scala languages ​​supported by Spark, I realized that the easiest method is Scala. So I tried to do the same thing with Scala by abandoning my applications in Java language. So I got a great advantage in terms of comprehensibility and convenience. First of all I want to show "Word Count" application which is famous as "Hello Word" in big data.

Bu hafta Python öğrenmeye başlayıp ordan Apache spark’a atılmıştım.Daha önceden java dilini bildiğim için onunla daha kolay olur sandım.Ancak Spark’ın desteklediği Python,Java ve Scala dillerini araştırdıktan sonra en kolay yöntemin Scala olduğunu anladım.O yüzden Java diliyle yaptığım uygulamalarımı bırakıp aynılarını Scala ile yapmaya çalıştım.Böylece anlaşılırlık ve kolaylık açısından büyük avantaj yakaladım.Öncelikle bid data’da “hello Word” kadar ünlü olan “Word count” uygulamasını göstermek istiyorum.

Word Count With Java :





First we create a JavaSparkContext object. We then upload a text file and create an RDD with each row being a String object. Each loaded line is separated from the gaps in the function we wrote in the flatMap method. The flatMap method also converts each element into a row. In order to get the number of words, we give these lines to the map method and convert them into a pair object in [word, 1] format. We then use the reduceByKey method to add the numbers corresponding to these words. Finally, we save the resultant text file.

 ilk olarak bir JavaSparkContext objesi oluşturuyoruz. Sonrasında bir metin dosyasını yükleyip her bir satır bir String objesi olacak şekilde bir RDD oluşturuyoruz. Yüklenen her bir satırı flatMap metodu içerisinde yazdığımız fonksiyonda boşluklarından ayrılıyor, flatMap metodu da her bir elemanı birer satır haline dönüştürüyor. Kelime sayısını alabilmek için bu satırları map metoduna vererek [kelime, 1] formatında bir pair objesi haline getiriyoruz. Sonrasında reduceByKey metodu ile bu kelimelere karşılık gelen sayıları topluyoruz. En sonunda da sonucu bir metin dosyasına kaydediyoruz.

**RDD Yapısı** RDD Structure

A resilient distributed dataset (RDD) is a component that allows us to compute data on a Spark cluster in general. At the same time, we can transfer other systems of data.

RDD( resilient distributed dataset) genel olarak Spark cluster üzerinde veriler üzerinde hesaplamalar yapmamızı sağlayan bir bileşendir.Aynı zamanda verileri diğer sistemleri aktarabiliriz.

With file from local computer

|  |  |
| --- | --- |
| 1 | JavaRDD<String> rdd = jsc.textFile("C:\\Users\\Ozgul\\Desktop\\test.txt"); |

Via hdfs

JavaRDD<String> rdd = jsc.textFile("hdfs://user/file");

Compressed files

JavaRDD<String> rdd = jsc.textFile("C:\\Users\\Serkan\\Desktop\\file\\\*.gz");

RDD Operations

RDD operasyonları genel olarak 2 bölüme ayrılır.

RDD operations are generally divided into 2 divisions.

***transformasyon(transformations)***

In this process, a new RDD is created via the existing RDD. If we give an example, we can create new RDD from map and filter methods

Bu işlemde mevcut RDD üzerinden yeni bir RDD oluşturulur.Örnek verirsek **map**ve **filter**metodlarından yeni RDD oluşturabiliriz 

***aksiyon(actions)***

In this process, calculations via RDD, saving data to external systems are performed. Count, first methods are examples

Bu işlemde RDD üzerinden hesaplama,dış sistemlere verileri kaydetme işlemleri yapılır. Örnek olarak **count,first**metodları örnek verilebilir

As you can see, our work with java is getting harder. Now you can see the writing with Scala.

Görüldüğü üzere java ile işimiz baya bir zorlaşıyor.Şimdi bir de Scala ile yazılışını görelim.

WordCount with Scala:

**val** textFile **=** sc.textFile("hdfs://...")

**val** counts **=** textFile.flatMap(line **=>** line.split(" "))

.map(word **=>** (word, 1))

.reduceByKey(**\_** + **\_**)

counts.saveAsTextFile("hdfs://...")

Our code lines are very long with Java and we have descended into 3-4 lines of code with the scala language. I do not know which language is better for deeper projects, but for Shan, Scala seems to be the best.

Java ile çok uzun olan kod satırlarımız scala dili ile 3-4 satırlık koda inmiş bulunmakta.Daha derin projeler için hangi dil daha iyi olur bilmiyorum ama şuan için Scala en iyisi gibi duruyor.