

TABLE OF CONTENTS

Sr. No.	Objective of the Practical	Page No.
1	Recording types of data and various file formats. Identifying data sources. Handling data traditionally to start with at a small scale.	3
2	Handling of data in a multidimensional view-point using the concept of data warehouses. To be aware of a form of analytics also known as data mining.	7
3	Configuring and experimenting Hadoop single node cluster. Start/stop script and monitoring.	15
4	Interfacing to Hadoop Distributed File System (HDFS) using command line interface (CLI) and graphical user interface (GUI) on a stand alone Apache Hadoop instance running locally. Storage as a Service or STaaS is cloud storage which can be architecture using hadoop-like frameworks which works upon horizontal scaling.	23
5	Learn concepts of mapreduce java programming using practicing Word Count by Apache. (Calculating number copies for mentioned book categories.)	29
6	Exploring NoSQL storage and processing solutions. Utilize known mapreduce framework supported by mongoDB (a NoSQL DB Store) + java script (functional programming), to demonstrate Word Count. (Finding mutual friends)	37
7	Establish SSH User Equivalence and configure to experiment with Hadoop multiple node's cluster. Start/stop script and monitoring. Know the backbone of cloud platform.	48
8	Solve Word Count program requirement using Apache Spark. Apply the learning on a custom problem at hand (i.e. Man of the series award).	54
9	Perform machine learning using Apache Mahout tool and explore possibilities for supervised and unsupervised learning. (Recommendation system)	59
10	Final step towards data analytics, "Reporting/Presentation"; Utilize reports/visualizations/charts/integration tools to be able to understand the findings. (Identifying trends using word/tag cloud).	64

Big Data & Analytics

Lab#1. Data Handling using traditional(relational) databases

1. **Aim :** Recording types of data and various file formats. Identifying data sources. Handling data traditionally to start with at a small scale.
2. **Objective :** Learn various types of digital data and how to deal with them. Develop an attitude towards observing the surroundings and be digitally aware of the generation of data and handling required for digitization/computerization/automation of the processes involved. Focus here is to compile and apply data handling techniques used in various domains traditionally. This shall sharpen your existing skills and prepare you before getting onto data analytics and data science journey.

3. Description:

Note that based on a high-level viewpoint of data, it can be categorized in multiple different ways. i.e. Based on the structureness, sources, formats, storage and usage, etc.

Based upon the structureness property can be categorized as below:

- Structured Data
- Semi-structured Data
- Unstructured Data

Based upon the sources:

- Human generated data
- Machine generated data

Based on the file formats(binary/textual):

- Spreadsheet (.odt .xls .xlsx)
- CSV/TSV
- XML
- JSON
- Configuration File (.ini)
- Properties file (.properties)
- MS Access (.mdb .accdb)
- Oracle Database Dump (.dmp)
- YAML (https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html)
- BSON (<https://docs.mongodb.com/manual/reference/bson-types/>) (<http://bsonspec.org/>), etc.

Based on the storage and usage:

- Data at rest
- Data in motion

4. Methodology:

Structured data is managed easily now by various techniques like indexing, hashing, searching, undo/redo logs for recovery, encryption, etc and primarily in tables/relations of databases.

Semi-structured data does not conform to any data model but has some structures like hierarchy, tags, etc. The actual data within meta data might have some randomness. Parsers help up to a certain level to process.

Unstructured data primarily has no formally defined structure, though it can be imposed via efforts but it may not be enough to process the newest different requirement. Very rarely it's also possible that the structure is so complex to understand or decode for the end agency that it assumes that data is unstructured, provided various sources of data are dumped or the structure is not publicized.

Be aware of terms OLTP (Online Transaction Processing), OLAP (Online Analytical Processing) and RTAP (Real Time Analytical Processing)

Note that Big Data characteristics various V as below also describes data.

- Velocity
- Volume
- Value
- Variety
- Veracity, etc.

5. Example:

Identifying Data Sources is vital to master data handling technicality.

To start, let's work Scenario-based:

Imagine “You are at the university library. You see a few students browsing through the library catalog on a kiosk. You observe the librarians busy at work issuing and returning books. You see a few students fill up the feedback form on the services offered by the library. Quite a few students are learning using e-learning content. “.

Think for a while on the different types of data that are being generated in this scenario. You shall be able to observe data generation, flow and types be it structured or unstructured or inbetween.

6. Implementation Notes:

Explore GUI and CLI both options for the selected tool. Note down / Think of challenges involved.

What is UIMA?

Unstructured Information Management applications are software systems that analyze large volumes of unstructured information in order to discover knowledge that is relevant to an end user. An example UIM application might ingest plain text and identify entities, such as persons, places, organizations; or relations, such as works-for or located-at.

Below list shows the tools widely available for various data handling purposes. Knowing your options ahead of time prepares you to complete the actual task at hand effectively(quality) and efficiently(quantity/timeline).

Associated Learning Tools:

Google Data Studio
Google Rich Card
Google Knowledge Graph
CAS-Content Addressable Storage
SQL Server Migration Assistant
DMA - Data Migration Assistant
mongoImport/mongoExport/Oracle Sqldr
python panda excel to csv
csvkit/sqlizer - convert files into databases
xml.etree.elementtree

datapine
datatables.net
Studio3T Migration Tool
php son_encode
json.stringify
.bacpac file windows, etc.

7. Exercise:

Know that roots of Big Data and Analytics still are with storage solutions primarily by relational databases followed by data warehouses and data mining. The listed exercises enable one to be able to understand data types, capacity, limitation, etc while handling data in the case of structured data to get started.

1. Given the spreadsheet file convert it into a csv
2. Import a csv into MySQL database table
3. Write a computer program to read records from the database and generate data file/s.
 - a. XML
 - b. JSON
4. Import XML/JSON file into another database/table. I.e MS Access. Oracle, etc.
5. Export database dump for data migration/archival
6. Validate/Map data types across different database systems when migrating from one to another

8. References:

Textbook. Big Data and Analytics – Seema Acharya and Subhashini Chellappan – Wiley India

Topics: Types of Digital Data

Textbook. Data Mining Concepts and Techniques. - Jiawei Han, Micheline Kamber, Jian Pei - Morgan Kaufmann. Web resources of Book: Data Mining <https://hanj.cs.illinois.edu/bk3/>

Topics: Introduction and Getting to Know Your Data

<https://www.phpmyadmin.net/>

<https://livesql.oracle.com/> (create free oracle account first)

CE720 : Big Data & Analytics

Lab#2. Data Handling using warehouse

1. **Aim :** Handling of data in a multidimensional view-point using the concept of data warehouses. To be aware of a form of analytics also known as data mining.
2. **Objective :** Learn various types of digital data and how to deal with them. The requirements of aggregation and reporting for decision making during the early analytics phase, lead the development of data warehousing and data mining. Experiment multidimensional mechanism of statistics data storage and handling.

3. Description:

The primary level of analytics started with aggregation functionalities. i.e find maximum, minimum, average, mode, mean, median, etc from mathematics. The flat/relational storage supports transactions and is able to perform aggregation like primitives. The need for a multidimensional view point of summarized data and being able to perform operations on the same runtime like slicing, dicing, drill down, etc is accomplished via data warehousing and data mining. Many legacy products are having their own establishment and it is important to be aware of the same. Know that Data Warehouse is in itself a separate data storage in parallel with raw data storage. It is built and maintained (nightly/weekly/monthly/yearly) as per requirements.

This again builds the base of data handling of modern times which are a lot advanced and sustainable to meet Millennials needs.

During the development, the following are types and purposes of analytics observed. It also has effect of Industrial Revolutions i.e. Industry 1.0. 2.0, etc, community wise.

HindSight to Insight to ForeSight

- What happened? (Descriptive Analytics)
- Why did it happen? (Diagnostic Analytics)
- What will happen? (Predictive Analytics)
- How can we make it happen? (Prescriptive Analytics)
- The most recent and advanced is "COGNITIVE ANALYTICS", using AI, ML, DL and such advanced technologies learning and improving to infer better and better and act autonomously.

4. Methodology:

Understand the multi-dimensionality and utilize the concept of star schema to create relations within Oracle/MySQL like relational databases. Understand clearly that either from internal tables or external data sources i.e. .csv,.xml, web service responses, the establishment of data warehousing is carried out after running some or other aggregation like queries.

5. Example:

Dimensional modeling for business process of “STUDENT ADMISSION FOR GRADUATION, AFTER 12TH SCIENCE”

STEPS TO BE FOLLOWED FOR DIMENSIONAL MODELING:

- 1) Identify the Business Process
- 2) Identify the Grain
- 3) Identify the Dimensions
- 4) Identify the Facts

- Identifying the business process is the process of determining the business process that the data warehouse represents.

Here the data warehouse is going to represent number of admission with average of percentage across branches (Eng-CS, Eng-EC, Med-MMBS, Med-Dental), across type of students (Male-Open, Male-SC, Female-Open, Female-SC), across period (2004-1RS, 2004-2RS, 2005-1RS, 2005-2RS), Where RS means reshuffling.

- Identifying the Grain is the process of identifying the level of detail of the fact table.

Here the grain is branch wise students with their average percentage per kind of student, per period of admission.

- Identifying the dimensions for the business process of interest is the process of representing characteristics such as who, what, where, when, how of a measurement.

In this case, the measurement is admissions with average percentage.

Characteristics:

Who is going to be admitted?

Male/Female student having cast OPEN/SC.

When is going to be admitted?

During 1st Reshuffling of year 2004

During 2nd Reshuffling of year 2004

During 1st Reshuffling of year 2005

During 2nd Reshuffling of year 2005.

Where is going to be admitted?

To the category of Engineering/Medical within stream CS/EC or MBBS/Dental accordingly.

Hence, here three dimension tables as per following:

Dimension 1: JD_BRANCH_ADM Table (WHERE)

Represents different branch types available for admission.

```
create table JD_Branch_ADM (Branch_ID integer not null,Branch_CATEGORY
varchar2(15),Branch_STREAM varchar2(20),
PRIMARY KEY(Branch_ID));
```

Dimension 2: JD_FELLOW_ADM Table (WHO)

Represents different fellow types admitted.

```
create table JD_Fellow_ADM (Fellow_ID integer not null,Fellow_GENDER
char,Fellow_CATEGORY varchar2(5),
PRIMARY KEY(Fellow_ID));
```

Dimension 3: JD_PERIOD_ADM Table (WHEN)

Represents different period of admission

```
create table JD_Period_ADM (Period_ID integer not null, Period_TYPE.
integer,Period_YEAR varchar2(5),
PRIMARY KEY(Duration_ID));
```

- Identifying the facts is the process of determining measurements for the business process of interest. (what)

The facts must be confirmed to the grain defined previously.

Here number of admission with average of percentage are the measurements and hence are the facts which are numeric.

Students admitted, with their average percentage, of the same fellow type and branch during a specific period.

FACT TABLE SCHEMA: JF_ADM


```
create table JF_ADM (Branch_ID integer references JD_Branch_ADM,
Fellow_ID integer references JD_Fellow_ADM,
Period_ID integer references JD_Period_ADM,
Fact_No_Adm integer,
Fact_Avg_Per integer,primary key (Branch_Id,Fellow_Id,Duration_Id));
```

Notice that the Fact table contains the composite primary key consisting of each dimension table's simple - non-composite primary key.

6. Implementation Notes:


Dimension 1: JD_BRANCH_ADM

(WHERE)

Field Name	Type	Primary key
Branch_ID	Integer	
Branch_CATEGORY	Varchar2	
Branch_STREAM	Varchar2	


Dimension 2: JD_FELLOW_ADM

(WHO)

Field Name	Type	Primary key
Fellow_ID	Integer	
Fellow_GENDER	Varchar2	
Fellow_CATEGORY	Varchar2	


Dimension 3: JD_PERIOD_ADM Table

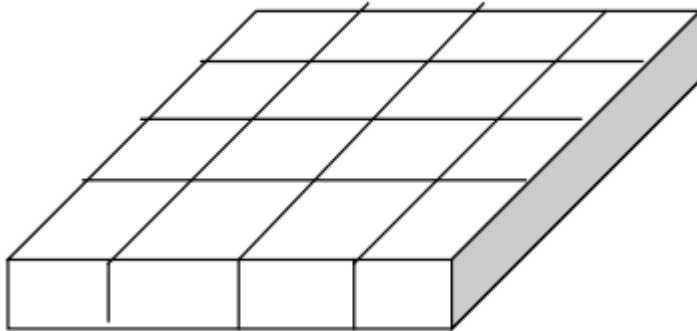
(WHEN)

Field Name	Type	Primary key
Period_ID	Integer	
Period_TYPE	Varchar2	
Period_YEAR	Varchar2	

Facts:

FACT TABLE SCHEMA: JF_ADM

Field Name	Type	Primary key
Branch_ID	Integer	
Fellow_ID	Integer	
Period_ID	Integer	
Fact_No_Adm	Integer	
Fact_Avg_Per	Integer	

OPERATIONS**Operation#1) SLICE**

-> Find and display number of students only those Male & Open type of fellow with their percentage average for each period and for each type of branch separately.

```
-> Select Fact_No_Adm, Fact_Avg_Per
from JD_Branch_ADM B,JD_Fellow_ADM F,JD_Period_ADM D,JF_ADM FACT
where (
B.Branch_Id = FACT.Branch_Id and
F.Fellow_Id = FACT.Fellow_Id and
D.Period_Id = FACT.Period_Id and
F.Fellow_Gender = 'M' and
F.Fellow_Category = 'OPEN');
```

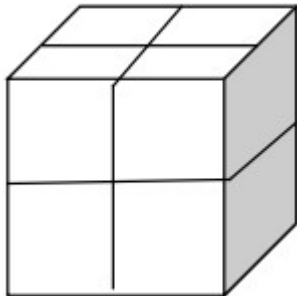
Resulting slice:

FACT_NO_ADM FACT_AVG_PER

1000	60
1100	70
1200	80
1300	90
2600	60
2700	70
2800	80
2900	90
4200	60
4300	70
4400	80
4500	90
5800	60
5900	70
6000	80
6100	90

16 rows selected.

Operation#2. DICE



- Find number of students with their average percentage which satisfies following:
 - Belong to Branch Category of Engineering
 - Male
 - Have reshuffled during year 2004

```
Select Fact_No_Adm, Fact_Avg_Per
from JD_Branch_ADM B,JD_Fellow_ADM F,JD_Period_ADM D,JF_ADM FACT
where (
B.Branch_Id = FACT.Branch_Id and
F.Fellow_Id = FACT.Fellow_Id and
D.Period_Id = FACT.Period_Id and
(F.Fellow_Id = 1 or F.Fellow_Id = 2) and
(B.Branch_Id = 1 or B.Branch_Id = 2) and
(D.Period_Id = 1 or D.Period_Id = 2));
```

FACT_NO_ADM	FACT_AVG_PER
3100	70
3000	60
1500	70
1400	60
2700	70
2600	60
1100	70
1000	60

8 rows selected.

Other Way of finding the same dice:

```
Select Fact_No_Adm, Fact_Avg_Per
from JD_Branch_ADM B,JD_Fellow_ADM F,JD_Period_ADM D,JF_ADM FACT
where (
B.Branch_Id = FACT.Branch_Id and
F.Fellow_Id = FACT.Fellow_Id and
```

D.Period_Id = FACT.Period_Id and
 F.Fellow_Gender = 'M' and
 B.Branch_Category = 'ENG' and
 D.Period_Year = 2004);

FACT_NO_ADM FACT_AVG_PER

1000	60
1100	70
1400	60
1500	70
2600	60
2700	70
3000	60
3100	70

8 rows selected.

Operation#3 Roll UP:

Find number of total students admitted to branch category of Engineering.

```

Select sum(Fact_No_Adm)
from JD_Branch_ADM B,JD_Fellow_ADM F,JD_Period_ADM D,JF_ADM FACT
where
B.Branch_Id = FACT.Branch_Id and
F.Fellow_Id = FACT.Fellow_Id and
D.Period_Id = FACT.Period_Id and
B.Branch_Category = 'ENG'
group by B.Branch_Category;
```

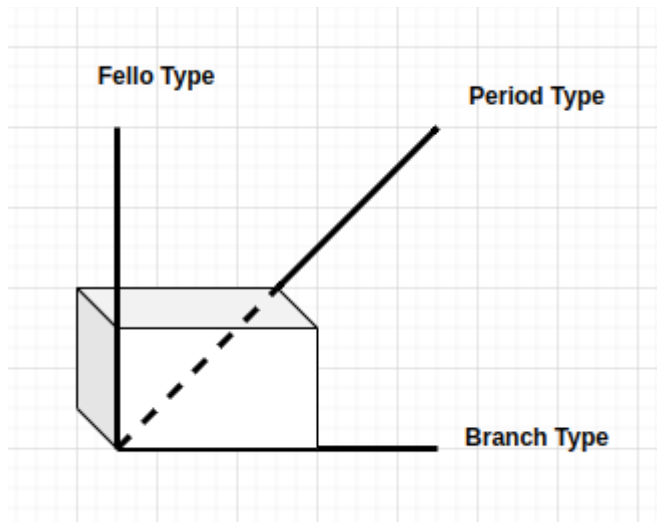
Result of the query:

SUM(FACT_NO_ADM)

 81600

7. Exercise:

1. Represent Data Cube OLAP - Data Warehouse either for the described business process or any other of your business domain interest having 3 dimensionals.



2. Perform operations on the built warehouse. Know that the raw data/source/transactions A is assumed to be a separate data source altogether where actual transactions are recorded. The data warehouse B is populated with entries (dummy) out of results by aggregation query on the transaction data source/raw table/s A.
3. Extend the business process to admit 4th and 5th dimensional data and idealize the changes/additions required to achieve more dimensionality in Data Warehouse.
4. Learn and explore tools/api to be able to generate pdf-like reports containing tabular/visualizations dashboards, etc.

8. References:

Textbook. Data Mining Concepts and Techniques. - Jiawei Han, Michelin Kamber, Jian Pei - Morgan Kaufmann. Web resources of Book: Data Mining <https://hanj.cs.illinois.edu/bk3/>
Topics: Data Warehousing and Online Analytical Processing

CE720 : Big Data & Analytics

Lab#3 Hadoop single node cluster

1. **Aim :** Configuring and experimenting Hadoop single node cluster. Start/stop script and monitoring.
2. **Objective :** Students will learn HADOOP and its components. To be able to practice Apache Hadoop, students are required to understand the architecture and perform setup on a computer to establish a single node cluster. Note that Hadoop is a distributed storage and processing system though.
3. **Description:** Hadoop Distributed File System (HDFS) was developed using distributed file system design. It is to run on commodity hardware. Unlike other distributed systems, HDFS is highly fault-tolerant and scalable. HDFS holds a very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available for parallel processing. Apache HDFS or Hadoop Distributed File System is a block-structured file system where each file is divided into blocks of a predetermined size. These blocks are stored across a cluster of one or several machines. Apache Hadoop HDFS Architecture follows a Master/Slave Architecture, where a cluster comprises a single NameNode (Master node) and all the other nodes are DataNodes (Worker nodes).

NameNode

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. NameNode is the master node in the Apache Hadoop HDFS Architecture that maintains and manages the blocks present on the DataNodes (slave nodes). In other words Namenode maintains metadata. NameNode is a very highly available server that manages the File System Namespace and controls access to files by clients. The HDFS architecture is built in such a way that the user data never resides on the NameNode. The data resides on DataNodes only.

Functions of Namenode:

- It is the master daemon that maintains and manages the DataNodes (worker nodes)
- It records the metadata of all the files stored in the cluster, e.g. The location of blocks stored, the size of the files, permissions, hierarchy, etc.

There are two files associated with the metadata:

- FsImage: It contains the complete state of the file system namespace since the start of the NameNode.
- EditLogs: It contains all the recent modifications made to the file system with respect to the most recent FsImage.

- It records each change that takes place to the file system metadata. For example, if a file is deleted in HDFS, the NameNode will immediately record this in the EditLog.
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live.
- It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.
- The NameNode is also responsible to take care of the replication factor of all the blocks.
- In case of the DataNode failure, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

DataNode:

DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is a commodity hardware, that is, a non-expensive system which is not of high quality or high-availability. The DataNode is a block server that stores the data in the local file ext3 or ext4.

Functions of DataNode:

- These are worker daemons or process which runs on each worker machine.
- The actual data is stored on here in DataNodes.
- The DataNodes perform the low-level read and write requests from the file system's clients.
- They send heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds.

Secondary NameNode/CheckpointNode:

Apart from these two daemons, there is a third daemon or a process called Secondary NameNode. The

Secondary NameNode works concurrently with the primary NameNode as a helper daemon. And don't be confused about the Secondary NameNode being a backup NameNode because it is not.

Functions of Secondary NameNode:

- The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes it into the hard disk or the file system.
- It is responsible for combining the EditLogs with FsImage from the NameNode.
- It downloads the EditLogs from the NameNode at regular intervals and applies to FsImage.

The

new FsImage is copied back to the NameNode, which is used whenever the NameNode is started the next time. Hence, Secondary NameNode performs regular checkpoints in HDFS. Therefore, it is also called CheckpointNode.

4. Methodology:

Here, the host operating system is assumed to have a Linux flavor. Steps mentioned herewith can have similar theoretical aspects on Windows based platforms too.

4.1 Check out the contents of .bashrc file using the following command.

Note that there are various environment variables required for hadoop to function properly on a given system. It starts with where the hadoop binary/software is installed. In our case

/opt/hadoop is the location where hadoop downloaded tarball is extracted. Hence /opt/hadoop becomes HADOOP_HOME

```
[hadoop@hadoop-clone Desktop]$ cat ~/.bashrc
```

Following output can be seen (on an instance where hadoop setup is already done).

```
# .bashrc
# Source global definitions
if [ -f /etc/bashrc ]; then
. /etc/bashrc
fi
# User specific aliases and functions
export JAVA_HOME=/usr/java/jdk1.8.0_131
export HADOOP_HOME=/opt/hadoop
export HADOOP_PREFIX=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar

export
PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

Note that JAVA_HOME can be configured as part of the internal file HADOOP_HOME/conf/hadoop-env.sh as well.

4.2 Also check the contents of PATH environment variable using following command:

```
[hadoop@hadoop-clone Desktop]$ echo $PATH
```

Following output can be seen (as hadoop setup is already done).

```
/opt/ns-allinone-2.35/bin:/opt/ns-allinone-2.35/tcl8.5.10/unix:/opt/ns-allinone-2.35/tk8.5.10/
unix:/usr/
lib64/qt-
3.3/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/java/jdk1.8.0_131/bin:/
opt/hadoop
/bin:/opt/hadoop/sbin:/opt/hbase/bin:/home/hadoop/bin:/usr/java/jdk1.8.0_131/bin:/opt/
hadoop/bin:/op
t/hadoop/sbin::
```

Also, to be able to run hadoop related commands/scripts from any location, HADOOP_HOME/bin and HADOOP_HOME/sbin have been added to linux PATH.

4.3 Configure Hadoop

Using any editor, view the contents of core-site.xml and hdfs-site.xml. To understand the contents of these files, open <https://hadoop.apache.org> website. Refer Configuration section located in the bottom-

left corner of the site to check various properties of various configuration files . Explore various properties. When hadoop is started, it sees the contents of these files and loads various configurations. Note that any change in configuration requires restart of server for its effect. It's not live load.

The screenshot shows the Apache Hadoop 3.2.1 documentation page for SingleCluster.html. The browser address bar shows the URL: hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html. The page content includes:

- Left Sidebar:**
 - Apps:** Aliyun OSS, Amazon S3, Azure Blob Storage, Azure Data Lake Storage, OpenStack Swift
 - Auth:** Overview, Examples, Configuration, Building
 - Tools:** Hadoop Streaming, Hadoop Archives, Hadoop Archive Logs, DistCp, GridMix, Ruven, Resource Estimator, Service, Scheduler Load Simulator, Hadoop Benchmarking
 - Reference:** Changelog and Release Notes, Java API docs, Unix Shell API, Metrics
 - Configuration:** core-default.xml, hdfs-default.xml, hdfs-rbf-default.xml, mapred-default.xml, yarn-default.xml, Deprecated Properties
- Main Content:**
 - etc/hadoop/core-site.xml:**

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```
 - etc/hadoop/hdfs-site.xml:**

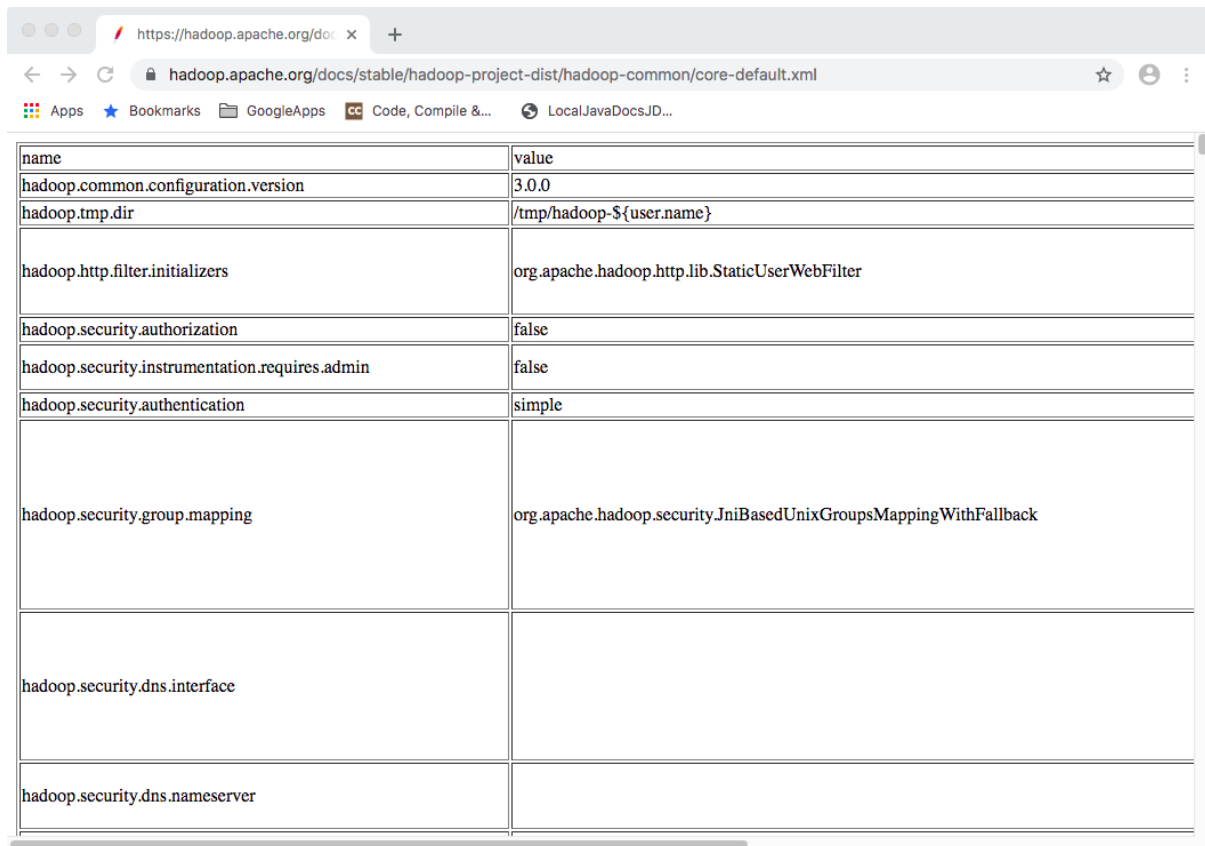
```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```
 - Setup passphraseless ssh:**

Now check that you can ssh to the localhost without a passphrase:

```
$ ssh localhost
```

If you cannot ssh to localhost without a passphrase, execute the following commands:

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```
 - Execution:** (Section header visible)



name	value
hadoop.common.configuration.version	3.0.0
hadoop.tmp.dir	/tmp/hadoop-\${user.name}
hadoop.http.filter.initializers	org.apache.hadoop.http.lib.StaticUserWebFilter
hadoop.security.authorization	false
hadoop.security.instrumentation.requires.admin	false
hadoop.security.authentication	simple
hadoop.security.group.mapping	org.apache.hadoop.security.JniBasedUnixGroupsMappingWithFallback
hadoop.security.dns.interface	
hadoop.security.dns.nameserver	

Basically, the `fs.defaultFS` property value indicates the uri of the server. Later when we will run a cluster of multiple computers, this property will be marked via ip of a unique designated namenode computer's ip instead of just value localhost as seen rightnow.

By default, temporary data, metadata and actual data blocks of virtual HDFS gets stored into host operating system's native file system. The below properties specify that location.

Note that below overrides the default location of temporary data from `/tmp/hadoop-${user.name}` to within hadoop setup location `HADOOP_HOME/tmp`

Configuration File: `HADOOP_HOME/etc/hadoop/core-site.xml`

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/opt/hadoop/tmp</value>
</property>
```

Note that below specifies the new location for storage of namenode and datanode onto Host operating system native file system location.

Configuration File: `HADOOP_HOME/etc/hadoop/hdfs-site.xml`

```
<property>
  <name>dfs.namenode.name.dir</name>
  <value>/opt/hadoop/namenode_storage</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>/opt/hadoop/datanode_storage</value>
```

</property>

4.4 After doing any modification, for the first time namenode has to be formatted using following command:

```
[hadoop@hadoop-clone hadoop]$ hadoop namenode -format
```

4.5 Once setup is ready, start hdfs using following command:

```
[hadoop@hadoop-clone Desktop]$ start-dfs.sh
```

4.6 After executing previous command, it's effect can be seen using following command.

```
[hadoop@hadoop-clone Desktop]$ jps
```

This command will display output similar to below:

3426 DataNode

3766 Jps

3302 NameNode

3642 SecondaryNameNode

This actually shows that these many java processes are currently running on the machine.

Hadoop monitoring and HDFS Web GUI can be accessed by the URL <http://localhost:50070/>. Explore

this Web GUI.

4.7 stop hadoop using following command:

```
[hadoop@hadoop-clone hadoop]$ stop-dfs.sh
```

5. Example:

Navigation Links (Default):

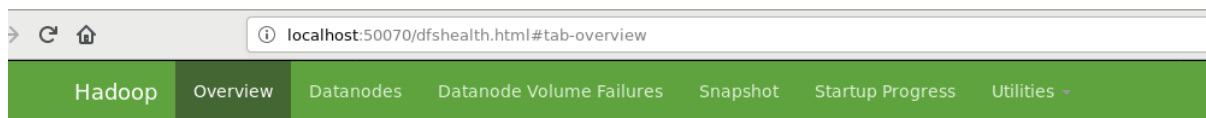
NameNode:

<http://localhost:50070/dfshealth.html#tab-overview>

DataNode

<http://localhost:50075/datanode.html>

<http://localhost:8088/cluster/nodes>



Overview 'hadoop-clone:9000' (active)

Started:	Wed May 08 17:07:16 +0530 2019
Version:	2.8.0, r91f2b7a13d1e97be65db92ddabc627cc29ac0009
Compiled:	Fri Mar 17 09:42:00 +0530 2017 by jdu from branch-2.8.0
Cluster ID:	CID-56d0833f-1cb4-4d20-87e8-e0d35fe4310b
Block Pool ID:	BP-1591587460-127.0.0.1-1557315413831

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 29.77 MB of 58.25 MB Heap Memory. Max Heap Memory is 966.69 MB.

6. Implementation Notes:

HDFS can be deployed on a broad spectrum of machines that support Java. Hadoop single node cluster is very much pseudo distributed mode where on the same one computer all the components of hadoop are configured to run. Though one can run several DataNodes on a single machine, in the practical world, these DataNodes are spread across various machines.

To install and configure Apache Hadoop, first download the binary from the official website. Extract it to a native file system location i.e. /opt/hadoop

The linux os user i.e. hadoop needs to establish a known host profile and user equivalence/passwordless ssh to be able to let inter process communication occur systemwide by hadoop components. You may verify the same via ssh to localhost for chosen username i.e. hadoop and it shall go through without asking password, if know host profile entry is already added and user equivalence using ssh-keygen and ssh-copy-id is done successfully.

7. Exercise:

1. Download and install hadoop on your workstation or seek alternatives. Monitor the same using version specific navigation links.
2. Locate the log directory and refer for troubleshooting.
3. Change the URL of HDFS Web UI to localhost:51234 instead of localhost:50070 and the access the web UI using new URL. [Hint: refer configuration files] [Know that traditionally configuration based servers need to be restarted after any server configuration change.]

4. Undo changes done in task1.
5. Find out what is the current location of namenode and datanode storage directories.

Normally whenever cleanup is required, tmp folder is one to be emptied up (not to confuse with formatting file system) and remove logs. Hence, it is advised data and metadata better be outside tmp.

Do the needful to have namenode and datanode directories in /opt/hadoop, if present within tmp.

6. Analyze the output of following command:
[hadoop@hadoop-clone hadoop]\$ ps -ef |grep NameNode
[hadoop@hadoop-clone hadoop]\$ ps -ef |grep DataNode
7. Find out the version of hadoop installed on your system using CLI.

8. References:

Textbook. Big Data and Analytics – Seema Acharya and Subhashini Chellappan – Wiley India

Topics: Introduction to Hadoop.

Apache Hadoop official website <https://hadoop.apache.org/>

https://www.tutorialspoint.com/hadoop/hadoop_hdfs_overview.htm

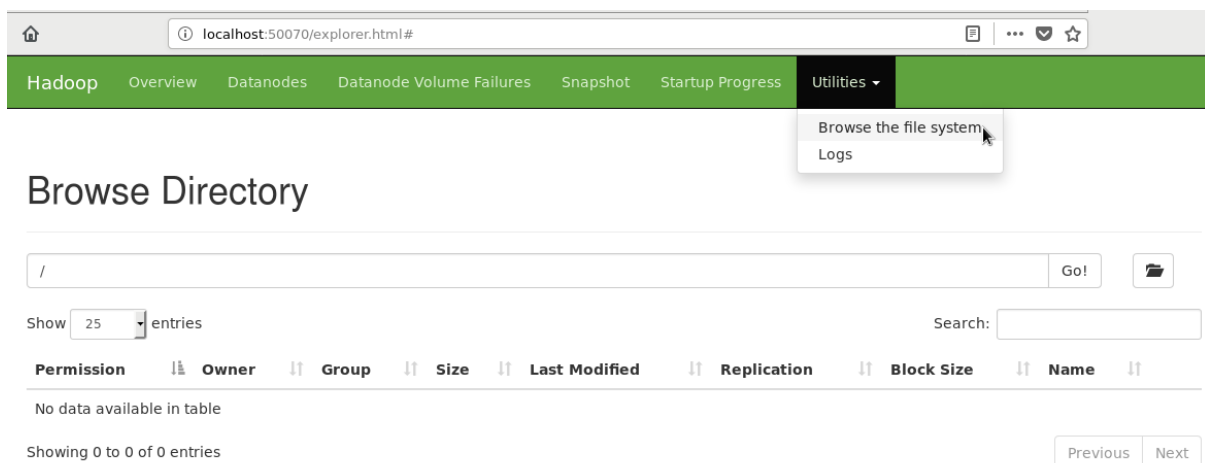
<https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/>

CE720 : Big Data & Analytics

Lab#4 Interfacing to Hadoop Distributed File System (HDFS).

1. **Aim :** Interfacing to Hadoop Distributed File System (HDFS) using command line interface (CLI) and graphical user interface (GUI) on a stand alone Apache Hadoop instance running locally. Storage as a Service or STaaS is cloud storage which can be architecture using hadoop-like frameworks which works upon horizontal scaling.
2. **Objective :** Students will learn HADOOP and its components. Commands to perform read and write directory and file structures and data content also with properties of the same. Understanding the abstraction imposed by HDFS being a guest file system on top of the host file system provided by the underlying operating system itself.
3. **Description:**
The setup and architecture of Apache Hadoop as understood from “Configuring and experimenting Hadoop single node cluster. Start/stop script and monitoring.” has to be utilized now for data storage in the form of a distributed file system within HDFS.
4. **Methodology:**
Hadoop monitoring and HDFS Web GUI can be accessed by the URL <http://localhost:50070/>. Explore this Web GUI.

Go to 'Browse the file system' under the 'Utilities' section. If no data is dumped in HDFS yet, it should display the same message.



Documentation/Help

Seek help using command “`hadoop fs -help ls`”

This will provide help about ls command and options. Just to see documentation about commands you don't need to have hdfs daemons started. Of course to practice, you will first need to have hdfs started and verified that the NameNode, DataNode and SecondaryNameNode daemons are up and running.

5. Example:

Sample command usage:

Create a directory named 'user' and create another directory named 'hadoop' inside 'user' directory:

To verify

```
[hadoop@hadoop-clone hadoop]$ hdfs dfs -ls /user/hadoop
```

To create complete path

```
[hadoop@hadoop-clone hadoop]$ hdfs dfs -mkdir -p /user/hadoop
```

Browse the file system again to see these directories. Both CLI and GUI ways.

Know that the name of the user here assumes 'hadoop' and hence the above location will be considered HDFS home location of user named 'hadoop' which is /user/hadoop.

You may use the relative path to hdfs user home as a third approach to referring files or directories. Ideally hadoop administrator shall do this job of creating home for all users.

The user home concept is similar to Linux user home i.e. /home/<nameofuser> but the location here in HDFS is like /user/<nameofuser> that's the only difference.

6. Implementation Notes:

The File System (FS) shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS) as well as other file systems that Hadoop supports, such as Local FS, HFTP FS, S3 FS, and others. The FS shell is invoked by 'hdfs dfs <args>'.

All FS shell commands take path URIs as arguments. The URI format is scheme://authority/path. For HDFS the scheme is hdfs, and for the Local FS the scheme is file. The scheme and authority are optional. If not specified, the default scheme specified in the configuration is used. An HDFS file or directory such as /parent/child can be specified as hdfs://namenodehost/parent/child or simply as /parent/child (given that your configuration is set to point to hdfs://namenodehost). Most of the commands in FS shell behave like corresponding Unix commands.

6.1. mkdir

Usage: hdfs dfs -mkdir [-p] <paths>

Takes path uri's as argument and creates directories.

Options:

The -p option behavior is much like Unix mkdir -p, creating parent directories along the path.

Example:

```
hdfs dfs -mkdir /user/hadoop/dir1 /user/hadoop/dir2
hdfs dfs -mkdir hdfs://nn1.example.com/user/hadoop/dir
hdfs://nn2.example.com/user/hadoop/dir
```

Exit Code:

Returns 0 on success and -1 on error.

6.2. ls

Usage: `hdfs dfs -ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] <args>`

Options:

- C: Display the paths of files and directories only.
- d: Directories are listed as plain files.
- h: Format file sizes in a human-readable fashion (eg 64.0m instead of 67108864).
- q: Print ? instead of non-printable characters.
- R: Recursively list subdirectories encountered.
- t: Sort output by modification time (most recent first).
- S: Sort output by file size.
- r: Reverse the sort order.
- u: Use access time rather than modification time for display and sorting.

For a file `ls` returns stat on the file with the following format:

```
permissions
number_of_replicas userid groupid filesize modification_date
modification_time filename
```

For a directory it returns a list of its direct children as in Unix. A directory is listed as:

```
permissions
userid groupid modification_date modification_time dirname
```

Files within a directory are order by filename by default.

Example:

```
hdfs dfs -ls /user/hadoop/file1
```

Exit Code:

Returns 0 on success and -1 on error.

6.3. put

Usage: `hdfs dfs -put [-f] [-p] [-l] [-d] [- | <localsrc1> ..]. <dst>`

Copy single src, or multiple srcs from local file system to the destination file system. Also reads input from stdin and writes to destination file system if the source is set to “-”

Copying fails if the file already exists, unless the -f flag is given.

Options:

- p : Preserves access and modification times, ownership and the permissions. (assuming the permissions can be propagated across filesystems)
- f : Overwrites the destination if it already exists.
- l : Allow DataNode to lazily persist the file to disk, Forces a replication factor of 1. This flag will result in reduced durability. Use with care.
- d : Skip creation of temporary file with the suffix `._COPYING_`.

Examples:

```
hdfs dfs -put localfile /user/hadoop/hadoopfile
hdfs dfs -put -f localfile1 localfile2 /user/hadoop/hadoopdir
hdfs dfs -put -d localfile hdfs://nn.example.com/hadoop/hadoopfile
hdfs dfs -put - hdfs://nn.example.com/hadoop/hadoopfile Reads the input from stdin.
```

Exit Code:

Returns 0 on success and -1 on error.

6.4. rm

Usage: `hdfs dfs -rm [-f] [-r] [-R] [-skipTrash] [-safely] URI [URI ...]`

Delete files specified as args.

If trash is enabled, file system instead moves the deleted file to a trash directory. Currently, the trash feature is disabled by default. User can enable trash by setting a value greater than zero for parameter fs.trash.interval (in core-site.xml).

Options:

The -f option will not display a diagnostic message or modify the exit status to reflect an error if the file does not exist.

The -R option deletes the directory and any content under it recursively.

The -r option is equivalent to -R.

The -skipTrash option will bypass trash, if enabled, and delete the specified file(s) immediately. This can be useful when it is necessary to delete files from an over-quota directory.

The -safely option will require safety confirmation before deleting directory with total number of files greater than `hadoop.shell.delete.limit.num.files` (in core-site.xml, default: 100). It can be used with -skipTrash to prevent accidental deletion of large directories. Delay is expected when walking over large directory recursively to count the number of files to be deleted before the confirmation.

Example:

```
hdfs dfs -rm hdfs://nn.example.com/file /user/hadoop/emptydir
```

Exit Code:

Returns 0 on success and -1 on error.

6.5. rmdir

Usage: `hadoop fs -rmdir [--ignore-fail-on-non-empty] URI [URI ...]`

Delete a directory.

Options:

--ignore-fail-on-non-empty: When using wildcards, do not fail if a directory still contains files.

Example:

```
hdfs dfs -rmdir /user/hadoop/emptydir
```

6.6. copyFromLocal

Usage: `hdfs dfs -copyFromLocal <localsrc> URI`

Similar to the `fs -put` command, except that the source is restricted to a local file reference.

Options:

-p : Preserves access and modification times, ownership and the permissions. (assuming the permissions can be propagated across filesystems)

-f : Overwrites the destination if it already exists.

-l : Allow DataNode to lazily persist the file to disk, Forces a replication factor of 1. This flag will result in reduced durability. Use with care.

-d : Skip creation of temporary file with the suffix `._COPYING_`.

6.7. copyToLocal

Usage: `hdfs dfs -copyToLocal [-ignorecrc] [-crc] URI <localdst>`

Similar to `get` command, except that the destination is restricted to a local file reference.

6.8. cp

Usage: `hdfs dfs -cp [-f] [-p | -p[topax]] URI [URI ...] <dest>`

Copy files from source to destination. This command allows multiple sources as well in which case the destination must be a directory.

'raw.*' namespace extended attributes are preserved if (1) the source and destination filesystems support them (HDFS only), and (2) all source and destination pathnames are in the `/.reserved/raw` hierarchy. Determination of whether `raw.*` namespace xattrs are preserved independent of the -p (preserve) flag.

Options:

The -f option will overwrite the destination if it already exists.

The -p option will preserve file attributes [topx] (timestamps, ownership, permission, ACL, XAttr). If -p is specified with no arg, then preserves timestamps, ownership, permission. If -pa is specified, then preserves permission also because ACL is a super-set of permission.

Determination of whether raw namespace extended attributes are preserved is independent of the -p flag.

Example:

```
hdfs dfs -cp /user/hadoop/file1 /user/hadoop/file2
```

```
hdfs dfs -cp /user/hadoop/file1 /user/hadoop/file2 /user/hadoop/dir
```

Exit Code:

Returns 0 on success and -1 on error.

6.9. get

Usage: hdfs dfs -get [-ignorecrc] [-crc] [-p] [-f] <src> <localdst>

Copy files to the local file system. Files that fail the CRC check may be copied with the -ignorecrc option. Files and CRCs may be copied using the -crc option.

Example:

```
hdfs dfs -get /user/hadoop/file localfile
```

```
hdfs dfs -get hdfs://nn.example.com/user/hadoop/file localfile
```

Exit Code:

Returns 0 on success and -1 on error.

Options:

-p : Preserves access and modification times, ownership and the permissions. (assuming the permissions can be propagated across filesystems)

-f : Overwrites the destination if it already exists.

-ignorecrc : Skip CRC checks on the file(s) downloaded.

-crc: write CRC checksums for the files downloaded.

6.10. moveFromLocal

Usage: hdfs dfs -moveFromLocal <localsrc> <dst>

Similar to put command, except that the source localsrc is deleted after it's copied.

11. mv

Usage: hdfs dfs -mv URI [URI ...] <dest>

Moves files from source to destination. This command allows multiple sources as well in which case the destination needs to be a directory. Moving files across file systems is not permitted.

Example:

```
hdfs dfs -mv /user/hadoop/file1 /user/hadoop/file2
```

```
hdfs dfs -mv hdfs://nn.example.com/file1 hdfs://nn.example.com/file2
```

```
hdfs://nn.example.com/file3 hdfs://nn.example.com/dir1
```

Exit Code:

Returns 0 on success and -1 on error.

7. Exercise:

1. Create/Delete a directory/file named 'Test' from HDFS web UI in '/user/hadoop'. It should give an error message. Read this message and solve the error.
2. Create a text file in the local file system and put the same file in HDFS using appropriate command.
3. Display the contents of this file of HDFS on the terminal.

4. Download this file of HDFS on a local machine and change its contents. Upload this modified file again on HDFS and verify its contents. Also make sure that this file should be deleted from the local machine automatically.
5. Find out a CLI way to display all file system commands supported by hdfs. Record the list.

p.s. “hadoop fs ...” and “hdfs dfs ...” can be used interchangeably but note that the “hadoop fs ...” is deprecated. Read documentation as per version to follow.

8. References:

Textbook. Big Data and Analytics – Seema Acharya and Subhashini Chellappan – Wiley India

Topics: Introduction to Hadoop.

Apache Hadoop official website <https://hadoop.apache.org/>

https://www.tutorialspoint.com/hadoop/hadoop_hdfs_overview.htm

<https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/>

<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html>

CE720 : Big Data & Analytics

Lab#5 Learn concepts of mapreduce programming.

1. **Aim :** Learn concepts of mapreduce programming using practicing Word Count. Apply the learning on a custom problem at hand. (i.e. Calculating number copies for mentioned book categories.)
2. **Objective :** Students will learn concepts of mapreduce programming. Word Count - A type of hello world program with respect to Map Reduce distributed processing framework utilizing data primarily from Hadoop Distributed File System (HDFS). Learn about the framework itself and identify the phases involved in which can be exploited for solutions about Big Data Analytics.

3. Description:

Word Count is a type of Hello World of MapReduce Framework - Java. Note that it does not just count the total number of words but bundles unique words into separate bins and gives unique words with their respective frequencies/counts line by line.

Whenever any programming language needs to be explored, how to compile and run a program is described by so called “Hello World” program.

With respect to MapReduce Java programming with Hadoop Distributed File System support, “Word Count” is considered the simplest example to understand the paradigm and also how to compile, execute and verify results, etc.

Input files containing plain text need to be stored into HDFS location. This HDFS location will be provided to the program as a command line argument - input. Also, the second command line argument -input will be the location of HDFS where the resulting files - output have to be stored. Know that this output folder at the mentioned location will be created by the framework itself. If it exists already, the program will terminate before storing output.

Java program jar based map-reduce job will be submitted to ResourceManager everytime wordcount is expected for set of input files.

Framework reads line by line file text and provides a single line to mapper function as value in the argument for that function call. Understand that for every line mapper function is called. Actually mapper function is because of first class citizen characteristics and locality of reference optimization very efficiently in use once per line even. Ideally lines are expected to be longer enough and to need complex logic due characteristics of Big Data.

Mapper processes and generates set of key-value pairs containing word with frequency 1 for every words in the current line for WordCount program. Don't confuse with key-value and having unique key requirement. Every pair has its own independent existence. There can be

multiple pairs having same key, in word count it is the word happening multiple times in the line possibly.

Mapper function of WordCount performs tokenization and generates plain simple pairs where every input word is associated as result of mapper `<word,1>` `<key,value>` showing just the fact that the current word has occurred 1 time just now irrespective of anywhere else in the line or files all over data.

Here, if for a different logic requirement this '1' can be something else also. Instead of tokenization of value (line) any other logic can also be performed based on requirement.

After the execution of mapper, the framework behind the scene shuffles all resulting pairs and sorts to generate `<key, list of values>` kind of pair where all values for the same key is added into values list. In word count program entries into values list will be number of 1's for a certain word, i.e. 'hello' has occurred 5 times overall then `<'hello',[1,1,1,1,1]>`.

For every such `<key,list of values>` reduce function is called which actually sums up all entries into the values list to produce total occurrences of that word. And finally the `<word, count>` pair will be dumped to the result file by multiple calls to mapper for various words.

4. Methodology:

4.1 To be able to run map-reduce, we need to specify that Yet Another Resource Negotiator (YARN) is chosen for below property value.

HADOOP_HOME/etc/hadoop/mapred-site.xml

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

4.2 Development of WordCount Program with Map-Reduce. The program and its binary stay in local file system only. The data utilized for processing will be provided to/from HDFS in the form of locations as command line arguments to the java class.

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).

The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.

The reduce task is always performed after the map job.

WordCount Program: WordCount.java

```

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount
{
    public static class TokenizerMapper extends Mapper < LongWritable,
        Text, Text, IntWritable >
    {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map (LongWritable key, Text value, Context context) throws
IOException, InterruptedException
        {
            StringTokenizer tokenizer = new StringTokenizer(value.toString());
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class IntSumReducer extends Reducer < Text, IntWritable,
        Text, IntWritable >
    {
        private IntWritable result = new IntWritable();
        public void reduce(Text key, Iterable < IntWritable > values, Context context)
throws
IOException, InterruptedException
        {
            int sum = 0;
            for (IntWritable val: values) {
                sum += val.get();
            }
            result.set (sum);
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception
    {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
    }
}

```

```

        job.      setMapperClass(TokenizerMapper.class);
        job.      setCombinerClass(IntSumReducer.class);
        job.      setReducerClass(IntSumReducer.class);
        job.      setOutputKeyClass(Text.class);
        job.      setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System. exit  (job.waitForCompletion(true) ? 0 : 1);
    }
}

```

Explanation

The program consist of 3 classes:

1. Map (i.e. TokenizerMapper) class which extends public class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the Map function.
2. Reduce (i.e. IntSumReducer) class which extends public class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the Reduce function.
3. Driver (i.e. WordCount) class (public static void main() - the entry point)

Mapper Class

Mapper <LongWritable, Text, Text, IntWritable>

The data types provided here are Hadoop specific data types designed for operational efficiency suited for massive parallel and lightning fast read-write operations.

All these data types are based out of java data types itself, for example LongWritable is the equivalent for long in java, IntWritable for int and Text for String.

When we use it as Mapper<LongWritable, Text, Text, IntWritable> , it refers to the data type of input and output key value pairs specific to the mapper or rather the map method, ie Mapper<Input Key Type, Input Value Type, Output Key Type, Output Value Type>.

In our example the input to a mapper is a single line, so this Text (one input line) forms the input value.

The input key would be a long value assigned by default based on the position of Text in input file.

Our output from the mapper is of the format “Word, 1” hence the data type of our output key value pair is <Text(String), IntWritable(int)>

The functionality of the map method is as follows

1. Create a IntWritable variable ‘one’ with value as 1
2. Convert the input line in Text type to a String
3. Use a StringTokenizer to split the line into words
4. Iterate through each word and a form key value pairs as
 - a. Assign each word from the tokenizer(of String type) to a Text ‘word’

b. Form key value pairs for each word as <word,one> and write it to the context

Reducer Class

Reducer<Text, IntWritable, Text, IntWritable>

The first two refers to data type of Input Key and Value to the reducer and the last two refers to data type of output key and value.

Our mapper emits output as <apple,1> , <grapes,1> , <apple,1> etc. This is the input for reducer so here the data types of key and value in java would be String and int, the equivalent in Hadoop would be Text and IntWritable.

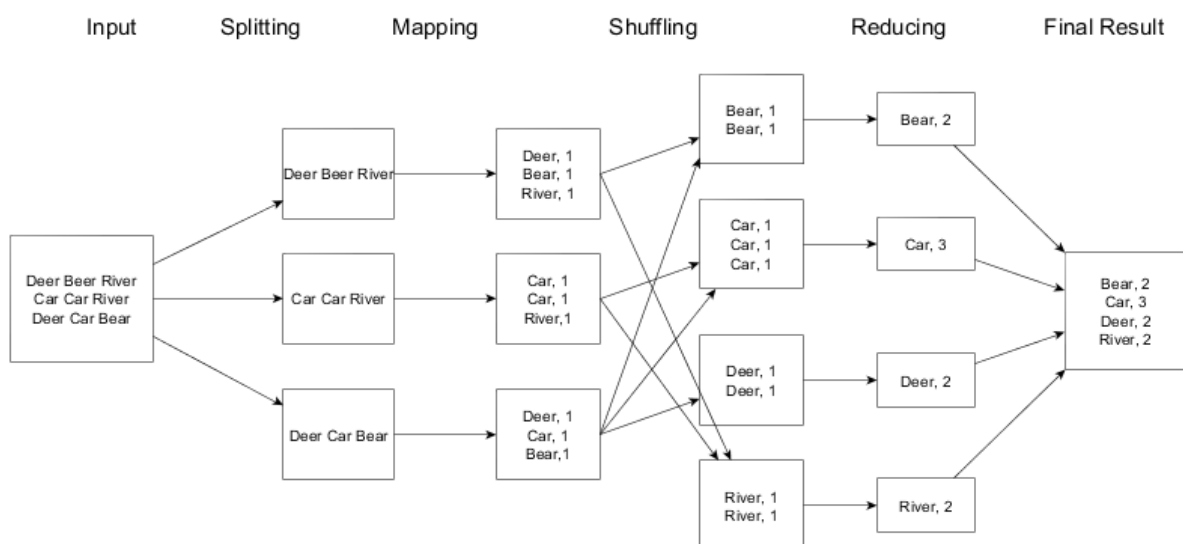
Also we get the output as <word, no of occurrences> so the data type of output Key Value would be <Text, IntWritable>

The functionality of the reduce method is as follows

1. Initialize a variable 'sum' as 0
2. Iterate through all the values with respect to a key and sum up all of them
3. Write the final result (number of occurrence) for each word (key) to the context.

5. Example:

Note that the below diagram (source: <http://stephanie-w.github.io/brainscribble/hadoop-word-count-explained.html>) depicts a visual representation of understanding how map-reduce phases works. It has more detail to it and few phases are optional and configured to work differently.



6. Implementation Notes:

Start hadoop entities once configuration is completed both for HDFS and Map-reduce/YARN.


```
start-dfs.sh
start-yarn.sh
mapred --daemon start historyserver
```

Run jps and verify the components' running status.

Assuming environment variables are set appropriately:

```
Compile WordCount.java and create a jar:
$ hadoop com.sun.tools.javac.Main WordCount.java
$ jar cf wc.jar WordCount*.class
```

As hadoop provides an option to generate and provide hadoop classpath as output runtime. We can utilize following command also on linux like platforms to substitute the output and handle classpath.

```
$javac -cp `hadoop classpath` WordCount.java
$jar cf wc.jar WordCount*.class
```

Run the application:

Assuming that:

/projects/wordcount/input - input directory in HDFS contains input text-files

Create using below commands :

```
hadoop dfs -mkdir -p /projects/wordcount/input
```

Put some text files within input folder to validate the result later in output folder:

```
hadoop dfs -put *.txt /projects/wordcount/input
hadoop dfs -chmod -R 775 /projects
```

or create dummy files as below:

```
echo "Hello World of Hadoop" > file01.txt
echo "Hello Again. Learning MapReduce programming with Hadoop. Hadoop enables
horizontal scalling." > file02.txt
```

```
hadoop dfs -put file*.txt /projects/wordcount/input
```

/projects/wordcount/output - output directory in HDFS. (output directory will be created by program itself). If you are running multiple times, either provide different name of output directory or remove using below command.

```
hadoop dfs -rm -r /projects/wordcount/output
```

Run following command to execute program

```
$ hadoop jar wc.jar WordCount /projects/wordcount/input /projects/wordcount/output
```

Notice the progress on CLI as well as using GUI (<http://localhost:8088/cluster>).

When completed the MapReduce task, run the following command to see the output:
\$ `hadoop dfs -cat /projects/wordcount/output/part-r-00000`

Alternatively, you may download the data files from web resources, CLI:

```
wget https://storage.googleapis.com/download.tensorflow.org/data/illiad/cowper.txt
wget https://storage.googleapis.com/download.tensorflow.org/data/illiad/derby.txt
wget https://storage.googleapis.com/download.tensorflow.org/data/illiad/butler.txt
```

7. Exercise:

1. Compile Word Count java program and be able to provide HDFS input directory containing text files, execute and verify output files from the resulting folder within HDFS.
2. Download sample data files from kaggle and run the WordCount for these input files.
3. Identify differences between Map-reduce library versions.
4. To practice MapReduce programming in Hadoop using Java Coding.

Step. 1.1: Create a file named 'pages.txt' in local file system. Store line by line content as shown below. Each line data represents number of pages of a sample book.

350
250
150
450
120

Step. 1.2: Put the file from local file system to hdfs with folder named 'input'.
Confirm the presence of above data.

Step. 1.3: Write map and reduce functions to split the books into the following two categories:

- (a) Big Books
- (b) Small Books

Books which have more than 300 pages should be in the big book category.
Books which have less than 300 pages should be in the small book category.

Count the number of books in each category.

Store the output as follow as result file within hdfs 'output' folder.

Book Category	Count of the books
"Big Books"	2
"Small Books"	3

8. References:

Textbook. Big Data and Analytics – Seema Acharya and Subhashini Chellappan – Wiley India
Topics: Introduction to MAPREDUCE programming

<https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

<http://stephanie-w.github.io/brainscribble/hadoop-word-count-explained.html>

CE720 : Big Data & Analytics

Lab#6. Exploring NoSQL storage and processing solutions

1. **Aim :** Exploring NoSQL storage and processing solutions

2. **Objective :** Students must understand big data, sources of big data, characteristics of big data storage and processing. Utilize mongoDB crud operations and known mapreduce framework supported by mongoDB (a NoSQL DB Store)+java script (functional programming), to demonstrate Word Count. Apply the learning on a custom problem at hand (i.e. Finding mutual friends). Hands on other NoSQL data stores such as HBase, Cassandra, etc.

3. **Description:**

NoSQL – Not Only SQL database/s have evolved because of the requirement of handling schema less/free data; There is more to it than structured query language when it comes to big data handling.

MongoDB

MongoDB is one of the NoSQL database management systems. It is a document oriented database. Learn the terminology of components in mongoDB.

Following table shows basics about CRUD operations on Students named collection with fields <StudRollNo,StudName,Grade,Hobbies,DOJ>.

Operation	RDBMS	MONGODB
Create	Create table Students (StudRollNo varchar2(20), StudName varchar2(50), Grade varchar2(5), Hobbies varchar2(50), DOJ date)	db.createCollection("Students")
INSERT	Insert into Students (StudRollNo, StudName, Grade, Hobbies, DOJ) values ('S101',' ','VII',...)	<pre> db.Students.insert({ _id:1, StudRollNo:'S101', StudName:'', Grade:'VII', Hobbies:'Net Surfing', DOJ:'10-OCT-2012', }) </pre>

Lab Manual: Big Data & Analytics

Update	<div> Update Students Set Hobbies='Ice Hockey' Where StudRollNo='S101' </div> <pre> db.Students.update({StudRollNo:'S101'}, { \$set: { Hobbies:'Ice Hockey' } },) </pre> <div> Update Students Set Hobbies='Ice Hockey' </div> <pre> db.Students.update({}, { \$set: { Hobbies:'Ice Hockey' } } </pre>
--------	--

	<pre> }, {multi:true},) </pre>
Delete	<div> Delete from Students where StudRollNo='S101' </div> <pre> db.Students.remove({ StudRollNo:'S101' }) </pre> <div> Delete from Students </div> <pre> db.Students.remove() </pre>

Select	<p>Select * from Students</p> <p>db.Students.find({}).pretty()</p>
	<p>Select * from Students</p> <p>Where StudRollNo='S101'</p> <p>db.Students.find({StudRollNo:'S101'})</p>
	<p>Select StudRollNo, StudName, Hobbies</p> <p>from Students</p> <p>db.Students.find(</p> <p>{},</p> <p>{StudRollNo:1,</p> <p>StudName:1,</p> <p>Hobbies:1,</p> <p>_id:0}</p> <p>)</p>
	<p>Select StudRollNo, StudName, Hobbies</p> <p>from Students</p> <p>Where Grade='VII' and Hobbies='Ice Hockey'</p> <p>db.Students.find({StudRollNo:'S101',</p> <p>Hobbies:'Ice Hockey'},</p> <p>{StudRollNo:1,</p> <p>StudName:1,</p> <p>Hobbies:1,</p> <p>_id:0}</p> <p>)</p>
	<p>Cross check syntax</p> <p>db.Students.find({\$or:[{StudRollNo:'S101',</p> <p>Hobbies:'Ice Hockey'}]}),</p> <p>{StudRollNo:1,</p> <p>StudName:1,</p> <p>Hobbies:1,</p> <p>_id:0}</p>
	<p>Select StudRollNo, StudName, Hobbies</p> <p>from Students</p> <p>Where Grade='VII' or Hobbies='Ice Hockey'</p>

	<p>)</p>
	<p>Select * from Students</p> <p>Where StudName like 'S%'</p> <p>db.Students.find({StudName:/^S/})</p>

4. Methodology:

The 'mongod' utility is a daemon process responsible to act as a server. The internal provided database client is 'mongo' utility. Optionally you may use other clients like Compass, Jupyter Notebook (imongo-kernel), and programmatically to interact with MongoDB.

MongoDB is not part of Hadoop EcoSystem.

Below steps are done as part of MongoDB trial:

Run MongoDB server using 'mongod' command; Either keep the window open or may start in the background using 'mongod &'. The letter 'd' in mongod can also be assumed to stand for 'daemon' - a service. On certain computers where mongod is configured as a service, you may run the command “systemctl status mongodb.service” to know the status and “systemctl start mongodb.service”.

Know that another utility 'mongo' as shown below is actually inbuilt client provided.

```
> show dbs;
admin    0.000GB
config   0.000GB
local    0.000GB
mng      0.000GB
> db;
test
> use demodb;
switched to db demodb
> db.createCollection('democollection');
{ "ok" : 1 }
> db.democollection.save({__id:1,name:"James"});
WriteResult({ "nInserted" : 1 })
> db.democollection.find({});
{ "_id" : ObjectId("5cecdc7c78529695a945ccab"), "__id" : 1, "name" : "James" }
> db.democollection.drop();
true
> db.dropDatabase();
{ "dropped" : "demodb", "ok" : 1 }
> use test;
switched to db test
> show dbs;
admin    0.000GB
config   0.000GB
local    0.000GB
mng      0.000GB
> quit();
```

There are many other mongodb clients available in the market. 'Compass' named GUI based client can also be installed and used as required. JuPyter can be configured with mongodb kernel and can be used in interactive mode with MongoDB.

Know that by default MongoDB is installed with no username and password to be used listening on 27017 port localhost. Authentication and authorization (security) of data is not in scope of this exercise here.

5. Example:

Python JuPyter to connect to MongoDBHandsOn

```
from pymongo import MongoClient
```

```

from pprint import pprint
connection = MongoClient("mongodb://localhost:27017/admin")
db = connection.test
db.inventory.insert_one(
    {"_id":1,
    "item": "canvas",
    "qty": 100,
    "tags": ["cotton"],
    "size": {"h": 28, "w": 35.5, "uom": "cm"}})

```

```

db.inventory.insert_one(
    {"_id":2,
    "item": "painting",
    "qty": 50,
    "tags": ["nature"],
    "size": {"h": 100, "w": 55, "uom": "cm"}})
cursor = db.inventory.find({})

```

```

for inventory in cursor:
    pprint(inventory)

```

Subdocument key order matters in a few of these examples so we have
to use bson.son.SON instead of a Python dict.

```

from bson.son import SON
db.inventory.insert_many([
    {"_id":3,
    "item": "journal",
    "qty": 25,
    "size": SON([("h", 14), ("w", 21), ("uom", "cm")]),
    "status": "A"},
    {"_id":4,
    "item": "notebook",
    "qty": 50,
    "size": SON([("h", 8.5), ("w", 11), ("uom", "in")]),
    "status": "A"},
    {"_id":5,
    "item": "paper",
    "qty": 100,
    "size": SON([("h", 8.5), ("w", 11), ("uom", "in")]),
    "status": "D"},
    {"_id":6,
    "item": "planner",
    "qty": 75,
    "size": SON([("h", 22.85), ("w", 30), ("uom", "cm")]),
    "status": "D"},
    {"_id":7,
    "item": "postcard",
    "qty": 45,
    "size": SON([("h", 10), ("w", 15.25), ("uom", "cm")]),

```



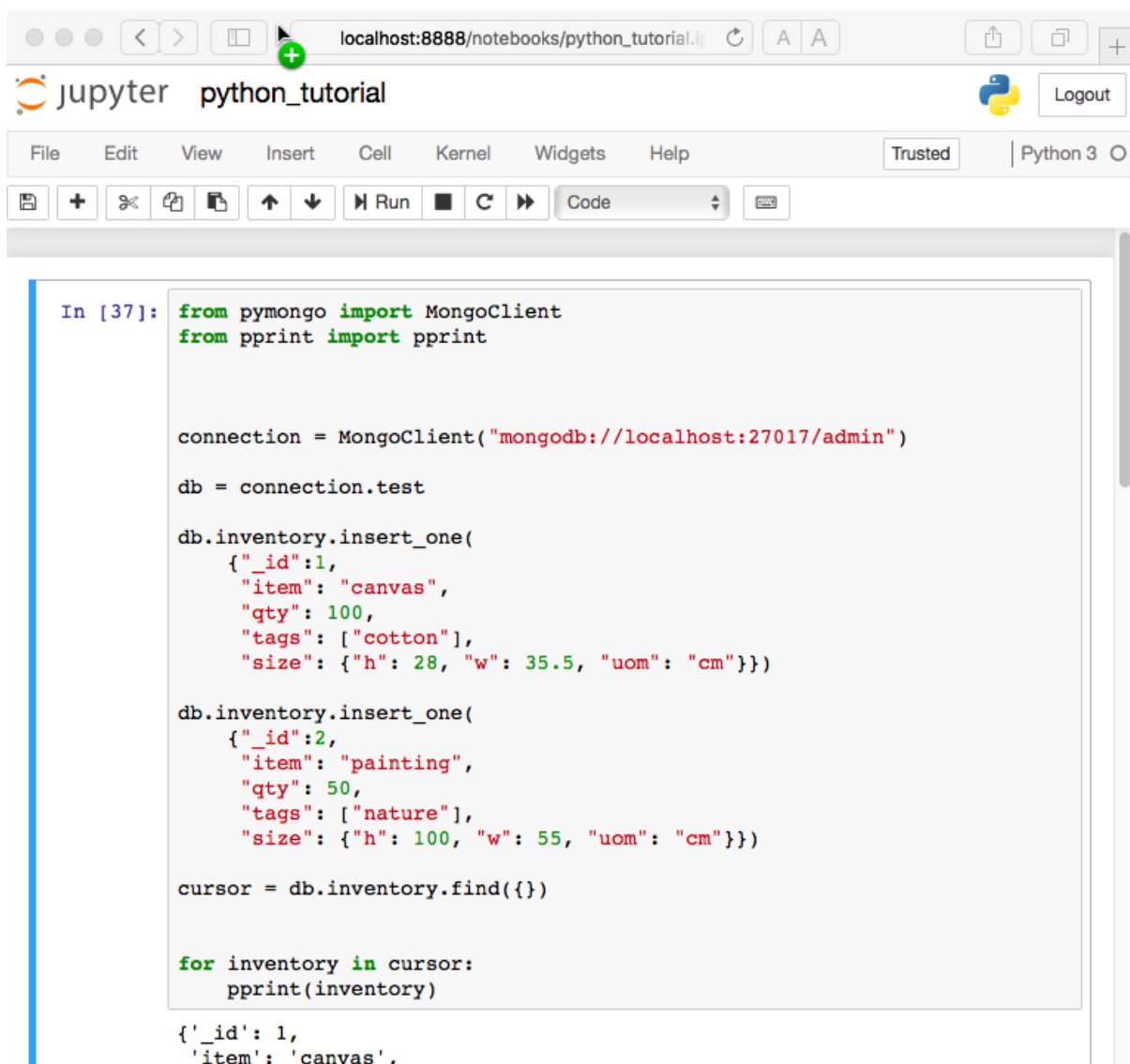
```
"status": "A"}})
```

```
cursor = db.inventory.find({"status": "D"})
```

```
cursor = db.inventory.find(
{"size": SON([("h", 14), ("w", 21), ("uom", "cm")])})
```

```
cursor = db.inventory.find({"size.uom": "in"})
```

```
from pprint import pprint
for inventory in cursor:
    pprint(inventory)
```



The screenshot shows a Jupyter Notebook titled 'python_tutorial' running on 'localhost:8888/notebooks/python_tutorial.ipynb'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and code execution. The code in the notebook is as follows:

```
In [37]: from pymongo import MongoClient
from pprint import pprint

connection = MongoClient("mongodb://localhost:27017/admin")
db = connection.test
db.inventory.insert_one(
    { "_id": 1,
      "item": "canvas",
      "qty": 100,
      "tags": ["cotton"],
      "size": {"h": 28, "w": 35.5, "uom": "cm"}})

db.inventory.insert_one(
    { "_id": 2,
      "item": "painting",
      "qty": 50,
      "tags": ["nature"],
      "size": {"h": 100, "w": 55, "uom": "cm"}})

cursor = db.inventory.find({})

for inventory in cursor:
    pprint(inventory)

{'_id': 1,
 'item': 'canvas',
```

The screenshot shows a Jupyter Notebook titled 'demo2' running on 'localhost:8888/notebooks/demo2.ipynb'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and code execution. The code cell contains the following input:

```
In [4]: db.inventory.({}).pretty()
```

The output of the code cell is displayed as follows:

```
Out[4]: {
  "_id" : 1,
  "item" : "canvas",
  "qty" : 100,
  "tags" : [
    "cotton"
  ],
  "size" : {
    "h" : 28,
    "w" : 35.5,
    "uom" : "cm"
  }
},
{
  "_id" : 2,
  "item" : "painting",
  "qty" : 50,
  "tags" : [
    "nature"
  ],
  "size" : {
    "h" : 100,
    "w" : 55,
    "uom" : "cm"
  }
}
```

Following code snippet with respect mongoDB client run shall help you know how map-reduce is achieved in mongoDB itself using javascript api:

```
var map = function(){ if ( this.pages > 300) emit ('Big Books',1); else emit ('Small Books',1); }
var reduce = function(key, values){ return Array.sum(values);}
db.books.mapReduce(map, reduce, {out:"Book_Result",query:{}});
```

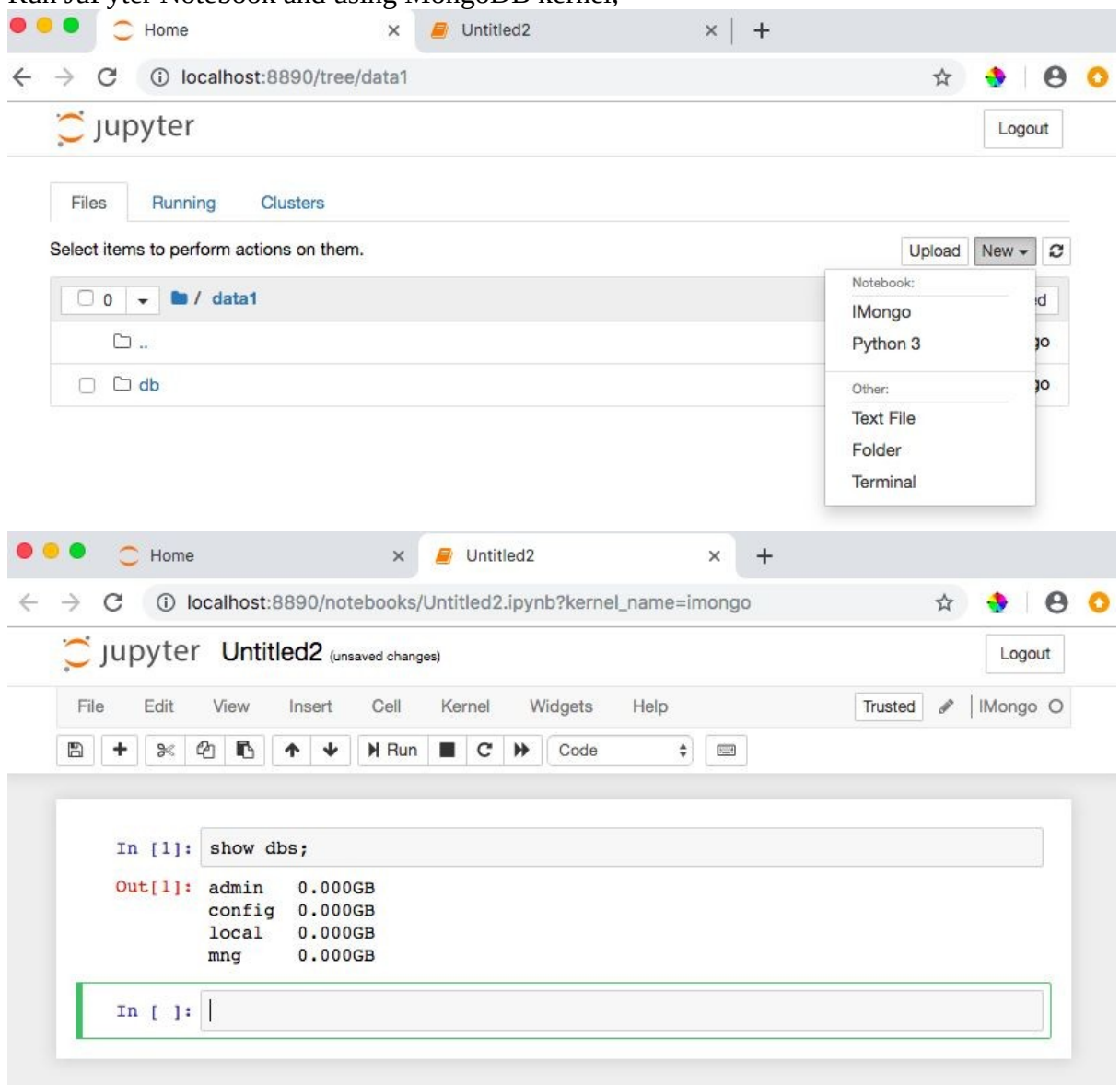
6. Implementation Notes:

Note that it is up to the data science team that how to store and process data. The technician has to first choose a data store based on the characteristics of data. Then he may choose how to interact with the data for processing purposes. i.e. Querying, scripting, programming, etc. Here, mongoDB query language (MQL) is used for querying. Also, a python api example has been provided for demonstration of connectivity between python and mongoDB. Also, note

that general statistics/aggregation is possible. You can take analytics forward via map-reduce like supporting solutions.

7. Exercise:

- Using any other client than mongo, i.e. compass/Jupyter notebook do the same exercise shown in the screenshot by mongo client above.
Run JuPyter Notebook and using MongoDB kernel,



Note that the folder location to create notebook file is irrelevant to mongoDB. Otherwise /data/db is the location where mongoDB is default configured to store metadata and database.

- Write the insert method to store the following document in MongoDB and verify. May practice crud operations.

Name : "Stephen More"
Address : {

```
"City" : "Banglore",
"Street" : "Electronics City",
"Affiliation" : "XYZ Ltd"
}
Hobbies : Chess, Lawn Tennis, Base Ball
```

3. To practice MapReduce programming in MongoDB.
Step. 3.1: Insert 5 documents as shown below in the collection named 'books'.

```
> db.books.find({}).pretty();
{
  "_id" : 1,
  "Category" : "Machine Learning",
  "BookName" : "Machine Learning for Hackers",
  "Author" : "Drew Conway",
  "qty" : 25,
  "price" : 400,
  "rol" : 30,
  "pages" : 350
}
{
  "_id" : 2,
  "Category" : "Business Intelligence",
  "BookName" : "Fundamentals of Business Analytics",
  "Author" : "Seema Acharya",
  "qty" : 55,
  "price" : 500,
  "rol" : 30,
  "pages" : 250
}
{
  "_id" : 3,
  "Category" : "Analytics",
  "BookName" : "Competing on Analytics",
  "Author" : "Thomas Davenport",
  "qty" : 8,
  "price" : 150,
  "rol" : 20,
  "pages" : 150
}
{
  "_id" : 4,
  "Category" : "Visualization",
  "BookName" : "Visualizing Data",
  "Author" : "Ben Fry",
  "qty" : 12,
  "price" : 325,
  "rol" : 6,
  "pages" : 450
}
{
```

```
"_id" : 5,  
"Category" : "Web Mining",  
"BookName" : "Learning R",  
"Author" : "Richard Cotton",  
"qty" : 5,  
"price" : 850,  
"rol" : 10,  
"pages" : 120  
}
```

Step. 3.2: Confirm the presence of above documents in the “books” collection.

Step. 3.3: Write map and reduce functions to split the books into the following two categories:

(a) Big Books

(b) Small Books

Books which have more than 300 pages should be in the big book category. Books which have less than 300 pages should be in the small book category.

Step. 3.4: Count the number of books in each category.

Step 3.5: Store the output as follow as documents in a new collection, called “Book_Result”.

Book Category Count of the books

“Big Books” 2

“Small Books” 3

4. To practice import, export and aggregation in MongoDB.

Step 4.1 Pick and public dataset from the site www.kdnuggets.com Convert it into CSV format. Make sure that you have at least two numeric columns.

Step 4.2 Use MongoImport to import data from the CSV format file into MongoDB collection named “MongoDBHandsOn” in test database.

Step 4.3: Identifying a grouping column.

Step 4.4: Compute the sum of the values in the first numeric column.

Step 4.5: Compute the average of the values in the second numeric column.

5. Exercise Python to MongoDB Connectivity using JuPyter Notebook.
6. Hands On any other NoSQL data stores of your choice such as HBase, Cassandra, etc.
7. Below is the example data set given and the logic explained to find mutual friends.

8. References:

Textbook. Big Data and Analytics – Seema Acharya and Subhashini Chellappan – Wiley India

Topics:

<https://www.mongodb.com>

<https://docs.mongodb.com/manual/>

<https://github.com/gusutabopb/imongo> for interacting to MongoDB by JupPyter as a client

[https://docs.mongodb.com/manual/reference/method/db.collection.aggregate/
#db.collection.aggregate](https://docs.mongodb.com/manual/reference/method/db.collection.aggregate/#db.collection.aggregate)

CE720 : Big Data & Analytics

Lab#7. Setting up a private cloud/cluster using commodity nodes.

1. **Aim :** Setting up a private cloud/cluster using commodity nodes.
2. **Objective :** Students will learn HADOOP and its components. Establish SSH User Equivalence and configure to experiment with Hadoop multiple node's cluster. Start/stop script and monitoring. Know the backbone of cloud platform.
3. **Description:** After experimenting with Single node/pseudo distributed cluster, moving ahead with a cluster of multiple nodes let us learn about internal engineering of cloud platforms. Know that it is now important to learn about distributed systems. Important thing about distributed systems is that they do not have common primary memory and a common cpu clock.

Know that in a cluster, all computers shall be able to communicate with itself over various protocols and to others internally without human intervention. User equivalence is one of the primary requirements of clusters. Communication mechanisms such as remote procedure call (rpc), message passing interface (mpi), secure shell (ssh), hypertext transfer protocol (http), etc are utilized for many services offered by cluster as a whole.

Many network fundamentals will be utilized when dealing with clusters. It involves naming the hosts, configuring ip, local dns via /etc/hosts file, trusting using ~/.ssh/known_hosts file, generating and adding keys using ssh-keygen, ssh-copy-id, etc.

4. Methodology:

Designate one of the computers as master (namenode) and others as workers (datanodes). Let us accept that master is having a local name server lookup enabled with name "hadoop-master" or having unique ip m.a.s.t. May replace everywhere in configuration name by ip. i.e. hadoop-master by m.a.s.t ip.

4.1 Make sure to have every computer uniquely identified in the network. To do so, as an admin you will need fix hostname and ip of every participating computers.

4.2 Next let local domain server be active via lookup file. /etc/hosts file must have entries of every computers as show in the sample below:

```
vi /etc/hosts
192.168.1.100 hadoop-master
192.168.1.101 hadoop-slave-1
192.168.1.102 hadoop-slave-2
192.168.1.103 hadoop-slave-3
```

4.3 Now generate keys on every computer and copy over to self and every other using ssh-keygen and ssh-copy-id.

Setup ssh in every node such that they can communicate with one another without any prompt for password. Execute following commands on each node.

```
$ ssh-keygen -t rsa -P
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-master
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave-1
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave-2
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop@hadoop-slave-3
$ chmod 0600 ~/.ssh/authorized_keys
```

This enables password less login internally by processes/modules. You shall test this manually or using a script to check all possible combinations and be able to ssh from one to the other without password/passphrase.

4.3 Edit the following configuration as compared to pseudo distributed mode.

/opt/hadoop/etc/hadoop.

core-site.xml

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://hadoop-master:9000</value>
</property>
```

hdfs-site.xml

```
<property>
  <name>dfs.hosts</name>
  <value>/opt/hadoop/etc/hadoop/slaves</value>
</property>
```

Do create a file /opt/hadoop/etc/hadoop/slaves containing ips of worker nodes line by line.

```
<property>
  <name>dfs.namenode.http-address</name>
  <value>hadoop-master:50070</value>
</property>

<property>
  <name>dfs.namenode.secondary.http-address</name>
  <value>hadoop-master:50090</value>
</property>
```

Only namenode will contain below directory storage in native/host file system:


```
<property>
  <name>dfs.namenode.name.dir</name>
  <value>/opt/hadoop/namenode_storage</value>
</property>
```

All datanodes will contain below directory storage in native/host file system:

```
<property>
  <name>dfs.datanode.data.dir</name>
  <value>/opt/hadoop/datanode_storage</value>
</property>
```

mapred-site.xml

```
<property>
  <name>mapreduce.jobhistory.address</name>
  <value>hadoop-master:10020</value>
</property>
<property>
  <name>mapreduce.jobhistory.webapp.address</name>
  <value>hadoop-master:19888</value>
</property>
```

yarn-site.xml

```
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>hadoop-master</value>
</property>
```

5. Example:

The diagram below shows that one of the clusters in laboratory once had 39 active nodes totaling storage capacity 11.8 TB (309.81 GB * 39)

Summary

Security is off.

Safemode is off.

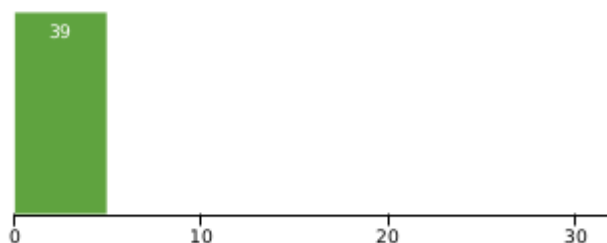
1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 198.83 MB of 404 MB Heap Memory. Max Heap Memory is 3.42 GB.

Non Heap Memory used 56.05 MB of 57.22 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	11.8 TB
Configured Remote Capacity:	0 B
DFS Used:	952 KB (0%)
Non DFS Used:	1.2 TB
DFS Remaining:	10 TB (84.76%)
Block Pool Used:	952 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	39 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	2 (Decommissioned: 0, In Maintenance: 0)

Datanode usage histogram



6. Implementation Notes:

If it causes trouble in internal or external services, you may want to check on hostname and canonical host name similarity using below java code for every participating node.

```
import java.net.InetAddress;
import java.net.UnknownHostException;

public class dns {

    public static void main(String[] args) throws UnknownHostException {
```

```

InetAddress addr = InetAddress.getLocalHost();

System.out.println(
String.format(
"IP:%s hostname:%s canonicalName:%s",
addr.getHostAddress(),      // The "default" IP address
addr.getHostName(),        // The hostname (from gethostname())
addr.getCanonicalHostName() // The canonicalized hostname (from resolver)
)
);
}

}

```

You may come up with shell scripts having loops, sshpass utility usage to perform cluster setup related operations.

i.e. To disable firewall may utilize following shell script:

```

export HISTCONTROL=ignorespace
labsubnet=192.168.32
passw=test
for i in {11..51};
do
    sshpass -p $passw ssh -o StrictHostKeyChecking=no root@$labsubnet.$i "systemctl stop
firewalld"
done;

```

If all configurations and network/ssh related settings are correct following shall show up the cluster up and ready:

```

cleanup
format
start-all-dfs
start-all-yarn

```

7. Exercise:

1. Monitor through browser urls.

```

http://hadoop-master:50070/dfshealth.html#tab-overview
http://hadoop-master:8088/cluster/nodes

```

2. To be able to store/retrieve files and directories into established private hadoop cluster of computers. Being able to judge that storage and retrieval is independent of the physical node where you are interacting from.
3. Run map-reduce program and check the distributed storage of results.

8. References:

<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html>

CE720 : Big Data & Analytics

Lab#8. Data analytics using Apache Spark

1. **Aim :** Learn and apply the architecture of Apache Spark for data analytics. Solve Word Count program requirement using Apache Spark. Apply the learning on a custom problem at hand (i.e. Man of the series award).
2. **Objective :** Students will learn various data cleaning, data transformation, data reductions techniques. Apache Spark is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters. Learn the architecture and working of Spark. Utilize the language of your choice to perform sample analytics. Note that when chosen HDFS as data storage behind the spark framework, integration of software components are learnt implicitly.

3. Description:

Spark is a robust and evolving engine for data analytics. Spark works with data using RDDs (Resilient Distributed Dataset) and data frames. The operations are immutable. The api allowed to process the data uses an approach of lazy processing. This allows spark to have various workflows checked out and optimize the processing. It uses graphs to represent the dependencies and operations, which are also utilized while optimizing.

4. Methodology:

Make sure to have Hadoop and Spark installed. As there happens to be data in hadoop distributed files systems which are retrieved and loaded as RDD/dataframe. It is then processed and results are recorded back to HDFS.

To create maven based scala example program structure, you may utilize below command:

```
mvn archetype:generate -DarchetypeGroupId=net.alchim31.maven -  
DarchetypeArtifactId=scala-archetype-simple
```

```
#Start Hadoop as we are using HDFS  
start-dfs.sh
```

```
#Start Spark master and monitor in the browser  
cd /opt/spark  
sbin/start-master.sh
```

```
http://localhost:8080
```

```
#Start Spark worker and attach to above master. Monitor in the browser about registered  
worker with the master.
```

```
sbin/start-worker.sh spark://localhost:7077
```

`http://localhost:8080`

Now know that Apache spark, as many other platforms do, support various ways to do implementation/data analytics.

Alternatively, you may download the data files from web resources, CLI:

`wget https://storage.googleapis.com/download.tensorflow.org/data/illiad/cowper.txt`

`wget https://storage.googleapis.com/download.tensorflow.org/data/illiad/derby.txt`

`wget https://storage.googleapis.com/download.tensorflow.org/data/illiad/butler.txt`

And put these files into HDFS corresponding locations. We are going to utilize the same for counting unique words and their frequencies.

Way#1 to run the scala word count

`spark-shell --master spark://localhost:7077`

Run the code within `src main App.scala` manually line by line.

Way#2

`spark-submit --master spark://localhost:7077 --deploy-mode=cluster --class=me.my.mine.App target/SparkWordCount-1.0-SNAPSHOT.jar`

Above, jar file can be generated using “mvn package” command line.

Graceful stopping of all spark and hadoop components shall be done as below:

`stop-worker.sh`

`stop-master.sh`

`stop-dfs.sh`

5. Example:

Word Count program using scala with Spark

`src/main/scala/me/my/mine/App.scala`

```
package me.my.mine
import org.apache.spark.SparkContext
import org.apache.spark.SparkConf
object App
{
    // Main Method
    def main(args: Array[String])
    {
```

```

        val conf = new SparkConf().setAppName("App").setMaster("spark://localhost:7077")
        val sc = new SparkContext(conf)
        val text_file = sc.textFile("hdfs://localhost:9000/user/sparkuser/my-dev/mapreduce/
WordCount/input")
        text_file.collect.foreach(println)
        val counts = text_file.flatMap(line => line.split("
")).map(word=>(word,1)).reduceByKey((a,b)=>a+b).sortByKey()
        counts.collect.foreach(println)
        counts.saveAsTextFile("hdfs://localhost:9000/user/sparkuser/my-dev/mapreduce/
WordCount/spark_output")
    }
}

```

6. Implementation Notes:

###Spark setup steps

```

chown -R sparkuser:sparkuser /opt/spark-3.3.0-bin-without-hadoop
ln -s /opt/spark-3.3.0-bin-without-hadoop /opt/spark
chown sparkuser:sparkuser /opt/spark

```

```

cd /opt/spark/conf
cp spark-env.sh.template spark-env.sh

```

```

vi spark-env.sh
export SPARK_LOCAL_IP=localhost
export HADOOP_HOME=/opt/hadoop
export HADOOP_CONF_DIR=/opt/hadoop/etc/hadoop
export YARN_CONF_DIR=/opt/hadoop/etc/hadoop
export SPARK_HOME=/opt/spark
export SPARK_CONF_DIR=/opt/spark/conf
export SPARK_LOG_DIR=/opt/spark/logs
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

```

```

chmod +x conf/spark-env.sh
export SPARK_HOME=/opt/spark
cp spark-defaults.conf.template spark-defaults.conf
vi spark-defaults.conf
spark.master                spark://localhost:7077
spark.eventLog.enabled      true
spark.eventLog.dir          hdfs://localhost:9000/user/sparkuser/spark/eventLog
spark.serializer            org.apache.spark.serializer.KryoSerializer
spark.driver.memory         1g
spark.executor.extraJavaOptions -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two
three"

```

7. Exercise:

1. Download dataset using url <https://www.kaggle.com/datasets/nowke9/ipldata?select=matches.csv> and perform data analytics. i.e. the man of the series award

8. References:

<https://spark.apache.org/>

<https://www.kaggle.com>

sample data source matches.csv:

```
id,season,city,date,team1,team2,toss_winner,toss_decision,result,dl_applied,winner,win_by_runs,win_by_wickets,player_of_match,venue,umpire1,umpire2,umpire3
1,2017,Hyderabad,05-04-2017,Sunrisers Hyderabad,Royal Challengers Bangalore,Royal Challengers Bangalore,field,normal,0,Sunrisers Hyderabad,35,0,Yuvraj Singh,"Rajiv Gandhi International Stadium, Uppal",AY Dandekar,NJ Llong,
2,2017,Pune,06-04-2017,Mumbai Indians,Rising Pune Supergiant,Rising Pune Supergiant,field,normal,0,Rising Pune Supergiant,0,7,SPD Smith,Maharashtra Cricket Association Stadium,A Nand Kishore,S Ravi,
3,2017,Rajkot,07-04-2017,Gujarat Lions,Kolkata Knight Riders,Kolkata Knight Riders,field,normal,0,Kolkata Knight Riders,0,10,CA Lynn,Saurashtra Cricket Association Stadium,Nitin Menon,CK Nandan,
4,2017,Indore,08-04-2017,Rising Pune Supergiant,Kings XI Punjab,Kings XI Punjab,field,normal,0,Kings XI Punjab,0,6,GJ Maxwell,Holkar Cricket Stadium,AK Chaudhary,C Shamshuddin,
5,2017,Bangalore,08-04-2017,Royal Challengers Bangalore,Delhi Daredevils,Royal Challengers Bangalore,bat,normal,0,Royal Challengers Bangalore,15,0,KM Jadhav,M Chinnaswamy Stadium,,,
6,2017,Hyderabad,09-04-2017,Gujarat Lions,Sunrisers Hyderabad,Sunrisers Hyderabad,field,normal,0,Sunrisers Hyderabad,0,9,Rashid Khan,"Rajiv Gandhi International Stadium, Uppal",A Deshmukh,NJ Llong,
7,2017,Mumbai,09-04-2017,Kolkata Knight Riders,Mumbai Indians,Mumbai Indians,field,normal,0,Mumbai Indians,0,4,N Rana,Wankhede Stadium,Nitin Menon,CK Nandan,
8,2017,Indore,10-04-2017,Royal Challengers Bangalore,Kings XI Punjab,Royal Challengers Bangalore,bat,normal,0,Kings XI Punjab,0,8,AR Patel,Holkar Cricket Stadium,AK Chaudhary,C Shamshuddin,
9,2017,Pune,11-04-2017,Delhi Daredevils,Rising Pune Supergiant,Rising Pune Supergiant,field,normal,0,Delhi Daredevils,97,0,SV Samson,Maharashtra Cricket Association Stadium,AY Dandekar,S Ravi,
10,2017,Mumbai,12-04-2017,Sunrisers Hyderabad,Mumbai Indians,Mumbai Indians,field,normal,0,Mumbai Indians,0,4,JJ Bumrah,Wankhede Stadium,Nitin Menon,CK Nandan,
```

p.s. Looks like last field of “umpire3” is empty for most records. That is why every line ends with comma delimiter.

CE720 : Big Data & Analytics

Lab#9. Machine learning using Apache Mahout.

1. **Aim** : Perform machine learning using Apache Mahout tool and explore possibilities for supervised and unsupervised learning. (i.e. Recommendation system)

2. **Objective** : Students will learn to apply various supervised and unsupervised algorithms of Machine Learning. Machine learning is a huge generic application with the growth of data sources and big data. The analytics performed is one way to utilize the data while learning from data using machine learning algorithms is another way. Experts involved in storing and processing data shall explore both the tracks.

3. Description:

Machine learning is trending for multiple reasons including vast usage of the World Wide Web

and services using smart devices. BigData and machine learning are connected by requirement that for machine learning underlying data source can itself be Big Data, which in turn distributed by nature.

Mahout is part of hadoop ecosystem to create scalable performance machine learning applications. Mahout is a distributed linear algebra framework for data scientists and others to quickly implement their own algorithms as quoted by the site.

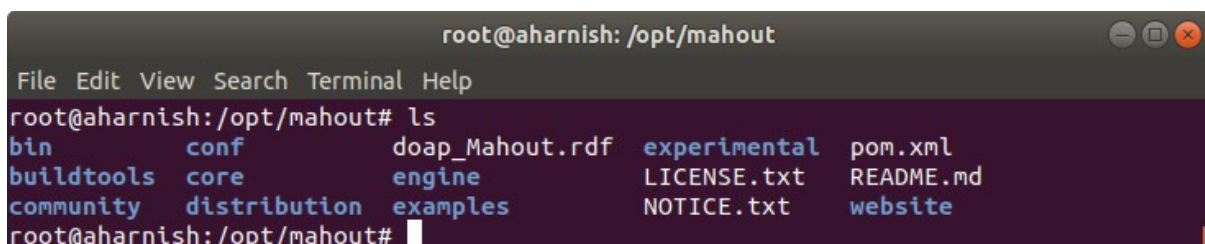
As of 0.13, Mahout achieves following algorithms:

- Distributed Linear Algebra
- Preprocessors
- Regression
- Clustering
- Recommenders

Optionally, you may download hadoop source and utilize or contribute on your personal computer. Mahout source uses apache maven as Software Project Management and comprehension tool.

4. Methodology:

Download and extract mahout.



```
root@aharnish: /opt/mahout
File Edit View Search Terminal Help
root@aharnish:/opt/mahout# ls
bin          conf         doap_Mahout.rdf  experimental  pom.xml
buildtools   core         engine           LICENSE.txt   README.md
community    distribution examples        NOTICE.txt   website
root@aharnish:/opt/mahout#
```

Mahout will build the following packages:

Apache Mahout
 Mahout Core
 Mahout Engine
 Mahout HDFS Support
 Mahout Community
 Mahout Spark CLI Drivers
 Apache Maven

pom.xml file is the configuration file located in the project folder. It contains information about

the dependencies, etc. It is a smart tool which does further dependency checks based on entries into pom.xml and fetches live from the web. Make sure to have connected to net if you are expecting live fetch. The appropriate version number of module can be identified quickly from mvn repository site. i.e. for mahout-core package

Mahout java library can be used within a project after adding dependencies into pom.xml

```

<dependency>
  <groupId>org.apache.mahout</groupId>
  <artifactId>mahout-core</artifactId>
  <version>0.9</version>
</dependency>
<dependency>
  <groupId>org.apache.mahout</groupId>
  <artifactId>mahout-math</artifactId>
  <version>0.13.0</version>
</dependency>
<dependency>
  <groupId>org.apache.mahout</groupId>
  <artifactId>mahout-integration</artifactId>
  <version>0.13.0</version>
</dependency>

```

5. Example:

The demo application by TutorialsPoint website can be downloaded and tested after project structure and mvn commands as below.

```

hadoop@hadoop-clone:~/JMP/mahoutdemo
File Edit View Search Terminal Help
[hadoop@hadoop-clone mahoutdemo]$ java -jar target/demo-mahout-0.0.1-SNAPSHOT.jar
19/08/31 01:20:00 INFO file.FileDataModel: Creating FileDataModel for file data
19/08/31 01:20:00 INFO file.FileDataModel: Reading file info...
19/08/31 01:20:00 INFO file.FileDataModel: Read lines: 22
19/08/31 01:20:00 INFO model.GenericDataModel: Processed 5 users
User Neighborhood information
User ID #2
RecommendedItem[item:3, value:4.5]
RecommendedItem[item:4, value:4.0]
[hadoop@hadoop-clone mahoutdemo]$

```

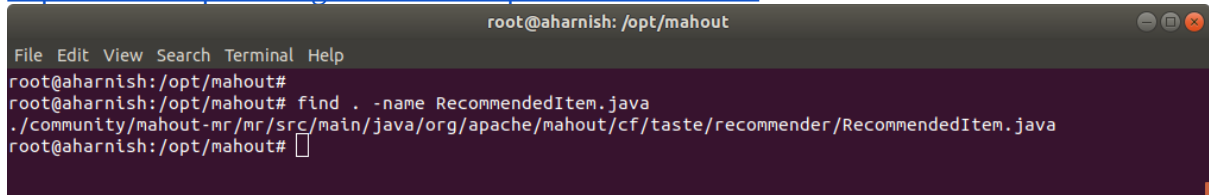
6. Implementation Notes:

May explore web based documentation and java source code from mahout downloads:

<http://mahout.apache.org/docs/0.13.0/api/docs/>

<http://mahout.apache.org/docs/0.13.0/api/docs/mahout-mr/org/apache/mahout/cf/taste/recommender/RecommendedItem.html>

<http://mahout.apache.org/docs/0.13.0/api/docs/mahout-mr/>

A terminal window titled 'root@aharnish: /opt/mahout' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
root@aharnish:/opt/mahout#  
root@aharnish:/opt/mahout# find . -name RecommendedItem.java  
./community/mahout-mr/mr/src/main/java/org/apache/mahout/cf/taste/recommender/RecommendedItem.java  
root@aharnish:/opt/mahout#
```

Setup Commands references:

apt install git

apt install maven

git clone <https://github.com/apache/mahout.git> mahout

cd /opt/mahout; find . -name RecommendedItem.java

Mahout

<https://www.tutorialspoint.com/mahout/>

<http://mahout.apache.org/>

<http://mahout.apache.org/docs/0.13.0/api/docs/>

<http://mahout.apache.org/docs/0.13.0/api/docs/mahout-mr/org/apache/mahout/cf/taste/recommender/RecommendedItem.html>

<http://mahout.apache.org/docs/0.13.0/api/docs/mahout-mr/>

Maven Commands:

<https://maven.apache.org/run.html>

mvn clean

mvn compile

mvn install

mvn package

java -jar target/demo-mahout-0.0.1-SNAPSHOT.jar

compile, package and java -jar .. mostly you would need to do if jars are already installed locally in target folder.

<https://maven.apache.org/>

[Maven need internet connectivity if the dependency added in pom.xml or any other updates need additional jar]

7. Exercise:

1. Run recommendation of a sample dataset using Mahout Library.
2. Try out a machine learning algorithm supported by Mahout for a dataset from kaggle.

8. References:

<https://mahout.apache.org/>

<https://mahout.apache.org/users/basics/algorithms.html>

<https://mahout.apache.org/general/downloads>

<https://maven.apache.org/>

<https://mvnrepository.com/artifact/org.apache.mahout/mahout-core>

https://www.tutorialspoint.com/mahout/mahout_environment.htm

<https://mahout.apache.org/docs/0.13.0/api/docs/>

<http://mahout.apache.org/developers/buildingmahout>

Sample data set as provided by tutorials point website:

User,Item,PreferencesDetailsOfAProduct

1,00,1.0

1,01,2.0

1,02,5.0

1,03,5.0

1,04,5.0

2,00,1.0

2,01,2.0

2,05,5.0

2,06,4.5

2,02,5.0

3,01,2.5

3,02,5.0

3,03,4.0

3,04,3.0

4,00,5.0

4,01,5.0

4,02,5.0

4,03,0.0

5,00,5.0

5,01,5.0

5,02,5.0

5,03,0.0

CE720 : Big Data & Analytics

Lab#10. Reporting and visualization for decision making.

1. **Aim** : Final step towards data analytics, “Reporting/Presentation”; Utilize reports/visualizations/charts/integration tools to be able to understand the findings. (Identifying trends using word/tag cloud).

2. **Objective** : Learn various types of digital data and how to deal with them till the last step of presentation. Big Data analytics related tasks to present findings to leadership or relevant stakeholders in an easy/convenient way. The reporting tools available shall be explored to meet the requirements. There are many obvious requirements of being able to deal with various format downloads, extracts out of reporting mechanisms, perform pagination, search and filter, grouping ,interactive sorting, etc.

3. **Description:**

Presentation of information in a required and user friendly manner is a must achievement for any software. Having data recorded into various data marts, data warehouses, etc is the first thing to have to be able to perform data analytics and second thing is preparing various reports out of information/knowledge extracted.

JasperSoft community edition provides an advanced way to achieve reporting which can also be embedded easily in java based applications. Here, the first example taken prepares the dataset and then using JasperSoft Studio achieves a feature-rich report.

4. **Methodology:**

- Start mongod

Know that JasperSoft Studio is going to communicate to mongoDB (in this exercise) for fetching data. 'mongod' utility has to be running all the time during the report development and preview.

- Load the data into mongoDB using mogoimport utility

mongoimport

--db=mng

--collection=student_master

--type=csv

--headerline --file="SourceData.csv"

```

> db;
mng
> db.student_master.find().count();
3796
> db.student_master.find().limit(1).pretty();
{
  "_id" : ObjectId("5cf6c2e623a0ea5de475874e"),
  "Candidate Id" : "C0002",
  "Employee Id" : "E0002",
  "First Name" : "Shailendra",
  "Middle Name" : "Kumar",
  "Last Name" : "",
  "DegreePercentage" : 72.55,
  "12th/DiplomaPercentage" : 75.77,
  "SSCPercentage" : 80.1,
  "University Name" : "RGPRV",
  "Native State" : "Madhya Pradesh",
  "Background(CS\NCS)" : "CS",
  "Industrial Training" : "Y",
  "TrainingExam1Percentage" : 78,
  "Exam1Grade" : "B",
  "TrainingExam2Percentage" : 0,
  "Exam2Grade" : "C",
  "IntoProduction" : 0,
  "Channel" : "Walk in Interviews"
}
> 

```

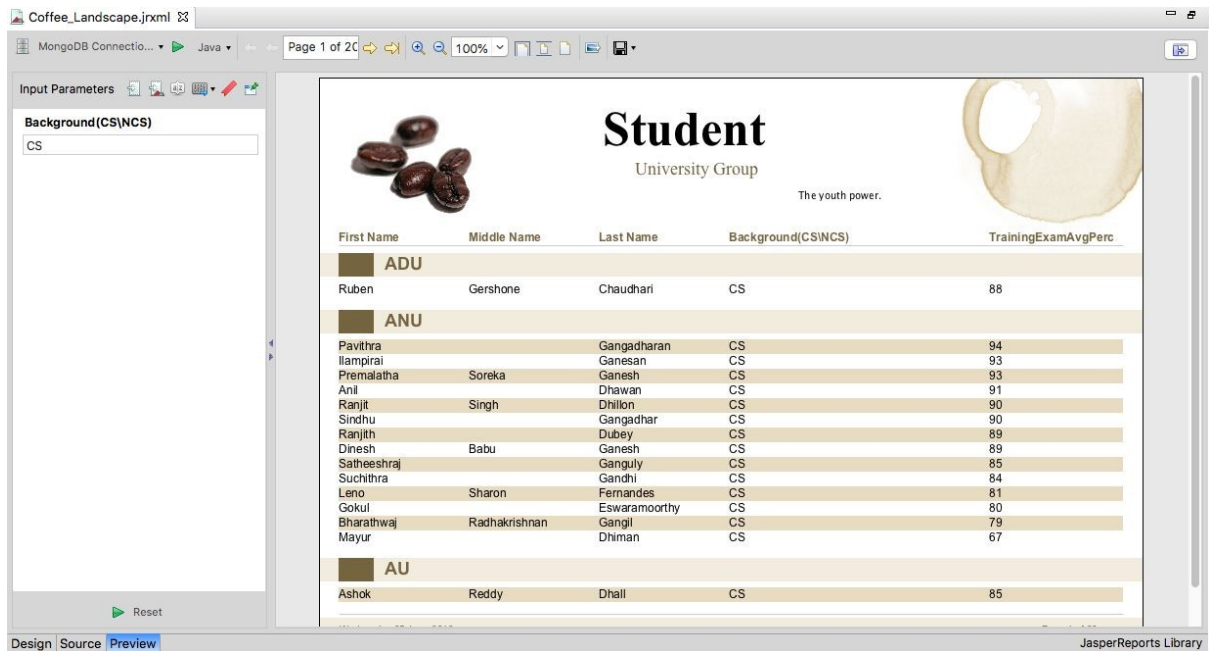
● Understanding Business Requirement

Provide multi-pages report of students having following features:

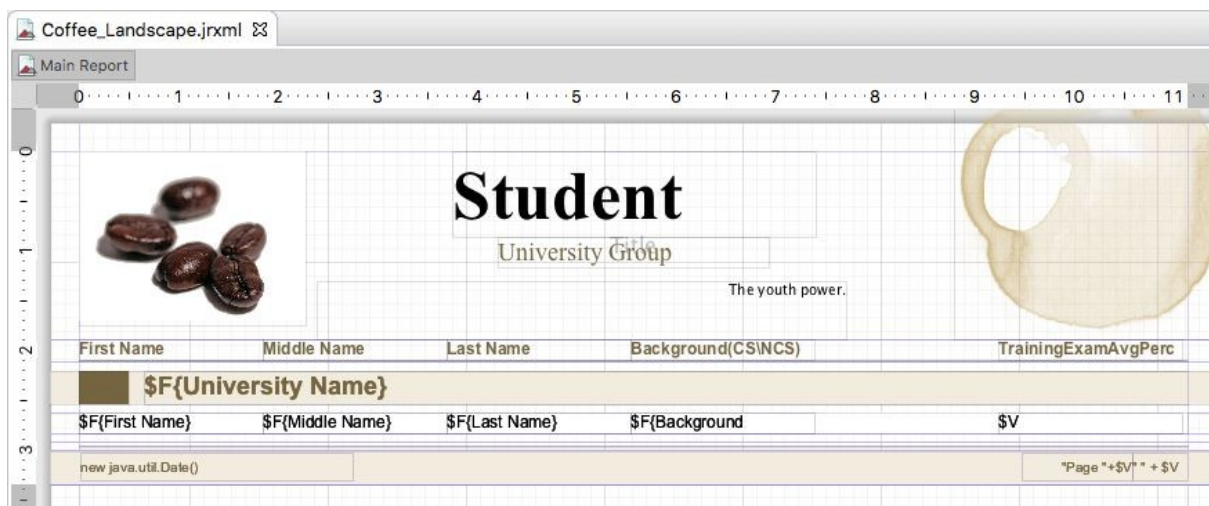
- Show First Name, Middle Name, Last Name, Background (CS/NCS) and Training Exam Average Percentage (Avg. of two exams).
- Records shall be grouped by the University Name.
- Records shall be sorted with the group based on Training Exam Average Percentage in descending order.
- User shall be able to run interactively for CS records or NCS records.
- Pagination shall be supported.

5. Example:

A report which allows end user/leadership to view the aggregated data with better perspective. Here, information about university students. Report enables graphical layout with aggregated results with sorting and pagination features.

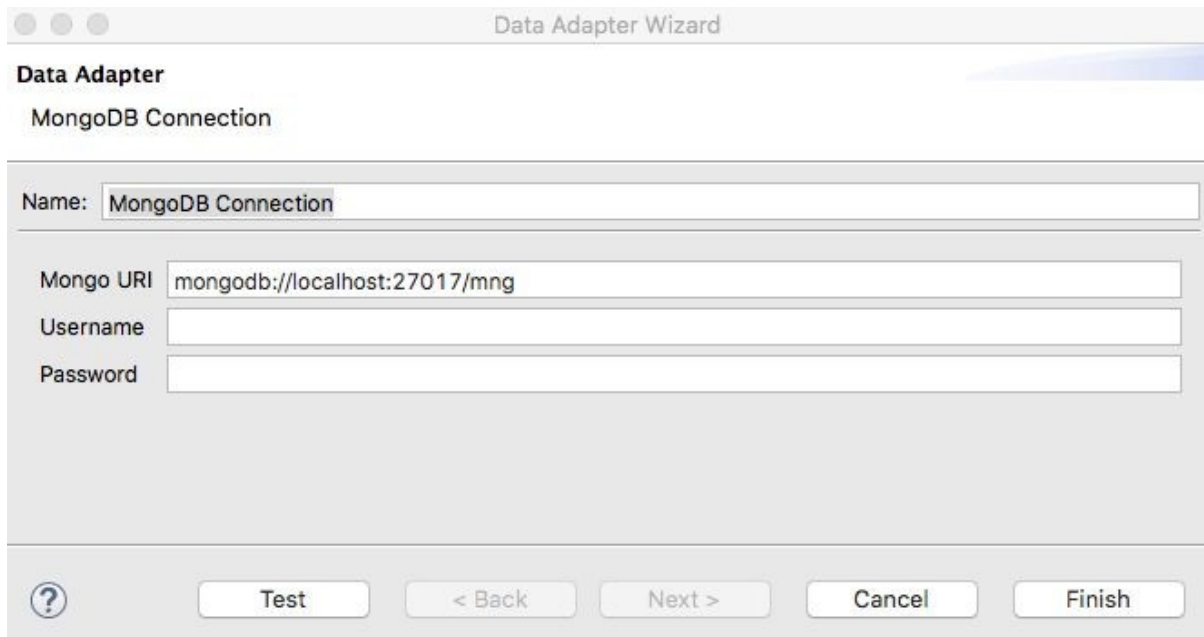


Start JasperSoft Studio IDE. Aim to achieve below like report. HighLevel To Do.



Understanding JasperSoft Report needed Technicality

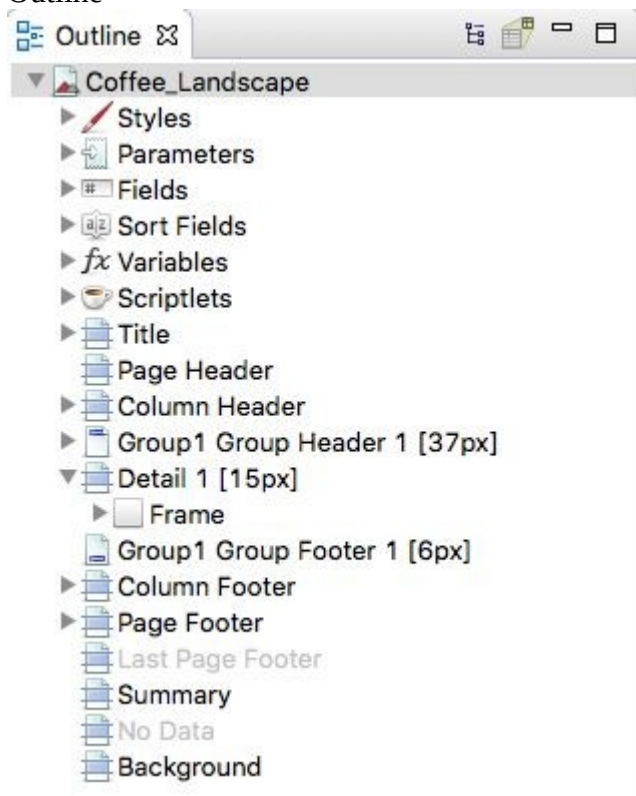
- JasperReports Project and Project Explorer name MyReports
- JasperReports “New Report Wizard” name Coffee_Landscape.jrxml
- Data Source
Data Adapter – MongoDB Connection
name MongoDB Connection
url mongodb://localhost:27017/mng



The image shows a 'Data Adapter Wizard' window with the title 'Data Adapter Wizard'. Below the title bar, it says 'Data Adapter' and 'MongoDB Connection'. There is a 'Name:' label followed by a text box containing 'MongoDB Connection'. Below this, there are three more text boxes: 'Mongo URI' with the value 'mongodb://localhost:27017/mng', 'Username' (empty), and 'Password' (empty). At the bottom, there is a row of buttons: a help button (question mark icon), 'Test', '< Back', 'Next >', 'Cancel', and 'Finish'.

- Repository Explorer

Outline



- MongoDB Query

May create parameter first named “ Background(CS\NCS” or write the query without \$P line.

Dataset (provided as csv towards end of manual) and Query Dialog


```
{
  'collectionName':'student_master',
  'findQuery' : {
    {
      'TrainingExam1Percentage':{'$ne':''},
      'TrainingExam2Percentage':{'$ne':''},
      'Background(CS\\NCS)':{'$P{Background(CS\\NCS)
    }
  }
}
```

- Fields (Dataset)

Fields are mapped to dataset. i.e. field of collection from mongoDB.

```
<field name= "University Name" class= "java.lang.String" / >
<field name= "First Name" class= " java.lang.String" />
<field name= " Middle Name" class= "java.lang.String" / >
<field name= " Last Name" class= "java.lang.String" / >
<field name= " Background(CS\\NCS)" class= " java.lang.String" />
<field name= " TrainingExam1Percentage" class= " java.lang.String" / >
<field name= " TrainingExam2Percentage" class= " java.lang.String" / >
```

- Group By

```
<groupExpression> <![CDATA[ $F{University Name} ]]> </groupExpression>
```

- Sort Field

```
<sortField name= "University Name" />
<sortField name= "TrainingExamAvgPercentage" order= "Descending" type= "
Variable" />
```

- Properties Window

- Variables – Expressions

Variables in JasperReport are extra memory created and used to do arithmetic on dataset. i.e. here Avg Percentage is not the actual field from dataset but is calculated runtime based on the expression provided.

```
<variable name= " TrainingExamAvgPercentage" class= "java.lang.String" >
  <variableExpression> <![CDATA[
    String.valueOf((Integer.parseInt($F{TrainingExam1Per
    centage}))+Integer.parseInt($F{TrainingExam2Percentage}))/2) ] ]>
  </variableExpression>
</variable>
```

- Report Parameters

Parameters in JasperReport are placeholders which can be mapped to query. Here the parameters are provided values at runtime if configured such a way. i.e. Report in here need user input 'CS' or 'NCS' to generate report of respective records only.

```
<parameter name= "Background(CS\NCS)" class= "java.lang.String" >
    <parameterDescription> <![CDATA[ CS Or NCS ] ]> </parameterDescription>
</parameter>
```

- Design-Source-Preview

The IDE provides a GUI drag and drop way to design the report. Internally which maps to xml-java source. Preview executes and displays embedded report within IDE itself. To be able to embed into external java application there are separate steps to look into.

- Palette

Composite Elements

The page count, page numbers, etc are considered to be composite elements. They can be used as required in the reporting.

6. Implementation Notes:

Note that the xml/source file of reporting is a document which will have various changes over the period of time. The way developers use git/svn/subversion/cvs like version control tools, even engineers working on reports are required to use version control softwares for the same reason.

7. Exercise:

1. Create the described report or any other to explore various functionalities.
2. University Name on X-axis, Avg Percentage by various students on Y-Axis. Use bar chart.
3. Embed the JasperReport to your java application/web.
4. Populate dropdown for parameters.

8. References:

Textbook. Big Data and Analytics – Seema Acharya and Subhashini Chellappan – Wiley India

Topics: JasperReport using Jaspersoft

<https://community.jaspersoft.com/wiki/designing-report-jaspersoft-studio>

<https://community.jaspersoft.com/wiki/jaspersoft-mongodb-query-language>

Sample Data Set from SourceData.csv with header line:-

CandidateId,EmployeeId,FirstName,MiddleName,LastName,DegreePercentage,12th/
DiplomaPercentage,SSCPPercentage,UniversityName,NativeState,Background(CS\
NCS),IndustrialTraining,TrainingExam1Percentage,Exam1Grade,TrainingExam2Percentage,Exam2Grade,IntroP
roduction,Channel

Lab Manual: Big Data & Analytics

C0001,E0001,Supriyo,,,74.4,75,79,VTU ,Madhya Pradesh,CS,Y,86,A,81,A,1,Campus Recruitment
C0002,E0002,Shailendra,Kumar,,72.55,75.77,80.1,RGPRV,Madhya Pradesh,CS,Y,78,B,0,C,0,Walk in Interviews
C0003,E0003,Prameet,,,73.7,72.6,80.4,RGPRV,Madhya Pradesh,CS,Y,91,A,88,A,1,Online
C0004,,Priyank,Kumar,,77.15,76.44,84.4,RGPRV,Madhya Pradesh,CS,Y,,,,,Campus Recruitment
C0005,,Aarti,,,75.1,65.2,88.4,RGPRV,Madhya Pradesh,CS,Y,,,,,Referral
C0006,E0006,Gaurav,,,68.65,78.67,87.2,RGPRV,Madhya Pradesh,CS,Y,80,A,85,A,1,Campus Recruitment
C0007,E0007,Shraddha,,,72.6,77.6,80.6,RGPRV,Madhya Pradesh,CS,Y,86,A,97,A,1,Campus Recruitment
C0008,E0008,Prachi,,,71.68,64,88.29,RGPRV,Madhya Pradesh,CS,Y,90,A,85,A,1,Campus Recruitment
C0009,,Shankey,,,66.1,63.6,61.33,HPUNS,Madhya Pradesh,CS,Y,,,,,Referral
C0010,E0010,Prateek,,,78.37,68.61,86.4,RGPRV,Madhya Pradesh,CS,Y,74,B+,93,A,1,Campus Recruitment
C0011,E0011,Neha,,,79.3,74.2,86.4,RGPRV,Madhya Pradesh,CS,Y,75,B+,90,A,1,Walk in Interviews
,,Rahul,,,65.05,85.33,91,RGPRV,Madhya Pradesh,,,,,,Walk in Interviews
C0013,E0013,K,Guru,,75.5,77.5,79.9,RGPRV,Madhya Pradesh,CS,Y,91,A,89,A,1,Walk in Interviews
C0014,E0014,Manoj,,,68.7,74.2,76.6,RGPRV,Madhya Pradesh,CS,N,92,A,87,A,1,Campus Recruitment
C0015,E0015,Sushant,,,67.6,67.6,65,RGPRV,Madhya Pradesh,CS,Y,77,B+,78,B+,1,Walk in Interviews
,,Tanuj,,,71.67,84,80.6,RGPRV,Madhya Pradesh,,,,,,Referral
C0017,,Sreekanth,Cheriyannath,,77.54,80.4,78.6,CALT ,Kerala,CS,N,,,,,Online
,,Rishy,,,71.2,79,87,CALT ,Kerala,,,,,,Online
C0019,,Chinnu,Elsa,,70.7,80.8,92.5,CALT ,Kerala,CS,N,,,,,Online
,,Sruthi,Lakshmanan,,64.3,82.66,76.8,CALT ,Kerala,,,,,,Referral
C0021,E0021,Sapnu,Sangeet,,68.26,86.5,88.83,CALT ,Kerala,CS,N,60,C,63,C,0,Online
C0022,E0022,Sumi,V,,73.2,79.67,91.6,CALT ,Kerala,CS,N,65,B,66,B,1,Referral
C0023,E0023,Neethu,,,69.88,86.5,85.5,CALT ,Kerala,CS,N,76,B+,80,A,1,Online
C0024,E0024,Aswathi,Davies,,72.7,84.8,84.4,CALT ,Kerala,CS,N,93,A,80,A,1,Referral
C0025,E0025,Sruthi,Saseendran,,72.52,74.4,87.73,CALT ,Kerala,CS,N,85,A,82,A,1,Online