Design and Implementation of a Data Lake

By

Ruoran Liu

10191244

Project Type: Thesis-Research

Course Code: CISC 499

Supervisor(s): Dr. Farhana Zulkernine and Dr. Haruna Isah

Date: 2019/04

Abstract

A data lake is a large storage repository that holds a vast amount of raw data in its native format until it is needed. Large organizations are seeking to create data lakes because they manage and use data with increased volume, variety, and a velocity rarely seen in the past.

The main benefit of a data lake is the centralization of data from disparate sources. A data lake also: Improves the capture, refinement, archival, and exploration of raw data within an enterprise. Allow users to query and explore data in different formats. In this paper, we compare the differences between traditional data warehouse and data lake, explained how a data lake can be constructed using Hadoop Data Platform and given a proposed solution to make connection of structured data and unstructured data.

Table of Contents

Chapter 1
Introduction

1.1. Motivation

Queen's University is using PeopleSoft(PS) information systems in order to manage student records. The short-comings of this system is that, for now, users generate data by PS query directly or through component interfaces. Also, the finance department need to extract data though data warehouses which is not clearly organized and is also limited by the output format. PeopleSoft is the only way to store data, which is not efficient when numerous queries must be executed against the storage each day to access and process the same data. In order to meet the advanced business requirements, the data platform need revamping to provide the following functionalities:

- Scalable and rapid ingestion of new data from multiple sources.
- Transform and integrate enterprise data, regardless of its source, structured or unstructured, inside or outside the enterprise.
- Define necessary metadata to describe data in the platform such that queries can be generated on the data based on customer requests.

Build a data lake or staging platform where cleaned and transformed data would reside that can serve as an integrated data source to answer customer queries.

1.2. Problem Description

Traditional data warehousing solutions pre-date the cloud. They were designed to run on small, static clusters of well-behaved machines, making them a poor architectural fit.[1] Most of them are still using the traditional pre-defined reporting models, based on the operational database.

Nowadays, data lake is becoming the answer for data handling. We are using Apache Hadoop to address the limitations of traditional computing, helps businesses overcome real challenges, and powers new types of Big Data analytics.

The Apache Hadoop allows distributed parallel processing of very large data sets across clusters of commodity machines (low-cost hardware computers) using simple programming models.[2] The solution framework is horizontally scalable with addition of worker nodes to the cluster. Hadoop was designed to process any kind of data from structured, semi-structured to unstructured.[3] Hadoop does not enforce on having a schema or a structure to the data that has to be stored. [4]

The administration requirements of University are mostly the same as the business requirements of any other industry. The information explosion will give rise to new issues and challenges in database and data mining area. The types of data vary a lot. No well-defined matrices exist and the potential data sources are many. In such a scenario, the performance of traditional data manipulation techniques will be restricted by the hardware. Different types of data, especially the unstructured data, need a special approach to be stored and analyzed. Data Lake is given a way to hold a vast amount of raw data in its native format until it is needed.[5]

1.3. Key contributions

This research is given a proposed architecture of a data lake that includes the functions of data importation, data storage, data analysis and data visualization. It provides a possible solution to transport data from existing database (relational database) to Apache Hadoop HDFS which addresses the limitations of traditional computing. The components being used for this implementation includes HDFS, Sqoop, Kafka, Nifi, Hive, Spark and Zeppelin.

1.4. Organization

The following paragraphs will make a comparison between multiple distribution storage platforms based on Apache Hadoop. And will provide a proposed architecture of Hadoop Data Lake that addresses the existing problems.

Chapter 2

Background

2.1 Background

Enterprises are using various big data technologies to process data and drive actionable insights. The enterprise data lake is a Big Data solution for storing vast amounts of data and analyzing disparate sources of data in their native formats. The goal is to break the information silos in the enterprise by bringing all the data into a single place for analysis without the restrictions of schema, security, or authorization. [6] More technically, the data lake is a set of tools for accessing, transforming, storing, recovering and analyzing the relevant data.

Hadoop Distributed File System (HDFS) has become the core storage system for enterprise data, including enterprise application data, social media data, log data, click stream data, and other Internet data. [7] Hortonworks Data Platform is one of the choices to run with. HDP addresses the complete needs of data-at-rest, powers and real-time customer applications and delivers robust big data analytics that help to make decision.

At the same time, enterprise data warehouses (EDWs) continue to support critical business analytics. EDWs are usually shared-nothing parallel databases that support complex SQL processing,[7] "Has Hadoop really replaced Data Warehouses in the enterprise?" is still the question needed to be figured out.

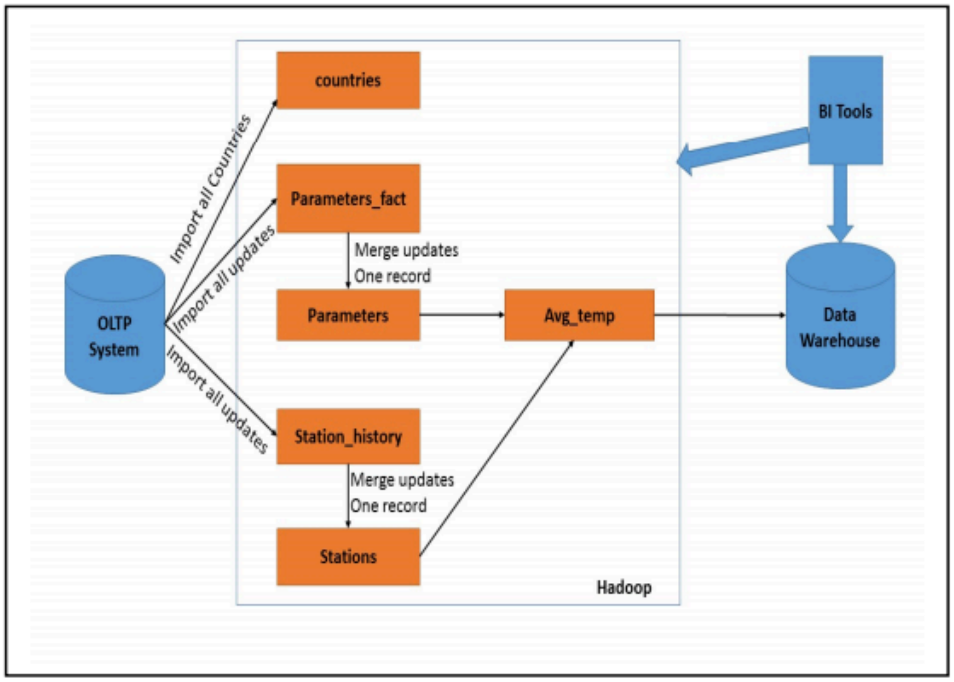
2.2 Literature Study

Hadoop vs Data Warehouse

"We don't see anybody today trying to build an IDW with Hadoop. This is a capability issue, not a cost issue. Hadoop is not an IDW. Hadoop is not a database." - said Bob Page, the VP of development at Hortonworks. The difference between Hadoop and Data warehouse is like a hammer and nail. []Data warehouse is an architecture for organizing data to ensure integrity, whereas Hadoop is a big data technology for storing and managing big data. It is not a good move to use Hadoop to process 20 GB of structured data as that can easily be tackled with a traditional data warehouse. In fact, many businesses do not have the required Hadoop skilled personnel and the resources to run and deploy a Hadoop cluster for simple data queries. The co-existence of the two technologies depends on the business problem.[8]

Hybrid Data Warehouse

Data warehouse is a Decision Support System (DSS) technology that allows extracting, grouping and analyzing historical data from different sources in order to discover information relevant to decision making. This paper is given a solution to build hybrid data warehouse by combining the traditional Data Warehouse approach with big data technology (Hadoop). [9]



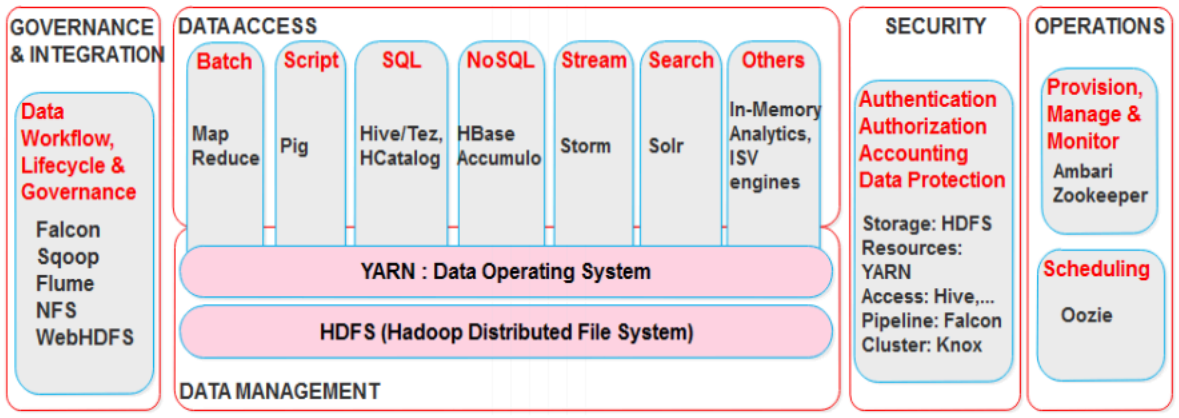
High Level of The Proposed Hadoop Data Warehouse Modol

One of the OLTP schema advantages is to allow the updates/deletes/inserts values of the records. On the other hand, Hadoop allows only deleting old records and write new records in the place. So, Hadoop does not allow updates for values of the records. Simple solution in Hadoop is to create tables in order to append only instead of overwriting old records in the place.

Comparison of Different Hadoop Distributions[10]

Hortonworks distribution (HDP)

The defined objectives of this distribution are to simplify the adoption of the Apache Hadoop platform. All the components of this distribution are open source and licensed from Apache.



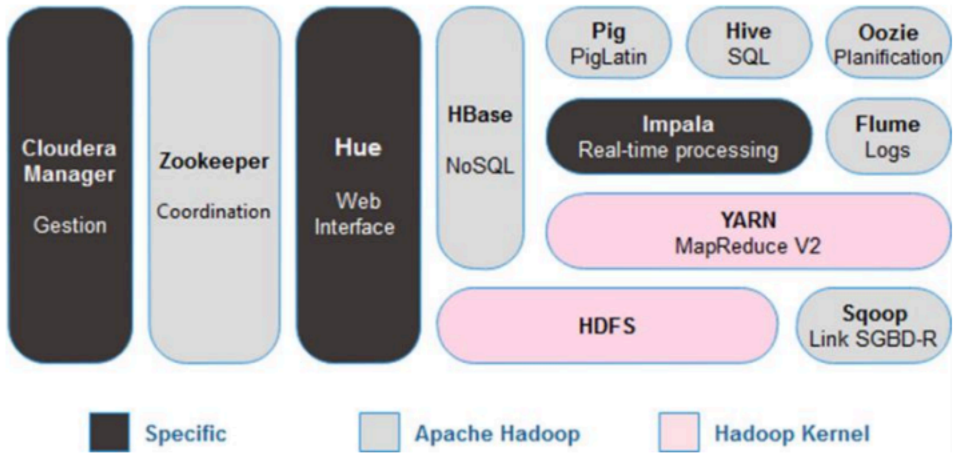
Hortonworks Hadoop Platform (HDP)

Cloudera distribution (CDH)

This distribution includes the components of Apache Hadoop and it succeeds to develop effectively house components for cluster management.

The components rather than Apache Hadoop:

- Impala Cloudera: "is Cloudera's open source massively parallel processing (MPP) SQL query engine for data stored in a computer cluster running Apache Hadoop" [26].
- Cloudera Manager: Deployment and management of Hadoop components.



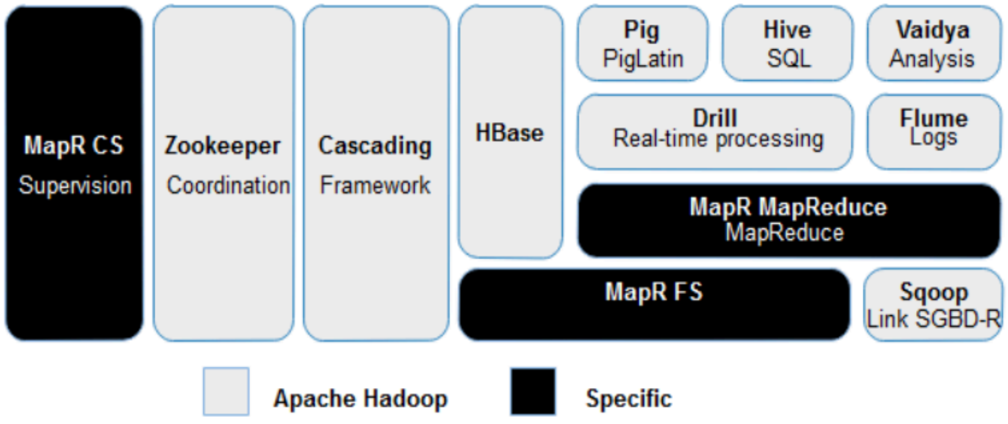
Cloudera Distribution of Hadoop Platform (CDH)

MapR distribution

They successfully propose their own version of MapReduce and distributed file system: MapR FS and MapR MR.

The components rather than Apache Hadoop:

- MapR FS: MapR proposes its own file system by replacing the HDFS.
- MapR Control System (MCS): MCS allows management and supervision of the Hadoop cluster. It is a web-based tool for managing cluster resources (CPU, Ram, Disk, etc.) as well as services and jobs.
- Apache Cascading: Java framework dedicated to Hadoop. It allows a Java developer to find his brands (JUnit, Spring, etc.) and to manipulate the concepts of Hadoop with a high-level language without knowing the API.
- Apache Vaidya: Hadoop Vaidya is a performance analysis tool for MapReduce jobs.
- Apache Drill: Drill completes MapReduce and is an API for faster query creation based on the SQL model.



MapR (M3)

Chapter 3
Comparative Analysis

3.1 List of Criteria [10, 11]

To compare MapR, Cloudera and Hortonworks platforms, we shall use the following criteria:

Disaster Recovery: It can prevent data loss in the event of a computer center failure. It has the ability to rebuild the infrastructure and to restart applications that support the activity a company. Therefore, Disaster Recovery must be able to take care of the computer needs necessary for the survival of the organization in case of a Big Data system disaster.

Replication: The different Big Data Hadoop distributions use a process of information sharing to improve reliability, fault tolerance and availability. They also aim to ensure data consistency across multiple redundant data sources. Data replication is called if the data is duplicated on multiple storage locations.

Management tools: These are management consoles used by different Hadoop solution providers to manage a Hadoop distribution. Thanks to these tools, you can effortlessly deploy, configure, automate report, track, troubleshoot, and maintain a Big Data system.

Data and Job placement control: It allows controlling the placement of data and jobs on a Hadoop cluster and, hence, permits to choose nodes to execute jobs presented by different users and groups.

DFS: Distributed file system for storage.

Data Ingestion: Data Ingestion is the process of importing and obtaining data for immediate use or storage in a database or HDFS. Thus, Data can be broadcast in real time or ingested in batches. When it is ingested in real time, it is imported as it is transmitted by the source. However, when data is ingested in batches, it is imported in discrete blocks at periodic time intervals.

Metadata Architecture: here we shall talk about two types of architectures used by the Hadoop distributions at Metadata level. The first one is a centralized architecture where everyone depends on the same authority. The second one is a decentralized architecture that has no center, no more and no less. This means that every entity can be a part of a network that has no main authority and

Apache Hadoop YARN (Yet Another Resource Negotiator) is a technology for managing clusters and making Hadoop more suitable for operational applications that cannot wait for the completion of batch processing. YARN is among the key features of Hadoop 2, the second generation of the distributed processing infrastructure of Apache Software Foundation.

Bulk Data Transfer between RDB and Hadoop: These are tools designed to transfer efficiently data between Apache Hadoop and structured data stores such as relational databases.

Data warehousing: means a database used to collect, order, log, and store information from operational databases. It also provides a basis for business decision support.

Extract, Transform and Load (ETL): It is a computer technology known as ETL. It allows massive synchronization of information from one data source to another.

Comparison Tables
Distributed Platform

Criteria \ Platform		Horton works	MapR	Cloudera
Disaster Recovery		-	+	+
Replication Data		+	+	+
Replication Meta Data		-	-	+
Management Tools		+	+	+
Data and Job Placement Control		-	-	+
DFS		+	+	+
Data Ingestion	Batch	+	+	+
	Streaming	-	-	+
Meta Data Architecture	Centralized	+	+	-
	Distributed	-	-	+
Non-Map Reduce Tasks(YARN)		+	+	+
Data Transfer Between RDB and Hadoop		+	+	+
Data Warehouse		+	+	+
Extract, Transform and Load (ETL)		+	+	+

Data Ingestion Tool [12, 13]

	Feature	Limitation	Use Case
Flume	- Configuration-based - Sources, channels & sinks - Interceptors	- Data loss scenarios when not using Kafka Channel - Data size - No data replication	- Collect, aggregating, and moving high-volume streaming event into Hadoop
Kafka	- Back-pressure - Reliable stream data storage - Kafka-Streams - Sources/sink with Kafka-Connect	- Custom coding often needed - Data size - Fixed protocol/ format/schema	- Streaming data - Messaging - Systems integration - Commit log
Nifi	- Configuration-Based UI - Many drag & drop processors - Back-pressure - Prioritized queuing - Data provenance - Flow Templates	- Not for CEP or windowed computations - No data replication	- Dataflow Management with visual control - Data routing between disparate systems - Arbitrary data size

Sqoop	- Full load - Incremental Load - Parallel import/export - Import results of SQL query - Connectors for all major RDBMS Databases - Load data directly into HIVE/HBase - Support for Accumulo	- Cannot be paused and resumed. It is an atomic step. If it is failed we need to clear things up and start again. - Export performance also depends upon the hardware - Slow because of MapReduce in backend processing. - JDBC connection to connection can be inefficient and less performance	- moving data in/out between Hadoop and RDBMS

Data Storage[4, 15]

	Feature	Use Case
Hive	Hive is not ideally a database but a mapreduce based SQL engine that runs on top of Hadoop	RDBMS professionals love apache hive as they can simply map HDFS files to Hive tables and query the data.
Hbase	HBase is a NoSQL database used for real-time data streaming. With flexible data models, cost effectiveness and no sharding (automatic sharding), HBase works well with sparse data.	For real-time querying of data. HBase is an ideal big data solution if the application requires random read or random write operations or both. If the application requires to access some data in real-time then it can be stored in a NoSQL database.

Query Engine [16, 17]

	Features	Use cases
Drill	Schema-free SQL Query Engine for Hadoop, NoSQL and Cloud Storage	- Document Store - Relational DBMS - Main objective is to only query data from disparate sources.

Spark SQL	General computation engine that happens to have SQL query capabilities. Allow working with data from stream processing to machine learning	- Relational DBMS - Objective is not just to query data but work with it in an algorithmic manner that requires complex math, statistics and Machine Learning

3.2 Discussion

The most difference between Hadoop and data warehouse is that Hadoop is a big data technology for storing and managing big data, whereas data warehouse is an architecture for organizing data to ensure integrity. A data warehouse is usually implemented in a single RDBMS which acts as a centre store, whereas Hadoop and HDFS span across multiple machines to handle large volumes of data that does not fit into the memory.[8]

Generally, ETL workloads are often offloaded to Hadoop. Hadoop is middleware infrastructure for parallelism and not an ETL solution, as it does not have what it takes to be data warehouse. [18] With many missing ETL subsystem features like data lineage, role based security, data quality and profiling subsystems, workflow management- Hadoop will be the one that exists together with data warehouse and make a perfect combination. A data warehouse makes the best use of relational and structured data whereas Hadoop excels in storing and managing unstructured data - which traditional data warehouses cannot handle. It is not a good choice to use Hadoop to process 20 GB of structured data as it can easily be tackled with SQL query, but traditional data warehouse systems cannot ingest complex hierarchical data types in polytrees or graphs.

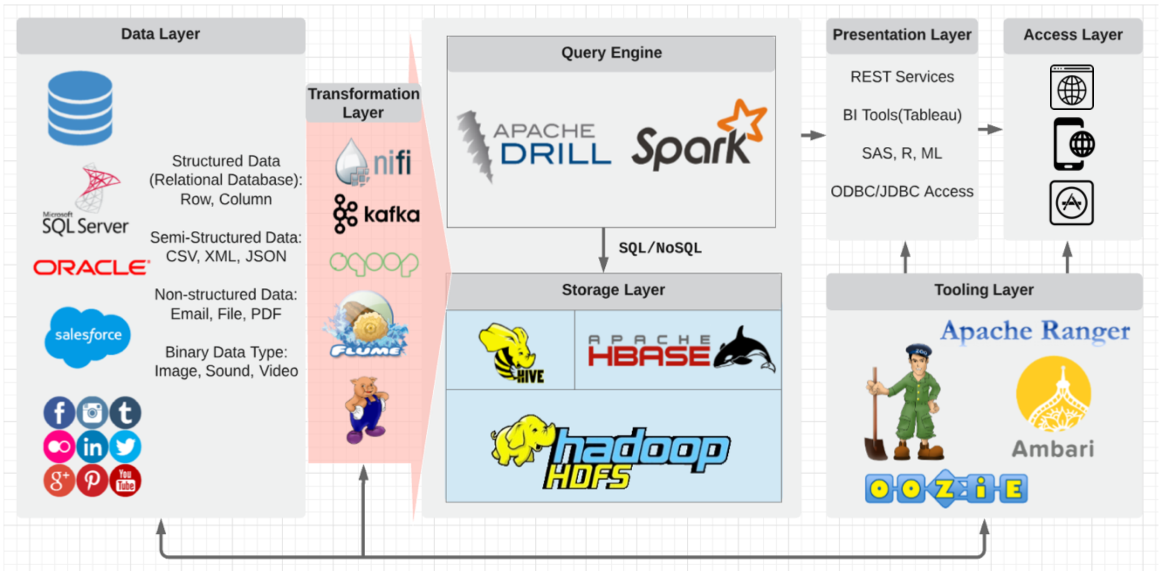
Here is a scenario that shows the motivation of the coexistence. Suppose - an IT company decides to pull data from its social networking website and merge it with the data from the data warehouses to update the social circle of a user's friends. Under these circumstances, using Hadoop to calculate the score of a person's social influence can be a cost effective and timely solution. This data is then sent back to the data warehouse where the campaign manager of the organization can see the user's social influence score and re-segment them accordingly.[8]

Combining use of multiple technologies or tools is really common in Big Data area, those tools are more functional because they are complementary to each other. NiFi was built to do one important thing well: data flow management. With both tools combined, NiFi can take advantage of Kafka's reliable stream data storage, while taking care of the dataflow challenges that Kafka was not designed to solve [12] Commonly HBase and Hive are also can be used together on the same Hadoop cluster. Hive worked as an ETL tool for batch inserts into HBase or to execute queries that join data present in HBase tables with the data present in HDFS files or in external data stores.[15]

Chapter 4
Implementation

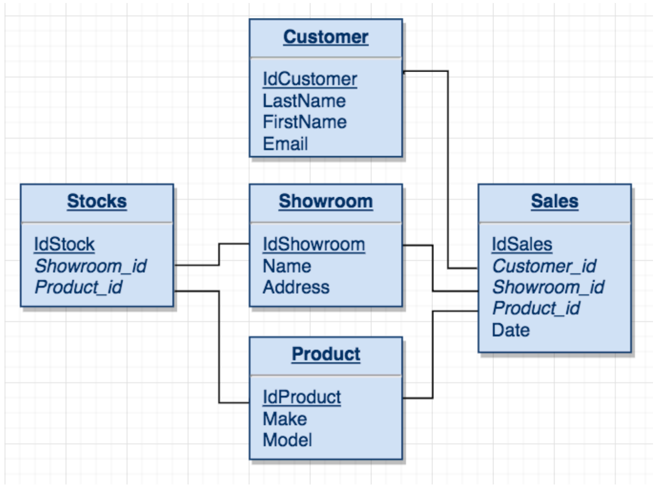
4.1 Overview

The Queen's University ITS needs a fully integrated data processing, reporting, analysis, and business intelligence services. This project designs and implements a data lake for a car trading firm to simulate ITS cases. There is a given scenario: How to analyze the students' attitude toward a new University policy given the student data stored in relational database and text data according to their tweets. The architecture of the data lake includes five layers: Data Layer, Transformation Layer, Storage Layer, Query Engine Layer and Presentation Layer. [19]



4.2 Data

a) Sample SQL database for the car trading firm with customer, product, showroom, sales and stocks tables.



b) Simulated student database and their posted tweets regarding car brands gathered using Twitter API.

4.3 Implementation Details

4.3.1 Implementation Environment

To implement the system, we used a SOSCIP cloud and the relevant services.

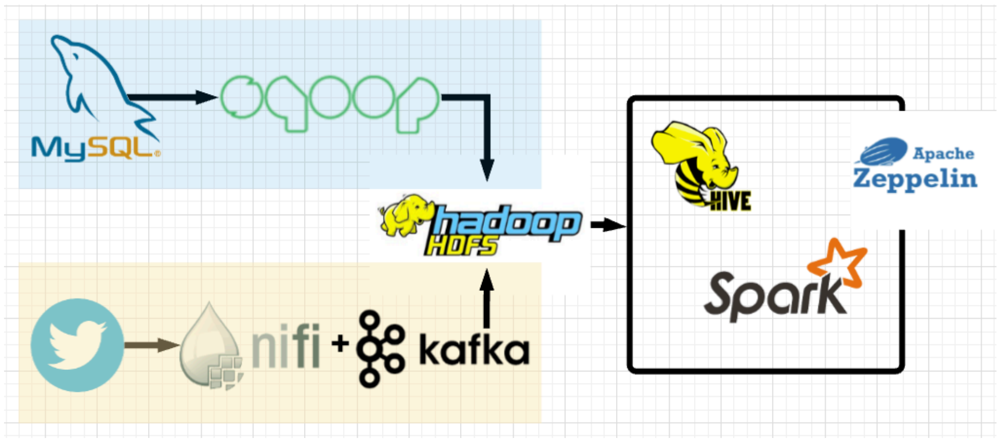
Platform: Hortonworks Data Platform

RAM: 48GB

VCPUs: 12 VCPUs

Disk: 120 GB

4.3.2 Validation



Transformation: Structured and un-structured data are transferred into same data storage system:

a) Extract each table from MySQL server (RDBMS) into HDFS using Sqoop.

```
sqoop import
--connect jdbc:mysql://hdp-bigsql.novalocal:3306/sales
--username=root
--password=root
--table customer
--target-dir /home/edw_user/sampledata
--hive-import
--hive-table sales.customer
```

The process of partitioning the data is used to reduce a large amounts of I/O operations by skip parts of unnecessary data to execute the query. Partitioning the data depends on the data access patterns. We need to retrieve the updates/inserts that were added to Customer and Sales tables in the last import. So, partitioning these tables is important since we do not want to query all the data.[9]

The partition size has an important role when storing the datasets in HDFS. The size of table is less than 100 MB which is not enough to fill a partition in HDFS. The partitioning strategies for all tables:

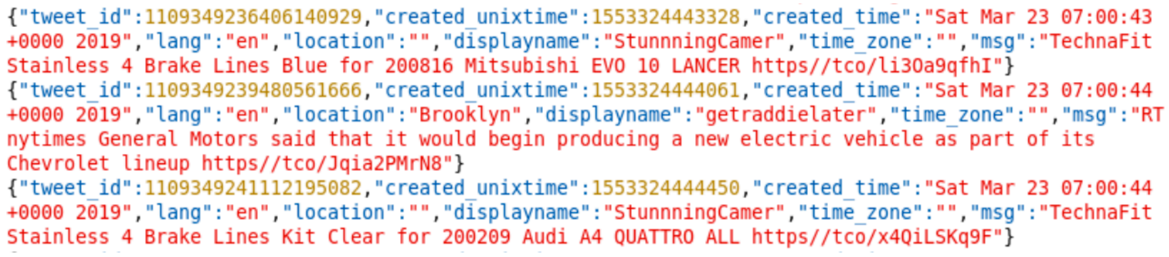
Data Set	Partitioned
Stocks	unpartitioned
Customer	Partitioned by Customer ID
Showroom	unpartitioned
Product	unpartitioned
Sales	Partitioned by month

Alternatively, extract table from MySQL server (RDBMS) with applied query

```
sqoop import
--connect jdbc:mysql://hdp-bigsql.novalocal:3306/sales
--username=root
--password=root
--query    "SELECT    sales.order_number,    sales.product_id,
product.make,      product.model  FROM  sales  LEFT  OUTER  JOIN
product ON    (sales.product_id     =     product.id)     WHERE
\$CONDITIONS AND sales.id !=  0"
--target-dir /home/edw_user/sampledata
--hive-import
--hive-table sales_analytics.jointable
-m 2
```

Specify the number of map tasks (parallel processes) to use to perform the import by using the -*m or –num-mappers* argument.

b) Gather text data from Twitter API through the combination of Nifi and Kafka into HDFS in JSON format.



```
{"tweet_id":1109349236406140929,"created_unixtime":1553324443328,"created_time":"Sat Mar 23 07:00:43
+0000 2019","lang":"en","location":"","displayname":"StunnningCamer","time_zone":"","msg":"TechnaFit
Stainless 4 Brake Lines Blue for 200816 Mitsubishi EVO 10 LANCER https//tco/li3Oa9qfhI"}
{"tweet_id":1109349239480561666,"created_unixtime":1553324444061,"created_time":"Sat Mar 23 07:00:44
+0000 2019","lang":"en","location":"Brooklyn","displayname":"getraddielater","time_zone":"","msg":"RT
nytimes General Motors said that it would begin producing a new electric vehicle as part of its
Chevrolet lineup https//tco/Jqia2PMrN8"}
{"tweet_id":1109349241112195082,"created_unixtime":1553324444450,"created_time":"Sat Mar 23 07:00:44
+0000 2019","lang":"en","location":"","displayname":"StunnningCamer","time_zone":"","msg":"TechnaFit
Stainless 4 Brake Lines Kit Clear for 200209 Audi A4 QUATTRO ALL https//tco/x4QiLSKq9F"}
```

Query: Apache Zeppelin is used as a multi-purposed web-based notebook which brings data ingestion, data exploration, visualization, sharing and collaboration features to Hadoop and Spark.

a) Query the best selling brands from table stored in Hive using SparkSQL

```
%scql
select brand, count(*) from sales_analytics.jointable Group by
brand order by count(*) DESC LIMIT 10
```

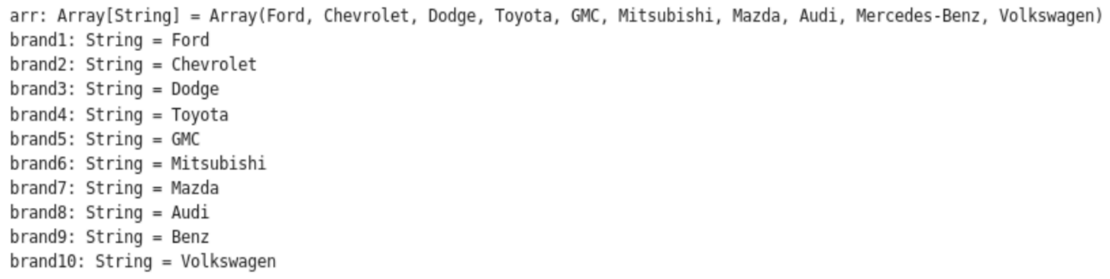
b) Convert the result into Spark RDD, map each row to get the string of each brand name

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.sql.hive.HiveContext
import sqlContext.implicits._

val resultDF = sqlContext.sql(“select brand, count(*) from
sales_analytics.jointable Group by brand order by count(*)
DESC LIMIT 10”)

val arr = resultDF.rdd.map(row=>row.getString(0)).collect;
val brand1 = arr(0);
val brand2 = arr(1);
val brand3 = arr(2);
val brand4 = arr(3);
val brand5 = arr(4);
val brand6 = arr(5);
val brand7 = arr(6);
val brand8 = arr(7);
val brand9 = arr(8);
val brand10 = arr(9);
```

The result shown as below:



```
arr: Array[String] = Array(Ford, Chevrolet, Dodge, Toyota, GMC, Mitsubishi, Mazda, Audi, Mercedes-Benz, Volkswagen)
brand1: String = Ford
brand2: String = Chevrolet
brand3: String = Dodge
brand4: String = Toyota
brand5: String = GMC
brand6: String = Mitsubishi
brand7: String = Mazda
brand8: String = Audi
brand9: String = Benz
brand10: String = Volkswagen
```

c) Count the number of messages in Json file. Convert the Twitter Json file into Spark RDD, label each row according to its msg content. Labeled results are written into Hive for later use.

```
   var messages = tweetDF.select(“msg”)
   println(“Total messages:” + messages.count())
```

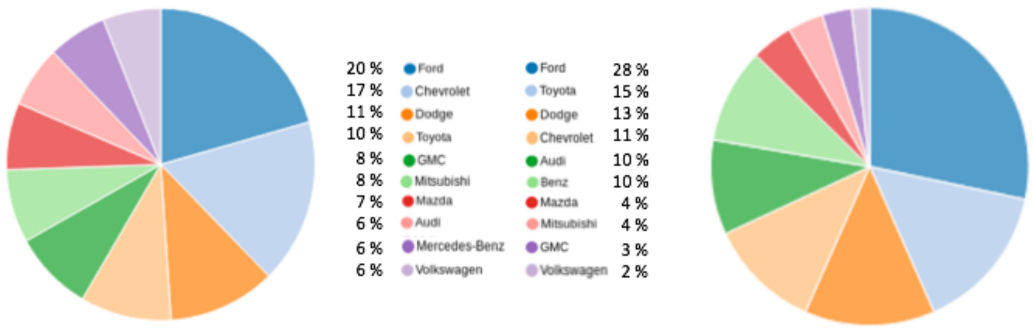
```
val messagesRDD = tweetDF.rdd
val BrandRecords = messagesRDD.map(
    row=> {
        Try{
            val msg = row.toString()
            var isBrand: String = “Other”
            if (msg.contains(brand1)){isBrand = brand1}
            else if (msg.contains(brand2)){isBrand = brand2}
                 ...
            else if (msg.contains(brand10)){isBrand = brand10}
            (isBrand,msg.split(“”).toSeq)
        }
    }
)

val exceptions = BrandRecords.filter(_.isFailure)
println("total records with exceptions: " + exceptions.count())
exceptions.take(10).foreach(x => println(x.failed))
var labeledTweets = BrandRecords.filter((_.isSuccess)).map(_.get)
println("total records with successes: "+ labeledTweets.count())

labeledTweets.toDF().write.saveAsTable(“rst”)
```

4.3.3 Results

The comparison between the sales of top ten brands and the frequency of each brand’s name contained in twitter messages.



JSON File Size	Number of Records	Analyzation Time
100 KB	320	17 secs
100 MB	320000	33 secs
1 GB	3530000	127 secs

4.2.1 Discussions

Because of the limitation of data set volume, it is hard to completely simulate Big Data environment. The multi-node Hadoop setup's performance might be subpar compared with a single SQL instance. This is a common situation in real world because many organizations do not have Big Data. You should know the purposes and capabilities of data lake technology beforehand compared with traditional data warehouse.

In order to speed up the Sqoop performance, we can apply the arguments given below:

- *–boundary-query* --- Boundary query to use for creating splits
- *–fetch-size* --- Number of entries to read from database at once.

In cases where computing resources are limited, Sqoop is not a viable option as it may have high resource consumption. Apache Spark SQL is the alternative tool migrating data from RDBMS to HDFS. JDBC driver source should be added in /lib directory in advance. [20]

Besides, security framework is also an important part needed to be considered in the future work. When building a data lake our goal is to have all data at one place and make it available for various teams. But when using the data lake, the goal changes to make the data generally unavailable (only available to those with permissions). The framework we built was for scientific usage without enterprise in mind.[21] Oozie and other tools can be used to organize the jobs and supervise the process. For example, a user can create a job to load some specific data using Sqoop metastore. Then other users can access from any node in the cluster the same job and just run it again. This is very convenient when using Sqoop in Oozie workflows.

Chapter 5

Conclusion and Future Work

5.1 Summary

In this paper, we outlined the concept of Big data technologies, the difference between traditional data warehouse and data lake, explained how a data lake can be constructed using Hadoop Data Platform. In addition to, we presented a proposed solution for the scenario - How to analyze the students' attitude toward a new University policy given the student data stored in RDBMS and the text data according to their tweets, which tests the conception of the data lake.

5.2 Limitation

The platform used in this project is not a commercial version. The advantages are that it is open-source software is created and maintained by a network of developers from around the world for free, while the performance of the platform is varied from time to time. Besides, some tools and technologies are updated frequently so that some online forums and research papers are no longer available as references. Finding the proper solution is a time-consuming process when applying cutting-edge technologies.

5.3 Future Work

In the future work, we plan to implement HBase and Hive together because they works on the same Hadoop cluster. Hive can be used as an ETL tool for batch inserts into HBase or to execute queries that join data present in HBase tables with the data present in HDFS files or in external data stores.[15]

Besides, define metadata such as catalog objects, which table consists of which files, statistics, locks, transaction logs, etc. in a scalable, transactional key-value database such as HBase is also a challenge.

REFERENCES

[1] Dageville, B., Huang, J., Lee, A. W., Motivala, A., Munir, A. Q., Pelley, S., . . . Hentschel, M. (2016). The Snowflake Elastic Data Warehouse. Proceedings of the 2016 International Conference on Management of Data - SIGMOD 16, 215-226. doi:10.1145/2882903.2903741

[2] What is Apache Hadoop? (n.d.). Retrieved from https://hortonworks.com/apache/hadoop/

[3] Hadoop, A., Inc, & Apache Software Foundation. (2018, December 28). Apache Hadoop core components. Retrieved from https://www.cloudera.com/products/open-source/apache-hadoop/hdfs-mapreduce-yarn.html

[4] Hadoop And Unstructured Data. (2018, November 30). Retrieved from https://analyticstraining.com/hadoop-and-unstructured-data/

[5] What is data lake? - Definition from WhatIs.com. (n.d.). Retrieved from https://searchaws.techtarget.com/definition/data-lake

[6] Mitra, S. (2018, February 20). Hadoop DataLake Implementation. Retrieved from https://dwbi.org/etl/bigdata/216-hadoop-datalake-implementation

[7] Tian, Y., Özcan, F., Zou, T., Goncalves, R., & Pirahesh, H. (2016). Building a Hybrid Warehouse. ACM Transactions on Database Systems, 41(4), 1-38. doi:10.1145/2972950

[8] Is Hadoop going to Replace Data Warehouse? (n.d.). Retrieved from https://www.dezyre.com/article/is-hadoop-going-to-replace-data-warehouse/256

[9] Doreswamy, Gad, I., & Manjunatha, B. R. (2017). Hybrid data warehouse model for climate big data analysis. 2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT). doi:10.1109/iccpct.2017.8074229

[10] Digging into Hadoop-based Big Data Architectures. (2017). International Journal of Computer Science Issues, 14(6), 52-59. doi:10.20943/01201706.5259

[11] Tyagi, P., & Demirkan, H. (n.d.). Data Lakes: The biggest big data challenges Why data lakes are an important piece of the overall big data strategy. Retrieved from http://analytics-magazine.org/data-lakes-biggest-big-data-challenges/

[12] Siciliani, T. (2018, November 12). Big Data Ingestion: Flume, Kafka, and NiFi - DZone Big Data. Retrieved from https://dzone.com/articles/big-data-ingestion-flume-kafka-and-nifi

[13] Top 18 Data Ingestion Tools - Compare Reviews, Features, Pricing in 2019. (2019, January 25). Retrieved from https://www.predictiveanalyticstoday.com/data-ingestion-tools/

[14] Neumann, S. (n.d.). Hive vs. HBase. Retrieved from https://www.xplenty.com/blog/hive-vs-hbase/

[15] Hive vs.HBase–Different Technologies that work Better Together. (n.d.). Retrieved from https://www.dezyre.com/article/hive-vs-hbase-different-technologies-that-work-better-together/322

[16] MatzzMatzz 346413, Tomer ShiranTomer Shiran 22623, & Ian DownardIan Downard 24137. (n.d.). Apache Drill vs Spark. Retrieved from https://stackoverflow.com/questions/29790655/apache-drill-vs-spark

[17] Spark SQL vs. Apache Drill-War of the SQL-on-Hadoop Tools. (n.d.). Retrieved from https://www.dezyre.com/article/spark-sql-vs-apache-drill-war-of-the-sql-on-hadoop-tools/234

[18] Mudit. (2016, August 21). Some thoughts on Data warehousing Architectures BI. Retrieved from https://medium.com/@muppal/some-thoughts-on-data-warehousing-architectures-bi-bfbd4707c7ef

[19] Phase 4 Optimization. (n.d.). Retrieved from https://mapr.com/ebooks/architects-guide/phase-4-optimization-copied.html

[20] Manjrekar, A. (n.d.). Data Migration from RDBMS to HDFS. Retrieved from https://explore.emtecinc.com/blog/data-migration-from-rdbms-to-hdfs

[21] Top 7 challenges of building a data lake. (2017, January 10). Retrieved from http://www.open-bigdata.com/top-7-challenges-of-building-a-data-lake/