VIA 504E – Big Data Technologies and Applications

Homework 2

528211011 – Hakan Can İpek

Q1: **Please discuss MapReduce logic in terms of one of the example datasets in detail? Try to discuss your own dataset schema/structure.**

A1: Büyük verileri işlemeye çalışırken yapmamız gereken şeylerden bazıları ölçeklenebilir(scalable) ve efektif(efficient) olması gerektiğidir. Bu durumları sağlamada Map-Reduce Algoritmasının oldukça etkili olduğunu görmekteyiz. Bunu bir örnek üzerinde Map-Reduce uygulayarak görelim.

Diyelim ki büyük miktarlarda veri işliyoruz ve kullanıcılarımızın yüzde kaçlık bir oranının oyunlar hakkında tartıştığını bulmaya çalışıyoruz. Öncelikle veri içerisinde eşlememiz(mapping) gereken anahtar kelimeleri(keyword) tanımlamamız gerekiyor. Bunu yapmamızın sebebi bu kelimeler ile oyunlarla ilişkisi olan bir sonuca ulaşabilme çabasıdır.

Sonrasında veri içerisindeki bu eşleşmeleri yakalayacak fonksiyonları yazmaya başlarız. Örneğin Football, Volleyball, Soccer, Cup, Medal vb.

Bu fonksiyon içerisinde verdiğimiz anahtar kelimelerin frekanslarını yakalamamız gerekir. Örneğin bir döngü içerisinde her cümleyi tek tek incelediğimizi düşünelim. Bu cümle elimizdeki bir anahtar kelimeyi içeriyor mu diye bakmamız gerekir. İçeriyor ise ilgili anahtar kelime miktarını(count) 1 arttırırız. Elimizdeki tüm veri bittikten sonra elimizde bir anahtar-değer objesi ve içerisinde anahtar olarak ilgili “keyword” isimleri, değer olarakta miktarları bulunacaktır. Bu kısım genel olarak algoritmanın “Map” kısmını oluşturuyor.

Peki "Reduce” kısmı nasıl?

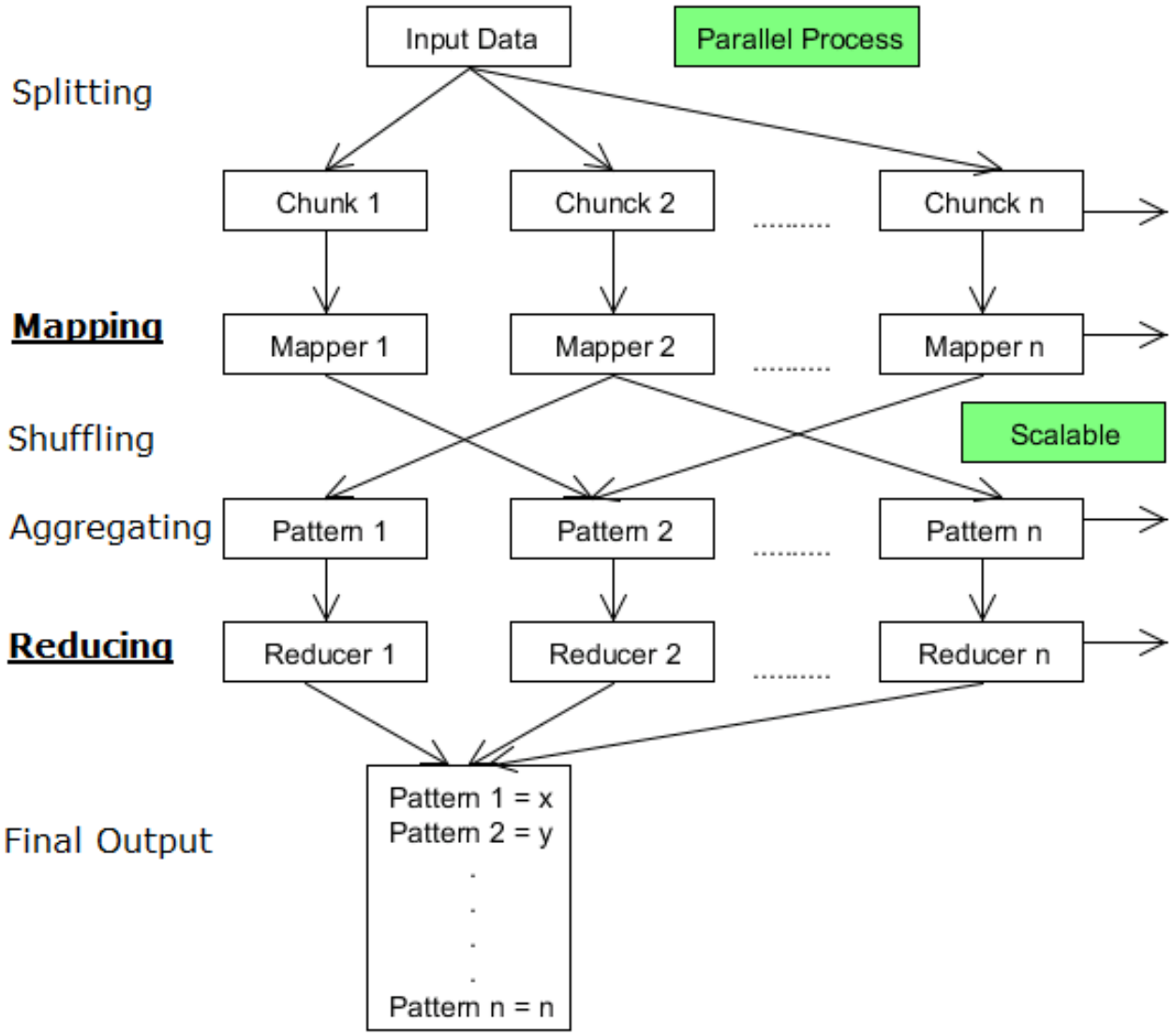
Reduce kısmında üstte bahsettiğimiz fonksiyon çıktısından gelen anahtar değer çiftlerini girdi olarak alıp işlemeye başlarız. Her çift için gelen değeri sayaç değerine ekleriz. En son olarak oyunlarla ilgili toplam sayaç değerini döneriz.

Olaya daha büyük bir pencereden bakalım. Burada n adet eşleme fonksiyonu yazabiliriz. Mesela bu seferde kimlerin birbirine iyi dileklerde bulunduğunu öğrenmek istiyoruz, doğum günü olabilir, yeni yıl olabilir, herhangi bir kutlama günü dileği olabilir. “Wish”, “Merry”, “Happy” gibi kelimelere(veya türkçe bir veri seti için Mutlu, Nice, İnşallah, Umarım) bakan bir eşleme ve bu eşlemenin reduce fonksiyonunu yazmamız gerekir.

Bu noktada artık bir karıştırıcı(Shuffling) fonksiyona ihtiyacımız olacak. Bu fonksiyon, oyunlar ve iyi dileklerle alakalı anahtarları ayrıştıracak ve ilgili reduce fonksiyonuna gönderecek.

Ayrıca Eşleme fonksiyonları öncesinde büyük veriyi daha küçük parçalara ayırarak Eşleme fonksiyonlarına daha küçük porsiyonlar halinde gönderecek bir Ayrıştırıcı fonksiyona ihtiyacımız olacak.

Bu sürecin akışı şu şekildedir.



Yukarıdaki akışa bakarak,

- Girdi gelen veri miktarı ve işleme kapasitesine bakılarak n adet porsiyonlara bölünmüştür.
- Sonrasında Eşleme fonksiyonlarına dağıtılmışlar. Unutulmaması gereken bir durumda tüm porsiyonların eşleme fonksiyonlarından aynı anda paralel işlemden geçtiğidir.
- Devamında karıştırma yapılır, bu da benzer kalıpların bir araya gelmesini sağlar.
- Son olarak Reduce fonksiyonları her mantığa uygun bir şekilde hepsini birleştirir.

- Bu algoritma girdi verisinin miktarına bağlı olarak bize ölçeklenebilirlik sağlamaktadır, çünkü paralel işleme ünitelerini arttırabiliriz.

Q2: **Please find research case studies on Big data tools/platforms(i.e Apache Hadoop, Apache Hive, Apache Spark (v2 vs v3), Apache Kafka, Apache Nifi, Apache Airflow, Logstash-Elastics Search-Kibana etc.) and find some topics for discussions (for your taste – according to design for your proposal).**

A2:

- Credit Card Fraud Detection with ***Apache Flink***

  As millions of people are using a credit card nowadays, so it has become very necessary to protect people from frauds. It has become a challenge for Credit card companies to identify whether the requested transaction is fraudulent or not.

  A credit card transaction hardly takes 2-4 seconds to completion. So the companies need an innovative solution to identify the transactions which may appear as fraud in this small time and thus protect their customers from becoming its victim.

  An abnormal number of clicks from the same IP address or a pattern in the access times — although this is the most obvious and easily identified form of click fraud, it is amazing how many fraudsters still use this method, particularly for quick attacks.

  They may choose a to strike over a long weekend when they figure you may not be watching your log files carefully, clicking on your ad repeatedly so that when you return to work on Tuesday, your account is significantly depleted. Part of this fraud might be unintentional when a user tries to reload a page.

  Again if you have made any transaction from Mumbai today and the very next minute there is a transaction from your card in Singapore. Then there are chances that this transaction may be fraud and not done by you.

So companies need to process the data in real time (Data in Motion analytics DIM) and analyze it against individual history in a very short span of time and identify whether the transaction is actually fraud or not. Accordingly, companies can accept or decline the transaction based on the severity.

To process the data streams we need streaming engines like ***Apache Flink***. The streaming engine can consume the real-time data streams at very high efficiency and process the data in low latency (without any delay).

- Sentiment Analysis with ***Apache Hadoop***

  Sentiment analysis provides substance behind social data. A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level — whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral.

  Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as "angry," "sad," and "happy."

  In sentiment analysis, language is processed to identify and understand consumer feelings and attitudes towards brands or topics in online conversations ie, what they are thinking about a particular product or service, whether they are happy or not with it, etc.

  For example, if a company is launching a new product, it can find what its customers are thinking about the product.

  Whether they are satisfied with it or not or they would like to have some modifications in it can be found out using Big data by doing sentiment analysis i.e, using sentiment analysis we can identify users' opinion about the same.

  Then the company can take action accordingly to modify or improve the product to increase their sales and to make customers fell happy with their product.

  Below is a real example of sentiment analysis:
  A large airline company started monitoring tweets about their flights to see how customers are feeling about upgrades, new planes, entertainment, etc. Nothing special there, except when they began feeding this information to their customer support platform and solving them in real-time.

One memorable instance occurred when a customer tweeted negatively about lost luggage before boarding his connecting flight. They collect the tweets (having issues) and offer him a free first class upgrade on the way back.

They also tracked the luggage and gave information on where the luggage was, and where they would deliver it. Needless to say, he was pretty shocked about it and tweeted like a happy camper throughout the rest of his trip.

With ***Hadoop***, you can mine Twitter, Facebook and other social media conversations for sentiment data about you and your competition, and use it to make targeted, real-time, decisions that increase market share.

With the help of quick analysis of customer sentiment through social media, company can immediately take decision and action and they need not wait for the sales report (which might take 6 or more months also)as earlier to run their business in a better manner.

- Data Processing(Retail) with ***Apache Hadoop***

  Let us now see an application for Leading Retail Client in India. The client was getting invoice data daily which was of about 100 GB size and was in XML format.

  To generate a report from the data, conventional method was taking about 10 hours time and client had to wait for this time to get the report from the data.

  This conventional method was developed in C and was taking a huge time which was not a feasible solution and the client was not happy with it.

  The invoice data was in XML format which needs to be transformed into a structured format before generating the report. This involved validation, verification of data and implementation of complex business rules.

  In today's world when things are expected to be available anytime when required, waiting for 10 hours was not a proper and acceptable solution.

  So the client approached Big data team of one of the companies with their problem and with a hope to get a better solution. The client was even able to accept time reduced from 10 hours to 5 hours or lil more also.

When Big Data team started working on their problem and approached them back with the solution, the client was amazed and could not believe that the report which they were getting in 10 hours could now be received in just 10 minutes using Big Data and ***Hadoop***.

The team used a cluster of 10 nodes for the data getting generated and now the time taken to process data was just 10 minutes. So you can imagine the speed and efficiency of Big Data in today's world.

- Orbitz.com with ***Apache Hadoop***

  Orbitz is a leading travel company using latest technologies to transform the way clients around the world plan the travel. They operate the customer travel planning sites Orbitz, Ebookers and CheapTickets.

  It generates 1.5mn flight searches and 1mn hotel searches daily and the log data being generated by this activity is approximately 500GB in size. The raw logs are only stored for a few days because of costly data warehousing.

  To handle such huge data and to store it using conventional data warehouse storage and analysis infrastructure was becoming more expensive and time consuming with time.

  For example to search hotel in database using conventional approach which was developed in Per/ Bash, extraction need to be done serially.

  The time it was taking to process and sort hotels based on just last 3 months data was also 2 hours which was again not acceptable and feasible solution today when customers are expecting results to be generated on just their click.

  This problem was again very big and needed some solution to protect the company from losing their customers. Orbitz needed an effective way to store and process this data, plus they needed to improve their hotel rankings. It was then tried using Big Data and ***Hadoop*** approach.

  Here **HDFS, Map Reduce and Hive** were used to solve the problem and just amazing results were received. A ***Hadoop*** cluster provided a very cost effective way to store vast amounts of raw logs. Data is cleaned and analyzed and machine learning algorithms are run.

  Earlier when it was taking time of about 2 hours to generate search result on hotel data of last 3 months, the time was reduced to just 26 minutes to generate the same result with Big Data.

Big data was able to predict hotel and flight search trends much faster, more efficiently and cheaper than the conventional approach.

- Sears Holding with ***Apache Hadoop***

  Now we are going to see how Sears Holding used ***Hadoop*** to personalize marketing campaigns. Sears is an American multinational department store chain with more than 4,000 stores with millions of products and 100mn customers.

  As of 2012, it is the fourth-largest U.S. department store company by retail sales and is the 12th-largest retailer in the United States, leading its competitor Macy's in 2013 in terms of revenue.

  With so many stores and customers, Sears has collected over 2PB of data so far. Now the problem came when legacy systems became incapable of analyzing large amounts of data to personalize marketing & loyalty campaigns.

  They wanted to personalize marketing campaigns, coupons, and offers down to the individual customer, but our legacy systems were incapable of supporting that which was leading to decline of their revenues.

  Improving customer loyalty, and with it sales and profitability, was desperately important to Sears due to large competition.

  Sears' conventional process for analyzing marketing campaigns for loyalty club members used to take six weeks on mainframe, Teradata, and SAS servers to analyze just 10% of customer data.

  Here came the cutting-edge implementation of ***Apache Hadoop***, the high-scale, open source data processing platform driving the big data trend.

  With the new approach of Big data, Sears shifted to ***Hadoop*** with 300 nodes of commodity servers. The new process running on ***Hadoop*** can be completed weekly for 100% analysis of customer data.

  Whereas the old models made use of 10% of available data, the new models run on 100%. For certain online and mobile commerce scenarios, Sears can now perform daily analyses. Interactive reports can be developed in 3 days instead of 6 to 12 weeks with this method.

  This move saved millions of dollars in mainframe and RDBMS cost and got 50 times better performance for Sears. They were even able to

increase revenues through better analysis of customer data in a timely and quickly manner.

- Market Basket Analysis with ***Apache Spark***

  In retail, inventory, pricing and transaction data are spread across multiple sources.

  Business users need to collect together this information to understand products, come up with reasonable pricing, find platforms to support so that their online users would have efficient performance, and where to target ads.

  Market basket analysis may provide the retailer with information to understand the purchase behaviour of a buyer what he is looking for and what other things he may be interested in buying along with this product.

  An obvious application of Market Basket Analysis is in the retail sector where retailers have large amounts of transaction data and often thousands of products.

  One of the recognizable examples is the Amazon, their recommendation system is one of the best: “people who bought a particular item also bought items X, Y, and Z”. Market Basket Analysis is applicable in many other industries and use cases

  For example, retail company leader in the Fashion industry analyzed sales data from the last three years. There were more than 100 million receipts but the results obtained can be used as an indicator for defining new promotional initiatives, Identifying optimal schemes for the layout of goods in stores etc.

  This information will enable the retailer to understand the buyer’s needs and rewrite the store’s layout accordingly, develop cross-promotional programs, or even capture new buyers. By analyzing the uses’ buying pattern they can identify what are items they bought together.

  To make store customer-friendly these items can be put together and relevant campaigns can be run to attract new buyers.

  Market Basket Analysis algorithm can be customized on users’ needs. To increase the sales, supermarkets are trying to make store more customer friendly. Now business users can deeply investigate on the effectiveness of marketing and campaigns.

Marketing and sales organizations across all industries are looking to analyze, understand and predict purchase behaviour towards achieving goals to reduce customer churn and maximize customer lifetime value (CLV).

Selling additional products and services to existing customers over their lifetime is key to optimizing revenues and profitability.

Market Basket Analysis association rules identify the products and services that customers typically purchase together, empowering organizations to offer and promote the right products to the right customers.

To implement this complex use-case ***Apache Spark*** is best solution which provides generalized framework to handle diverse use-cases. Market basket analysis needs to be developed using machine learning algorithms.

Apache Spark provides MLlib which is a rich machine learning library. ***Spark*** run iterative algorithm (Machine Learning executions are iterative in the nature) very efficiently.

- Customer Churn Analysis with ***Apache Flink***

Churn analysis is the calculation of the rate of attrition in the customer base of any company. It involves identifying those consumers who are most likely to discontinue using your service or product.

Losing the customer is not liked by any industry. In the current market all the customer facing industries are facing the issue of customer churn due to huge competition in the market. Industries like Retail, Telecom, Banks, etc. are facing this issue severely.

The best way to manage these issues will be to predict the subscribers who are likely to churn, well in advance so that business can take required measures to mitigate it and win-back the customers or re-activate the sleeping base.

The industries are interested in finding root cause of customer churn; they want to know why customer is leaving them and which the biggest factor is. To find out the root cause of Customer churn companies need to analyze following data. This data might be in the range of TBs to PBs

Companies need to go through billions of customer complains which are stored for years and get them resolved with immediate effect.

Data from social media, where users write their opinion about the product they are using from which companies can identify if customers are liking their products or not.

Let us take an example of Call Centre Analysis. Here the data used is Call Log and Transactional data. Many banks are integrating this call centre data with their transactional data warehouse to reduce churn, and increase sells, customer monitoring alerts and fraud detection.

Apache Flink (4G of Big Data) offers an opportunity to tap into the many internal and external customer interaction and behavioural data points to detect, measure and improve the desired but illusive objective of consistent and rewarding Customer Experience success.

Q3: **Install Hadoop (anyversion) as a pseudocluster mode and Show Word counting example with your own document file.**

A3: Bu kısmı mevcut instance kullanarak yaptım.

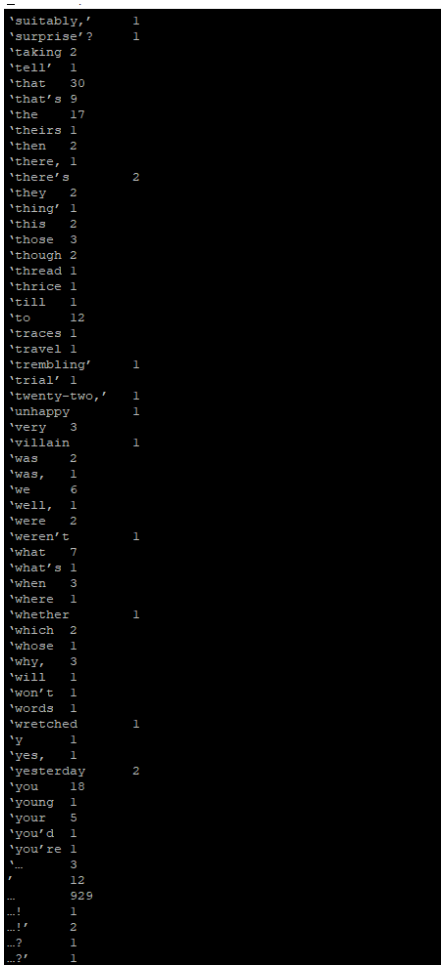
Öncelikle VIA504E altında kendi klasörümü oluşturup buraya map ve reduce fonksiyonlarımı oluşturdum. Ardından Putty ile bağlanarak ilgili klasor altına gittim. Burada aşağıdaki komutu çalıştırdım.

```
cat hakan.txt | python mapper.py | sort -k1,1 | python reducer.py
```

Devamında tüm kelimeler ve miktarlarını içeren bir çıktı aldım. Bazı ekran görüntüleri aşağıdadır.

ubuntu@ip-172-31-44-192: ~/VIA504E/h

```
‘new    1
‘no     2
‘no,    1
‘nobler’—       1
‘not    4
‘nothing        1
‘nothing,       1
‘now    1
‘now,’  1
‘of     4
‘oh     1
‘oh,    1
‘old    1
‘on     2
‘one    2
‘or     1
‘ordinary’      1
‘ose    1
‘ough   3
‘our    3
‘paper  1
‘percentage’    1
‘perhaps        1
‘pierced        1
‘please 1
‘politely       1
‘poof!  1
‘poor   1
‘poppet,’       1
‘poverty        1
‘problem’       1
‘prophet’       2
‘protect        3
‘protests’      1
‘question’      1
‘rash   1
‘read   1
‘reassure’      1
‘ree    1
‘rehearsal’     1
‘remember       1
‘remember,’     1
‘renews,’       1
‘rough  1
‘run    1
‘running        1
‘save   1
‘say    1
‘schwach’       1
‘seems’ 1
‘serious        1
‘serviceable    1
‘set    1
‘she    9
‘show   1
‘six    1
‘so     1
‘some   2
‘spontaneous’   1
‘stern  1
‘stop   1
```



Q4: **Install Spark (v2) as a pseudocluster mode and Show Word counting example with your own document file (same as Hadoop).**

A4: Bu kısmı mevcut instance kullanarak yaptım.

Aşağıdaki komutla Jupiter Notebook açtım.

jupyter notebook --no-browser --port=8945

Notebook üzerinden aşağıdaki kodu çalıştırdım.

from pyspark import SparkConf, SparkContext

```
conf = SparkConf().setMaster("local").setAppName("WordCount")
sc = SparkContext(conf = conf)

input = sc.textFile("hakan.txt")
words = input.flatMap(lambda x: x.split())
wordCounts = words.countByValue()

for word, count in wordCounts.items():
    cleanWord = word.encode('ascii', 'ignore')
    if (cleanWord):
        print(cleanWord.decode() + " " + str(count))
```

Aldığım sonuç şu şekildedir.(uzun bir çıktısı var, başlangıcını koydum)