**CCT College Dublin**

**Assessment Cover Page**

| **Module Title:** | Storage Solutions for Big Data |
| --- | --- |
| **Assessment Title:** | CA2 |
| **Lecturer Name:** | Dr. Muhammad Iqbal |
| **Student Full Name:** | Kavi Patak |
| **Student Number:** | sba22391 |
| **Assessment Due Date:** | 26 May 2024 |
| **Date of Submission:** | 26 May 2024 |

****

**Declaration**

| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |
| --- |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# Storage Solutions for Big Data:

Continuous Assessment 2

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

by Kavi Patak

sba22391

Storage Solutions for Big Data

Lecturer: Dr. Muhammad Iqbal

CCT College, Dublin

May 26, 2024

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# Table of Contents

[**Storage Solutions for Big Data: Continuous Assessment 2**](#_3iddv1ojtzjk)

[**Table of Contents**](#_ytyrg5mbj69k) **3**

[Question 1:](#_usiluud1xtj5) 5

[Define Big Data and outline its key characteristics.](#_l90b5kk1iqyh) 5

[Discuss the potential for banks to increase profits through big data processing and analysis.](#_pnfuy4ljevl8) 5

[Identify three businesses that have successfully leveraged big data storage solutions in recent times.](#_39296nz0rdbh) 5

[Question 2: Word Count A (Smaller file)](#_6v3q9s8f4v4a) 6

[Provide the screenshots based on your VM (hduser@studentusername:~$) or Google cloud platform (we check the Google cloud has been used or not, if not used, zero marks) for the following processes](#_9r0rujli3652) 6

[a) Start all five processes of hadoop distributed file system](#_kvao7nidgxdu) 6

[b) Find the text dataset of size (500 MB at least) and create a folder on hadoop named CA2. Copy the dataset from VM to the hdfs directory (CA2) the data.](#_o57biulv9hxn) 7

[c) Execute Mapper and Reducer programmes by using loaded input file.](#_hkzik1kyrpsj) 8

[d) Display the frequency of each word obtained from the dataset](#_ctcigno2jdfp) 14

[e) Download the output file from hadoop and upload it on Moodle. If you could not provide the screenshots for all commands or related files, no marks will be awarded.](#_emvdowj9i3ey) 14

[Question 2: Word Count A (Larger file)](#_7u0z4vmwfwto) 16

[Provide the screenshots based on your VM (hduser@studentusername:~$) or Google cloud platform (we check the Google cloud has been used or not, if not used, zero marks) for the following processes](#_mo5uw2hz7pxm) 16

[a) Start all five processes of hadoop distributed file system](#_k7h3qnmf2khs) 16

[b) Find the text dataset of size (500 MB at least) and create a folder on hadoop named CA2. Copy the dataset from VM to the hdfs directory (CA2) the data.](#_2nnw73gdae6i) 16

[c) Execute Mapper and Reducer programmes by using loaded input file.](#_cw6rvqj524a7) 17

[d) Display the frequency of each word obtained from the dataset](#_sr67lyn0f3pc) 20

[e) Download the output file from hadoop and upload it on Moodle. If you could not provide the screenshots for all commands or related files, no marks will be awarded.](#_9vqxv85q0iwc) 20

[Question 2: Top Ten](#_2w6az52pl4mj) 21

[Provide the screenshots based on your VM (hduser@studentusername:~$) or Google cloud platform (we check the Google cloud has been used or not, if not used, zero marks) for the following processes](#_c3syi1kzxmyg) 21

[a) Start all five processes of hadoop distributed file system](#_7cit48kk6c0) 21

[b) Find the text dataset of size (500 MB at least) and create a folder on hadoop named CA2. Copy the dataset from VM to the hdfs directory (CA2) the data.](#_k4mgd0u9mh1v) 21

[c) Execute Mapper and Reducer programmes by using loaded input file.](#_i4jyqsrisrc3) 22

[d) Display the frequency of each word obtained from the dataset](#_y6bwbhx6rgaf) 25

[e) Download the output file from hadoop and upload it on Moodle. If you could not provide the screenshots for all commands or related files, no marks will be awarded.](#_gfzkzhtr7bd) 25

[Question 3:](#_f47nfinc58d2) 26

[Discuss and demonstrate a comparison of MySQL and Apache Hive based on the architecture and performance.](#_515xav9xb5qu) 26

[Consider a dataset and perform a query on both systems with at least 5,000 rows and at least 5 features. Show the duration of query execution by displaying screenshots obtained from a virtual machine (VM).](#_j1sowpo606l0) 27

[Setting up Hive](#_i6wn8u8kqg44) 27

[Setting up MySQL Database and Table](#_ipsebcfzarzq) 29

[Select All Comparison](#_lwnu8e57x74b) 30

[Aggregation Comparison](#_v0zj90o5559p) 31

[Filtering Comparison](#_62ugu1nuasiz) 33

[Joining Comparison](#_kgidl03b5j9j) 36

[Question 4:](#_m4dsfqps71ot) 37

[Using Apache Pig and the provided dataset with columns Invoice, StockCode, Description, Quantity, InvoiceDate, Price, Customer ID, and Country, write a Pig script to calculate the total sales generated by customers in each country, as well as identify the top 10 products sold overall. Your script should compute the total sales for each country and then list the top 10 products based on the total sales amount. Ensure to handle any null or invalid values appropriately in the dataset. Provide the Screenshots for the execution of Pig Script on VM and explain the purpose of all these steps.](#_8gbg64qhaoxe) 37

[Question 5:](#_jv0anja79xcr) 44

[Explain Apache Flink architecture and illustrate with your own conceptual diagram (Use of online/ book images is prohibited, Use draw.io to create the image).](#_dfx1p1del3z3) 44

[What is Apache Storm, and how does it differ from other distributed computing systems?](#_72jjuvlb75on) 46

[Consider a text file comprising at least 20,000 words and write a wordcount program (Java/ Python) to count the frequency of words and related aggregation functions.](#_r58z3dwfi8pu) 46

[Word Count Topology:](#_trqmspt6gxg7) 46

[pom.xml (one of many version attempted)](#_71edx5s438mj) 49

[**References**](#_kya26suve4eo) **53**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

### Question 1:

#### Define Big Data and outline its key characteristics.

Big Data may be defined in many ways, depending on who you ask. However, most definitions of big data reflect the growing technological ability to capture, aggregate, and process an ever increasing volume, velocity, and variety of data (Iqbal, n.d.). Volume, Velocity and Variety, have become known as the 3 V’s of Big Data. Big Data comes in a variety of forms including pictures, video, audio, and social media, and hence a variety of formats, whether structured, semi structured, or unstructured. Volume and variety refer to data that is too large or complex for storage and analysis by traditional methods. Since the advent of the Internet and proceeding developments in technologies like the Internet of Things(IoT), Sensors, GPS devices, Machine Learning (ML) and Artificial Intelligence (AI), the volume and variety of data being produced is on an upward and seemingly unbound trajectory (Agnellutti, 2014). This data is being produced at an exponential rate, with velocity referring to this, and the requirement for its instant processing for immediate use and inferred insight. Other interpretations of Big Data include the addition of veracity and value, with veracity referring to the messiness and trustworthiness of data, while value refers to the potential insight or gain, hidden within the data (Yaqoob et al., 2016).

#### Discuss the potential for banks to increase profits through big data processing and analysis.

The banking industry, like most others, stands to benefit greatly from the storage, processing and analysis of big data.

It is an extremely competitive industry, with most banks offering similar financial products and services. The opportunity exists for banks to harness big data through their daily transactions, e-banking, real-time feeds, social media posts and customer service records, and from this develop analytic models to optimise outcomes and derive insights for a competitive advantage (Bedeley, 2014). Specific areas in which banks may benefit include through customer segmentation and personalised services, including target marketing campaigns. By leveraging AI for the analysis of transactional data, banks can improve their fraud detection capabilities as well as develop models for advanced risk assessments and credit scoring, for loan and service approvals. Additionally, predictive models may aid in analysing the financial landscape for economic indicators thereby aiding informed investment decisions.Through the processing and analysis of big data, banks may improve every day operational efficiency, including optimising processes, reducing expenditure and resource management. Finally, the analysis of big data such as social media would lead to insights into customer behaviour, brand loyalty, competitor actions, customer services interactions and customer experiences through sentiment analysis.

#### Identify three businesses that have successfully leveraged big data storage solutions in recent times.

The first business that not only leverages big data storage solutions, but offers its own is Google. Google uses Big data to better understand its users and to improve its services (Kumari, 2021). Google also -provides storage solutions through google cloud platform and workspace .

Amazon is another powerhouse in the e-commerce sector that both utilises and provides Big Data solutions. Amazon collects and stores boundless data related to products and customers (Kumari, 2021). They too provide a cloud computing platform with a diverse array of services through Amazon Web Services (AWS)(Emergen Research, https://www.emergenresearch.com, 2024).

A third company leveraging big data storage solutions would be IBM. IBM is a global technology and consulting giant that has played an influential role in the ever evolving technological landscape (Emergen Research, https://www.emergenresearch.com, 2024). They are one of the biggest vendors of big data related products and services, with their storage solutions known for their scalability, integrated data management and analytics capabilities (www.linkedin.com, n.d.).

Section Word Count: 544

### Question 2: Word Count A (Smaller file)

#### Provide the screenshots based on your VM (hduser@studentusername:~$) or Google cloud platform (we check the Google cloud has been used or not, if not used, zero marks) for the following processes

#### a) Start all five processes of hadoop distributed file system

| # Open Terminal  Ctrl + Alt + t  # Change to Desktop  Cd Desktop  # Start HDFS  ./hdfs |
| --- |

#### b) Find the text dataset of size (500 MB at least) and create a folder on hadoop named CA2. Copy the dataset from VM to the hdfs directory (CA2) the data.



| # Creating a new folder  hadoop fs -mkdir /CA2    # Change Directory  cd  cd Downloads  # Move the text file from Downloads to CA2  hadoop fs -put ./A\_Tale\_of\_Two\_Cities.txt /CA2  # List the contents of CA2 to confirm move  hadoop fs -ls /CA2 |
| --- |

#### c) Execute Mapper and Reducer programmes by using loaded input file.

|  |
| --- |

|  |
| --- |

| # Change Directory  cd  cd Desktop/Tutorial\_3  # Confirming mapper and reducer files are listed  ls  # Initiating Hadoop Streaming  hadoop jar $HADOOP\_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -mapper ./mapper.py -reducer ./reducer.py -input /CA2/A\_Tale\_of\_Two\_Cities.txt -output /CA2WordCount |
| --- |

#### d) Display the frequency of each word obtained from the dataset

| # Checking contents of CA2WordCount (output file)  hadoop fs -ls /CA2WordCount  # Checking word count  hadoop fs -cat /CA2WordCount/part-00000 |
| --- |

#### e) Download the output file from hadoop and upload it on Moodle. If you could not provide the screenshots for all commands or related files, no marks will be awarded.

| # Copying the CA2WordCount (output file) to local drive  Hadoop fs -copyToLocal /CA2WordCount ./  # Listing contents of Tutorial\_3 folder to confirm copy  ls  # Changing directory to CA2WordCount  Cd CA2WordCount  # Listing contents of CA2WordCount  ls  # Checking word count  cat part-00000 |
| --- |

### Question 2: Word Count A (Larger file)

#### Provide the screenshots based on your VM (hduser@studentusername:~$) or Google cloud platform (we check the Google cloud has been used or not, if not used, zero marks) for the following processes

#### a) Start all five processes of hadoop distributed file system

| # Open Terminal  Ctrl + Alt + t  # Change to Desktop  Cd Desktop  # Start HDFS  ./hdfs |
| --- |

#### b) Find the text dataset of size (500 MB at least) and create a folder on hadoop named CA2. Copy the dataset from VM to the hdfs directory (CA2) the data.



| # Creating a new folder  hadoop fs -mkdir /CA2  # Change Directory  cd  cd Downloads  # Move the text file from Downloads to CA2  hadoop fs -put ./200mb-example-dummy-file\_1.txt /CA2  # List the contents of CA2 to confirm move  hadoop fs -ls /CA2 |
| --- |

#### c) Execute Mapper and Reducer programmes by using loaded input file.

|  |
| --- |

|  |
| --- |

| # Change Directory  cd  cd Desktop/Tutorial\_3  # Confirming mapper and reducer files are listed  ls  # Initiating Hadoop Streaming  hadoop jar $HADOOP\_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -mapper ./mapper.py -reducer ./reducer.py -input /CA2/200mb-example-dummy-file\_1.txt -output /CA2WordCountLarge |
| --- |

#### d) Display the frequency of each word obtained from the dataset

| # Checking contents of CA2WordCount (output file)  hadoop fs -ls /CA2WordCountLarge  # Checking word count  hadoop fs -cat /CA2WordCountLarge/part-00000 |
| --- |

#### e) Download the output file from hadoop and upload it on Moodle. If you could not provide the screenshots for all commands or related files, no marks will be awarded.

| # Copying the CA2WordCount (output file) to local drive  hadoop fs -copyToLocal /CA2WordCountLarge ./  # Listing contents of Tutorial\_3 folder to confirm copy  ls  # Changing directory to CA2WordCount  cd CA2WordCountLarge  # Listing contents of CA2WordCount  ls  # Checking word count  cat part-00000 |
| --- |

### Question 2: Top Ten

#### Provide the screenshots based on your VM (hduser@studentusername:~$) or Google cloud platform (we check the Google cloud has been used or not, if not used, zero marks) for the following processes

#### a) Start all five processes of hadoop distributed file system

| # Open Terminal  Ctrl + Alt + t  # Change to Desktop  Cd Desktop  # Start HDFS  ./hdfs |
| --- |

#### b) Find the text dataset of size (500 MB at least) and create a folder on hadoop named CA2. Copy the dataset from VM to the hdfs directory (CA2) the data.



| # Creating a new folder  hadoop fs -mkdir /CA2  # Change Directory  cd Downloads  # Move the text file from Downloads to CA2  hadoop fs -put ./diamonds.txt /CA2  # List the contents of CA2 to confirm move  hadoop fs -ls /CA2 |
| --- |

#### c) Execute Mapper and Reducer programmes by using loaded input file.

|  |
| --- |

#### 

|  |
| --- |

| # Change Directory  cd  cd Desktop/Tutorial\_3  # Confirming mapper and reducer files are listed  ls    # Initiating Hadoop Streaming  hadoop jar $HADOOP\_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -mapper ./topTenDiamondsByPriceMapper.py -reducer ./topTenDiamondsByPriceReducer.py -input /CA2/diamonds.txt -output /CA2TopTenDiamondCuts |
| --- |

#### d) Display the frequency of each word obtained from the dataset

| # Checking contents of CA2TopTenDiamondsByPrice (output file)  hadoop fs -ls /CA2TopTenDiamondsByPrice  # Checking word count  hadoop fs -cat /CA2TopTenDiamondsByPrice/part-00000 |
| --- |

#### e) Download the output file from hadoop and upload it on Moodle. If you could not provide the screenshots for all commands or related files, no marks will be awarded.

| # Copying the CA2WordCount (output file) to local drive  hadoop fs -copyToLocal /CA2TopTenDiamondsByPrice ./  # Listing contents of Tutorial\_3 folder to confirm copy  ls  # Changing directory to CA2TopTenDiamondsByPrice  cd CA2TopTenDiamondsByPrice  # Listing contents of CA2TopTenDiamondsByPrice  ls  # Checking top ten  cat part-00000 |
| --- |

### Question 3:

#### Discuss and demonstrate a comparison of MySQL and Apache Hive based on the architecture and performance.

MySQL is a commonly used open-source relational database management system (RDBMS), that supports predefined data types like date and float. MySQL provides Structured Query Language (SQL) statements including DDL (Data Definition Language) and DML (Data Manipulation Language). DDL provides the ability to define, create and modify database objects such as tables, indexes, and views. DML allows for manipulating data in a database, like inserting, updating, and deleting records. MySQL supports secondary indexing, which is a way to efficiently access records in a database without the use of the primary key. It also offers user-defined functions and the integration of map-reduce. MySQL follows ACID properties in Atomicity, Consistency, Isolation, and Durability (db-engines.com, n.d.). Other features include sharding or horizontal partitioning that separates large databases into smaller, faster, more easily manageable database shards as well as various replication methods.

Apache Hive on the other hand provides data warehouse functionality, built on top of Hadoop, for querying and managing large distributed datasets. It is a system or framework for managing and querying unstructured data as if it was structured. With regards to data types, Hive is slightly more limited and does not support spatial data (Quora, n.d.). This can however be facilitated with the use of external libraries. Similarly to MySQL, Hive offers SQL like syntax through HiveQL (HQL). Hive too supports secondary indexing and user defined functions to manipulate data and support data-mining, while using map reduce for execution (Iqbal, n.d.). Apache Hives architecture consists of three core parts in Hive Clients, Hive Services and Hive Storage and Computing, and seven main components in the Metastore, Driver, Command Line Interface (VLI), Web Interface, Thrift Server, JDBC (Java database connectivity) and ODBC (Open Database Connectivity) (www.guru99.com, n.d.).

For communicating with different applications, Hive provides different drivers.Thrift client is provided for communication with Thrift based applications, JDBC is provided for Java applications, with ODBC drivers provided for other applications. These Clients and drivers themselves communicate with Hive server within Hive Services.

Client interactions with Hive including query related operations can be performed through Hive Services. DDL operations are performed through CLI with all drivers communicating with Hive server and to the main driver in Hiver services. This main driver communicates with all applications and will process different requests to the Meta store and field systems for further processing.

Metastore, File system, Job Client and other services then communicate with Hive storage. Hive's Meta storage database stores Metadata information of tables that are created in Hive, while query results and any data loaded in tables is stored in HDFS (www.guru99.com, n.d.).

With regards to performance, both tools perform similar actions in retrieving data but in very different ways. Hive is designed as a convenient interface for querying data within HDFS while MySQL is intended for online operations requiring many reads and writes. Hive applies serialisation and deserialisation adaptors which allow users to redefine tables to match data without actually touching the data using a method called ‘schema on read’ (Iqbal, n.d.). This makes it less suitable for online tasks which require heavy read and write traffic (Rathbone, 2015). MySQL on the other hand defines table schemas before adding data using ‘schema on write’, which allows for data reading and writing in an optimal way. Hive provides eventual consistency while MySQL offers immediate consistency. Hive is better utilised when performing complex querying and analytics on large datasets already stored on Hadoop. It offers extensibility, interoperability and performance through query optimisation and hash joining.

MySQL performs better on smaller datasets, for frequently updating and modifying records and for online transactions (Rathbone, 2015).

#### Consider a dataset and perform a query on both systems with at least 5,000 rows and at least 5 features. Show the duration of query execution by displaying screenshots obtained from a virtual machine (VM).

##### Setting up Hive

| # Open Terminal  Ctrl + Alt + t  # Change to Desktop  Cd Desktop  # Start HDFS  ./hdfs          CREATE TABLE IF NOT EXISTs diamondsTable (carat FLOAT, cut STRING, color STRING, clarity STRING, depth FLOAT, ‘table’ FLOAT, price INT, x FLOAT, y FLOAT, z FLOAT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';    LOAD DATA INPATH 'hdfs://localhost:9000/CA2/diamonds.txt' INTO TABLE diamondsTable; |
| --- |

##### Setting up MySQL Database and Table

|  |
| --- |

##### Select All Comparison

| SELECT \* FROM diamondsTable; |
| --- |

| SELECT \* FROM diamonds\_table; |
| --- |

##### Aggregation Comparison

| SELECT AVG(price) FROM diamondsTable; |
| --- |

| SELECT AVG(price) FROM diamonds\_table; |
| --- |

| SELECT AVG(carat) FROM diamondsTable; |
| --- |

| SELECT AVG(carat) FROM diamonds\_table; |
| --- |

| SELECT COUNT(\*) FROM diamondsTable; |
| --- |

| SELECT COUNT(\*) FROM diamonds\_table; |
| --- |

##### 

##### Filtering Comparison

| SELECT \* FROM diamondsTable WHERE carat > 1.0; |
| --- |

| SELECT \* FROM diamonds\_table WHERE carat > 1.0; |
| --- |

| SELECT \* FROM diamondsTable WHERE carat > 2.0 ORDER BY price DESC LIMIT 10; |
| --- |

| SELECT \* FROM diamonds\_table WHERE carat > 2.0 ORDER BY price DESC LIMIT 10; |
| --- |

| SELECT \* FROM diamondsTable WHERE cut = '"Ideal"'; |
| --- |

| SELECT \* FROM diamonds\_table WHERE cut = '”Ideal”'; |
| --- |

##### Joining Comparison

| SELECT d.\*, dummy.\* FROM diamondsTable d JOIN (SELECT 1 AS dummy\_col) dummy ON 1=1;    First execution:    Second execution: |
| --- |

| SELECT d.\*, dummy.\* FROM diamonds\_table d JOIN (SELECT 1 AS dummy\_col) dummy ON 1=1; |
| --- |

Across all queries, whether for aggregation, filtering, or for joins, MySQL consistently outperforms Apache Hive. The overall results are as expected, with MySQL proving more efficient for querying a relatively small dataset of 53939 rows and 10 columns. Both tools have similar querying structure as we discussed earlier, with Hive providing SQL like syntax for querying HDFS. With this, Hive's architecture introduces additional overheads due to the requirement of translating SQL like queries into MapReduce Jobs. MySQL on the other hand, benefits from its query optimisation, execution engine and relational database design. We find Hive's architecture and MapReduce execution to be considerably slower for join queries as tested with a dummy table above. Apache Hive is however designed for large scale data processing and analysis, executable on Hadoop clusters, for which it should outperform MySQL.

Section Word Count: 715

### Question 4:

#### Using Apache Pig and the provided dataset with columns Invoice, StockCode, Description, Quantity, InvoiceDate, Price, Customer ID, and Country, write a Pig script to calculate the total sales generated by customers in each country, as well as identify the top 10 products sold overall. Your script should compute the total sales for each country and then list the top 10 products based on the total sales amount. Ensure to handle any null or invalid values appropriately in the dataset. Provide the Screenshots for the execution of Pig Script on VM and explain the purpose of all these steps.

| # Open Terminal  Ctrl + Alt + t  # Change to Desktop  Cd Desktop  # List Desktop contents to ensure files are present  ls  # Start HDFS  ./hdfs |
| --- |

| # Move the csv files from Desktop to Hadoop CA2 storage  hadoop fs -put ./online\_retail\_II\_A /CA2  hadoop fs -put ./online\_retail\_II\_B /CA2  # Check contents of Hadoop CA2 to confirm move  Hadoop fs -ls /CA2 |
| --- |

| # Creating a Pig script file of code for processing on the Desktop  nano pig\_sales\_processing\_script.pig |
| --- |

| -- Loading csv data files  dataA = LOAD 'hdfs://localhost:9000/user1/CA2/online\_retail\_II\_A.csv' USING PigStorage(',') AS (  Invoice: int,  StockCode: chararray,  Description: chararray,  Quantity: int,  InvoiceDate: chararray,  Price: float,  CustomerID: int,  Country: chararray  );  dataB = LOAD 'hdfs://localhost:9000/user1/CA2/online\_retail\_II\_B.csv' USING PigStorage(',') AS (  Invoice: int,  StockCode: chararray,  Description: chararray,  Quantity: int,  InvoiceDate: chararray,  Price: float,  CustomerID: int,  Country: chararray  ); |
| --- |

Above, we load in the two csv files, each representing a different worksheet from the provided online\_retail\_II file, using the Apache Pig LOAD,USING and AS operators. LOAD specifies and loads the data. USING is a keyword that specifies the PigStorage delimiter function( the data is comma separated ) and AS the specified schema with data types.

| -- Combining datasets(Excel Worksheets)  all\_data = UNION dataA, dataB; |
| --- |

Here we combine/merge the two dataset (worksheets) using the UNION operator.

| -- Filtering out invalid or missing values  cleaned\_data = FILTER all\_data BY (Invoice IS NOT NULL AND StockCode IS NOT NULL AND Description IS NOT NULL AND Quantity IS NOT NULL AND InvoiceDate IS NOT NULL AND Price IS NOT NULL AND Country IS NOT NULL); |
| --- |

Using the FILTER and IS NOT NULL operators we create a new dataset by filtering/removing null values for each of the specified fields(columns).

| -- Calculating and adding a total sale (SaleAmount) for each entry  data\_with\_sales = FOREACH cleaned\_data GENERATE Country, StockCode, Description, (Quantity \* Price) AS SaleAmount; |
| --- |

Creating a new field SaleAmount for total sale amount for each entry(row) by multiplying the price with the quantity purchased.The FOREACH operator iterates over each sales entry and calculates the total sale amount, while GENERATE specifies the fields that should be included or added.

| -- Calculating the total sales by Country  sales\_by\_country = GROUP data\_with\_sales BY Country;  total\_sales\_by\_country = FOREACH sales\_by\_country GENERATE group AS Country, SUM(data\_with\_sales.SaleAmount) AS CountryTotalSales; |
| --- |

In calculating the total sales by country we first use the GROUP operator to group the new dataset with SaleAmount above into a new relation sales\_by\_country based on the country using the BY keyword. We then iterate through each country group created in the previous line in “sales\_by\_country” using the FOREACH operator and aggregate the total sum of “SaleAmount” using the SUM operator, to calculate the total sales amount for each country group. GENERATE again specifies what fields to include or add, in this case Country and CountryTotalSales respectively.

| -- Finding the top 10 products sold overall  products\_sales = GROUP data\_with\_sales BY (StockCode, Description);  total\_sales\_by\_product = FOREACH products\_sales GENERATE FLATTEN(group) AS (StockCode, Description), SUM(data\_with\_sales.SaleAmount) AS ProductTotalSales;  top\_10\_products = LIMIT (ORDER total\_sales\_by\_product BY ProductTotalSales DESC) 10; |
| --- |

Similar to the previous block of code, in calculating the top 10 products by sales overall, we first group the data\_with\_sales data by StockCode and Description into a new relation called product sales using the GROUP and BY operators. Both StockCode and Description are specified for grouping for more precision as some products may be unique but share the same StockCode. We then create a new relation called total\_sales\_by\_product by iterating through each of the unique product sales created in the previous line by grouping and total the sales amounts for each of the groups into a newly generated field called ProductTotalSales using GENERATE. FLATTEN is also used to flatten the previously grouped tuple for easier use and to specify the fields to add to the total\_sales\_by\_product relation. The top ten products by sales are then ordered in descending order for ProductTotalSales using the ORDER and DESC operators and limited to include just ten entries using the LIMIT operator.

| -- Storing the results to HDFS  STORE total\_sales\_by\_country INTO 'hdfs://localhost:9000/user1/CA2/pig\_output/total\_sales\_by\_country' USING PigStorage(',');  STORE top\_10\_products INTO 'hdfs://localhost:9000/user1/CA2/pig\_output/top\_10\_products' USING PigStorage(','); |
| --- |

Finally, we store the previous calculations for total sales by country and top ten products by total sales to the specified HDFS pathways using the STORE operator while specifying how the data should be stored (comma separated) with USING PigStorage(‘,’).

| # Move the pig script to HDFS and confirm move  hadoop fs -put ./pig\_sales\_processing\_script.pig /CA2  Hadoop fs -ls /CA2 |
| --- |

| # Starting Pig  Pig -x local |
| --- |

| # Executing sales processing pig script  exec hdfs://localhost:9000/CA2/pig\_sales\_processing\_script.pig |
| --- |

| # Checking output in hadoop  hadoop fs -ls /CA2/  hadoop fs -ls /CA2/pig\_output  hadoop fs -ls /CA2/pig\_output/total\_sales\_by\_country |
| --- |

| hadoop fs -cat /CA2/pig\_output/total\_sales\_by\_country/part-r-00000 |
| --- |

The output for total sales by country(region) appears to be successful. The country(region) is displayed with the respective total sales value beside it. Some values, as with for United Kingdom appear to be in scientific notation.

| hadoop fs -ls /CA2/pig\_output/top\_10\_products |
| --- |

The output for the top 10 products overall appears to be successful. As per the pig script, we specified the inclusion of the stock code and description, which is displayed followed by the product's totals sales. They are also listed in descending order.

| # Copy output files to local drive  hadoop fs -copyToLocal /CA2/pig\_output ./ |
| --- |

Section Word Count: 508

### Question 5:

#### Explain Apache Flink architecture and illustrate with your own conceptual diagram (Use of online/ book images is prohibited, Use draw.io to create the image).

Apache Flink is a powerful open source processing engine for stateful computations over unbounded and bounded data streams. It enables the processing of large-scale data streams with low latency and high throughput, while providing communications, fault tolerance, and distributed computations (Iqbal, n.d.). Its architecture is designed to handle both batch and stream processing efficiently, and is similar to Hadoop in that it follows a master-slave design (Iqbal, n.d.). Additionally, Flink is designed to run in all common cluster environments and execute at any scale, while performing computations at in-memory speed.

Apache Flink works on a Kappa architecture as opposed to Lambda, in which a single processor-stream handles both batch and real-time data through a single engine(www.tutorialspoint.com, n.d.).

The main components within the Flink architecture for job execution include the Program, Optimiser/Graph Builder, Client, Job Manager, Task Manager, Runtime and State Backend Storage.

The Program or program application, is a piece of code, usually written by the developer, which runs on the Flink cluster.

The Optimiser and Graph builder handle the parsing of the code, its optimisation and the conversion of every job to a data flow graph with the aid of the client (Iqbal, n.d.). The Client, although not part of the runtime and program execution, is responsible for preparing and sending a dataflow to the JobManager. Following this, it can disconnect, or be set to stay connected, monitor interactions and receive progress reports. The Job Manager is one of the main components in the Flink architecture, with some of its responsibilities including creating execution graphs, job scheduling, resource allocation, checkpointing, and recovery.

The main components within the Job Manager are the Resource Manager which manages resources, task slots and communicates with the cluster manager, the Dispatcher which manages job submissions and interactions with the REST API and WebUI to provide information on job executions, the Job Master of which there is always at least one, is responsible for the execution of a single JobGraph, the Scheduler for assigning tasks to Task Managers and the Checkpoint Coordinator which handles state snapshots for fault tolerance (nightlies.apache.org, n.d.). The Task Manager or worker node, is responsible for executing all of the tasks of a dataflow that have been assigned by the Job manager, while maintaining an update status (www.tutorialspoint.com, n.d.). Components of the Task Manager include task slots which are the smallest unit of resource scheduling and the Network Stack which facilitates communication between Tasks Managers and data exchange during task execution. As with the Job Master, there must always be at least one Task Manager (nightlies.apache.org, n.d.). A conceptual diagram of Apache Flinks architecture, and the aforementioned components has been created, using Microsoft Powerpoint, and can be seen below.

|  |
| --- |

#### What is Apache Storm, and how does it differ from other distributed computing systems?

Apache Storm is an open source, distributed big data processing system, that excels at processing large streams of data in real-time.

Storm was originally developed by Nathan Marz at Backtype, for use on social search applications before being acquired by Twitter, for whom it became an essential part of their infrastructure for various critical computations at scale and in real-time (Gupta, 2023).

Apache Storm is now considered an industry leader for distributed, real-time data processing. Its success, and what makes it different from competitors, lies in its ability to provide numerous features such as scalability, fault tolerance, high throughput, cluster isolation, modularity, continuous computation, real-time stream processing and ease of use, all within one package (Iqbal, n.d.).

Technological developments continue to improve the volume, velocity, variety and veracity of data available to organisations of all sizes, and with it, comes the need for its immediate processing and insight, as well as the ability to swiftly respond and adapt to changing situations (Iqbal, n.d.).

Apache Storm provides the solution. Apache Storm is ideal for real time analytics, as it processes stream data in realtime through continuous computation, leveraging Distributed Remote Procedure Calls (DRPCs) to parallelize the computation of intense queries. DRPC is not so much a feature of Storm as it is a pattern expressed from Storm's primitives of streams, spouts, bolts, and topologies (storm.apache.org, n.d.). Storm is robust, reliable and horizontally scalable, through the addition of nodes to a cluster, without any downtime. This allows Storm to maintain performance, even under increasing load through the linear addition of resources. Storm is reliable and fault tolerant, even following node failures. It ensures no data loss, guaranteeing that every message or piece of data is processed at least once (www.tutorialspoint.com, n.d.). Additionally, it is user friendly and flexible, easily integrating with various data sources and in a variety of programming languages. Finally, Apache Storm is fast and powerful, with an incredibly high ingestion rate and very low latency (www.tutorialspoint.com, n.d.).

#### Consider a text file comprising at least 20,000 words and write a wordcount program (Java/ Python) to count the frequency of words and related aggregation functions.

Numerous attempts are made to execute the code but are ultimately unsuccessful. Issues arise with dependencies within the jar and pom files. Multiple attempts are made, including using or altering the pom.xml that successfully executes from a class example. Proof of attempts can be found in the accompanying document, “SSBD CA2 Q5 Wordcount with Storm”.

##### Word Count Topology:

| package com.example;  import org.apache.storm.Config;  import org.apache.storm.LocalCluster;  import org.apache.storm.StormSubmitter;  import org.apache.storm.spout.SpoutOutputCollector;  import org.apache.storm.task.OutputCollector;  import org.apache.storm.task.TopologyContext;  import org.apache.storm.topology.BasicOutputCollector;  import org.apache.storm.topology.OutputFieldsDeclarer;  import org.apache.storm.topology.TopologyBuilder;  import org.apache.storm.topology.base.BaseBasicBolt;  import org.apache.storm.topology.base.BaseRichSpout;  import org.apache.storm.tuple.Fields;  import org.apache.storm.tuple.Tuple;  import org.apache.storm.tuple.Values;  import org.apache.storm.utils.Utils;  import java.io.BufferedReader;  import java.io.FileReader;  import java.util.HashMap;  import java.util.Map;  public class WordCountTopology {  public static class TextFileSpout extends BaseRichSpout {  private SpoutOutputCollector collector;  private BufferedReader reader;  private String line;  @Override  public void open(Map conf, TopologyContext context, SpoutOutputCollector collector) {  this.collector = collector;  try {  reader = new BufferedReader(new FileReader("input/A\_Tale\_of\_Two\_Cities.txt"));  } catch (Exception e) {  e.printStackTrace();  }  }  @Override  public void nextTuple() {  try {  if ((line = reader.readLine()) != null) {  collector.emit(new Values(line));  } else {  Utils.sleep(100); // Sleep if no more lines to read  }  } catch (Exception e) {  e.printStackTrace();  }  }  @Override  public void close() {  try {  reader.close();  } catch (Exception e) {  e.printStackTrace();  }  }  @Override  public void declareOutputFields(OutputFieldsDeclarer declarer) {  declarer.declare(new Fields("line"));  }  }  public static class WordSplitBolt extends BaseBasicBolt {  @Override  public void execute(Tuple tuple, BasicOutputCollector collector) {  String line = tuple.getStringByField("line");  String[] words = line.split("\\s+");  for (String word : words) {  collector.emit(new Values(word));  }  }  @Override  public void declareOutputFields(OutputFieldsDeclarer declarer) {  declarer.declare(new Fields("word"));  }  }  public static class WordCountBolt extends BaseBasicBolt {  private Map<String, Integer> wordCounts = new HashMap<>();  @Override  public void execute(Tuple tuple, BasicOutputCollector collector) {  String word = tuple.getStringByField("word");  int count = wordCounts.getOrDefault(word, 0) + 1;  wordCounts.put(word, count);  collector.emit(new Values(word, count));  }  @Override  public void declareOutputFields(OutputFieldsDeclarer declarer) {  declarer.declare(new Fields("word", "count"));  }  }  public static void main(String[] args) throws Exception {  TopologyBuilder builder = new TopologyBuilder();  builder.setSpout("text-file-spout", new TextFileSpout(), 1);  builder.setBolt("word-split-bolt", new WordSplitBolt(), 2).shuffleGrouping("text-file-spout");  builder.setBolt("word-count-bolt", new WordCountBolt(), 2).fieldsGrouping("word-split-bolt", new Fields("word"));  Config conf = new Config();  conf.setDebug(true);  if (args != null && args.length > 0) {  conf.setNumWorkers(3);  StormSubmitter.submitTopologyWithProgressBar(args[0], conf, builder.createTopology());  } else {  conf.setMaxTaskParallelism(3);  LocalCluster cluster = new LocalCluster();  cluster.submitTopology("word-count-topology", conf, builder.createTopology());  Thread.sleep(10000);  cluster.shutdown();  }  }  } |
| --- |

##### pom.xml (one of many version attempted)

| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.example</groupId>  <artifactId>word-count-topology</artifactId>  <version>1.0-SNAPSHOT</version>  <packaging>jar</packaging>  <name>word-count-topology</name>  <url>http://maven.apache.org</url>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <maven.compiler.source>1.8</maven.compiler.source>  <maven.compiler.target>1.8</maven.compiler.target>  </properties>  <dependencies>  <!-- Apache Storm dependencies -->  <dependency>  <groupId>org.apache.storm</groupId>  <artifactId>storm-core</artifactId>  <version>2.4.0</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>org.apache.storm</groupId>  <artifactId>storm-kafka-client</artifactId>  <version>2.2.0</version>  </dependency>  <!-- Logging framework -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>1.7.32</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>1.7.32</version>  </dependency>  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.17</version>  </dependency>    <!-- JUnit -->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.13.1</version>  <scope>test</scope>  </dependency>  </dependencies>  <build>  <plugins>  <!-- Maven Compiler Plugin -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.1</version>  <configuration>  <source>1.8</source>  <target>1.8</target>  </configuration>  </plugin>  <!-- Maven Shade Plugin -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-shade-plugin</artifactId>  <version>3.2.4</version>  <executions>  <execution>  <phase>package</phase>  <goals>  <goal>shade</goal>  </goals>  <configuration>  <transformers>  <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">  <mainClass>com.example.WordCountTopology</mainClass>  </transformer>  </transformers>  </configuration>  </execution>  </executions>  </plugin>  <!-- Maven Jar Plugin -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-jar-plugin</artifactId>  <version>3.2.0</version>  <configuration>  <archive>  <manifest>  <mainClass>com.example.WordCountTopology</mainClass>  </manifest>  </archive>  </configuration>  </plugin>  </plugins>  </build>  </project> |
| --- |

Section Word Count: 808

Total Word Count: 2575

Github Repository Link : <https://github.com/KaviCCT/Storage-Solutions-for-Big-Data-CA2>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# References

Bedeley, R. (2014). BIG DATA OPPORTUNITIES AND CHALLENGES: THE CASE OF BANKING INDUSTRY. SAIS 2014 Proceedings. [online] Available at: <https://aisel.aisnet.org/sais2014/2/>.

CloudDuggu. (n.d.). Apache Storm Project. [online]

Available at: <https://www.cloudduggu.com/storm/project-01/>.

db-engines.com. (n.d.). Hive vs. MySQL Comparison. [online]

Available at: <https://db-engines.com/en/system/Hive%3BMySQL>.

flink.apache.org. (n.d.). Architecture. [online]

Available at: <https://flink.apache.org/what-is-flink/flink-architecture/>.

Gölzer, P., Cato, P., Amberg, M. and Amberg (2015). Data Processing Requirements of Industry 4.0 -Use Cases for Big Data Applications. [online]

Available at: <https://core.ac.uk/download/pdf/301365475.pdf>.

Gupta, H. (2023). Insights from paper — Storm @Twitter. [online] Medium.

Available at: <https://hemantkgupta.medium.com/insights-from-paper-storm-twitter-cc2dc1b4c70a>.

Gutenberg.org. (2021).

Available at: <https://www.gutenberg.org/cache/epub/98/pg98.txt>.

Iqbal, Dr.M. (n.d.). Tutorial 3 Hadoop Streaming.

Iqbal, Dr.M. (n.d.). Tutorial 9 Apache Hive Step 1: Download and Untar Hive.

Iqbal, Dr.M. (n.d.). Tutorial 10 Apache Storm.

Iqbal, M. (n.d.). CCT Storage Solutions for Big Data Higher Diploma in Science in AI Applications CCT College Dublin Apache Hive Week 10.

Iqbal, M. (n.d.). CCT Storage Solutions for Big Data Higher Diploma in Science in AI Applications CCT College Dublin Apache Storm Week 11.

Iqbal, M. (n.d.). Storage Solutions for Big Data H Dip in AI Applications HCI CCT College Dublin Introduction to Storage Solutions for Big Data (SSBD) Week 1.

Iqbal, M. (n.d.). Storage Solutions for Big Data Higher Diploma in Science in AI Applications CCT College Dublin Apache Flink Week 9.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

nightlies.apache.org. (n.d.). Flink Architecture. [online]

Available at: <https://nightlies.apache.org/flink/flink-docs-release-1.19/docs/concepts/flink-architecture/>.

online.visual-paradigm.com. (n.d.). What is a Concept Map Diagram? [online]

Available at: <https://online.visual-paradigm.com/knowledge/concept-map-diagram/what-is-concept-map-diagram/>.

Products, D. (2023). Introduction To Stateful Stream Processing. [online] Medium.

Available at: <https://medium.com/@dataproducts/introduction-to-stateful-stream-processing-49447863931a>.

Quora. (n.d.). What are the main differences between MySQL and Hive syntax? [online]

Available at: <https://www.quora.com/What-are-the-main-differences-between-MySQL-and-Hive-syntax>.

Rathbone, M. (2015). Apache Hive vs MySQL - What are the key differences? [online] Matthew Rathbone’s Blog.

Available at: <https://blog.matthewrathbone.com/2015/12/08/hive-vs-mysql.html>.

Reddy, J. (2019). Hive Architecture in Depth. [online] Plumbers Of Data Science.

Available at: <https://medium.com/plumbersofdatascience/hive-architecture-in-depth-ba44e8946cbc>.

storm.apache.org. (n.d.). Distributed RPC. [online]

Available at: <https://storm.apache.org/releases/current/Distributed-RPC.html>.

storm.apache.org. (n.d.). Tutorial. [online]

Available at: <https://storm.apache.org/releases/current/Tutorial.html>.

www.guru99.com. (n.d.). What is Hive? Architecture & Modes. [online]

Available at: <https://www.guru99.com/introduction-hive.html>.

www.tutorialspoint.com. (n.d.). Apache Flink - Quick Guide. [online]

Available at: <https://www.tutorialspoint.com/apache_flink/apache_flink_quick_guide.htm>.

www.tutorialspoint.com. (n.d.). Apache Storm - Introduction - Tutorialspoint. [online]

Available at: <https://www.tutorialspoint.com/apache_storm/apache_storm_introduction.htm>.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**