BİL401

Final Report

Kerem Elma - 211101079

Mehmet Eski - 221104095

1. **Introduction**

This report compares of Avro, Parquet, XML, JSON data formats under the concept of big data. In the comparisons, different metrics were tried and it was tried to determine which one is more optimal to use. With the help of various technologies such as PySpark, Hadoop, Apache Hive; the scope has been expanded and metrics have been diversified.

1. **Codebase and Materials**

The first version of the data was in JSON format. with a file converter python script we wrote ourselves, we created the same data in other formats and added it to our datasets. In these Python scripts, libraries such as pandas, pyarrow, json was used which help in conversions.

In the same way, with Spark's PythonAPI, called PySpark, we wrote reader scripts to read these files, these scripts were responsible for reading the dataset on the local machine.

We used Hadoop’s HDFS for the storage part and again observed the processing of this big data for various metrics.

We also used Apache Hive to analyze this data using SQL-like queries.

The complete codebsae is available at the following github link. In the main path, the **file\_converter.py** file converts the original json dataset into different formats. In the **datasets/** path, there are the files created by the converter. In the path named **readers/**, there are pyspark scripts that read the files in these formats.

* <https://github.com/kelma01/dataComparer>

1. **Platform Details**

For the implementation, we used Python language and pyspark provided by spark. We chose Python because it is easy to write, flexible and Spark provides such a possibility.

For storage, we chose Hadoop’s HDFS because it is quite common and convenient to use for storage and management in big data processing environments. Its distributed processing is suitable for parallel processing. It is also important that it is compatible with the data formats we will be using. Since we are using Hadoop, we used a cluster infrastructure. Hadoop's big data processing power comes from its ability to perform parallel processing on multiple machines (nodes).

For data management, we chose Apache Hive, a SQL-like query language used for big data analytics on Hadoop. It stores data on HDFS and performs fast analysis with distributed processing power. Hive is a tool built on top of the Hadoop cluster. Therefore, because of Hive, we used the distributed computing infrastructure in the Hadoop cluster in the background.

1. **Results**

The tests were performed on a local machine without any storage or database, and metrics such as memory usage, read speed, and total size were used: If we examine them in order:

* **Read Rate and Memory Usage:**

The following outputs show how many seconds it took to read the file and the memory usage during that time.

BURADAKİ DEĞERLER DEĞİŞECEKTİR.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir. metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir. metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir. metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

When we look at the outputs, it is seen that the performance of Avro and Parquet is much superior to the JSON format. This is because Avro and Parquet are binary formatted and schema based. At the same time, reading times are much shorter thanks to their compressed format. Therefore, when looking at the read speed metric, using Avro and Parquet can be seen as an optimal solution. At the same time, the XML formatted file is also read in a very short time, but in scenarios where the data grows even larger, it will be more difficult to read a text-based file, so again Avro and Parquet will be more optimal.

Parquet, dördünün arasında bellek kullanımı en az olan formattır. Diskten okuma süresi optimize bir biçimde çalışmaktadır çünkü schema bazlı ve ilgili kolonları okumasından dolayıdır. Düşük bellek tüketimi ile büyük veri setleri ile çalışırken avantajlı olabilmektedir. Diğerlerine baktığımızda ise Avro da JSON ve XML kadar çok bellek kullanmıştır ama yine de okuma süresi diğerlerine göre düşüktür çünkü yine Parquet gibi schema bazlı olup sıkıştırılabilir oluşu bu alanda onu öne çıkarmaktadır.

* **File Size:**

After the files are created by the file converter, they keep the same data content in just different formats. After this conversion, the sizes of the files are as follows:

BURADAKİ DEĞERLER DEĞİŞECEKTİR.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

The numbers written next to the files are the space the files hold in memory in KB. If we interpret this output, the format that can be stored in the smallest size is Parquet, while the format that can be stored in the largest size is XML.

If we interpret this output, the format that can be stored in the smallest size is Parquet, while the format that can be stored in the largest size is XML.

The reason why Parquet and Avro are the best formats here is because of their compressibility feature, which we mentioned before. This provides advantages in storage and reading performance.

JSON and XML, on the other hand, can take up extra space because they are text-based and because of the key-value structure in JSON and the tag structure in XML.

As mentioned, we used Hadoop's HDFS for storage parts. With HDFS; processing, storing and managing big data has become much easier. In the tests we conducted using HDFS, the metrics were generally writing, reading and the size we used.

* **Read Rate and Memory Usage:**

The following outputs show how many seconds it took to read the file and the memory usage during that time.

<resim verileri eklenecek>